IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.047 ⁜ Vol. 10, Issue 9, September 2022

DOI: 10.17148/IJIREEICE.2022.10915

SMART CROP HARVESTING SYSTEM

Aasya P A[1], Alfiya M B[2], Namitha K Nair[3], Rahiya N H[4], Dr. Abhiraj T K[5]

Engineer, Electrical and Electronics Engineering, ICET, Ernakulam, India[1-4]

HOD, Electrical and Electronics Engineering, ICET, Ernakulam, India[5]

Abstract: The proposed system is a crop harvesting system based on the Internet of Things (IoT), object recognition and automation technologies. The system helps to reduce the manual labour of the farmers by automating the identification and plucking of ripened fruit without a direct visit to the field. The system aims to reduce the tiresome tasks in the agricultural scenario. Harvesting using robotic arms reduces the amount of time spend by farmers in performing repetitive task. The proposed system includes object detection in the server side and harvesting using robotic arm in the hardware side.

Keywords: IoT (Internet of Things), Robotic arm

I. INTRODUCTION

Crop harvest and crop profits are indirectly affected by factors such as competition brought on by globalization, extreme climate change, and population aging. Coupled with aggressive changes in agriculture in recent years, fierce competition between markets has led to demands for higher product quality. Accordingly, an increasing number of farmers are being forced to adapt to new technologies to meet rising quality demands and achieve lower costs. The agricultural sector is being forced to adopt modern methods, among which the combined use of sensors with IoT technologies is the most common method. This method enables farmers to quickly understand the conditions of vast farmlands and immediately implement appropriate measures.

Farming requires extensive experience, and every agricultural skill is the outcome of a farmer's dedicated labour. However, the trends of modern technology prevent some of the skills from being documented and passed on to the next generation; as a result, many are being lost. Accordingly, this study endeavoured to develop a system that can be easily adopted by the next generation of farmers. Through smart recognition models trained using neural networks, the proposed system can determine whether crops are ready for harvest, after which robotic arms can be employed to harvest the crops.

II. OBJECTIVES OF THE PROJECT

Smart agriculture is a future trend of technology. The crop maturity can be determined through object detection by training neural network modals. The robotic arms can be employed to harvest the matured crops. This saves farmers time and energy while achieving effective plant growth control. Smart agriculture is an approach towards the better agricultural management aims to the present utilization of advanced technologies towards the better management of agriculture requirements. Smart agriculture proved its viability for the better management of agricultural requirements. Proposed system can be easily adopted by the next generation of farmers. Robotic systems can be used to replace a wide range of tedious tasks that require manual work, including tiling of land, planting, watering, and harvesting. While not entirely robotic, a combine harvester is an example of machinery that can automatically cut wheat, and separate the grain from the stems. Automation increases the productivity of agricultural machinery by increasing efficiency, reliability, and precision, and reducing the need of human intervention. This is achieved by adding sensors and controls.

III. PROPOSED SYSTEM

The proposed system is a harvesting system based on the Internet of Things technology and Smart image recognition. Crop maturity can be determined through object detection by training neural network models, and mature crops can then be harvested using robotic arms. Following the execution of object detection on images, the pixel coordinates of the central point of the target crop in the image are used as neural network input, whereas the robotic arm is regarded as the output side. The model performs crop detection by collecting and tagging images. This study adopted a neural network to predict the movement of a robotic arm resolve problems concerning robotic arm designs instead of using kinematic chain algorithm. Success rate of picking task is more accurate.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.047 ⁜ Vol. 10, Issue 9, September 2022

