Brilliant Pad: Poo Vision

Final Report

Richard Chum, Derek Lam, Kevin Tain

CSE 145: Embedded Systems Design Project

Professor Ryan Kastner

Advisor: Alan Cook

Winter 2018

I. Abstract

The Brilliant Pad is a self-cleaning dog potty that removes the hassle of cleaning up after pets through its automatic pad dispensing and sealing system. Although convenient, the Brilliant pad still has some pet peeves: it currently advances the pad at a user-defined time interval regardless of the amount of waste on the pad. This leads to a waste of clean pads, prolonged odor, and higher maintenance. We aim to improve the existing product with a vision system that can tell the pad to advance when a user-defined waste threshold is reached. Our vision system makes use of a Raspberry Pi, Pi Camera, and OpenCV to determine the level of waste detected. To make the system more robust, we also explore methods to more reliably detect dog presence. These additions to the Brilliant Pad will help assess the feasibility of a camera system to improve the user experience and serve as a basis for potential commercial adoption.

II. Introduction

Brilliant Pad is the world's first self-cleaning, indoor dog potty, and has been out on the market for over a year. The purpose of a self-cleaning, indoor dog potty is to provide a convenient solution to an everyday problem dog owners have, when and where the dog can use the restroom.[1] Not all dog owners can offer their new family member a place to go, and this is because most people live in smaller accommodations, such as apartments or condominiums, and there are unforeseen natural events that make it unfavorable to let one's dog out (i.e. it is raining, too hot, snowing, etc.), and plenty other reasons. The Brilliant Pad removes the hassle of cleaning after one's pet, and offers the user a sense of confidence in knowing that their pet has a nice, appropriate place to defecate. Due to the Brilliant Pad's ability to automatically advance, one can be sure that the pad will advance at least a certain amount of time before one returns home. Furthermore, having the pad indoor makes it opportunate to allow the dog to use the restroom whenever it is necessary - effectively eliminating the concern of knowing when and where one's dog can use the restroom.

The Brilliant Pad is able to automate its cleaning process with the assistance of a control module that houses an infrared (IR) sensor and a small motor. When plugged in, the user is able to set an interval of how often the pad will advance throughout the day by pressing the Automatic Advance button. In doing so, the user can set the pad to advance either one, two, or three times throughout the day. Additionally, the user is able to advance the pad manually by pressing and holding the Pad Advance button, which signals the internal motor to start reeling in the pad into the main waste container. The IR sensor within the control module is used to detect whether or not the dog is on the pad, so the pad will not advance while your friend is using it.

Although the Brilliant Pad is advantageous for dog owners, there are still some flaws with its current design. Because the pad advances at a known interval and a dog needs to use the restroom at unknown times, it makes it difficult for the Brilliant Pad to determine when it is best to advance. Currently, the pad will advance a set amount of time, but consider if one's dog does not use the restroom as often as anticipated, i.e. the dog goes more often or not as often as expected. If the dog uses the restroom more often than anticipated, meaning the pad is currently dirty and the dog needs to go again, it is possible that the dog will not use the pad and instead defecate anywhere he/she deems fit. On the other hand, if the dog does not use the restroom as often as expected, then it goes without saying that the pad will advance when there is nothing on the pad. With the user's inability to consistently predict when one's dog uses the restroom, there is an opportunity for a new solution. Along with the inconsistent prediction of dog waste management, the IR sensor of the control module has a small scope of vision. The IR sensor can reach about half of the pad but has a narrow viewing angle. The IR sensor works great when the

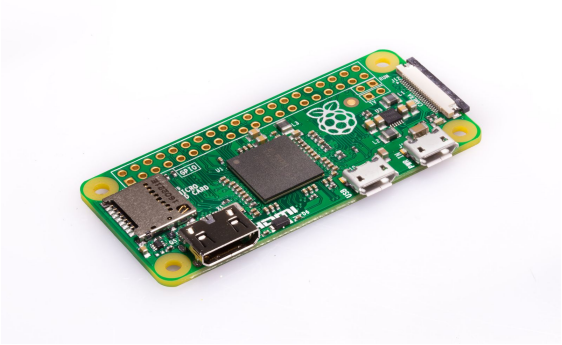
dog is where the sensor needs it to be, but anywhere else, the dog is not detected by the IR sensor. It is like looking through a narrow hole.

