Contents

[Intro 1](#_Toc6154022)

[Generate GripPipeline.java based on field lighting at the venue 2](#_Toc6154023)

[Using GRIP to generate GripPipeline.java 3](#_Toc6154024)

[Get vision software from GitHub and build it on the vision system 4](#_Toc6154025)

[Install vision software so it will run automatically when power is cycled 5](#_Toc6154026)

[Collect sample pictures from the vision system after practice and qualification matches 6](#_Toc6154027)

[Examine and edit the vision code using Eclipse 7](#_Toc6154028)

[Use Debug Annotation in the Browser Image 8](#_Toc6154029)

[Replay a Match using saved sample images 9](#_Toc6154030)

# Intro

This document provides an overview of operation and programming of the vision system used by team 4276. The target audience is a high school student team member, not a mentor, but feel free to ask questions.

You will need a laptop - preferably your own but a team laptop will do. You need admin password access for setting up the first time but (probably) won’t need for running at the venue. All of the following must be installed on the laptop:

* Git <https://git-scm.com/>
* TortoiseGit <https://tortoisegit.org/>
* Putty <https://www.putty.org/>
* GRIP <https://wpiroboticsprojects.github.io/GRIP/#/>
* YouCam <https://www.cyberlink.com/downloads/trials/youcam/download_en_US.html>

(Get YouCam “Essential” version - it is completely free, not a trial

* Eclipse <https://www.eclipse.org/downloads/>

You will need a long network cable, 15 feet or more. The team has a gray one for use as tether, but if you can find another it would be better. The driver station can use USB if no alternate is available.

The vision specialist job takes a significant amount of time at competitions to accomplish the following:

* Generate GripPipeline.java based on field lighting at the venue
* Get vision software from GitHub and build it with the new GripPipeline.java
* Run experimental software on the vision system without affecting what it does when power is cycled
* Install new software so it will run automatically when power is cycled
* Collect sample pictures from the vision system after all practice and qualification matches

# Generate GripPipeline.java based on field lighting at the venue

* Vision calibration is allowed on the field early on the first (practice) day, probably 9am.
* First thing upon arrival you must find the vision camera and battery pack in whatever tote(s) it was packed.
* Connect the battery pack to the LED ring and make sure it lights up when turned on
* Plug the camera into your laptop and make sure it displays OK in YouCam. Turn off auto exposure and experiment with the exposure setting to find the 3 or 4 darkest settings that still produce a non-black image.
* Use a tape measure to get the approximate height above the ground that the camera is mounted on the robot
* Bring laptop, camera and tape measure to the driver meeting that will probably take place before vision calibration is allowed on the field
* Once on the field
  + Use tape measure to sit 2 feet from the hatch feeder.
  + Use tape measure to estimate how high to hold the camera
  + Take pictures with YouCam at the 3 or 4 darkest exposure levels
  + Repeat at 4 and 6 feet distance
  + If have more time repeat collection for cargo bay and rocket
* After collecting calibration pictures find a table somewhere you can work more comfortably to generate GripPipeline.java

# Using GRIP to generate GripPipeline.java

* Sort through images from vision calibration or collected from the vision system during a match. Look for images that have a clear view of the vision target and copy them to a separate folder
* Start GRIP, and if a previous GRIP profile is available open it. Otherwise create a basic pipeline:
  + Blur (3)
  + HSV Threshold
  + Find contours
  + NTPublishContoursReport
* Click “Add Source” and select “Images”
  + Browse to the folder containing the selected images, and use Ctrl-A to select all of these images
* Adjust pipeline to produce a good contour result

<https://wpilib.screenstepslive.com/s/currentCS/m/vision/l/463566-introduction-to-grip>

* Tools 🡪 “Generate Code”
  + Java
  + (Don’t check the box for WPILib VisionPipeline)
  + GripPipeline
  + SaveLocation (Somewhere you can find it, like your Documents folder)
  + Package Name: frc.robot

# Get vision software from GitHub and build it on the vision system

* Right click on your ‘Documents folder’ and select ‘Git Clone…’
  + <https://github.com/Team4276/2019_VisionSystem>
* Turn on the robot to power on the vision system.
* Disconnect the network cable from the vision system and connect the long network cable to your laptop
* Configure the wired network port of your laptop to connect to the robot subnet 10.42.76.xxx
  + Open network and sharing center
  + Click ‘Change adapter settings’ (use admin password)
  + Right click the wired network and select ‘Properties’
  + Double click IPv4 settings
  + Select “Use the following IP Address”

IP Address: 10.42.76.222

Subnet: 255.255.255.0

Gateway: 10.42.76.8

* In Windows File Explorer (e.g. where you find “Documents”) enter the following in the address bar at the top:

[\\10.42.76.8\rpi3share](file:///\\10.42.76.8\rpi3share)

* U: pi
* PW: viking
* This will give you full access to all of the files on the vision system so you can easily copy files to and from
* Copy source code from GitHub to the vision system
* Browse to [\\10.42.76.8\rpi3share\home\pi\2019\_VisionSystem\raspi](file:///\\10.42.76.8\rpi3share\home\pi\2019_VisionSystem\raspi)
* Rename the camsvr folder to camsvr2
* Copy C:\Users\xxx\Documents\2019\_VisionSystem\raspi\camsvr 🡪 [\\10.42.76.8\rpi3share\home\pi\2019\_VisionSystem\raspi](file:///\\10.42.76.8\rpi3share\home\pi\2019_VisionSystem\raspi)
* Copy GripPipeline.java from where you saved it from GRIP to replace the currently checked in version
  + Copy it here: [\\172.16.0.8\rpi3share\home\pi\2019\_VisionSystem\raspi\camsvr\src\frc\robot](file:///\\172.16.0.8\rpi3share\home\pi\2019_VisionSystem\raspi\camsvr\src\frc\robot)
* Start Putty
  + Connect to 10.42.76.8
  + U: pi
  + PW: pi
  + This will give you a command line prompt the executes on the vision system
* Stop currently running vision software
  + In Putty, list running processes named “java”

ps -Al | grep java

* + The number in the fourth column is Process ID or “PID”. For example in the following line the PID is 943