DOI: 10.17148/IJIREEICE.2022.10915

IV. BLOCK DIAGRAM

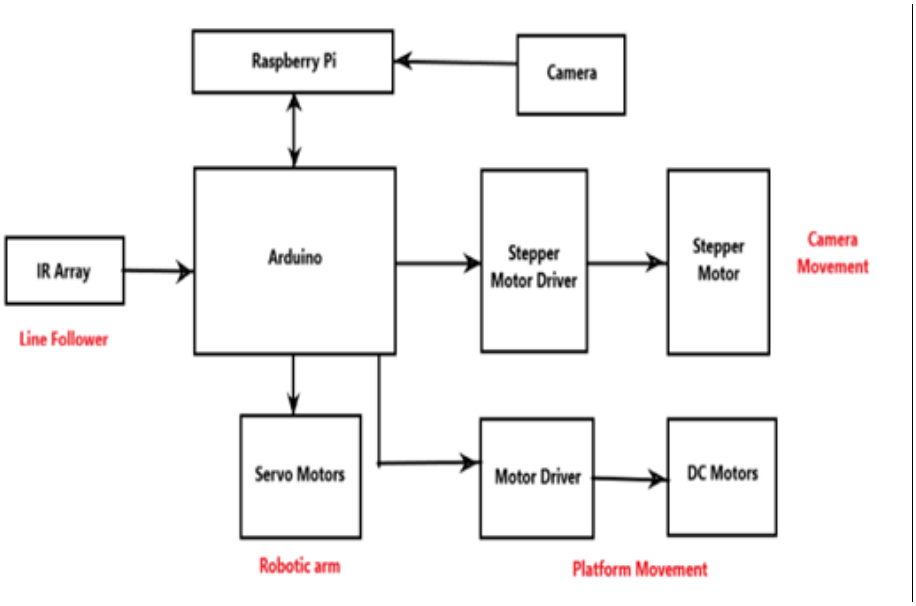


Fig. 1 Block diagram of the system

V. HARDWARE REQUIREMENTS

1. Raspberry pi 4 model B
2. Arduino Uno R3
3. TR213 Servomotors
4. 28BYJ-48- 5V Stepper Motor
5. D C motors
6. Logitech C-170-5 MP- Black Webcam
7. IR sensor array

VI. WORKING OF THE SYSTEM

The architecture can be divided into two parts, namely object detection and arm control. The object detection system is largely responsible for image detection. Firstly camera captures the image of tomato and sends to Raspberry Pi. We are using USB Webcam, and from webcam the signals will pass through the USB port in the form of array of bytes and thus this data will reach the Raspberry Pi. Then Raspberry Pi gives the result to Arduino. There is bidirectional communication between Arduino and Raspberry Pi and serial communication is taking place between them. Here also USB cable is providing for connecting between Arduino and Raspberry Pi. Object detection is performed and system tolerance is attained primarily by adjusting training images and training different numbers of images in batches to test the model network's detection accuracy. Object detection is performed by YOLO v3.

Two classifications of tomatoes are made based on the size and color. Then robotic arm can be employed to harvest the matured crops. Servomotor is used in all joints of robotic arm for enabling arm movement. Between servomotor and Arduino there is PWM (Pulse Width Modulation) signal transmission. And we can handle pulse width by using the servo library. Stepper motor is used for up and down movement of camera. Here we have to use stepper motor driver for interfacing stepper motor with Arduino. Because we cannot directly connect stepper motor with Arduino. Direct interfacing without using driver may cause damage to the board, and also Arduino cannot source that much current required to drive motor. Another problem is that arduino faces problem due to back emf of motor if directly connected. So to driver, we are giving high and low input. Then driver will convert it and provide the current and voltage necessary for stepper. Here line follower is performed using IR arrays. We are expecting a digital value from IR Array. Means, for