We aim to improve the Brilliant Pad by adding an extra camera module. The module includes a Raspberry Pi Camera and a Raspberry Pi. We use the Pi Camera to continuously capture images of the pad which is then processed with an open source computer vision library called OpenCV. By actively taking pictures of the pad area, we test to see if we can improve the Brilliant Pad's ability to determine when to advance the pad by measuring the current amount of waste on the pad. Moreover, we intend to use the camera to improve the Brilliant Pad's ability to detect the dog - instead of looking through a hole, we break the barrier that prevented us from properly determining whether or not the dog is currently on the pad. In the following paper, one will see how we were able to use the Pi camera and algorithms implemented through OpenCV to enhance the Brilliant Pad's ability to determine when to advance and when not to advance.

III. Technical Material

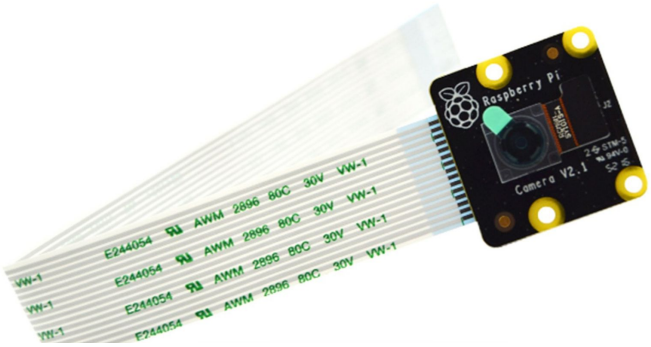
A. Hardware

1. Raspberry Pi Zero



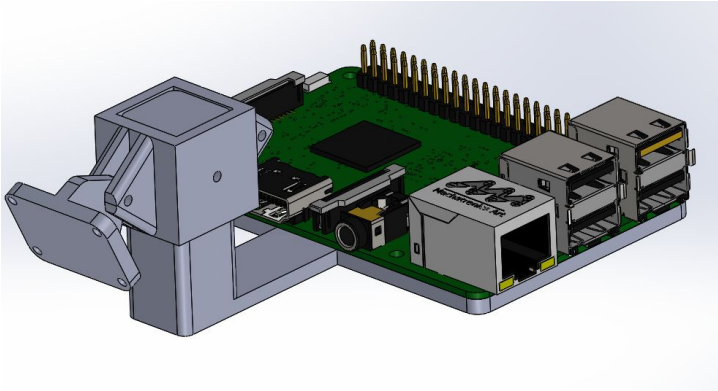
The Raspberry Pi Zero is a low profile computer featuring a 1 GHz single-core CPU and 512 MB of ram. It is a cheaper and smaller version of the most popular model, the Raspberry Pi 3, created by the Raspberry Pi Foundation. This model was chosen as its specifications were sufficient for prototyping the vision system for the Brilliant Pad, and it contained a port for the Raspberry Pi Camera.

2. Raspberry Pi Camera



The goal of our project revolved around being able to detect dog waste and dog presence on the Brilliant. To achieve our goal, we opted to use the Raspberry Pi NoIR Camera Module. This Camera Module adds the ability to take 8 Megapixel images, in resolutions of up to 3280 x 2464 pixels. To increase processing time on the Raspberry Pi Zero, these images were downscaled to 320 x 240.

3. 3D Printed Mount



We created a 3D printed mount, attached on top of the lid closest to the control module on the Brilliant Pad. The mount was designed in Solidworks with the intention of creating a platform for either the Raspberry Pi Zero or the Raspberry Pi 3, and was printed using standard PLA material. It features adjustable angle for the Pi Camera for testing purposes, and a low profile to minimize the overall height of the vision system.

B. Software

1. OpenCV

The core of our software revolved around the OpenCV library, an open source computer vision library available in the scripting language Python. OpenCV possesses many functions that aid in image processing, and was essential in meeting our requirements. OpenCV 3.3.0 was used for all processing in this project.

