Object Detection

Difficulty: 3/5

Video Link: https://www.youtube.com/watch?v=YmKg95CTtbc

Introduction

As a human, we naturally look at the things around us and recognize the different objects we see. Your phone, a can of soda, a pencil. Any of these things are easy to identify. But how does a computer do that? We can plug in a camera and tell the computer to take a picture. It will then look at that picture, compare it to a database of millions of other objects that have been identified in other pictures, and tell you what it believes the object is. Amazon AWS provides a computer to do the image recognition for you, all you need to do is provide the picture.

Parts List

For this project, you will need:

- Raspberry Pi
- USB Camera Module

AWS Services

This project uses the following AWS modules:

- Amazon Rekognition – Amazon AWS's image recognition service.
- Amazon S3 – Store files in the cloud

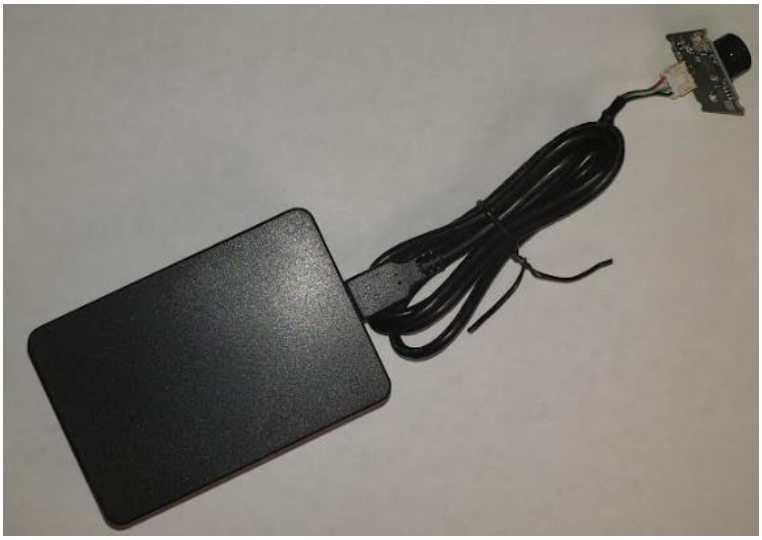
Code Requirements

This project uses the following libraries:

- Boto3 – Library for easy communication with AWS
- OpenCV – Open source computer vision library
- Datetime – Library for working with dates and times

Part One: The Circuit

The connections to be made in this circuit are simple, since the only device we are connecting is the USB Camera. The Raspberry Pi has four USB ports available. Connect the camera to any of these 4 connections.



Part Two: The Code

This section will describe the though process behind the sample code provided with this document. It is important to note that the code provided is not the only possible solution.

Writing the Raspberry Pi Python Code

Before we begin writing code, let's think about what we want the code to do. Planning is an important part of the code writing process. Making a clear step by step guide can make it easier to manage what features to work on.

The steps the code will follow in this tutorial are as follows:

1. Perform initial setup of devices
2. Create an S3 bucket if it doesn't already exist
3. Take a picture
4. Upload the image to the bucket
5. Tell Rekognition to label the S3 image
6. Report the detected items to the user's console

With this plan in place, we are now able to begin programming. The first step is to setup and import the needed libraries as well as configure devices and connections. This step begins on line 2 of the sample code.

1. Import boto3, OpenCV (cv2), and datetime
2. Determine a name for the S3 bucket and store in a variable
3. Determine region name and store in a variable
4. Setup the camera in video capture mode
5. Setup boto3 clients for both S3 and Rekognition

The next step is to create an S3 Bucket. However, we only want to create a bucket if the bucket does not exist. This step begins on line 21 of the sample code.

1. Get all the currently existing buckets

2. Iterate through the list of buckets, comparing names to the desired name
    a. If the desired name is found, then we don't need to create the bucket, and we can stop iterating
3. If the desired name is not found
    a. Create a bucket with the proper name and region
    b. Make the new bucket private

The next step is to take a picture. The library cv2 makes this easier. This step begins on line 54 of the sample code.

1. Read the current frame of video from the camera
2. Create a string representation of the current time to use as the filename. Store in a variable
3. Using the cv2 library, write the image to a file.

Next, we need to take the buffered image and send it to the S3 bucket. Boto3 makes this simple. This step begins on line 61 of the sample code.

1. Open the saved image file in binary read mode
2. Put the image in the S3 bucket

Next, we need to tell Rekognition to label the S3 image. When we do this, S3 will report the labels back to us. This step begins on line 73 of the sample code.

1. Send a label command to Rekognition, including the bucket name and image name.

Finally, we need to take the reported data and display it to the user. For simplicity in this project, we will only be writing the data to the console in an easy to read format. If desired, you could create a GUI and display the data there.

1. Loop through the array response of labels, and print out each label to the console

Working sample code has been provided and is outlined in steps so that it is easier to follow.

Setting up AWS

Amazon Rekognition can receive images in two ways. First is a direct image upload. The bytes of an images get sent directly to Rekognition, and Rekognition replies with all the labels that it can find. The second method, which we will use in this tutorial, has you upload an image to an S3 Bucket. Then, Rekognition can be told to label the image from S3. With this method, we can view the picture whenever we want and can even add custom labels later.

There is no specific setup required for using Rekognition. With S3 buckets, we can either manually create the bucket or create the bucket from code. The code that we've written creates the bucket for us, so there is no setup required for S3 either.

It is important to note that you will need to decide what region you want the S3 bucket to be in, and this region will also need to be the region for your Rekognition calls.

Part Three: Result

Now that everything has been setup, we can run the code and see it in action. To run your code, open your command line to the location of the python file. Then type “python” followed by the name of your file. For instance, if your file was named “ObjectDetection.py” you would type “python ObjectDetection.py” followed by pressing the enter key. If your code has errors, you will get notified. Do not let this scare you! The errors describe what happened and where, so take the time to read the message.

If your code does not have errors, it is now time to verify the proper performance of your program. Did you get labels back? Do they make sense? If they don’t, make sure the image in S3 is correct, and make sure you are telling Rekognition the correct picture to label.

If you are receiving labels that make sense, then congratulations! You have completed this tutorial! It is important to note that more than one test needs to be run in order to truly verify that the code works properly. Here is an example of what the output might look like:

```
Bucket pete-object-detection-bucket already exists
Reading file...
Uploading file...
I am 99.99% sure that I see a Furniture
I am 99.36% sure that I see a Table
I am 98.87% sure that I see a Desk
I am 93.05% sure that I see a Computer
I am 93.05% sure that I see a Electronics
I am 83.23% sure that I see a Pc
I am 73.46% sure that I see a Drawer
I am 72.97% sure that I see a Screen
I am 69.21% sure that I see a Monitor
I am 69.21% sure that I see a Display
I am 65.43% sure that I see a Bed
I am 59.87% sure that I see a Interior Design
I am 59.87% sure that I see a Indoors
```

Test your knowledge

The following challenges are some ideas to test your understanding of the current tutorial and information from the other tutorials.

- Connect a speaker to the Raspberry Pi and find use a text to speech library to audibly communicate the objects detected in the image.
- Use Amazon SNS to send a text message or email alert when a specific object is identified.