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

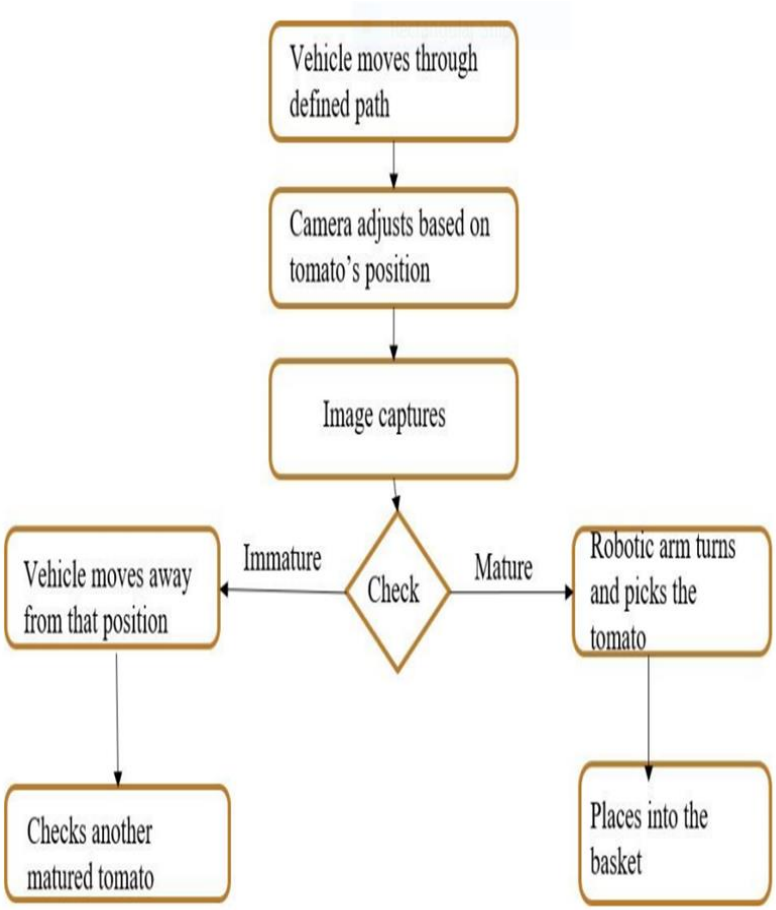
International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.047 ⁜ Vol. 10, Issue 9, September 2022

DOI: 10.17148/IJIREEICE.2022.10915

a black line we get 0 volt as output, then for a white line we get 5 volt as output. In this way, line follower is working. Platform movement is enabled by DC motors. Here also we are using driver for the interfacing between Johnson motor and Arduino. To driver we are giving high low signal input. And then driver will provide voltage and current to DC motors.

VII. FLOW CHART



VIII. PROGRAM CODES

A. PROGRAM CODE FOR LINE FOLLOWER

```
#include <AFMotor.h>
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_BMP280.h>
#define BMP_SCK (13)
#define BMP_MISO (12)
#define BMP_MOSI (11)
#define BMP_CS (10)

const int RPI_IN=13;//input from raspberry pi
const int LDR= A0;//ldr sensor at pin AO
const int RPI_OUT=2;// output to raspberry pi

// naming all the motors
```

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.047 ⁜ Vol. 10, Issue 9, September 2022

DOI: 10.17148/IJIREEICE.2022.10915

```
AF_DCMotor rightBack(1);
AF_DCMotor rightFront(2);
AF_DCMotor leftFront(3);
AF_DCMotor leftBack(4);

Adafruit_BMP280 bmp; // I2C communication for pressure sensor

void setup() {
 Serial.begin(9600);
 Serial.println("Initializing....");

 pinMode(LDR,INPUT);// initializid ldr sensor as input
 pinMode(RPI_IN, INPUT);
 pinMode(RPI_OUT, OUTPUT);

 //Initialization for pressure sensor
 Serial.println(F("BMP280 test"));
 if (!bmp.begin()) {
  Serial.println(F("Could not find a valid BMP280 sensor, check wiring!"));
  while (1);
 }
 /* Default settings from datasheet. */
 bmp.setSampling(Adafruit_BMP280::MODE_NORMAL, /* Operating Mode. */
            Adafruit_BMP280::SAMPLING_X2, /* Temp. oversampling */
            Adafruit_BMP280::SAMPLING_X16, /* Pressure oversampling */
            Adafruit_BMP280::FILTER_X16, /* Filtering. */
            Adafruit_BMP280::STANDBY_MS_500); /* Standby time. */

 //Setting speed for all motor
 rightBack.setSpeed(200);
 rightFront.setSpeed(200);
 leftFront.setSpeed(200);
 leftBack.setSpeed(200);

 //Releasing power of all motor
 rightBack.run(RELEASE);
 rightFront.run(RELEASE);
 leftFront.run(RELEASE);
 leftBack.run(RELEASE);

 delay(1000);
 Serial.println("System Ready");
 delay(1000);
}

// Function to check ldr sensor
int Line(){
 int ldr;
 if(analogRead(LDR)<100){
  ldr=1;
  Serial.println("Out of Path");
 }
 else{
  ldr=0;
  Serial.println("Right on Track");
 }
 return ldr;
}
```