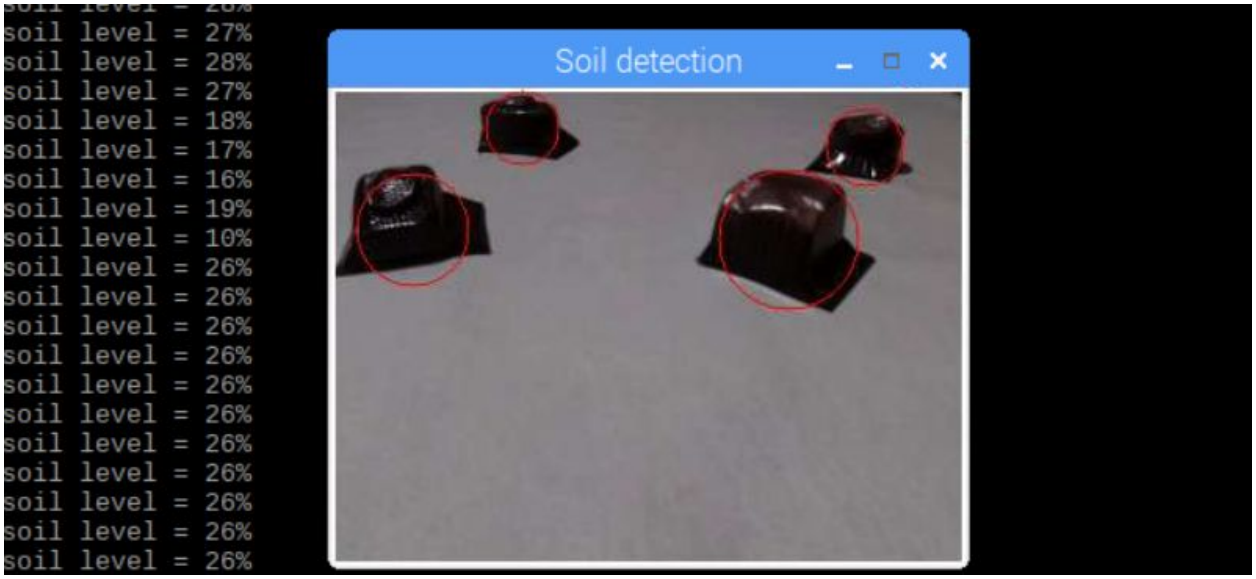
2. Algorithms

a) Blob Detection

To obtain a soil level for the Brilliant Pad to determine whether or not to advance the pad, we needed some way to differentiate between waste and the pad. OpenCV has a very helpful built-in class for detecting blob-like objects in an image, known as blob detection. A blob, which stands for binary large object, is simply a group of pixels connected in an image such that they possess similar features,

A simple blob detector can be initialized in Python simply by initializing a detector as detector = cv2.SimpleBlobDetector().[2] According to the documentation on the OpenCV website, blob detection is able to filter blobs based on color, area, circularity (defined by $4\pi Area/Perimeter^2$), ratio of minimum inertia to maximum inertia (another form of circularity that checks how elongated the blob is), and convexity.[3] For our blob detection algorithm, we opted to only filter by area and inertia.

Once a blob has been found in an image, we are able to retrieve the center and radius of each blob. The size of each blob is added and then divided by the size of the pad to determine a soil level. The soil percentage is calculated as follows: $100 * \frac{\sum_{i}^{n} b_i}{A}$, where b_i is the area of the i[th] blob. n is the total number of blobs, and A is the area of the pad. Shown below is example of blob detection, in which the detected blobs are marked by a red circle.



b) Differential Images

One of the objectives of the Brilliant Pad vision system is to detect whether or not a dog is present on the pad. This serves the purpose of ensuring that the pad will not automatically advance while the dog is on it. To accomplish this objective, we used a form of motion detection using OpenCV to implement differential images. If motion is detected on the Brilliant Pad, we mark that the dog is present on the pad, and the pad should not advance for X amount of seconds. This amount is variable and can change with additional user testing, but our current implementation has this timer reset to 10 seconds every time motion is detected on the pad.

Differential images is done by subtracting two images.[4] We first convert the RGB images taken from the Raspberry Pi Camera and convert them to grayscale. As each image is a matrix of size 320 x 240, filled with intensity values, a difference image can be generated using the formula $\Delta I = I_{diff}(x,y) = I_1(x,y) - I_2(x,y)$, though this is done using the absdiff() function in OpenCV.[5] To detect motion, three images in our video stream are taken and analyzed. A difference image is taken between the first and second image, and another difference image is taken between the second and third image. Once the two difference images are determined, a final image is created by taking the logical operation bitwise and between the two difference images using the bitwise_and() function in OpenCV such that $\Delta I_{final} = \Delta I_1 \wedge \Delta I_2$.

