Jordan Lewis
Mr. Tim Clark
ARDrone Ball Tracking
Semester 2 Year 10 IST

Tracking with OpenCV on an ARDrone

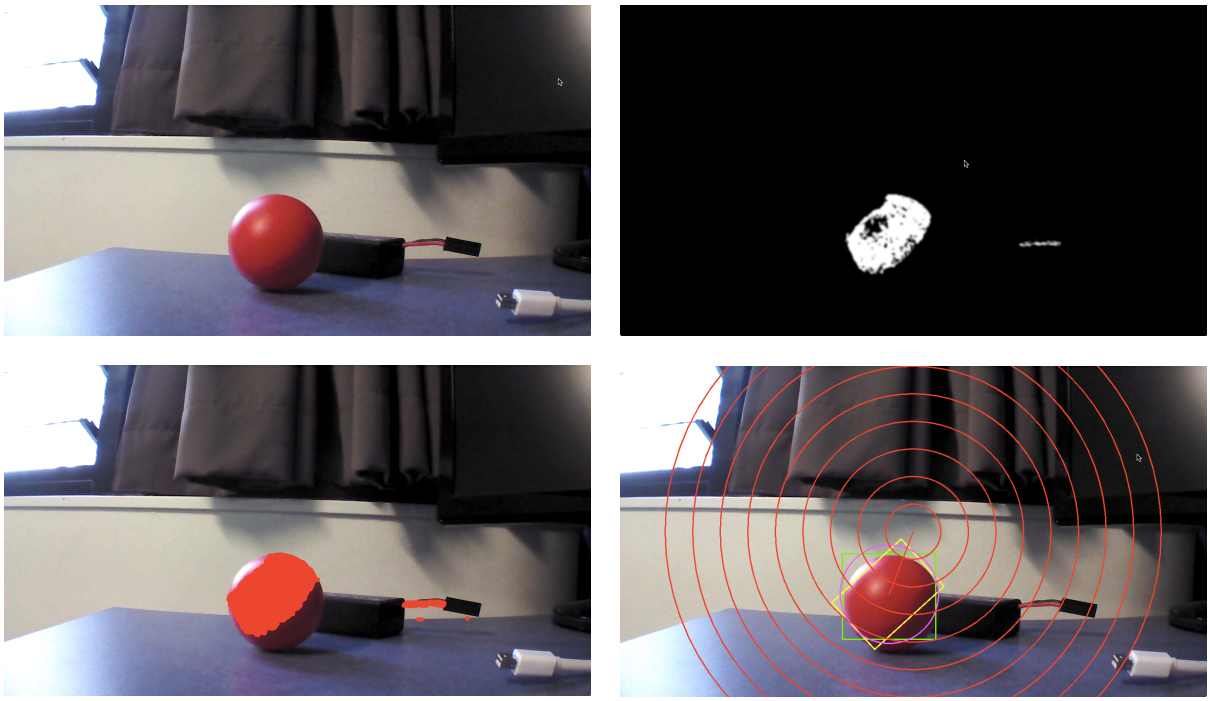
The Relatable Edition for Beginners

This is the documentation on tracking a coloured object using an ARDrone 2.0 with OpenCV 2.4.9 and Python 2.7.

The following is by no means the only way to accomplish the mission of tracking a ball with an ARDrone. It's just how I did it.

This document assumes you have some previous knowledge or very little knowledge of programming in Python.

The whole project can be found on Github here:
https://github.com/JJLewis/RedBallTracking-ARDrone2.0-Python



This is how computers see objects.

Dependencies

The following need to be installed on the controlling computer. Following must be completed in order.

- Python -> 2.7+ Not 3
- Homebrew
- OpenCV -> 2.4.9
- PIP (Optional, but handy)
- Python Imaging Library (PIL) -> 1.1.7+
- Mock -> 1.0.1+
- Numpy -> 1.7.1+ (May be installed with OpenCV)
- ffmpeg
- libardrone

Notes Before Continuing

Some of the terminal commands may need administrator privileges to run. To get past this, I would either suggest you get an authorised person to enter their password for you. I you do not have administrator access on the computer, I suggest copying the downloaded file(s) to a USB and log into an account with administrate using the 'login' command. Then open a terminal window and 'cd' into the USB, it should be something like:

```
cd /Volumes/<USB NAME>/<Directory to relevant files>
```

You can get admin on your computer by... Sorry, trade secret. Oh, and it might be a good idea to check if some of the Python modules are already installed on your computer.

Testing if some python modules are already installed

1. Open a terminal window
2. Type in the following:
    1. `python`
    2. `import <module name>`

If you don't get an error, and python moves to the next line, the bundle is installed. (Case Sensitive)

Install Python

1. Use the following link to download Python (19.4 MB)
    1. https://www.python.org/ftp/python/2.7.8/python-2.7.8-macosx10.6.dmg
2. Mount the disk image
3. Double click the Python.mpkg to launch the installer. If an alert pops up like the one to the right; right-click on the Python.mpkg and click open, this time, the alert should include an 'Open' option. Click that to open.
4. Step through the install process and enter an administrator's password when prompted.
5. Once the installer finishes you have successfully installed Python on your Mac!

