Design Lab Project

EEE department, IITG

Smart Gardener

Guided: Chitralekha Ma'am

Instructor: Rafi Ahamed Sir

Submitted by Group 23:

Deepak Meghwal- 150108009

Devesh Prajapat - 150102015

Pranav Totala - 150102051

Abstract and Summary:

The main aim of this project was to construct a mobile bot (with the camera and some sensors) which itself will be able to detect plants using its camera. After detecting the plant the bot will move near to the plant and detect the level of water in the plant and if water is required in the plant, it will water the plant according to the data collected of water level in soil. And it will continue the process in whole garden to all the plants.

The work completed till now is, we have made the bot and completed the image processing part from which the plant will be detected. The image processing was done by using the openCV and python in Raspberry pi 3. The camera module of raspberry takes image every 3 seconds and process it in python and then if the plant is detected it gives command to motor driver to move forward towards the plant. If the plant is not detected in that frame, the Raspberry pi commands the motor driver to rotate in the right direction and it will keep on rotating until a plant is detected.

After moving in the direction of plant, we stopped the bot by measuring the distance between bot and plant using the UV sensor. The distance measured by the UV sensor gives command to raspberry to stop the motor driver. After reaching the bot near to the plant it will detect the water level of the plant using the humidity sensor. And again start rotating to detect more plants.

Acknowledgement:

This project was fully supported by Dr. Chitralekha Mahanta. We are thankful to Chitralekha Ma’am who provided expertise that greatly assisted the research, although they may not agree with all of the interpretations provided in this paper.

We are also grateful to Dr. Shaik Rafi Ahamed for assistance with the course, and moderating this paper and in that line improved the manuscript significantly.

We have to express our appreciation to the doctorate seniors, Lab assistants for sharing their pearls of wisdom with us during the course of this research, although any errors are our own and should not tarnish the reputations of these esteemed professionals.

Index:

Literature review:

A. Problem Statement:

Our aim of the project is to create a autonomous bot that can perform the basic task of a gardener, i.e watering plants. Since we are trying to make it a smart robot, we cannot give it a command even from a distance.

The robot should do the following tasks to be called a smart gardener. The task of detecting the plant, moving towards it and watering it should all be integrated on the bot itself. The only command it will receive will be to start or stop the search of plants via a remote laptop.

The problem can be divided into three main parts:

1. Image processing: To identify the potted plant we need to capture an image from the surrounding of the bot and use the trained data on it.
2. Built and control bot: We need to make a bot to navigate our gardener around. It should be capable to move in every direction and rotate at point.
3. Processor and Sensors: Lastly we need to integrate the camera, humidity sensor, ultrasonic sensor and motor driver on the bot and control it using the processor that we will use for the image processing

B. Ideation and Research:

The main questions that required research were Image processing, which processor to use to control the bot, main design of the robot and movability in congested places.

Raspberry Pi seemed to be a efficient as well as a cheap processor to be capable to handle all the processes and support all the sensors. There is also a modular camera available with the raspberry pi so compatibility with camera will not be a problem.

Bot design can be varied a lot. We had an option of making a 4 wheeled, 3 motor bot or a 4 motor bot. But to make the project more cost effective, we made a bot with 2 motors and a all-directional wheel. This design also proves to be more compact and satisfies all requirements.

For Image processing we researched on caffemodel and Tensorflow models as these models are efficient and can be used in python and processed on the raspberry pi 3.

C. Similar projects:

In this project [1] we learned to use the GPIO pins in the raspberry pi pins for communicating with motors and sensors.