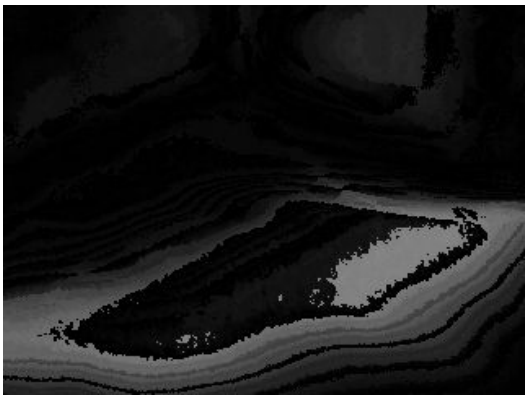
Finally, motion is detected by summing up all the values in the final image using countNonZero() in OpenCV() such that $M = \sum_x \sum_y \Delta I_{final}(x,y)$. Because the size of the images obtained from the Raspberry Pi Camera are 320 x 240, the maximum value of M is equal to $320 * 240 = 76800$. The Raspberry Pi Camera is inherently noisy, so M will range between 15000 - 25000, even when nothing is moving on the pad. Therefore, we have set the threshold for

motion as $M = 31000$. An example of differential images is shown below, with three images from the video stream and the resulting final difference image.

$image_{t-1}$

$image_t$

$image_{t+1}$



$$\Delta I = \Delta I_1 \wedge \Delta I_2, \text{ where } \Delta I_1 = image_{t+1} - image_t \text{ and } \Delta I_2 = image_t - image_{t-1}$$

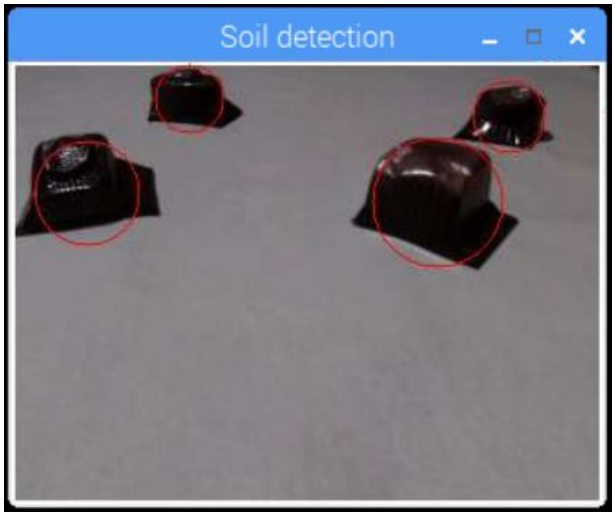
Differential Images were also looked into as a form of waste detection. Because we had an issue detecting liquid waste on the pad, one idea that we had was to compare two images: the first image being an image from before motion is detected, and one image 10 seconds after motion is no longer detected. By taking the difference between these two images, anything new on the pad, i.e. waste, could be seen. While this idea did work, as shown in the example below, this idea was not fully integrated along with blob detection due to time constraints.

C. Analysis

1. Camera Angle

The optimal camera position would be directly above the pad. In the example below, it was determined that the pad was about 16% soiled using the optimal camera position whereas the non optimal angle shown on the right registered the pad to be about 30% soiled. The actual camera position is much closer to the pad and thus the waste appears to be bigger in the image. A solution to having a skewed image would be to transform the image, but to create an optimal transformation the camera has to have view of all four corners - which we could not achieve given our mechanical specifications. Another solution would have been to try a fisheye lens - though we did not consider this idea until later in the project. With the current orientation of the camera, blob detection registers the pad to be about twice as soiled than it actually is. Given the constraint that the camera needed to be above the main waste container, the camera's actual optimal position was directly centered on the container and facing an angle that would have the pad completely in its field of view. We measured that we had the camera at 125^{o}, with this being the angle that had the most area of the pad covered.

Optimal



Actual

Milestones

- **Defining system specifications**
  The system specifications are:
  1. The added system should be able to determine how soiled the pad is.
  2. The added system should be able to detect whether or not a dog is on the pad.
  3. The system must be constrained to not alter the pad but be an addition to the pad; we could not change the pad area, the base of the pad cannot be changed (we thought of adding a weight sensor underneath the pad, but we cannot due to limits in manufacturing), camera must not be obstructive to the the Brilliant Pad system
- **Assembling list of parts**
  We decided that the parts we needed are as follows:
  Brilliant Pad and pads

SHCHV Raspberry Pi Zero
Raspberry Pi NoIR Camera Module V2
Raspberry Pi 3
3D Prints

- **Create block diagram/system architecture**
  Deliverable:

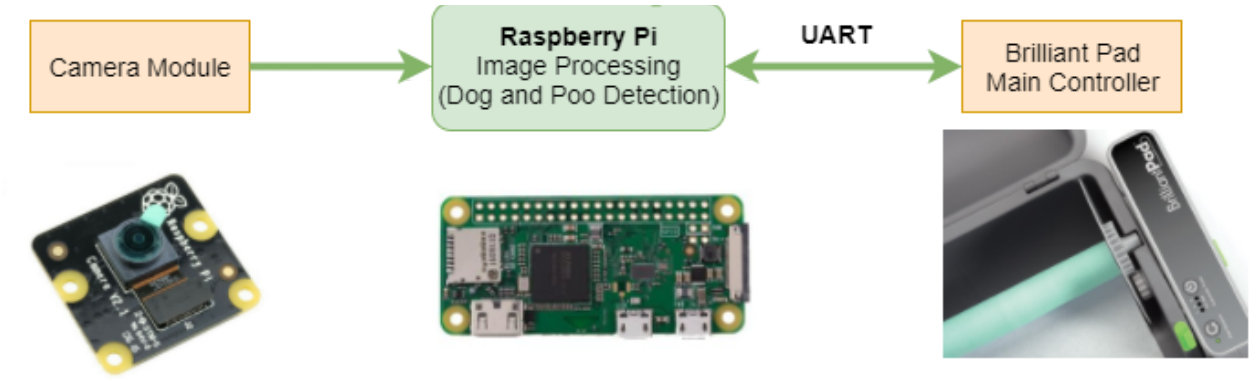


Figure 1.1 Block Diagram

- **Creating a mount and putting the system together**
  We used the dimensions of the Raspberry Pi NoIR Camera to create a mount that will be positioned on top of the main waste container cover. The camera had 4 holes with dimensions displayed in Figure 1.2. Furthermore, the base of the mount is designed to hold the Raspberry Pi 3 as seen in Figure 1.3. We also created the CAD of the mount as seen in Figure 1.5.