Install Homebrew

1. Launch a terminal window
2. Run the following command:
    1. `ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"`
3. If the installation finishes without any errors, you have successfully installed Homebrew on your Mac!

How to do Jordan's Ghetto Install

1. Download from the provided link
2. Unzip the file
3. Open the unzipped folder and locate the specified file or folder
4. Copy that file/folder
5. Navigate to where your python packages are located and paste the folder in there. You may be prompted for an administrator's password, enter it.

Installing OpenCV on OSX

1. Launch a terminal window
2. Run the following commands in order
   1. `brew tap homebrew/science`
   2. `brew info opencv`
      (Optional: If you want to see other possible options such as when compiling (such as using tbb) you can use info)
   3. `brew install opencv`
3. Navigate to your python path, if you don't know where it is when you installed Python you can find it in your .bash_profile or using:
   1. `cat ~/.bash_profile | grep PYTHONPATH`
4. Then run the following commands in order
   1. `cd /Library/Python/2.7/site-packages/`
   2. `ln -s /usr/local/Cellar/opencv/2.4.9/lib/python2.7/site-packages/cv.py cv.py`
   3. `ln -s /usr/local/Cellar/opencv/2.4.9/lib/python2.7/site-packages/cv2.so cv2.so`

Installing Python Pip

1. Go to the following link: https://bootstrap.pypa.io/get-pip.py
2. Right click the page and save the file to a directory on your computer. The desktop should be fine.
3. Open a terminal window and use 'cd' to change directory to the one with the get-pip.py file.
4. Then type into the terminal:
   1. `python get-pip.py install`

Installing the Python Image Library

There are four ways to install the Python Image Library (that I know of).

A. Via pip
B. Via easy_install
C. Via setup.py
D. Jordan's Ghetto Install

I will go through installing the Python Image Library via methods A, C, and D, as we will not install easy_install.

PIP

1. Open a terminal window
2. Type in: pip install Pillow

setup.py

1. Download the Python Image Library from the following link: https://pypi.python.org/packages/source/P/Pillow/Pillow-2.6.0.zip#md5=611b51542236f0d876ba0c8a97c259e8
2. Unzip the download
3. Open a terminal window and use the 'cd' command to change directories to the folder containing the unzipped folder
4. Then use 'cd' again to move into the unzipped folder
5. Type in:
   ```
   1. python setup.py install
   ```

Jordan's Ghetto Install

1. Follow the Ghetto Install Instructions from above with the download link: https://pypi.python.org/packages/source/P/Pillow/Pillow-2.6.0.zip#md5=611b51542236f0d876ba0c8a97c259e8 and specified folder PIL

Installing Mock

There are two ways to install mock, either by using the setup.py file or by using Jordan's Ghetto Install Method.

For both you must start with the following steps.

1. Download mock by going to the following link: https://pypi.python.org/packages/source/m/mock/mock-1.0.1.zip#md5=869f08d003c289a97c1a6610faf5e913
2. Unzip the file

setup.py

1. Open a terminal window and use the 'cd' command to change directories to the folder containing the unzipped folder
2. Then use 'cd' again to move into the unzipped folder
3. Type in:
   ```
   1. python setup.py install
   ```

Jordan's Ghetto Install

1. Open the unzipped folder and locate a file named mock.py, follow the rest of the Ghetto Install Instructions

Installing Numpy

In my case numpy was already installed on my computer after completing the above steps. Before continuing check if it is installed on your computer.

To Install Numpy via pip

1. Open a terminal window
2. Type in:
    ```
    1. pip install numpy
    ```

Maybe ghetto install -> http://sourceforge.net/projects/numpy/files/ (Download does not work at Joeys)

Installing ffmpeg

Via Homebrew

1. Open a terminal window
2. Type in:
    ```
    1. brew install ffmpeg
    ```

Via Install.md Instructions

1. Download the compressed ffmpeg file from: http://ffmpeg.org/releases/ffmpeg-2.4.2.tar.bz2
2. Uncompress the file
3. Open a terminal window and use the 'cd' command to change directories to the folder containing the unzipped folder
4. Then use 'cd' again to move into the unzipped folder
5. Then type in the following in order:
    ```
    1. ./configure
    2. make
    3. make install
    ```

Install libardrone

There are two ways to install libardrone, either by using the setup.py file or by using Jordan's Ghetto Install Method.
For both you must start with the following steps.