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.047 ⁜ Vol. 10, Issue 9, September 2022

DOI: 10.17148/IJIREEICE.2022.10915

```
//Function to run forward
void roll(){
  rightBack.run(FORWARD);
  rightFront.run(FORWARD);
  leftFront.run(FORWARD);
  leftBack.run(FORWARD);
  Serial.println("Moving Forward");
}

//Function to stop all motors
void brake(){
  rightBack.run(RELEASE);
  rightFront.run(RELEASE);
  leftFront.run(RELEASE);
  leftBack.run(RELEASE);
  Serial.print("Movement Halted");
}

//Function to turn right
void right(){
  rightFront.run(FORWARD)

 rightBack.run(FORWARD);
  leftFront.run(BACKWARD);
  leftBack.run(BACKWARD);
  Serial.println("Turning Right");
}

//Function for pressure sensor
int pressure(int threshold){
  int pres,data;
  Serial.print(F("Temperature = "));
  Serial.print(bmp.readTemperature());
  Serial.println(" *C");
  Serial.print(F("Pressure = "));
  Serial.print(bmp.readPressure());
  Serial.println(" Pa");
  Serial.print(F("Approx altitude = "));
  Serial.print(bmp.readAltitude(1013.25)); /* Adjusted to local forecast! */
  Serial.println(" m");
  Serial.println();
  data=int(bmp.readPressure());
  if (data>threshold){
    pres =1;
  }
  else {
    pres=0;
  }
  return pres;
}

void loop() {
  if(Line()==1)//checking the path
  {
    //dis-engaging all motors before turning
    brake();
    while(Line()==1){
```

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.047 ⁜ Vol. 10, Issue 9, September 2022

DOI: 10.17148/IJIREEICE.2022.10915

```
        //turning right till it find the path
        right();
        }
      }
      else{
       roll();
      }
      if (RPI_IN==1)//brake when raspberry pi sees the object
      {
       brake();
      }
      else{
       roll();
      }
      if (pressure(200000)==1){
      //checking arm has grabbed object to inform rpi

       digitalWrite(RPI_OUT,HIGH);
      }
      else{
       digitalWrite(RPI_OUT,LOW);
      }
  }
```

B. PROGRAM CODE FOR OBJECT DETECTION

```
import argparse
import sys
import time

import cv2
from tflite_support.task import core
from tflite_support.task import processor
from tflite_support.task import vision
import utils


def run(model: str, camera_id: int, width: int, height: int, num_threads: int,
        enable_edgetpu: bool) -> None:
  """Continuously run inference on images acquired from the camera.
  Args:
    model: Name of the TFLite object detection model.
    camera_id: The camera id to be passed to OpenCV.
    width: The width of the frame captured from the camera.
    height: The height of the frame captured from the camera.
    num_threads: The number of CPU threads to run the model.
    enable_edgetpu: True/False whether the model is a EdgeTPU model.
  """

  # Variables to calculate FPS
  counter, fps = 0, 0
  start_time = time.time()

  # Start capturing video input from the camera
  cap = cv2.VideoCapture(camera_id)
  cap.set(cv2.CAP_PROP_FRAME_WIDTH, width)
  cap.set(cv2.CAP_PROP_FRAME_HEIGHT, height)
```