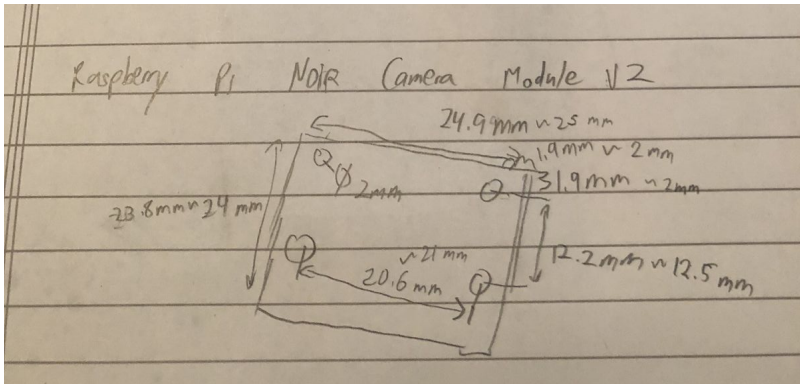


Figure 1.2 Dimensions of the Raspberry Pi NoIR Camera

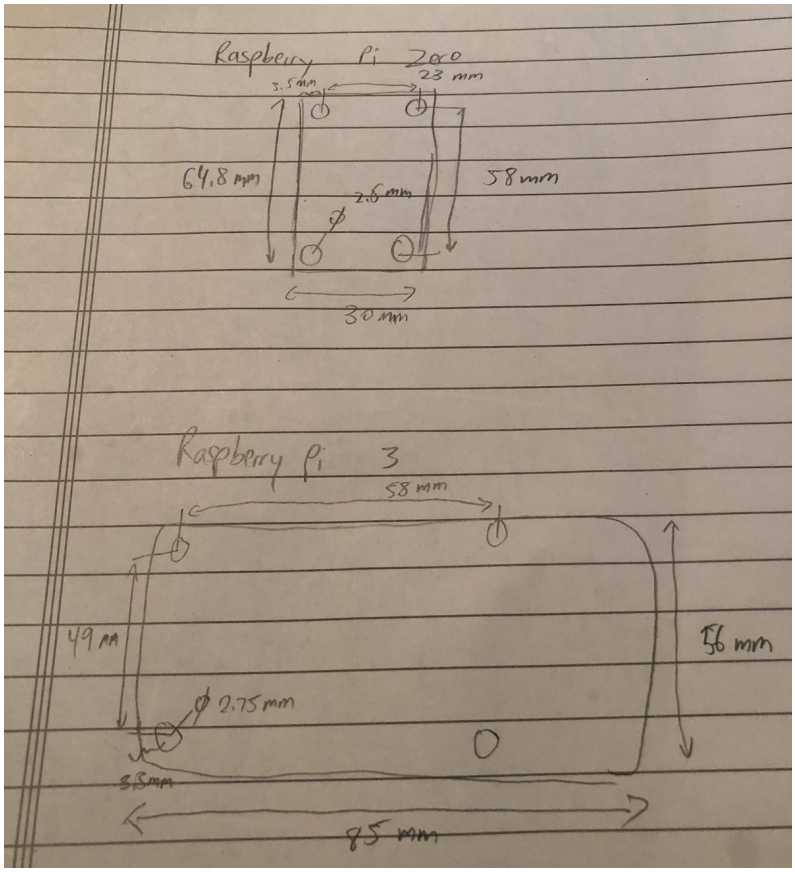


Figure 1.3 Dimensions of the Raspberry Pi's

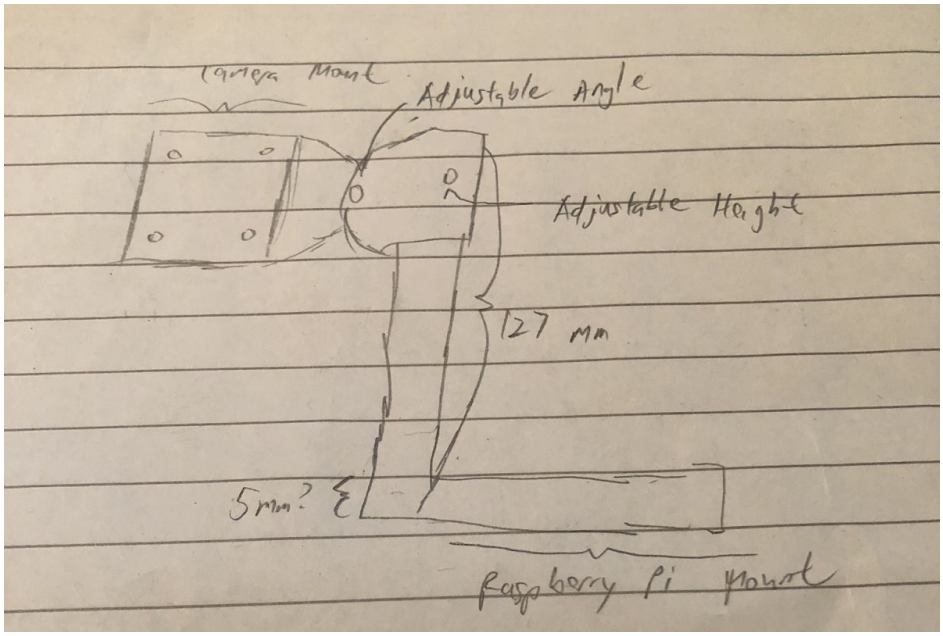


Figure 1.4 Visualization of how the camera will be mounted with the Raspberry Pi

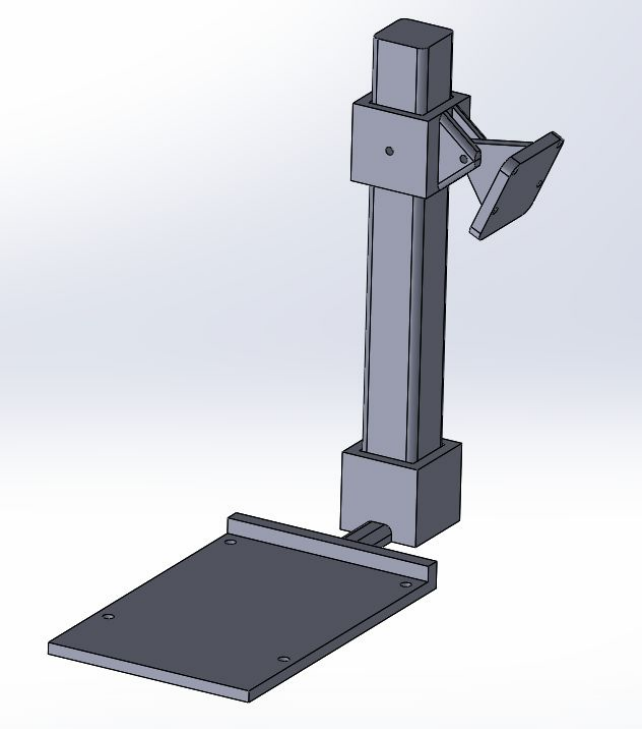


Figure 1.5 CAD of the mount that will hold the camera and the Raspberry Pi

- **Create a web presence**
  The following is a link to the github for this project: https://github.com/ktain/poovision
- **Capture images with the subsystem**
  The basic script to take a picture, given that we have set the camera and pi up properly, is
  raspistill -o filename.png
  but within our algorithm we use the command
  For frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
  Image = frame.array
  To continuously capture images and put it into a variable called image.

