TrackerBox: Object Tracking and Autonomous Robot Steering for FIRST Robotics using a Raspberry Pi and a Webcam.

Mike Ounsworth
Mentor
Team 296 & Team 3710
ounsw@cim.mcgill.ca

Alex Huctwith
Student
Team 3710
kngkeeper@gmail.com

January 6, 2014

Abstract

Our goal is to provide easy access to visual object tracking to teams who do not have sufficient programming knowledge to implement computer vision. We created a software and hardware system that allows teams to train a vision target, say an orange basketball on the ground or a green square on the wall, and have the robot autonomously steer to keep the target in the centre of the image. The system we built should be simple enough that even teams with little programming knowledge can assemble and use it, and cheap and light enough that teams can use one or multiple of them on their robot.

Contents

1 Introduction

Tracking the position of an object in a video feed is a well studied problem in computer vision and there are many algorithms which, given a target image, will track similar-looking objects in live video. The FIRST Robotics Game Design Committee (GDC) does an excellent job of making the game pieces and vision targets bright and distinct colours, which is great for computer vision! Despite this, very few teams use vision in a significant way. One problem is that while FIRST does provide some basic vision in its C++ / Java / Labview APIs, doing anything more complicated requires heavy-weight vision libraries such as OpenCV. To the knowledge of the authors there is no good way to get OpenCV running on the cRIO. Even if we had access to our vision library of choice there is still the problem of writing the code, which requires significant computer vision knowledge on the part of the team's programmers. We hope to solve both of these problems by providing a pre-compiled software package for Raspberry Pi. You simply buy the hardware ($25 for a Raspberry Pi and $20 for a webcam), download our software onto

the Pi and you're all set!

Our system - which we will refer to as the 'TrackerBox' - acts like a custom sensor meaning that it runs separately from the cRIO and sends its signal back over digital IO pins.

This paper will show some results in section 2 to give you an idea of what it's capable of, then we discuss the hardware you need in section 3 and finally we discuss downloading and using the software as well as give some tips for programming the cRIO in section 4.

2 Overview

This section shows examples of the two different graphical user interfaces (gui's) that are available on the TrackerBox; a full resolution colour gui, and an ASCII gui. By default if a team plugs in a monitor they will see the ASCII gui, we made this decision because the full graphics are quite heavy and lower the framerate to an unacceptable level. If you want to run the full gui and are comfortable with the Linux command line then please follow the instructions in section 4.4.

As an example I trained the system with an image of a yellow tape measure, seen in figure 1. There are two ways to capture a training image: hold the object in front of the camera and click the limit switch (discussed in section 3), or insert a USB stick with a file called 'target.jpg' on it. The USB stick method can be useful if you want to crop the image to centre the target or otherwise clean up the image. Keep in mind that computer vision is very sensitive to changes in lighting and colour, so always take a training image using the same camera in the same environment where you will be using it. A good trick is to unplug the camera from the Pi and plug it into your laptop so that you can be sure to get a good training image.

When the system is running we can see that it tracks the tape measure and draws a blue arrow from the centre of the screen to the centre of the target, fig 2. This blue arrow is the steering signal for the robot in both the X and Y directions.

In the gui pressing 'b' shows us the back-projection image which means "how strongly do we believe that

Figure 1: A good training image of a tape measure.

Figure 2: TrackerBox tracking the tape measure. The arrow does not start quite in the centre because the image has been cropped.

each pixel belongs to the target?" - the more white, the stronger the match. In this algorithm it takes a small window around each pixel and checks if all the pixels in the window match the colours we are looking for. Notice how the yellow parts of the tape measure appear strong white and everything else is ignored, fig 3.