1. Download mock by going to the following link: https://github.com/adetaylor/python-ardrone/archive/master.zip
2. Unzip the file

Via setup.py

1. Open a terminal window and use the 'cd' command to change directories to the folder containing the unzipped folder
2. Then use 'cd' again to move into the unzipped folder
3. Finally type into the terminal:
```
1. python setup.py install
```

Jordan's Ghetto Install Method

1. Locate a folder named libardrone in the unzipped folder and follow the rest of the Ghetto Install instructions.

Test if everything has been installed properly

1. Open a terminal window and type in:
```
1. python
```
2. Then the following:
```
1. import cv2
2. import numpy
3. import libardrone.libardrone
4. import PIL.Image
5. import mock
```
3. Once you have typed the above and have not received any errors, press: Control+Z. You should get an output like: [1]+ Stopped python
4. Then type in:
```
1. kill %<Number between the square brackets>
```
5. Now to check if ffmpeg is there, type in: ffmpeg. If you get an error like 'Command does not exist', then it has not been installed. Rather you should get information on ffmpeg such as the version.

The Fun Stuff

Introduction to my tracking program

Here you will be stepped through each major part of my ball tracking program. I will explain parts I believe may be hard to understand for new programmers.

Newington Imports

We installed all those dependancies for a reason... Well, we should probably use them then.
The program starts by importing the necessary modules for this program to run. It looks something like this:

```
import numpy as np
import cv2
import sys
import libardrone.libardrone as libardrone
import PIL.Image as Image
import time
```

You might be wondering, what the f*ck? I did't install 'sys' and/or 'time'! And why even bother using 'as'? Or if your really new, maybe even PIL.Image? libardrone.libardrone? How does that work? Well... To answer your questions. The 'sys' and 'time' come from the Python's belly and was installed with the Python. I used 'as' to be able to access the functions of the modules by using typing less, such as instead of 'libardrone.libardrone.takeoff()' it can be 'libardrone.takeoff()', just something small that can be handy. Remember some of the modules we installed were folders? Well, when you import PIL, you import the folder and its contents, in this case, I only want a single file in the folder, so I import that single file by going a step further and typing in 'import PIL.Image', here I imported just the Image.py file.

Everybody Has Limits

After importing our dependencies, we must set a threshold for the colour we will be targeting. This is done like so:

```
TARGET_COLOR_MIN = np.array([0,100,100], np.uint8)
TARGET_COLOR_MAX = np.array([5,255,255], np.uint8)
```

The three values in each line are HSV Values for the colour range we will be tracking.

The Main Course

```
def main():
```

Here we declare are main function (the only function) in this program. From here on in, all lines of code that are in this function will be indented once.

Setting Up for the Fun

Fun stuff, does not just happen. A fair bit of setting up occurs behind the scenes.

```
W, H = 640, 360
screenMidX = W/2
screenMidY = H/2
speedCirclesDiff = H/12

drone = libardrone.ARDrone(True, False)
time.sleep(0.5)
drone.reset()
time.sleep(0.5)
#drone.set_camera_view(True) # False for bottom camera
#sleep(0.5)
drone.set_speed(0.3)
time.sleep(0.5)
drone.takeoff()
print "take off"
time.sleep(0.5)
drone.hover()
time.sleep(0.5)

running = True
```

Here, we tell the program the resolution of the video being fed to us from the drone and we also set the drone up. Most of the code is self explanatory, we will apply settings to the drone and give it half a second to process the previous command.

Looper

We will now start an 'infinite loop', one that should run forever (obviously!).

```
while running:
```

Anything in this loop will be indented once more. Remember that variable we made called 'running' and we set to 'True'? Well, as long as 'running = True', the loop will keep looping.

Videos are made of Frames

Now we will try to fetch a frame from the drone and convert it to an OpenCV usable image.

```
try:
      # This should be an numpy image array
      pixelarray = drone.get_image()
      if pixelarray != None:
            # Convert RGB to BGR & make OpenCV Image
            frame = pixelarray[:, :, ::-1].copy()
```

Here we use the get_image function in the libardrone module to fetch the current frame from the drone as a numpy array. Once we make sure that the numpy array is not nothing, we will then convert it to an OpenCV usable variable and change it from RGB to BGR.

The Breaking Point of Frames

Now, each frame must be analysed to find the pixels that are within the target color range.

```
_frame = np.asarray(frame)
frameHSV = cv2.cvtColor(_frame, cv2.COLOR_BGR2HSV)
frameThreshold = cv2.inRange(frameHSV, TARGET_COLOR_MIN,
TARGET_COLOR_MAX)

element = cv2.getStructuringElement(cv2.MORPH_RECT,(1,1))
frameThreshold = cv2.erode(frameThreshold,element, iterations=1)
frameThreshold = cv2.dilate(frameThreshold,element, iterations=1)
frameThreshold = cv2.erode(frameThreshold,element)

# Blurs to smoothen frame
frameThreshold = cv2.GaussianBlur(frameThreshold,(9,9),2,2)
frameThreshold = cv2.medianBlur(frameThreshold,5)
```

Here we take the converted OpenCV image and convert it back into a numpy array. We will then convert each frame's colour format into HSV (Hue Saturation Value) for analysing. The HSV frame will then be put into the inRange function to isolate the sections in the frame that contain our target colour. The result will be a variable named frameThreshold which is a purely black and white frame, where sections within the colour threshold are white and the rest is black.
After getting this black and white frame, we will apply some effects to attempt to remove as much of the noise as we can, a standard method of doing this is to erode, dilate, then erode again.
Now, circles are smooth round objects, but our black and white frame will likely look quite blocky. To try to smoothen it out as much as we can, we will apply two blur effects, the gaussian and median blurs, one after the other.

