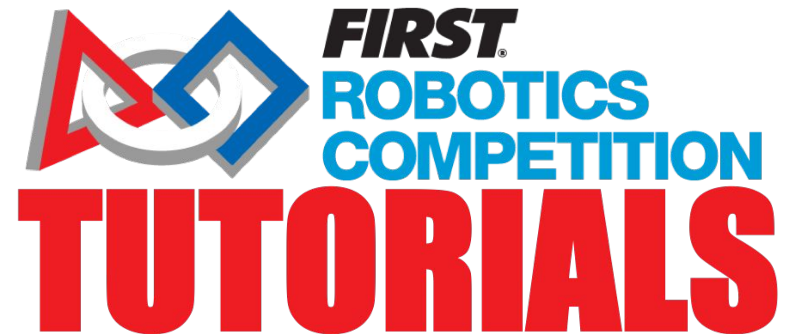
Intro To Vision Targeting

TEAM 4150



Overview

There are many different ways to use vision on your robot, including

- Targeting
- Monitoring Your Robot's Subsystems
- Viewing the Field from the robot's perspective

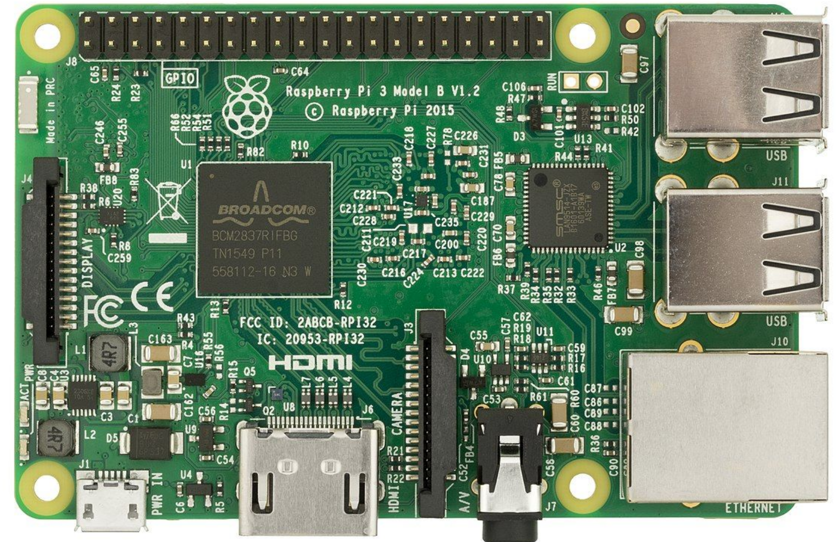
. We're going to be focusing on vision targeting in this tutorial. If you are looking to try and make a driver or monitor camera, FIRST FRC Documentation covers this topic in detail (link). This tutorial is to help teams unfamiliar with vision targeting to get started.

The main way a vision program works is by shining a bright light at a target, filtering out the bright spots made by the reflective tape, and working off the contour of that shape to harvest data. This tutorial is focused on setting things up, collecting useful resources, and giving tips for the data processing step.

Please bear in mind this is a complicated topic. Our team used C++ and the specific procedure can vary a lot depending on programming language, camera, etc. and it requires a lot of incremental testing to get right. Since it's impossible to cover every language and configuration a good bit of nuance is left out of this tutorial, but if you have a problem ***don't give up.*** you are likely to encounter roadblocks while getting started with this, reach out for help, look at the docs or just keep at it in some way, don't become discouraged.

Step 1: The Raspberry Pi

Before we get started we have to buy and set up our ***Raspberry Pi***. The ***Raspberry Pi*** is a little computer that will be running our vision program. There are a couple different models you can get, anything above a model 3 (3, 3a+, 3b, 4, etc.) is generally recommended and can be picked up from Amazon or the Raspberry Pi Foundation's website (link).



Raspberry Pi, (Credit Wikimedia Foundation)

For setting up your Pi the FRC Docs has a wealth of information on the subject including wiring configurations and such which is frankly beyond the scope of this tutorial. We recommend you go through their tutorial on this topic (link).

Step 2 - Software Setup

Github Logo (Courtesy of Github)

Now that you have your Raspberry Pi primed and ready, we can turn to the software you'll need for your vision workflow.