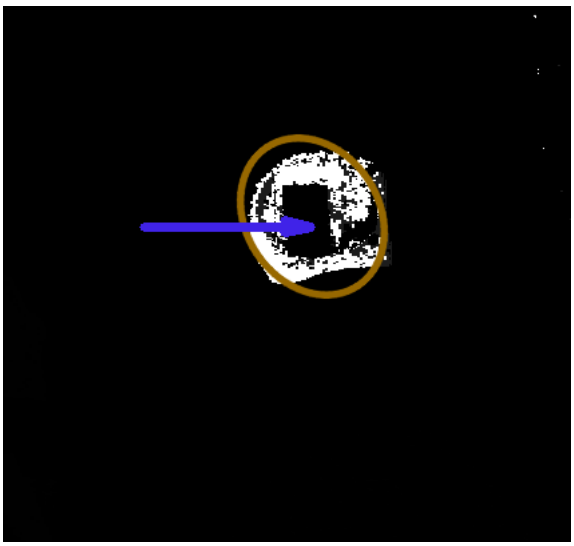

Figure 3: The back-projection image.

For competition use we expect that the Raspberry Pi will not have a monitor attached to it and so it also outputs a steering signal over the digital output pins, see section 3.1 for more information on the pinouts and format of this signal. We decided to leave the ASCII gui running in case a team wants to connect an HDMI monitor to the Pi for debugging, however the full gui shown here takes too much of the Pi's CPU time and slows the system down to about 1 frame per second (fps) which is too slow for robotics use. Instead we made an ASCII console gui of the back-projection where a low probability is shown with a '.' and a high probability is shown with a 'X', an example of this is in figure 4. If you plug in a monitor this is the default display that you should see. We found that with the ASCII gui we get an acceptable framerate of 4 - 5 fps.

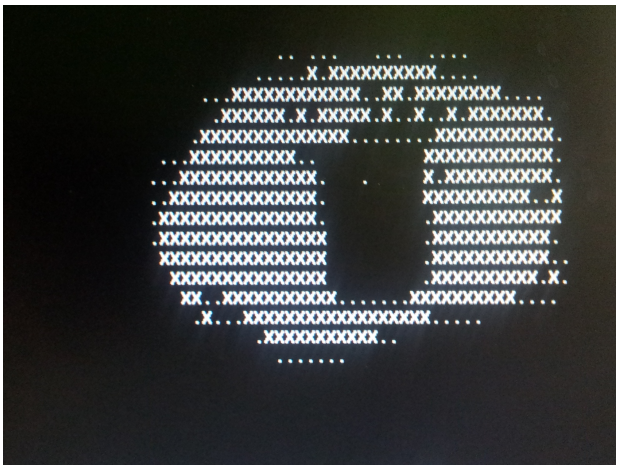

Figure 4: The back-projection image displayed in an ASCII console gui.

3 Hardware

The TrackerBox only has 4 hardware components: a Raspberry Pi, an SD card (which acts as the hard-drive for the Pi), a USB webcam (any old webcam will do) and a simple wire harness for a limit switch (training button) and the pinouts. See section 3.1 and figure 8 for wiring instructions and pin assignments.

The limit switch acts like a camera button, when you press it the TrackerBox snaps a new image to use as the target. This does not have to be a limit switch, it could be any button as long as it is open-off, meaning that when it is un-pressed it reads a 0. In fact when we were developing the system we just touched the wire ends together instead.

3.1 Digital Output Pins

The TrackerBox outputs whether or not it has found the target and where on the screen the centre of the target is using the digital general purpose input / output pins (GPIO pins) on the Raspberry Pi. We chose to treat the X-axis and the Y-axis independently since those will generally be controlled by different motors on the robot, and in many cases only one axis is controllable. For each axis we break the

Figure 5: The TrackerBox only has 4 hardware components.

screen into five regions: Far Left, Left, Centre, Right, Far Right for the X-axis and Far Up, Up, Centre, Down, Far Down for the Y-axis. Figure 6 shows the 25 regions that the screen is broken into, and what signal will be on each of the 6 pins (how to read this: for example "Y_Up, !Y_Ctr" means that the 'Y_Up' pin will have a 1 and the 'Y_Ctr' pin will have a 0). When "Y_Ctr" reads 1 the other pins should be ignored. Note that figure 6 is not to scale, figure 7 shows the actual scale. If you want to change this scale and are an advanced linux user, see section 4.5.