  Deliverable:

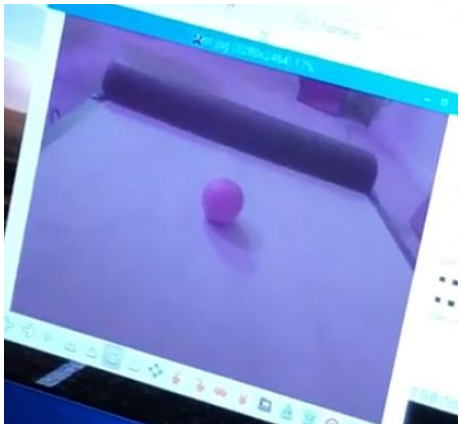


Figure 1.6 Sample picture taken by Raspberry Pi Camera

- **Isolate Pad Area from Non-Pad Area**
  We tested this using contours to isolate the pad, and we were successful but we decided to use a different algorithm called blob detection which didn't require us to isolate the pad from non-pad areas.[6]

  Deliverable:

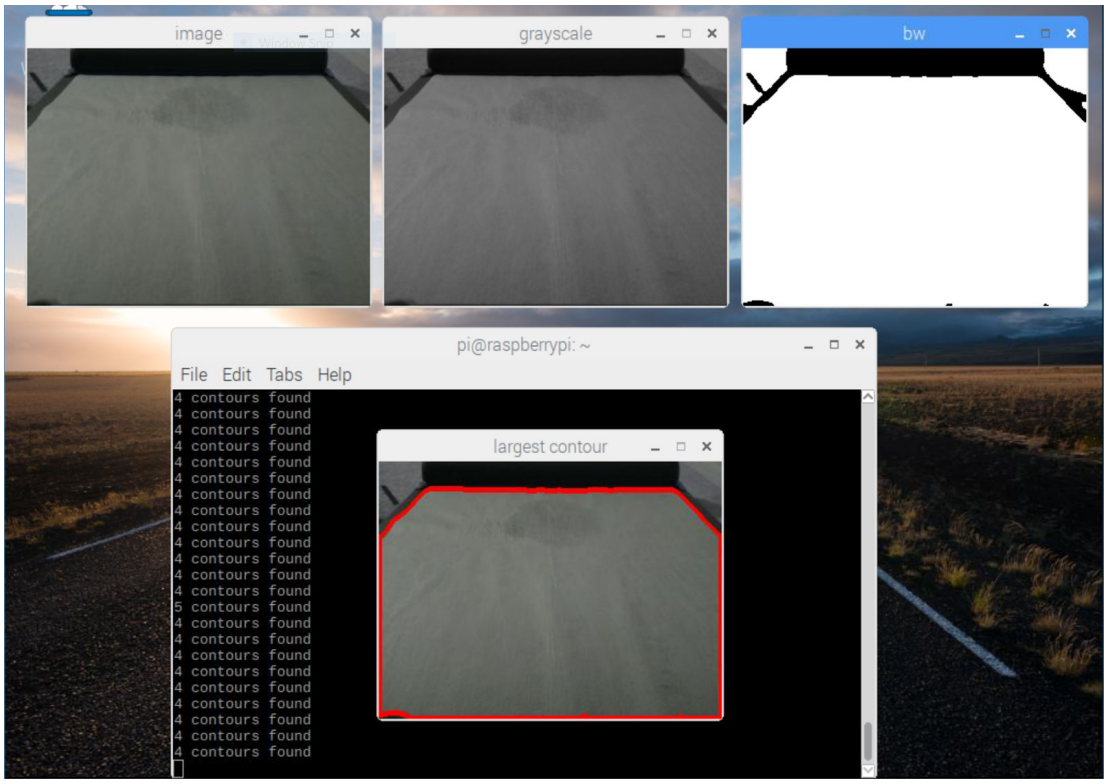


Figure 1.7 Using contours to isolate the pad from non-pad areas

- **Soil Level**
  We attempted to determine how soiled the pad is using blob detection, although we were able get a size from the blob we didn't know what how the size of the blob was scaled relative to the image. We also determined that the size of the blob was not proportional to the size of the pad so we used an arbitrary number that gave us a sound soil level based on the blobs detected. This module is partially completed. We were able to detect objects on the pad using blobs but the size of the blob was not proportional to the pad and the placement of the camera was only able to view about 75% of the pad. Due to our constraints, we were able to output a soil level that isn't completely accurate, but is able to demonstrate the feasibility of getting a soil level. We tried using contours, which gave us a better size based on the total pixels in the contour, but when using contours it was difficult to process the image in such a way that we could isolate the actual waste from its shadow or other shadows from other objects.