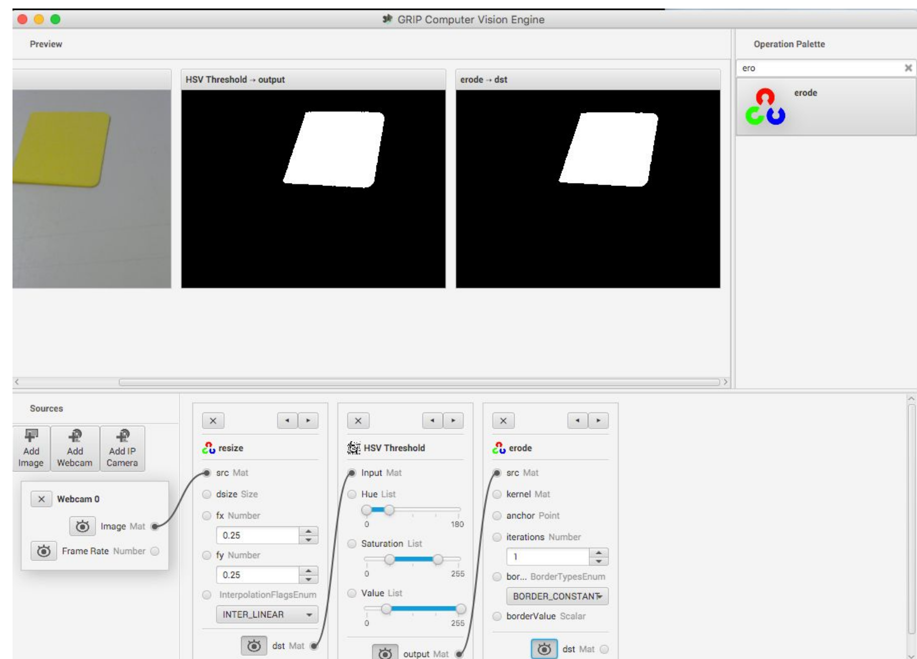
- GRIP (github) (releases)
- Compiler:
  - Raspberry Pi Toolchain (For compiling C++ Code) (github) (releases)
  - More information for Python and Java is mentioned in the docs, though it may require some testing (link)
- Any text editor of your choosing
- Web Browser

Step 3 Vision Workflow - GRIP

The vision program is split up into three parts the first being:

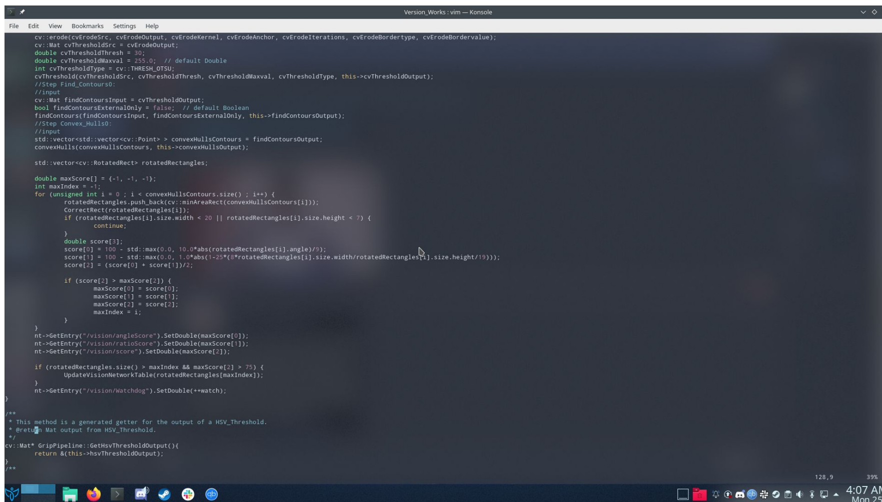
The GRIP Code

This is where the magic happens with vision processing. This takes the raw video from the camera and applies filters to it to get our "blobs" that contain data about what we think are vision targets. This is done in the GRIP program and then exported into a language of your choosing (Java, Python, C++). The FRC Docs covers this in depth, **we suggest you read this before progressing in this tutorial** since some of the concept taught in the grip tutorial will be mentioned and you may get lost (link).



GRIP Interface (Courtesy of WPILib Docs)

Step 3 Vision Workflow - Data



Screenshot of A bit of the Data Handling Program (Note part of this is the autogenerated Grip program so don't worry if it looks complicated).

The second part is:

Data Handling

Here we take the data from the blobs in from the GRIP code and we calculate the distance from the target, angle to the target, and any other information we need. Although the exact algorithm used will depend on what data you collected we'll go over generally how you want to be processing the data. This step is done in either Python, Java, or C++ and is built off a template that can be downloaded off of the Raspberry Pi vision server under "Application."

NOTE: You can write this in a language other than the one used for the RoboRIO e.g. if you write your robot code in labview you can write your vision code in something else

Step 3: Vision Workflow - Controller

The last part of the Vision Workflow is

The Robot Controller

This is when we take the data we've collected and act on it. This can be used for getting your bearings during autonomous, making auto aiming systems for shooters and placement subsystems and much more. This is part of your standard RoboRIO program so if you're using LabVIEW or some other language you'd still use that here.

To transfer the data from your Raspberry Pi to your RoboRIO you use the network table. A network table is a giant grid where you have a **"key"** which is the name of a piece of data, for instance "angle", and a **"value",** which in this case could be 90°. In our program all of the values you need to control the robot should be posted to the network table. Specifics on how to use the network table in C++ and Java can be found in the WpiLib docs under "WPILib Java/C++ API Docs" (link). Information for Python (link).

