GreenVision Documentation

Ian McVann, Ethan Emmons

Jun 21, 2019

Table of Contents:

Hello, and welcome to the Green Machine's Vision Processing Docs for the 2019 season.

CHAPTER 1

About

Hello, and Welcome to team 1816's Vision Code for the 2018-2019 Season. This year, we attempted to make our code as transparent as possible, and in order to help other teams use or code, we created this documentation. Our code this year is a one-file python program, with flags for various features, including:

- Image Capture
- Vision Processing
- Distance Recognition
- Yaw Determination
- And Video Capture

In order to get started using our code, click the Installation button on the side of the screen. These docs will guide you from setup, to install, to usage, and will hopefully answer all of your questions.

Good Luck, and see you on the field!

CHAPTER 2

Installation

In order to get setup with GreenVision Python, follow the steps below:

2.1 Getting Setup

To start, we need to clone the files off of Github. In order to do that, install Git from one of the links below:

- Git for Windows
- Git for Mac
- Git for Linux

Once Git is installed, we need to clone the GreenVision Repo. In order to do so, open git bash in whatever directory you want to clone to and run the following:

```
git clone https://github.com/TheGreenMachine/GreenVision.git
```

Next, you are going to want to install python if you haven't already. Download and install it from one of the links below. On Windows, make sure to check the "Add to Path" checkbox and press "Disable Path Limit" after install:

- Python 3.7.2 for Windows
- Python 3.7.2 for Mac
- Python 3.7.2 for Linux

Once Python finishes installing, navigate to the GreenVision folder and run the following command to install the required libraries:

```
pip3 install -r requirements.txt
```

This will install all of the required packages. Once that's done, move on to the next step of the install in order to calibrate your camera.

2.2 Calibration

Now that you've cloned and installed the requirements for GreenVision, you now need to change the values in **values.json** to match your current setup.

First of all, the server IP. This is the ip of the Robo-Rio, so that the program can connect to network tables and push values. Change this to match your IP.

```
{
  "server-ip": "10.18.16.2",
}
```

Second, color range. You really shouldn't need to change these, but if contour recognition is not working with your green light, you may need to.

```
{
  "lower-color-list": [
    50.0,
    55.03597122302158,
    174.28057553956833
  ],
  "upper-color-list": [
    90.60606060606061,
    255,
    255
  ],
}
```

Third, camera horizontal field of view. This should be what your camera says its FOV is in its tech specs.

```
{
  "fish-eye-cam-HFOV": 90,
}
```

Finally, image resolution. The lower this is set, the less extra contours your camera will pick up. However, if you go too low, it won't pick up anything.

```
{
  "image-width": 640,
  "image-height": 480,
}
```

Once you've tuned all of these, move on to the next section, usage.

CHAPTER 3

Usage

Now that you've installed and setup everything, let's move on to how to actually use GreenVision. First things first, that GreenVision is broken up into different categories:

- Vision Processing - This includes our actual robot image recognition code for the season.
- Image Capture - A utility to take pictures for calibration and off-robot use.
- Video Capture - A utility to take videos from the camera.
- Distance Table - A utility to log distance and contour area values in order to calibrate distance.

Since each of the categories has their own CLI flags, click on the links below to see info on each one.

3.1 Vision Processing

The Vision Processing module is used on the actual pi during matches to detect retro-reflective tape and send data back to Network Tables.

The Vision Processing module can be used through the following command:

```
python3 GreenVision.py vision [arguments]
```

The supported OPTIONAL arguments are:

```
-v or --view: Toggle contour and mask window. This allows you to see what the pi is␣
↪seeing.

-d or --debug: Toggles debug output to console. This shows coordinates of all the␣
↪contours, as well as additional data.

-mt or --multithread: Toggles multi-threading. Increases processing speed.

-th [double] or --threshold [double]: Adjusts color thresholds by 50.0 or less. Use␣
↪if recognition is not working.
```

(continues on next page)

(continued from previous page)

```
-nt or --networktables: Toggles sending data to Network Tables. Default is false.

-h: Shows command help.
```

The supported REQUIRED arguments are:

```
-src [input] or --source [input]: Sets source for image processing - [int] for camera,
↪ [path] for file.
```

3.2 Image Capture

The Image Capture module is used to capture a picture form the camera with specific settings.

The Image Capture module can be used through the following command:

```
python3 GreenVision.py image_capture [arguments]
```

The supported OPTIONAL arguments are:

```
-cw [int] or --width [int]: Sets width of camera resolution. Default is defined as␣
↪image-width in values.json.

-ch [int] or --height [int]: Sets height of camera resolution. Default is defined as␣
↪image-height in values.json.

-n [string] or --name [string]: Sets a name for output file. Can use {} to input␣
↪distance.

-h: Shows command help.
```

The supported REQUIRED arguments are:

```
-s [input] or -src [input] or --source [input]: Sets source for image processing -␣
↪[int] for camera.

-d [int] or --distance [int]: Sets distance from target in inches. Required for␣
↪naming.
```

3.3 Video Capture

The Video Capture module is used to record video form the camera with specific settings.

The Video Capture module can be used through the following command:

```
python3 GreenVision.py video_capture [arguments]
```

The supported OPTIONAL arguments are:

```
-f [int] or --fps [int]: Sets FPS for video. Default is 30.

-cw [int] or --width [int]: Sets width of camera resolution. Default is defined as␣
↪image-width in values.json.
```

(continues on next page)

(continued from previous page)

```
-ch [int] or --height [int]: Sets height of camera resolution. Default is defined as␣
↪image-height in values.json.

-n [string] or --name [string]: Sets a name for output file.

-h: Shows command help.
```

The supported REQUIRED arguments are:

```
-s [input] or -src [input] or --source [input]: Sets source for image processing -␣
↪[int] for camera.
```

3.4 Distance Table

The Distance Table module reads files in from the Test Images folder following the default naming scheme of *opencv_image_{}*, and then outputs a csv file with contour area on x and distance on y.

The Distance Table module can be used through the following command:

```
python3 GreenVision.py distance_table [arguments]
```

The supported OPTIONAL arguments are:

```
-c or --capture: Toggles capture of new images. Default is False.

-cw [int] or --width [int]: Sets width of camera resolution. Default is defined as␣
↪image-width in values.json.

-ch [int] or --height [int]: Sets height of camera resolution. Default is defined as␣
↪image-height in values.json.

-o [string] or --output [string]: Sets a name for output csv file. Default is␣
↪distance_table.

-th [double] or --threshold [double]: Adjusts color thresholds by 50.0 or less. Use␣
↪if recognition is not working.

-h: Shows command help.
```

The supported REQUIRED arguments are:

```
-s [input] or -src [input] or --source [input]: Sets source for image processing -␣
↪[int] for camera.
```