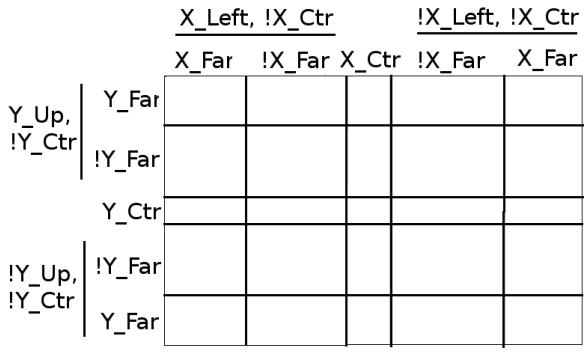

Figure 6: A not-to-scale drawing showing the 25 regions of the screen.

Figure 7: The scale of the screen regions.

There are three more pins not shown in figure 6; two pins for the camera button, "Camera Btn +" and "Camera Btn -", and a pin indicating whether or not it has found a target. Figure 8 shows a map of the GPIO header with the pins we have used.

One thing to be aware of is that the Raspberry Pi uses a 3.3V signal for *true* (written 3V3) while the cRIO expects a 5V signal. According to our knowledge the cRIO should behave normally with a 3V3 digital input, but we have net actually tested this yet. If you know more about this we would love to hear from you! I'll give another warning even though

it's not relevant to this project - if you put a 5V signal into the Raspberry Pi's GPIO pins you are guaranteed to destroy the Raspberry Pi, one of the ways they keep the Pi so cheap is by not having internal protection, so be careful with 5V signals!

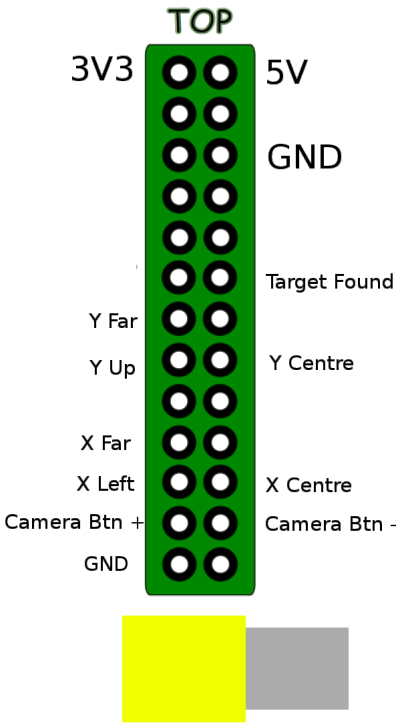

Figure 8: Pin layout on the Rasberry Pi's GPIO header.

4 Software

The first thing to know is that the TrackerBox is a colour-based system, meaning that it takes a colour profile of the target (called a histogram) and then looks for clumps of those colours in the video feed. This leads to a couple non-intuitive points about how to take a good training image which are explained in section 4.2.

The second thing to know is that this is a tracker, meaning that it looks for the target around the position on the screen where it was found it in the previous frame, this has advantages and disadvantages. The main advantage is that trackers tend to have higher frame rates than a brute-force search, but sometimes it loses the target if it moves too quickly or is temporarily hidden behind another object. The main difference comes when there are multiple target objects in the frame - say multiple orange basketballs on the ground; a tracker should keep following the same object and ignore the others. Another behaviour quirk is that if the object it is tracking goes out of frame - say off the left edge - the tracker will keep searching around the left edge and if a new object enters on the right side it may not find it right away.

The underlying tracker is a stock implementation of Gary Bradski's CAMSHIFT algorithm found in the OpenCV library [1].

4.1 Installing the Operating System

The Raspberry Pi runs Linux. We took one of the standard Raspberry Pi linux distributions, added our software and changed various launch scripts to run the tracker at boot, listen for USB sticks and disable the graphical environment. We have made all these changes available as a bundled image file that you can download and get working with no linux knowledge. Currently we have not found a webserver that will host files as large as we need, we will update this document as soon as we have the image file on a public website. In the meantime please email the authors for the image.

There are may guides online which explain how to install an image file onto a Raspberry Pi, for convenience we include short instructions here.