Step 4: Scoring

We're now going to dig in a bit deeper into the data handling part of the vision workflow, starting with scoring.

Our list of potential targets from GRIP will include a lot of "noise," reflections, objects in the background, etc. that the program filters out as a target. In order to whittle down this list we can employ a ***scoring algorithm.***

How it works is it scores a blob based on how much it seems like a target based on its attributes and then filters out blobs with low scores. Some possible scored data points include:

- ***Width/Height Ratio*** (measure the target's width/height as a benchmark)
- ***Size*** (Things that are too small probably aren't the target)
- ***Skew angle*** (Most targets won't be tilted)

By adding together the subscores you can normally filter out everything but the target. You will need to fiddle around with the scoring formula for each subscore (for instance you can change the degrees/coeffients to make them drop off more or less along with other things)

Step 5: Calculations

Most of the data you need such as:

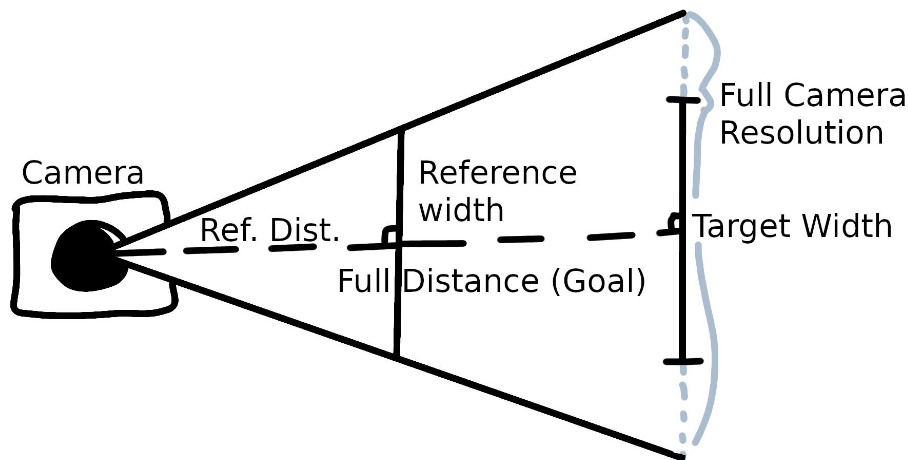
- ***Angle*** *offset from the target*
- ***Angle*** *offset vertically from the target*
- *The* ***Screw*** *of the target*

Can all be directly gathered from the grip data once you know which blob is actually the target via scoring. You may need some ***opencv*** calls, which is a vision processing library that powers ***GRIP***, to get certain data points (link), but this is advanced and normally ***GRIP*** will suffice. Once collected these data points just need to be published to the network table for use by the RoboRIO code. The main other variable that you can't get via this method is distance, and it requires a good amount of geometry as explained on the next slide.

Step 5: Calculations - Distance

To find the distance of the target distance, we first need some comparison between feet and pixels. One strategy is to make use of similar triangles. If we take a reference object, say a yard stick, and we move it away from the camera until it completely takes up the width of the camera's vision, the ratio of the distance to that stick and the width of that stick equals the ratio between the distance to any line parallel to the camera's vision and the distance to that line, we call this ratio R.

Now if we take the width of our screen and divide that by the width of our vision target in pixel, we get how many times wider the full screen (shown as full camera resolution (***FRC***) in the diagram) is than the target. Since we know the width of the target in feet if we multiply that by how many times larger the ***FRC*** is than the target, we get the width in feet of the ***FRC***. Since the FRC is parallel to the camera like the reference width, the ratio R, between the width and distance is the same. Therefore to solve for distance we multiply that FRC width by R.



Diagram

Long Story Short here's the equation:

$$Dist = \frac{Ref\ \ Dist(ft)}{Ref\ \ Width(ft)} * \frac{Camera\ \ Horizontal\ \ Resolution(px)}{Target\ \ Width(px)} * Target\ \ Width\ \ (ft)$$

Step 6: Tweaking

A large part of vision programming is incremental, you'll find an issue and tweak your program until it works, as mentioned early ***don't give up.*** There are great resources out there on forums and discord full of people willing to help. To finish things up here are a couple tweaks that we found useful while working on vision in general:

- Use green colored LEDs so that the reflective tape on targets glow green, it will make it much easier for your vision program to filter it out that way by sorting via color
- If you don't have a green led, make an led mount and put a green gel over it to color the light
- For the GRIP function that filters out color, add dashboard sliders to be able to remotely change the HSV range filtered out since competition in workshop vs play field can be drastically different

Credits

- This lesson was written by FRC 4150 in partnership with FRC 8027 for FRCTutorials.com
- You can contact the author at froboticsteam4150@gmail.com.

- More lessons for FIRST Robotics Competition are available at www.FRCtutorials.com

This work is licensed under a
Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.