Ebony and Ivory

As Michael Jackson tried to put through to us, black and white should work together. In this case, maybe make grey? But here they simply can't. We will have to find their edges and contours.

```
# Find Contours
contours, hierarchy =
cv2.findContours(frameThreshold,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE
)
```

Here we just use an OpenCV function to find the contours and store them into two variables.

Size DOES Matter

Your parents may have tried to tell you that size does not matter. They couldn't be more wrong. Size DOES matter very much, we would want a whole heap of small things all over would we? Let's do something about that.

```
showingCNTs = [] # Contours that are visible
areas = [] # The areas of the contours

# Find Specific contours
for cnt in contours:
      approx = cv2.approxPolyDP(cnt,0.1*cv2.arcLength(cnt,True),True)
      area = cv2.contourArea(cnt)
      if area > 300:
            areas.append(area)
            showingCNTs.append(cnt)

# Only Highlight the largest object
if len(areas)>0:
      m = max(areas)
      maxIndex = 0
      for i in range(0, len(areas)):
            if areas[i] == m:
                  maxIndex = i
            cnt = showingCNTs[maxIndex]
```

Here we will go through all of the contours and pick out the ones which have a pixel area of greater than 300, that's our minimum. We'll pop the ones that have an area larger than 300 and pop them into arrays; one for its area and one for the object itself, the by-catch can go.
Now, once we check that we have caught at least 1 object that has an area greater than 300, we will search for the largest in the bunch. Once we have found the largest object/blob of colour, we will store that contour into a variable for later use.

Trap dat Beast! & Give it a Leash

How would we know if our computer is seeing the target if we don't have a visual? Now let's make sure we can see what the computer can see. Note that the code below should be indented 5 times.

```
# Highlight the Object Red
#cv2.drawContours(frame,[cnt],0,(0,0,255),-1)

# Draw a bounding rectangle
x,y,w,h = cv2.boundingRect(cnt)
cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2)

# Draw a rotated bounding rectangle
rect = cv2.minAreaRect(cnt)
box = cv2.cv.BoxPoints(rect)
box = np.int0(box)
cv2.drawContours(frame,[box],0,(0,255,255),2)

# Draw a circumcircle
(x,y),radius = cv2.minEnclosingCircle(cnt)
center = (int(x),int(y))
radius = int(radius)
cv2.circle(frame,center,radius,(255,0,255),2)
```

Here we receive the largest contour from above and we will draw a bounding rectangle (always flat), minimum area bounding rectangle, and a circumcircle around the object. Although not all of it is necessary, why not? It looks cooler. You know what? Lets even add a line from the centre of the screen to the centre of the object. Again, note that the code below should be indented 5 times.

```
# Draw line to centre of screen
cv2.line(frame, (screenMidX, screenMidY), center, (0,0,255),2)
```

Because I Can -> Inception

Because I can, and for the sake of being able to see everything, I will now show you how to draw what some may find rather annoying to see, while others may find it handy to see; the inception of circles from the centre of the screen. Note that the code below should be indented 5 times before the subsequent indents.

```
# Draw Speed Limit Circles and Determine Speed
diff = speedCirclesDiff # Increment of radius of the Speed Limit Circles

# Loop for drawing speed circles
for i in range(1, 10):
      cv2.circle(frame, (screenMidX, screenMidY), diff*i,(0,0,255),2)
```

This will simply loop 10 times to draw each circle, incrementing its radius by the 'diff'. This was meant to be used for variable speeds while as the drone centres the ball on the screen, but the libardrone module only allows for one speed setting to control the whole drone, so it's kind of irrelevant now. BUT!, if you were interested in knowing how I would have done such a work of God, read the next section. If your not interested, read it anyways because I took the time to type it, and some of the code is relevant to the project.

I Pity the Fool!

Ok, so you decided to read this section, smart choice. Note that the code below should be indented 5 times before the subsequent indents.

```
# Centre of the object
x = center[0]
y = center[1]
'''
hor_speed = 0.0
vert_speed = 0.0

# Loop for checking speed
for i in range(1, 10):
      _diff = diff*i
      if (640-_diff)<x<(640+_diff): # On the x axis
            hor_speed = (i-1)/10
            break

for i in range(1, 10):
      if (360-_diff)<y<(360+_diff): # On the y axis
            vert_speed = (i-1)/10
            break
'''
```

From the top, we get the centre of the object we are tracking... thats all that we actually need in this section. From the triple single quotes on wards, we are inside a multi-line comment until the next triple single quotes. In here I wrote how I would have done speed management through having the speed boundaries we made above. I actually felt pretty proud of it. Basically it will loop up to 10 times for the horizontal and vertical speeds and work its way from the inner circle to the outer most circle. Once it finds that the value on either axis is within one of the circles, it will set that speed by checking at which point in the loop that it fit the conditions, then it will exit the loop. BAAAT, the real stupid part I only noticed after I finished this section was that in circles, you could be really close to the centre on the x-axis yet be far away on the y-axis but still move horizontally at the speed meant for vertical movement. 'I pity the fool!'. So, while feeling pretty terrible, I commented it out, in hopes that one day, it may be used to work (although it probably won't). ;(

Left, Left, Left, Right, Left! My Pack is Up! My Balls are Down!