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.047 ⁜ Vol. 10, Issue 9, September 2022

DOI: 10.17148/IJIREEICE.2022.10915

```
# Visualization parameters
row_size = 20  # pixels
left_margin = 24  # pixels
text_color = (0, 0, 255)  # red
font_size = 1
font_thickness = 1
fps_avg_frame_count = 10

# Initialize the object detection model
base_options = core.BaseOptions(
    file_name=model, use_coral=enable_edgetpu, num_threads=num_threads)
detection_options = processor.DetectionOptions(
    max_results=3, score_threshold=0.3)
options = vision.ObjectDetectorOptions(
    base_options=base_options, detection_options=detection_options)
detector = vision.ObjectDetector.create_from_options(options)

# Continuously capture images from the camera and run inference
while cap.isOpened():
  success, image = cap.read()
  if not success:
    sys.exit(
        'ERROR: Unable to read from webcam. Please verify your webcam settings.'
    )

  counter += 1
  image = cv2.flip(image, 1)

  # Convert the image from BGR to RGB as required by the TFLite model.
  rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

  # Create a TensorImage object from the RGB image.
  input_tensor = vision.TensorImage.create_from_array(rgb_image)

  # Run object detection estimation using the model.
  detection_result = detector.detect(input_tensor)

  # Draw keypoints and edges on input image
  image = utils.visualize(image, detection_result)

  # Calculate the FPS
  if counter % fps_avg_frame_count == 0:
    end_time = time.time()
    fps = fps_avg_frame_count / (end_time - start_time)
    start_time = time.time()

  # Show the FPS
  fps_text = 'FPS = {:.1f}'.format(fps)

text_location = (left_margin, row_size)
  cv2.putText(image, fps_text, text_location, cv2.FONT_HERSHEY_PLAIN,
              font_size, text_color, font_thickness)

  # Stop the program if the ESC key is pressed.
  if cv2.waitKey(1) == 27:
    break
  cv2.imshow('object_detector', image)
```

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.047 ⁜ Vol. 10, Issue 9, September 2022

DOI: 10.17148/IJIREEICE.2022.10915

```
    cap.release()
    cv2.destroyAllWindows()


def main():
    parser = argparse.ArgumentParser(
        formatter_class=argparse.ArgumentDefaultsHelpFormatter)
    parser.add_argument(
        '--model',
        help='Path of the object detection model.',
        required=False,
        default='efficientdet_lite0.tflite')
    parser.add_argument(
        '--cameraId', help='Id of camera.', required=False, type=int, default=0)
    parser.add_argument(
        '--frameWidth',
        help='Width of frame to capture from camera.',
        required=False,
        type=int,
        default=640)
    parser.add_argument(
        '--frameHeight',
        help='Height of frame to capture from camera.',
        required=False,
        type=int,
        default=480)
    parser.add_argument(
        '--numThreads',
        help='Number of CPU threads to run the model.',
        required=False,
        type=int,
        default=4)
    parser.add_argument(
        '--enableEdgeTPU',
        help='Whether to run the model on EdgeTPU.',
        action='store_true',
        required=False,
        default=False)
    args = parser.parse_args()

    run(args.model, int(args.cameraId), args.frameWidth, args.frameHeight,
        int(args.numThreads), bool(args.enableEdgeTPU))
    if __name__ == '__main__':
    main()
```