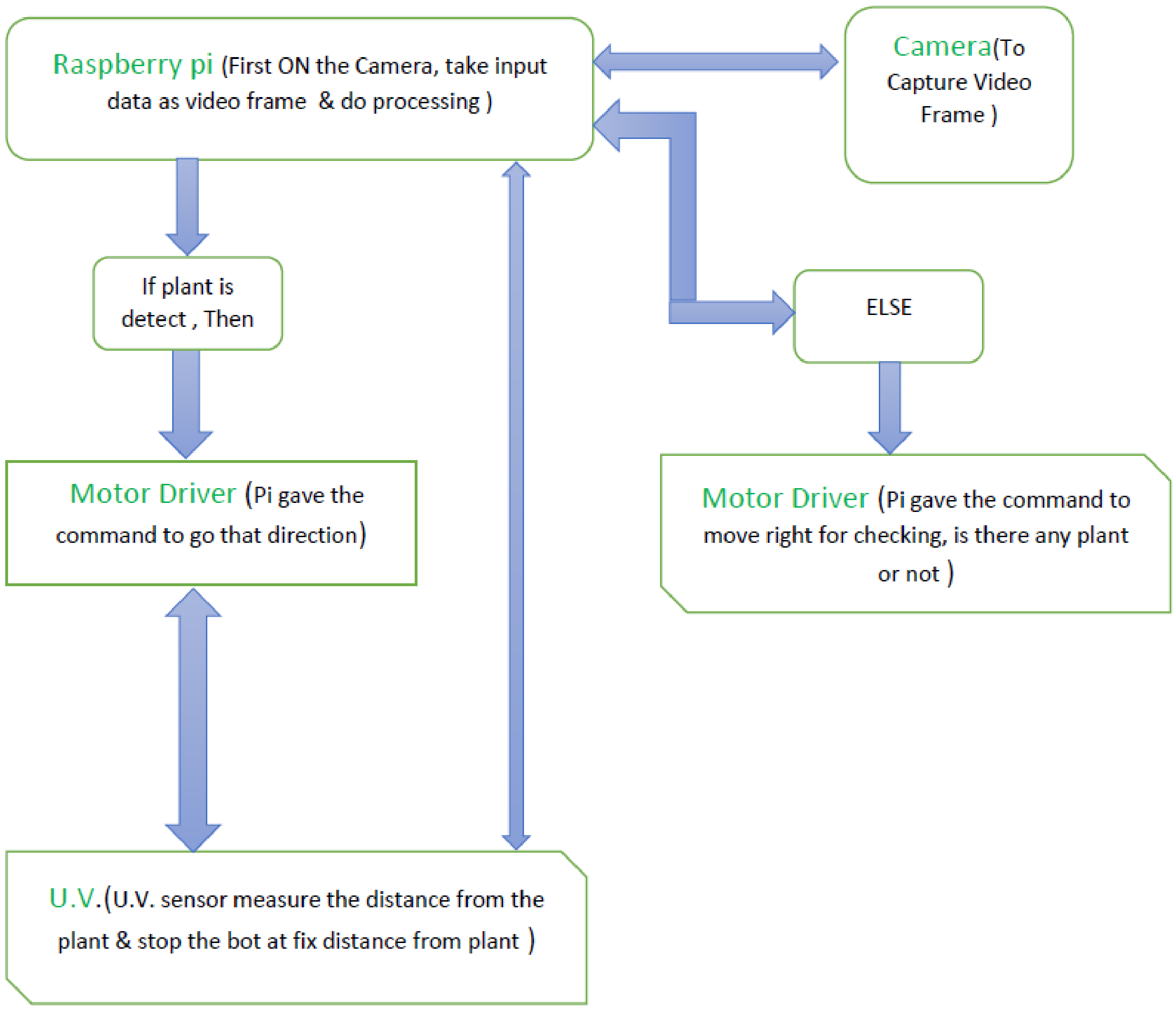
This project [2] is a stationary model of smart gardener and does the work of processing values of the sensors on the plant

Approach:

A. Implementation of Research:

From the research, we concluded that a raspberry pi would be requiresd to run the bot along with all the sensors and camera, plus being capable of processing the image on the bot itself.

B. Block flow diagram:



Raspberry pi (First ON the Camera, take input data as video frame & do processing)
Camera(To Capture Video Frame)
If plant is detect , Then
ELSE
Motor Driver (Pi gave the command to go that direction)
Motor Driver (Pi gave the command to move right for checking, is there any plant or not)
U.V.(U.V. sensor measure the distance from the plant & stop the bot at fix distance from plant)

C. Materials used:

1. Raspberry Pi 3
2. Raspberry camera module
3. Humidity sensor HS12sp
4. 12V DC motors 100rpm
5. 12V battery
6. Motor driver L298N
7. Jumper wires
8. Acrylic sheet
9. Ultrasonic sensor

The Materials used in the making the projects are:

1. Raspberry Pi3 - The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundations to promote the teaching of basic computer science . It does not include peripherals(such as keyboards,mice and cases).However ,some accessories have been included in several official and unofficial bundles. All models feature a Broadcom system on a chip with an integrated ARM compatible central processing unit(CPU) and on chip graphics processing unit(GPU).