Left, Left, Left, Right, Left! My Pack is Up! My Balls are Down, My d*ck is swinging left to right! (Repeat a couple times and read on).
So, now the most important part. Actually moving the drone so that it centres the ball on the screen. Note that the code below should be indented 5 times before the subsequent indents.

```
# Check direction of the ball
centerThresh = 50 # The 'centre' threshold for the ball
# On the X axis -> Note: Inverse
if (screenMidX-x) < -centerThresh: # To the left to the left
      drone.turn_right()
      #print "Left"
if (screenMidX-x) > centerThresh: # To the right to the right
      drone.turn_left()
      #print "right"
if -centerThresh < (screenMidX-x) < centerThresh:
      #drone.hover()
      #print "Stop Rotating"
      pass

# On the Y axis
if (screenMidY-y) < -centerThresh: # Down!
      drone.move_down()
if (screenMidY-y) > centerThresh: # Up!
      drone.move_up()
if -centerThresh < (screenMidY-y) < centerThresh:
      #drone.hover()
      #print "Stop Moving up or down"
      pass
```

Here we simply check if the distance between the x value of the centre of the object and of the screen are greater than the negative centre threshold or less than the centre threshold; we do the same for the y values except we replace the x values with y values. Then we move the drone accordingly.
You might be reading the code and wondering why the comments and what the drone is told to do is the opposite on the x-axis. This is because what the camera sees in inverted, but not for the y-axis.

Preparing for the Black Out

Black outs are bad, things can get damaged. Don't ask me how. Anyways, we should always be prepared for one. Maybe set some candles around the place and have a lighter and a flash light ready. Your phone's camera flash can be a good flashlight, just remember that it will eat up your phone's battery. So, it might be a good idea to prepare for a black out. Let's do that. Note that the code below should be indented 4 times before the subsequent indents.

```
# Get battery status of the drone
bat = drone.navdata.get(0, dict()).get('battery', 0)
if bat < 20:
      running = False
      print "Low Battery: "+str(bat)
```

Here we just ask the drone for its battery percentage and check if it is less than 20%. If it is, we'll murder the looper and close the loop. Why not just print out the batter percentage too.

Rubber Necking

Rubber Necking is terrible, if you do that, just know that you are hated by every driver behind you for slowing down and most likely being the cause of a traffic jam, sure to irritate the f*ck out of every driver caught in it. Plus the guys that were in the accident you were looking at will think your a total *ssh*le. The j*ck*ss*s wanted to see what was happening. Why don't we become one of them and have a look at what's happening.

```
# Display the Image
cv2.imshow('Drone', frame)
```

Okay, so there's one comment and one line of code. WTF, all that reading for that? It probably seems very self explanatory, but for the special ones, I'll explain. Here we make use of OpenCV's built in function to show an image in a window with a window title. The functions name is imshow and it takes to arguments, the first being the title of the window and the second being the image. Remember that videos were made of frames? Well did you know that a frame is a still image, so I'm not breaking any rules.

Except... No Excuses!

Sometimes things don't always work out and fail. People tend to like to make excuses to cover up their mistakes. Python likes to do that too, but we're not going to let it make an excuse.

```
except:
      print "Failed"
```

Here if some thing in the tracking process fails, it will jump to this section, where it will print "Failed" into the console. No excuses. Just Failed.

Practice Safe Tracking

Putting a thin sheet rubber on the propellers of the drone or on the whole drone itself is not going to save it from a potential crash. The next best thing is to pull out. Let's add sum code to flytus interuptus.

```
# Listen for Q Key Press
if cv2.waitKey(1) & 0xFF == ord('q'):
      running = False
```

Here, every time the loop loops, we are checking if the Q key on the keyboard has been pressed, if it has, it will murder the looper and close the loop. Just know that this method is not a hundred percent effective in preventing lasting damage.

All Good Things Must Come to an End (Cleaning Up)

Do you like getting screamed at by your mother or wife after having some fun to clean up you mess? I personally hate it, you have to listen to screaming, plus you have to clean up feeling like sh*t. Why don't we have a system of automatically cleaning up our mess.

```
# Shutdown
print "Shutting Down..."
drone.land()
drone.halt()
print "Ok."
```

Here we will print "Shutting Down..." before doing any cleaning up, just so we know it has started to clean up, it like screaming "I'm cleaning up know!". This usually shuts up the screaming woman. Then we land the drone and disconnect from it. Once that finishes without an error we will print "Ok." and the program will end, which is like leaving your room in a pristine condition and asking the woman for a cookie for doing so well.