Windows:

1. Insert your SD card and find it's drive letter in "My Computer" (it will be something like E:\)

2. Right click on the file "Win32DiskImager.exe" in the windows directory of your download folder and select "Run as Administrator"

3. Open the image file "TrackerBoxImage.img" from Disk Imager

4. Select the drive letter of your SD card (be sure to get this right, or you will destroy the data on your computer's hard drive)

5. Click write if you are ready (note that this will wipe everything already on the SD card) wait until it finishes (this can take some time)

6. Exit Disk Imager and remove the SD card

7. Put the SD card in the SD slot of your Raspberry Pi. It will autostart the tracker whenever it is plugged in

Ubuntu (graphical):

1. Install the usb-imagewriter package (available in Software Centre, apt://usb-imagewriter)

2. Select the image file "TrackerBoxImage.img" in imagewriter

3. Select the drive to write it to. You can identify the SD card using the "Disks" utility (it will be something like /dev/sdb or /dev/mmcblk0) be sure to get this right, or you will destroy the data on your computer's hard drive

4. Click "Write to Device" and wait for it to finish (this can take some time)

5. Exit imagewriter and remove the SD card

6. Put the SD card in the SD slot of your Raspberry Pi. It will autostart the tracker whenever it is plugged in

Linux (command line):

1. Use the "cd" command to change your working directory to your download folder (probably /home/<Username>/Downloads/ TrackerBoxImage)

2. Insert your SD card and run the command "df -h" to list your drives. Find the name of the SD card (it will be something like /dev/sdb or /dev/mmcblk0)

3. Run the command "sudo dd bs=4M if=TrackerBoxImage.img of=<your full device name from above>" be sure to get the device name right, or you will destroy the data on your computer's hard drive

4. Wait for dd to finish (this can take time, it will return you to a bash prompt when done) and remove the SD card

5. Put the SD card in the SD slot of your Raspberry Pi. It will autostart the tracker whenever it is plugged in

Mac:

1. Run "PiFiller.app" from the Mac directory of your download folder

2. PiFiller will automatically identify your SD card and guide you through the flashing process. Just remember to select the image file "TrackerBox-Image.img"

3. When finished remove the SD card

4. Put the SD card in the SD slot of your Raspberry Pi. It will autostart the tracker whenever it is plugged in

For improved performance we recommend that you overclock your Raspberry Pi. There are many good guides on the internet so we will not include one here.

4.2 Training the System

Before we talk about how to train it, a few words of warning about working with a colour-based vision system: changes in lighting effect everything! For example most indoor lights have a yellow or blue tint, so if you train the system in your shop then don't expect it to work in natural light or arena floodlights which are very white and much brighter. Also each camera has a slightly different colour profile, so make sure your training image is taken with the same camera that you intend to use. Finally pay attention to shadows, for example if your TrackerBox is buried under your chassis then make sure you take the training image from that location. The bottom line is that like

any other sensor, make sure you calibrate it every time you change environments, and make sure you calibrate it in a realistic setting.

Another thing to be aware of is that the TrackerBox can not track objects which are white or black, it only deals in rainbow colours. There is a good reason for this in that when an image saturates or is too dark it is very difficult to tell colours appart. For this reason it is far more robust to chop off the brights and the darks and ignore them. The downside is that the TrackerBox is fundamentally incapable of tracking a white or black gamepiece.

We have tried to make it as easy as possible to train the system and we provide two different ways to do it: by clicking the camera button on the Pi, and by inserting a USB thumbdrive with a file called 'target.jpg' on it.

When you click the camera button (this is the limit switch, see section 3) the TrackerBox will save the current frame to its SD card and use it as the new target. The most important thing is to make sure the target is in the proper lighting, and that it is (roughly) in the centre of the frame. When training it uses a Gaussian filter over the image to make the centre pixels more important than the outsides, so don't worry if there is some background or other object around the edges.

For best results we recommend that you unplug the webcam from the Pi and plug it into your laptop and take an image with your favourite program. This way you can see what you are taking a picture of, and also you can crop out extraneous background objects. To get the image onto the Pi, first rename it 'target.jpg' and either copy it directly onto the SD card into the folder '/root/withOpenCV' or put it onto a USB stick and plug it into the Pi (we wrote a background script that listens for USB sticks and automatically copies 'target.jpg' to the correct folder).