   The Raspberry Pi 3 include :

   - CPU: Quad-core 64-bit ARM Cortex A53 clocked at 1.2 GHz
   - GPU: 400MHz VideoCore IV multimedia
   - Memory: 1GB LPDDR2-900 SDRAM (i.e. 900MHz)
   - USB ports: 4
   - Video outputs: HDMI, composite video (PAL and NTSC) via 3.5 mm jack
   - Network: 10/100Mbps Ethernet and 802.11n Wireless LAN
   - Peripherals: 17 GPIO plus specific functions, and HAT ID bus
   - Bluetooth: 4.1
   - Power source: 5 V via MicroUSB or GPIO header
   - Size: 85.60mm × 56.5mm
   - Weight: 45g (1.6 oz)

2. Raspberry Pi Camera Module - The Raspberry Pi Module is a 5Mp CMOS camera with a fixed focus lens that is capable of capturing still images as well ashigh definitions videos. Stills are captured at a resolution of 2592x1944 ,while video is supported at 1080p at 30 FPS, 720p at 60 FPS and 640x480 at 60 or 90 FPS. The camera is supported in the latest version of Raspbian, Raspberry Pi's preferred operating system.

3. Motor Driver L298N - L293D is a typical Motor driver or Motor Driver IC which allows DC motor to drive on either direction. L293D is a 16-pin IC which can control a set of two DC motors simultaneously in any direction. It means that you can control two DC motor with a single L293D IC.In a single L293D chip there are two h-Bridge circuit inside the IC which can rotate two dc motor independently.H-bridge is a circuit which allows the voltage to be flown in either direction.H-bridge IC are ideal for driving a DC motor.Due its size it is very much used in robotic application for controlling DC motors.

4. Ultrasonic sensor - An Ultrasonic sensor is a device that can measure the distance to an object by using sound waves. It measures distance by sending out a sound wave at a specific frequency and listening for that sound wave to bounce back.

5. Humidity sensor DHT-22 - The DHT-22 (also named as AM2302) is a digital-output, relative humidity, and temperature sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and sends a digital signal on the data pin.

6. 12V DC motors 100rpm- The DC motor is a machine that transforms electric energy into mechanical energy in form of rotation. Its movement is produced by the physical behavior of electromagnetism. We use 12V DC motor for our project.

7. 12V DC Battery - We use 12V DC battery to power supply our bot, its mainly supply the DC motors , Raspberry Pi is take power from power-bank for processing.

8. Acrylic sheet - Acrylic plastic sheet is completely transparent, flexible, and exhibits pretty good resistance to breakage. Acrylic is an excellent material to use in making bot.

Steps in Work:

- Transporting Robot:

    We completed the making to the robot that will navigate the system around the garden and move towards plants for the rest of the process.

    We took a acrylic sheet of 25 x 20 square cm and attached two DC motors at the rear end of the sheet, like two rear wheels of a car. Then we drilled a multi-directional wheel (like the one in moving chairs) in the middle of the front side so the bot can turn in any direction possible.

- Motor Driver:

    We attached the motors to the L298N motor driver and tested it at variable rpm and speed.

- Humidity Sensor:

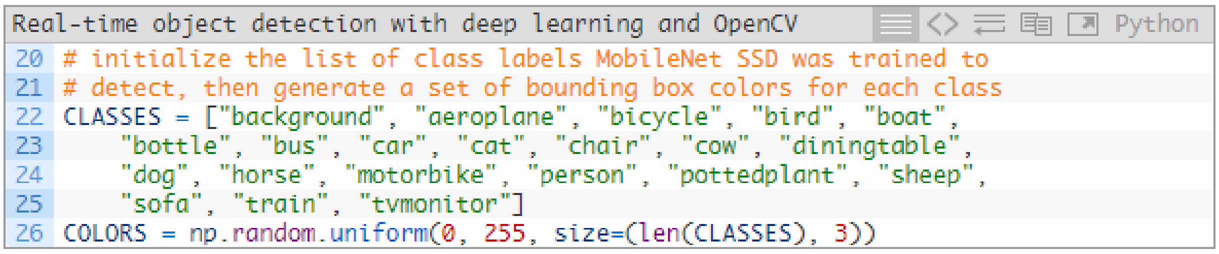
    To detect weather the bot requires water or not, the humidity sensor is placed on the bot that will check the humidity of the plants and give the value to the raspberry so that it can identify a dry or wet plant. We took some sample readings of the humidity sensor to check the same.

- Image Processing Code:

    Object detection in video with deep learning and OpenCV :-

    To build our deep learning-based real-time object detector with OpenCV we'll need to (1) access our webcam/video stream in an efficient manner and (2) apply object detection to each frame.

We begin by importing packages on Lines 2-8. We will need imutils and OpenCV 3.3. OpenCV 3.3 have deep neural networks(dnn) modules.