0 S 1000 943 920 40 80 0 - 123856 futex\_ pts/0 00:00:04 java

* + Kill all of the currently running vision processes, for example:

kill -9 943

* Build the vision software

cd /home/pi/2019\_VisionSystem/raspi/camsvr

ant clean

ant build

* Run an experimental version of the vision software

cd /home/pi/2019\_VisionSystem/raspi/camsvr

ant Main

# Install vision software so it will run automatically when power is cycled

* The following takes a minute or two. The entire contents of the camsvr folder will be copied to a release area and built there

cd /home/pi/2019\_VisionSystem/raspi/camsvr

sudo ./do\_install.sh

# Collect sample pictures from the vision system after practice and qualification matches

Connect the network cable to the vision system and turn on the robot power as soon as possible after the robot returns to the pit following a match.

Map a network drive letter to the vision system

* In Windows File Explorer, right click on “This PC” and select “Map network drive…”
* Choose “R:\” (for robot)
* Folder: [\\10.42.76.8\rpi3share](file:///\\10.42.76.8\rpi3share)
* Check ‘Reconnect…’ and ‘Connect using different credentials’
* Click Finish
* U: pi
* PW: viking

Open a Windows command prompt

* Click the windows start button in the lower left and enter ‘cmd’ in the search window

In the Windows command prompt create a folder to receive image samples from the match, copy the images, and then delete the images from the vision system so the next match will be recorded separately. For example copying images for qualification match 21:

cd C:\Users\xxx\Documents\FRC2019\Champs

mkdir Q21

cd Q21

copy R:\home\pi\log\\*.\* C:\Users\xxx\Documents\FRC2019\Champs\Q21

delete R:\home\pi\log\\*.\*

(Answer “Y” for ‘Are you sure?’)

# Examine and edit the vision code using Eclipse

* Start Eclipse, set workspace to [\\10.42.76.8\rpi3share\home\pi\2019\_VisionSystem\raspi](file:///\\10.42.76.8\rpi3share\home\pi\2019_VisionSystem\raspi)
* File --> ‘Open Projects from File System…’
* Import Source: [\\10.42.76.8\rpi3share\home\pi\2019\_VisionSystem\raspi\camsvr](file:///\\10.42.76.8\rpi3share\home\pi\2019_VisionSystem\raspi\camsvr)
* Check only box for ‘camsvr’
* Expand tree control camsvr 🡪 src 🡪 frc.robot

You can now browse and edit the source code for the vision system.

If you change anything and want to test it, save using Ctrl-S or File 🡪 ‘Save All’ and the build with ant in the putty prompt as described above.

Interesting source files:

* Main.java (Where camera parameters are set)
* QGripThreadRunnable.java (Where the GripPipeline is used)
* CargoBayFinder.java (Uses the output of GripPipeline to first narrow down the choices (too small, too big, too far apart, rectangles wider than tall, etc.) and finally recognize as a cargo bay when the rectangles tilt toward each other.

# Use Debug Annotation in the Browser Image

When the vision program is running you can view raw camera output in a browser at <http://10.42.76.8:1185/> and annotated camera output at <http://10.42.76.8:11856/>

The annotated output for normal operation is:

* Green tick mark on the right signifying that all of the pixels above it are ignored
* Yellow highlighted rectangles for the “found” cargo bay, (if any) and a yellow plus sign in the center between them.

You can turn on more annotation output for debugging purposes:

* White text at the top of the display indicating number of rectangles found and reasons for not using them
* Additional cargo bay rectangles (outlined in green)
* Too small/big/far rectangles (outlined in blue)
* Bad tilt rectangles (outlined in orange)

To turn this annotation on or off, edit JTargetAnnotation.java and look for this line:

public Boolean m\_showDebugAnnotation = true;

TRUE 🡪 Show debug annotation

FALSE 🡪 normal operation

# Replay a Match using saved sample images

Sort through sample images saved from a previous match, e.g. C:\Users\xxx\Documents\FRC2019\Champs\Q21 to find images just from start to end of the match.

Copy just the match images to the vision system “test” folder at \\10.42.76.8\home\pi\test

In Main.java look for this line:

private static ImageSourceType m\_imageSourceType = ImageSourceType.IMAGE\_SOURCE\_CAMERA;

Change to IMAGE\_SOURCE\_JPEG\_FOLDER and it will cycle through the images in the ‘test’ folder instead of using camera input.

**\*\*\* WARNING \*\*\* Do not forget to change this back to camera input before checking in any changes!**

Note that match samples are a mix of raw and annotated frames, so the previously annotated frames will get annotated again during the replay which can be confusing for debugging. This will be less so if you have debug annotation turned off during matches and on only for the replay, although you may see more than one “Found” cargo bay in yellow.

Note that the image shown in the browser is reduced in size by a factor of 4 to reduce bandwidth use during a match. You can delete the contents of the “log” directory ([\\10.42.76.8\home\pi\log](file:///\\10.42.76.8\home\pi\log)) and the images collected during the