IX. FUTURE WORKS SUGGESTED

The architecture can be extended into one more system that is communication system. A humanmachine interface can be installed in the server side. And through user webpage farmers can perform remote control and displays various information that influences harvesting decisions, such as streamed images, post detection results, the distance determined by ultrasonic sensors that are provided, arm control schematics, arm control buttons, and the GPS position of tracked mobile vehicles etc. We can add this communication system which mainly serves as a bridge for communication between the front and back ends of the system and performs some complex computations, including image streaming, detection, transmission of detection results to the webpage, and the reception or issuance of movement commands from the webpage. Raspberry Pi transferred data through a wireless network, controlled movement of the tracked mobile vehicle, and received ultrasonic measurement data and the geographic coordinates of the system from the GPS. The ultrasonic sensors measures the distance between the vehicle and the objects, and these measurements can be served as the reference value; the GPS module sent current coordinates to the server.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.047 ⁜ Vol. 10, Issue 9, September 2022

DOI: 10.17148/IJIREEICE.2022.10915

X. CONCLUSION

This system used an object detection algorithm with IoT technology to develop a remote crop harvesting system. An arm movement prediction model also designed. Through this system, younger farmers can quickly acquire practical farming knowledge. Using deep learning on collected images to perform smart image recognition increases convenience and the number of potential applications. The system can be used by various populations and can provide different functions according to user needs. The simple design of the system also enables experienced older farmers to operate the system easily. For novice farmers, the system can enable them to quickly learn the trade and develop their career. This system is highly reliable. The extensibility, scalability and usefulness of the system facilitate its integration into various industries.

REFERENCES

[1] Effendi Andoko, Hua-Jing, Zeng, Agnes Sjöblom, Wan-Yu Liu “Agricultural Statistics Visualized Query System”, Council of Agriculture, Executive Yuan, Taiwan. Environments 2017, 4, 50; doi:10.3390/environments4030050

[2] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, "ImageNet classification with deep convolutional neural networks", ACM Communications Magazine, Vol. 60, Issue 6, June 2017, pp. 84-90.

[3] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, "Gradient-based learning applied to document recognition", Proceedings of the IEEE, Vol. 86, Issue: 11, 1998.

[4] Matthew D., ZeilerRob Fergus, "Visualizing and Understanding Convolutional Networks", European Conference on Computer Vision (ECCV): Computer Vision, 2014, pp 818- 833.

[5] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, "Going deeper with convolutions", 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), DOI: 10.1109/CVPR.2015.7298594.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Deep Residual Learning for Image Recognition", 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), DOI:10.1109/CVPR.2016.90.

[7] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", 2014 IEEE Conference on Computer Vision and PatternRecognition.

[8] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, "You Only Look Once: Unified, Real-Time Object Detection", 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779-788.

[9] B. Kristyanto, B. B. Nugraha, A. K. Pamosoaji, K. A. Nugroho, "Analysis of human arm motions at assembly work as a basic of designing dual robot arm system", 2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM).

[10] Muhammad Ayaz 1, (Senior Member, IEEE), Mohammad Ammad-Uddin 1,(Senior Member, IEEE), Zubair Sharif2, Ali Mansour3, (Senior Member, IEEE),And El-Hadi M. Aggoune1, (Senior Member, IEEE) 1Sensor Networks and Cellular Systems Research Center, University of Tabuk, Tabuk71491, Saudi, “ Internet-of-Things (IoT)-Based Smart Agriculture: Toward Making the Fields Talk” IEEE Access (Volume: 7), Digital Object Identifier 10.1109/ACCESS.2019.2932609.

[11] Jinhui Li1, Yunchao Tang 2, (Member, IEEE), Xiangjun Zou 1, Guichao Lin1, And Hongjun Wang 11College of Engineering, South China Agricultural University, China College of Urban and Rural Construction, Zhongkai University of Agriculture and Engineering, Guangzhou 510225, China Corresponding authors: Yunchao Tang (ryan.twain@zhku.edu.cn) and Xiangjun Zou. “Detection of Fruit-Bearing Branches and Localization of Litchi Clusters for Vision Based Harvesting Robots” IEEE Access (Volume: 8), DOI: 10.1109/ACCESS.2020.3005386 .