The Black One Runs Faster

You might be really happy with your self for reading though my documentation and having finally put together your program. Well, try running it. I'll give you 5 minutes to work it out.

1.....2.....3.....4.....5

Ok, so your program doesn't do anything or is giving you indentation errors. I'm not fixing your indentation errors because it is a b*tch of a task to do, and I personally think that whoever made up the Python syntax to use indentation instead of brackets was/is high and retarded. I will tell you how to make your program run though. Type the following into the bottom of your program.

```
if __name__ == '__main__':
      main()
```

This code basically says that is the program is running from the console, call the function main, which will in turn run MY ball tracking program inside the main function.

If Only, If Only

A Feature

If there was one feature I think I should have used was the Hough Circles function built into OpenCV to track circles instead of large blobs of colour. The only issue with doing that would have been that firstly, the ball may not appear to be a circle in the eye/camera of the ARDrone as it has a wide angle lens, which can stretch the ball to appear like an oval and secondly, that if your hand covers part of the ball, it will no longer be a circle. So that idea was aborted faster than it was conceived.

A Function

After having completed my program, the world noticed that the motion of the drone was jaggy. This was because as soon as the direction to turn changed it would suddenly change at a set speed instead of ease into the ball. To fix this Mr. Clark recommended using a method known as PID (Proportional Integral Differentiation) to have smooth movements when attempting to centre the object on camera. This method will likely take up a lot of horse power and some devices have a built in controller for use for PID so I will leave it out for this program.