Deliverable:

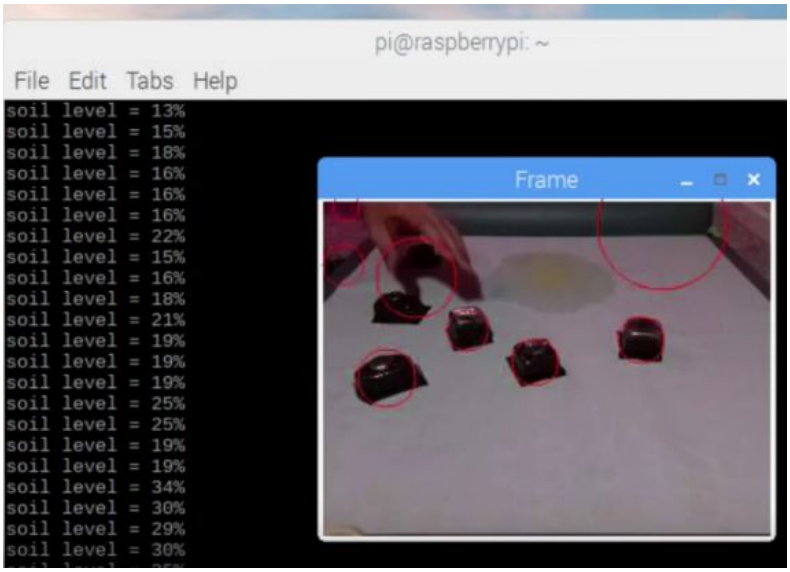


Figure 1.8 Demonstrating the feasibility of obtaining a pad soiled level

- **Reiterate**
  The final version of the 3D printed mount is shown in the Hardware section.
- **Detecting presence of the dog**
  We completed this milestone by using differential images as mentioned above.
- **Ability to send signals**
  We tested the feasibility of this milestone by hooking up an LED to the Raspberry Pi. The LED would turn on when there is a dog on the pad determined through motion detection, and the LED would turn off when there is no dog present. In doing this test, we were able to demonstrate that Raspberry Pi can send a signal indicating whether or not the pad is ready to advance. We conclude this because sending a signal to the Brilliant Pad via UART would be done in a similar fashion.
- **Communicate with Brilliant Pad**
  We were not able to complete this milestone because we didn't have the hardware required, an FTDI usb to serial cable, and we didn't have the updated version of the control module that would have been able to communicate with us.
- **Integrate vision system with wireless system**
  We were not able to complete this milestone because of time constraints.

Conclusion

In this project, we aimed to improve the Brilliant Pad by fixing its main downsides and constraints. The Brilliant Pad's IR sensor has a narrow field of view making it easy for the dog to avoid being detected, and it had trouble determining when to advance the pad. Through the help of the Raspberry Pi and OpenCV to process images obtained from a Raspberry Pi Camera, we were able to detect waste and to improve the Brilliant Pad's ability to detect the dog. The camera module could view about 75% of the pad which is approximately ten times more than what the current IR sensor could see. Additionally, with the ability to detect waste, the pad can now advance when needed, and not be constrained to only a set amount of times. With these improvements, the Brilliant Pad has more tools to combat against a dog owner's greatest fear, knowing when and where the dog can go to the restroom.

References

[1] Brilliant Pad Website
https://www.brilliantpad.com/

[2] Blob Detection using OpenCV
https://www.learnopencv.com/blob-detection-using-opencv-python-c/

[3] OpenCV: SimpleBlobDetector
https://docs.opencv.org/3.3.1/d0/d7a/classcv_1_1SimpleBlobDetector.html

[4] Motion detection using a webcam, Python, OpenCV and Differential Images
http://www.steinm.com/blog/motion-detection-webcam-python-opencv-differential-images/

[5] OpenCV: Operations on Arrays
https://docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html

[6] OpenCV: Contours
https://docs.opencv.org/3.3.1/d4/d73/tutorial_py_contours_begin.html