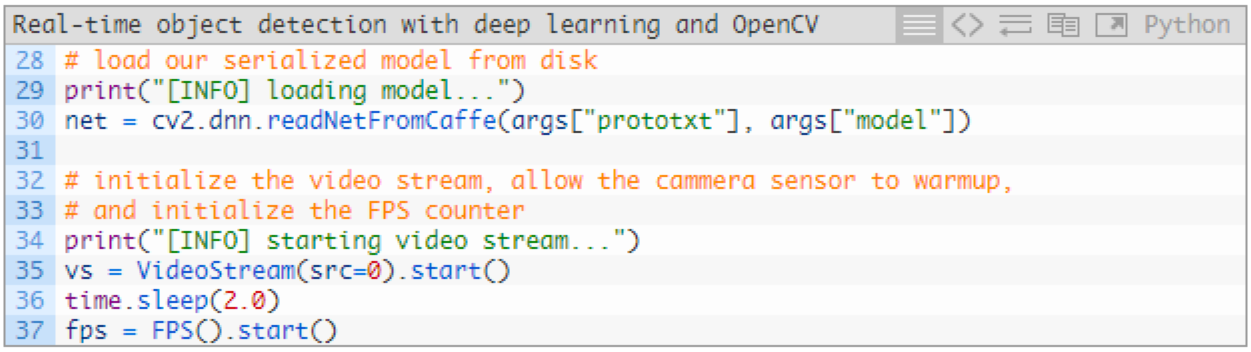

Real-time object detection with deep learning and OpenCV

```python
20 # initialize the list of class labels MobileNet SSD was trained to
21 # detect, then generate a set of bounding box colors for each class
22 CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
23     "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
24     "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
25     "sofa", "train", "tvmonitor"]
26 COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))
```

On Lines 22-26 we initialize CLASS labels and corresponding random COLORS .

For more information on these classes (and how the network was trained)

Now, let’s load our model and set up our video stream:


Real-time object detection with deep learning and OpenCV

```python
28 # load our serialized model from disk
29 print("[INFO] loading model...")
30 net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])
31 
32 # initialize the video stream, allow the cammera sensor to warmup,
33 # and initialize the FPS counter
34 print("[INFO] starting video stream...")
35 vs = VideoStream(src=0).start()
36 time.sleep(2.0)
37 fps = FPS().start()
```

We load our serialized model, providing the references to our prototxt and model files on Line 30 — notice how easy this is in OpenCV 3.3.

Next let’s initialize our video stream (this can be from a video file or a camera). First we start the VideoStream (Line 35), then we wait for the camera to warm up (Line 36), and finally we start the frames per second counter (Line 37). The VideoStream and FPS classes are part of my imutils package.

Real-time object detection with deep learning and OpenCV Python

```python
39 # loop over the frames from the video stream
40 while True:
41     # grab the frame from the threaded video stream and resize it
42     # to have a maximum width of 400 pixels
43     frame = vs.read()
44     frame = imutils.resize(frame, width=400)
45 
46     # grab the frame dimensions and convert it to a blob
47     (h, w) = frame.shape[:2]
48     blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)),
49         0.007843, (300, 300), 127.5)
50 
51     # pass the blob through the network and obtain the detections and
52     # predictions
53     net.setInput(blob)
54     detections = net.forward()
```

First, we read a frame (Line 43) from the stream, followed by resizing it (Line 44) Since we will need the width and height later, we grab these now on Line 47. This is followed by converting the frame to a blob with the dnn module (Lines 48 and 49). Now for the heavy lifting: we set the blob as the input to our neural network (Line 53) and feed the input through the net (Line 54) which gives us our detections.

Real-time object detection with deep learning and OpenCV Python

```python
    # loop over the detections
    for i in np.arange(0, detections.shape[2]):
        # extract the confidence (i.e., probability) associated with
        # the prediction
        confidence = detections[0, 0, i, 2]

        # filter out weak detections by ensuring the `confidence` is
        # greater than the minimum confidence
        if confidence > args["confidence"]:
            # extract the index of the class label from the
            # `detections`, then compute the (x, y)-coordinates of
            # the bounding box for the object
            idx = int(detections[0, 0, i, 1])
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")

            # draw the prediction on the frame
            label = "{}: {:.2f}%".format(CLASSES[idx],
                confidence * 100)
            cv2.rectangle(frame, (startX, startY), (endX, endY),
                COLORS[idx], 2)
            y = startY - 15 if startY - 15 > 15 else startY + 15
            cv2.putText(frame, label, (startX, y),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)
```

We start by looping over our detections , keeping in mind that multiple objects can be detected in a single image. We also apply a check to the confidence (i.e., probability) associated with each detection.