The Source Code

The font size of the source code has been reduced to fit the page. Copy and Paste it into another document for use.

```
import numpy as np
import cv2
import sys
import libardrone.libardrone as libardrone
import PIL.Image as Image
import time

TARGET_COLOR_MIN = np.array([0,100,100], np.uint8)
TARGET_COLOR_MAX = np.array([5,255,255], np.uint8)

def main():
                        W, H = 640, 360
                        screenMidX = W/2
                        screenMidY = H/2
                        speedCirclesDiff = H/12

                        drone = libardrone.ARDrone(True, False)
                        time.sleep(0.5)
                        drone.reset()
                        time.sleep(0.5)
                        #drone.set_camera_view(True) # False for bottom camera
                        #sleep(0.5)
                        drone.set_speed(0.3)
                        time.sleep(0.5)
                        drone.takeoff()
                        print "take off"
                        time.sleep(0.5)
                        drone.hover()
                        time.sleep(0.5)

                        running = True
                        while running:
                                                try:
                                                                        # This should be an numpy image array
                                                                        pixelarray = drone.get_image()
                                                                        if pixelarray != None:
                                                                                                # Convert RGB to BGR & make OpenCV Image
                                                                                                frame = pixelarray[:, :, ::-1].copy()

                                                                                                _frame = np.asarray(frame)
                                                                                                frameHSV = cv2.cvtColor(_frame, cv2.COLOR_BGR2HSV)
                                                                                                frameThreshold = cv2.inRange(frameHSV, TARGET_COLOR_MIN, TARGET_COLOR_MAX)

                                                                                                element = cv2.getStructuringElement(cv2.MORPH_RECT,(1,1))
                                                                                                frameThreshold = cv2.erode(frameThreshold,element, iterations=1)
                                                                                                frameThreshold = cv2.dilate(frameThreshold,element, iterations=1)
                                                                                                frameThreshold = cv2.erode(frameThreshold,element)

                                                                                                # Blurs to smoothen frame
                                                                                                frameThreshold = cv2.GaussianBlur(frameThreshold,(9,9),2,2)
                                                                                                frameThreshold = cv2.medianBlur(frameThreshold,5)

                                                                                                # Find Contours
                                                                                                contours, hierarchy = cv2.findContours(frameThreshold,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)

                                                                                                showingCNTs = [] # Contours that are visible
                                                                                                areas = [] # The areas of the contours

                                                                                                # Find Specific contours
                                                                                                for cnt in contours:
                                                                                                                        approx = cv2.approxPolyDP(cnt,0.1*cv2.arcLength(cnt,True),True)
                                                                                                                        #if len(approx)==4:
                                                                                                                        area = cv2.contourArea(cnt)
                                                                                                                        if area > 300:
                                                                                                                                                areas.append(area)
                                                                                                                                                showingCNTs.append(cnt)

                                                                                                # Only Highlight the largest object
                                                                                                if len(areas)>0:
                                                                                                                        m = max(areas)
                                                                                                                        maxIndex = 0
                                                                                                                        for i in range(0, len(areas)):
                                                                                                                                                if areas[i] == m:
                                                                                                                                                                        maxIndex = i
                                                                                                                        cnt = showingCNTs[maxIndex]

                                                                                                                        # Highlight the Object Red
                                                                                                                        #cv2.drawContours(frame,[cnt],0,(0,0,255),-1)

                                                                                                                        # Draw a bounding rectangle
                                                                                                                        x,y,w,h = cv2.boundingRect(cnt)
                                                                                                                        cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2)

                                                                                                                        # Draw a rotated bounding rectangle
                                                                                                                        rect = cv2.minAreaRect(cnt)
                                                                                                                        box = cv2.cv.BoxPoints(rect)
                                                                                                                        box = np.int0(box)
                                                                                                                        cv2.drawContours(frame,[box],0,(0,255,255),2)

                                                                                                                        # Draw a circumcircle
                                                                                                                        (x,y),radius = cv2.minEnclosingCircle(cnt)
                                                                                                                        center = (int(x),int(y))
                                                                                                                        radius = int(radius)
                                                                                                                        cv2.circle(frame,center,radius,(255,0,255),2)

                                                                                                                        # Draw line to center of screen
                                                                                                                        cv2.line(frame, (screenMidX, screenMidY), center, (0,0,255),2)

                                                                                                                        # Draw Speed Limit Circles and Determine Speed
                                                                                                                        diff = speedCirclesDiff # Increment of radius of the Speed Limit Circles

                                                                                                                        # Loop for drawing speed circles
                                                                                                                        for i in range(1, 10):
                                                                                                                                                cv2.circle(frame, (screenMidX, screenMidY), diff*i,(0,0,255),2)

                                                                                                                        # Center of the object
                                                                                                                        x = center[0]
                                                                                                                        y = center[1]
                                                                                                                        '''
                                                                                                                        hor_speed = 0.0
                                                                                                                        vert_speed = 0.0

                                                                                                                        # Loop for checking speed
                                                                                                                        for i in range(1, 10):
                                                                                                                                                _diff = diff*i
                                                                                                                                                if (640-_diff)<x<(640+_diff): # On the x axis
                                                                                                                                                                        hor_speed = (i-1)/10
                                                                                                                                                                        break

                                                                                                                        for i in range(1, 10):
                                                                                                                                                if (360-_diff)<y<(360+_diff): # On the y axis
                                                                                                                                                                        vert_speed = (i-1)/10
                                                                                                                                                                        break

                                                                                                                        '''
                                                                                                                        # Check direction of the ball
                                                                                                                        centerThresh = 50 # The 'centre' threshold for the ball
                                                                                                                        # On the X axis -> Note: Inverse
                                                                                                                        if (screenMidX-x) < -centerThresh: # To the left to the left
                                                                                                                                                drone.turn_right()
                                                                                                                                                #print "Left"
                                                                                                                        if (screenMidX-x) > centerThresh: # To the right to the right
                                                                                                                                                drone.turn_left()
                                                                                                                                                #print "right"
                                                                                                                        if -centerThresh < (screenMidX-x) < centerThresh:
                                                                                                                                                #drone.hover()
                                                                                                                                                pass
                                                                                                                                                #print "Stop Rotating"

                                                                                                                        # On the Y axis
                                                                                                                        if (screenMidY-y) < -centerThresh: # Down!
                                                                                                                                                drone.move_down()
                                                                                                                        if (screenMidY-y) > centerThresh: # Up!
                                                                                                                                                drone.move_up()
                                                                                                                        if -centerThresh < (screenMidY-y) < centerThresh:
                                                                                                                                                #drone.hover()
                                                                                                                                                pass
                                                                                                                                                #print "Stop Moving up or down"

                                                                                                # Get battery status of the drone
                                                                                                bat = drone.navdata.get(0, dict()).get('battery', 0)
                                                                                                #print str(bat)
                                                                                                if bat < 20:
                                                                                                                        running = False
                                                                                                                        print "Low Battery: "+str(bat)

                                                                                                # Display the image
                                                                                                cv2.imshow('Drone', frame)
                                                except:
                                                                        print "Failed"

                                                if cv2.waitKey(1) & 0xFF == ord('w'):
                                                                        print "Take Off!!"
                                                                        drone.reset()
                                                                        drone.takeoff()
                                                                        drone.hover()

                                                # Listen for Q Key Press
                                                if cv2.waitKey(1) & 0xFF == ord('q'):
                                                                        running = False

                        # Shutdown
                        print "Shutting Down..."
                        drone.land()
                        drone.halt()
                        print "Ok."

if __name__ == '__main__':
                        main()
```

Personal Thoughts

After having completed the task within two weeks, I am fairly happy with the program and its performance. To me, the hardest part was finding out and installing the required dependencies. This was likely because I tried to do this on my school computer, which did not have the same permissions on some directories as the rest of the world. The logic in my program was fairly straight forward. The only issues I ran into being the drone turned the wrong way the first time I tested it, this was quickly fixed after recognising that what the drone sees was inverted. Another issue is that the drone will not always take off when the program starts, I still unsure at the time of this typing as to why it won't take off, but it will if you stop the program and try again (a couple times sometimes). Other than those two issues, I am quite proud of what was accomplished in a fairly short time, while other students struggled to get a robot to move fifty centimetres.