4.3 Controlling the Robot

The output pins on the Raspberry Pi need to be wired to pins on the cRIO's Digital Sidecar which allows you to read their values in your control software. We suggest that to control a motor with the TrackerBox input you have the motor on two speeds: fast when the object is far, slow when it is close to the centre. If you come up with a more clever control scheme it would be great if you shot us an email telling us what you did! You can adjust the thresholds for Far and Centre by following the instructions in section 4.5.1.

4.4 Getting the GUI (Advanced)

This section contains instructions to run the graphical user interface shown in section 2.

This process will likely require a USB hub since you need a mouse, keyboard and webcam all plugged in at the same time. The first step is to disable the auto-running of our program. This can be done either by inserting the SD card into a laptop or by booting the Pi and switching to tty2 by pressing "ctrl + alt + F2", log in with user: root, pass: root, and use your favourite command line editor. Either way you need to open the file '/etc/profile' and comment out the last line

```
sh /root/launchTracker.sh
```

Then restart the Raspberry Pi. Remember to undo this change when you are finished or the TrackerBox won't do anything! Once the system comes up type

```
startx
```

to start the Xorg graphical environment. Then right-click anywhere to get the menu, select

```
Terminals > Xterm
```

into the terminal type

```
withOpenCV/tracker gui
```

and you're off! Note that the gui allows you to play with the parameters Vmin, Vmax, and Smin which refer to cutoff values for pixel values in HSV space - pixels with V or S values above / bellow this will be ignored. There is also the confidenceThreshold which effects how certain the system must be before it reports that it has found the object. Any changed you make using the sliders will be lost on restart. We have picked a set of values that should work in most lighting conditions, but when the lighting changes significantly (for example from your shop onto a playing field) it is advisable to adjust these parameters. You

may play with the sliders here to see the different effects but note that changes made to the sliders are not permanent. To change the values permanently see section 4.5.2.

When you are finished, remember to re-enable the auto-start in '/etc/profile'.

4.5 Changing Parameters (Advanced)

As with the previous section, the first step is to disable the auto-run in '/etc/profile' by commenting out the last line

```
sh /root/launchTracker.sh
```

This can be done either by inserting the SD card into a laptop, or on the Pi (boot, switch to tty2 by pressing "ctrl + alt + F2"). Similarly all files can be edited either on by inserting the SD card into a laptop, or directly on the Pi, however the Make must be done on the Pi. Remember to undo the change to '/etc/profile' once you are done making changes, or the TrackerBox won't do anything!

4.5.1 Far and Centre thresholds

To change the thresholds for 'Far' and 'Centre' in the X and Y directions, open the file '/root/withOpenCV/trackerMain.cpp' and you will find those constants around line 185 (note that line numbers may change as we do bug fixes and things).

Finally make the compile the code by typing: (on the Pi)

```
>> cd /root/withOpenCV
>> make
```

4.5.2 HSV thresholds and confidenceThresh

The HSV parameters can be found in the file '/root/withOpenCV/tracker.cpp' at line 38 and the confidenceThresh can be found in 'tracker.h' at line 28. These parameters effect how sensitive the system is to similar-looking objects and to various lighting conditions. We have picked a set of values that should work in most lighting conditions, but when the lighting changes significantly (for example from your shop onto a playing field) it is advisable to adjust these parameters.

Finally make the executable by typing: (on the Pi)

```
cd /root/withOpenCV
make
```

When you are finished, remember to re-enable the auto-start in '/etc/profile'.

5 Final Comments

If you use our system and love it, or find a bug, or would like to see additional features feel free to email the authors.

This code is 100% free and open source so feel free to edit and redistribute it.

References

[1] G R Bradski. Real time face and object tracking as a component of a perceptual user interface. *Proceedings Fourth IEEE Workshop on Applications of Computer Vision WACV98 Cat No98EX201*, pages 214–219, 1998.