If the confidence is high enough (i.e. above the threshold), then we'll display the prediction in the terminal as well as draw the prediction on the image with text and a colored bounding box. Let's break it down line-by-line:

Looping through our detections , first we extract the confidence value (Line 60).

If the confidence is above our minimum threshold (Line 64), we extract the class label index (Line 68) and compute the bounding box coordinates around the detected object (Line 69).

Then, we extract the (x, y)-coordinates of the box (Line 70) which we will will use shortly for drawing a rectangle and displaying text.

We build a text label containing the CLASS name and the confidence (Lines 73 and 74).

Let's also draw a colored rectangle around the object using our class color and previously extracted (x, y)-coordinates (Lines 75 and 76).

In general, we want the label to be displayed above the rectangle, but if there isn't room, we'll display it just below the top of the rectangle (Line 77).

Finally, we overlay the colored text onto the frame using the y-value that we just calculated (Lines 78 and 79).

The remaining steps in the frame capture loop involve (1) displaying the frame, (2) checking for a quit key, and (3) updating our frames per second counter:

Real-time object detection with deep learning and OpenCV Python

```
80      # show the output frame
81      cv2.imshow("Frame", frame)
82      key = cv2.waitKey(1) & 0xFF
83  
84      # if the `q` key was pressed, break from the loop
85      if key == ord("q"):
86          break
87  
88      # update the FPS counter
89      fps.update()
```

The above code block is pretty self-explanatory — first we display the frame (Line 82). Then we capture a key press (Line 83) while checking if the ‘q’ key (for “quit”) is pressed, at which point we break out of the frame capture loop (Lines 86 and 87).

Finally we update our fps counter (Line 90).

If we break out of the loop (‘q’ key press or end of the video stream)

Real-time object detection with deep learning and OpenCV Python

```
91  # stop the timer and display FPS information
92  fps.stop()
93  print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))
94  print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))
95  
96  # do a bit of cleanup
97  cv2.destroyAllWindows()
98  vs.stop()
```

When we’ve exited the loop, we stop the fps counter (Line 93) and print information about the frames per second to our terminal (Lines 94 and 95).

We close the open window (Line 98) followed by stopping the video stream (Line 99).

If you’ve made it this far, you’re probably ready to give it a try with your webcam — to see how it’s done, let’s move on to the next section.

To identify weather the bot is pointing at a plant, it should be able to process the image and apply machine learning to differentiate a plant from surrounding.

For processing this image we need to communicate the Raspberry pi to MATLAB and transfer data to and from matlab. We completed communicating Arduino Uno with matlab. Since a change of boards, we are working on the process to transfer and receive data from the raspberry through its inbuilt wifi.

Result:

A. Achievements:

We are successful in creating a cost efficient bot that can process images and navigate fully autonomously

B. Uses:

The bot which we planned to make have many uses and applications in the field of agriculture and plants.

A. We can place a tank and pump on the top of the bot so that it can water all the plants it detects.

B. We can put various sensors and note the health condition of plants. The bot can report any critical message directly to your laptop through wifi connection.

C. Limitations:

We could not make the robot differentiate between two potted plants. This limitation was due to the use of single camera and no defined path for the robot. Hence the bot will continue to detect the plants over and over again until it is manually stopped. With more time and resources, the limitation can be removed.

References:

1. We use this link for image processing :-
https://www.pyimagesearch.com/2017/09/18/real-time-object-detection-with-deep-learning-and-opencv/

2. We use this link for install OpenCV and all required libraries :-
https://www.pyimagesearch.com/2017/09/04/raspbian-stretch-install-opencv-3-python-on-your-raspberry-pi/
http://pklab.net/?id=392&lang=EN
http://www.life2coding.com/install-opencv-3-4-0-python-3-raspberry-pi-3/

3. We use this link for setup camera module:-
https://thepihut.com/blogs/raspberry-pi-tutorials/16021420-how-to-install-use-the-raspberry-pi-camera

4. We use this link for step wifi connection of Pi:-
https://www.raspberrypi.org/documentation/configuration/wireless/wireless-cli.md

5. We use this link for connecting pi with laptop:-
https://raspberrypi.stackexchange.com/questions/30144/connect-raspberry-pi-to-pc-ubuntu-with-ethernet?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa

6. Used this project to understand working of GPIO pins in the raspberry pi
https://www.raspberrypistarterkits.com/accessories/best-sensors-raspberry-pi/

7. A stationary smart gardener, in similar projects
https://business.tutsplus.com/tutorials/controlling-dc-motors-using-python-with-a-raspberry-pi--cms-20051