Copyright and Licensing

Creative Commons - Attribution - ShareAlike 3.0 License

1. Definitions

"Adaptation" means a work based upon the Work, or upon the Work and other pre-existing works, such as a translation, adaptation, derivative work, arrangement of music or other alterations of a literary or artistic work, or phonogram or performance and includes cinematographic adaptations or any other form in which the Work may be recast, transformed, or adapted including in any form recognizably derived from the original, except that a work that constitutes a Collection will not be considered an Adaptation for the purpose of this License. For the avoidance of doubt, where the Work is a musical work, performance or phonogram, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered an Adaptation for the purpose of this License.

"Collection" means a collection of literary or artistic works, such as encyclopedias and anthologies, or performances, phonograms or broadcasts, or other works or subject matter other than works listed in Section 1(f) below, which, by reason of the selection and arrangement of their contents, constitute intellectual creations, in which the Work is included in its entirety in unmodified form along with one or more other contributions, each constituting separate and independent works in themselves, which together are assembled into a collective whole. A work that constitutes a Collection will not be considered an Adaptation (as defined below) for the purposes of this License.

"Creative Commons Compatible License" means a license that is listed at http://creativecommons.org/compatiblelicenses that has been approved by Creative Commons as being essentially equivalent to this License, including, at a minimum, because that license: (i) contains terms that have the same purpose, meaning and effect as the License Elements of this License; and, (ii) explicitly permits the relicensing of adaptations of works made available under that license under this License or a Creative Commons jurisdiction license with the same License Elements as this License.

"Distribute" means to make available to the public the original and copies of the Work or Adaptation, as appropriate, through sale or other transfer of ownership.

"License Elements" means the following high-level license attributes as selected by Licensor and indicated in the title of this License: Attribution, ShareAlike.

"Licensor" means the individual, individuals, entity or entities that offer(s) the Work under the terms of this License.

"Original Author" means, in the case of a literary or artistic work, the individual, individuals, entity or entities who created the Work or if no individual or entity can be identified, the publisher; and in addition (i) in the case of a performance the actors, singers, musicians, dancers, and other persons who act, sing, deliver, declaim, play in, interpret or otherwise perform literary or artistic works or expressions of folklore; (ii) in the case of a phonogram the producer being the person or legal entity who first fixes the sounds of a performance or other sounds; and, (iii) in the case of broadcasts, the organization that transmits the broadcast.

"Work" means the literary and/or artistic work offered under the terms of this License including without limitation any production in the literary, scientific and artistic domain, whatever may be the mode or form of its expression including digital form, such as a book, pamphlet and other writing; a lecture, address, sermon or other work of the same nature; a dramatic or dramatico-musical work; a choreographic work or entertainment in dumb show; a musical composition with or without words; a cinematographic work to which are assimilated works expressed by a process analogous to cinematography; a work of drawing, painting, architecture, sculpture, engraving or lithography; a photographic work to which are assimilated works expressed by a process analogous to photography; a work of applied art; an illustration, map, plan, sketch or three-dimensional work relative to geography, topography, architecture or science; a performance; a broadcast; a phonogram; a compilation of data to the extent it is protected as a copyrightable work; or a work performed by a variety or circus performer to the extent it is not otherwise considered a literary or artistic work.

"You" means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

"Publicly Perform" means to perform public recitations of the Work and to communicate to the public those public recitations, by any means or process, including by wire or wireless means or public digital performances; to make available to the public Works in such a way that members of the public may access these Works from a place and at a place individually chosen by them; to perform the Work to the public by any means or process and the communication to the public of the performances of the Work, including by public digital performance; to broadcast and rebroadcast the Work by any means including signs, sounds or images.

"Reproduce" means to make copies of the Work by any means including without limitation by sound or visual recordings and the right of fixation and reproducing fixations of the Work, including storage of a protected performance or phonogram in digital form or other electronic medium.

2. Fair Dealing Rights.

Nothing in this License is intended to reduce, limit, or restrict any uses free from copyright or rights arising from limitations or exceptions that are provided for in connection with the copyright protection under copyright law or other applicable laws.

3. License Grant.

Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

to Reproduce the Work, to incorporate the Work into one or more Collections, and to Reproduce the Work as incorporated in the Collections;
to create and Reproduce Adaptations provided that any such Adaptation, including any translation in any medium, takes reasonable steps to clearly label, demarcate or otherwise identify that changes were made to the original Work. For example, a translation could be marked "The original work was translated from English to Spanish," or a modification could indicate "The original work has been modified.";
to Distribute and Publicly Perform the Work including as incorporated in Collections; and,
to Distribute and Publicly Perform Adaptations.
For the avoidance of doubt:

Non-waivable Compulsory License Schemes. In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme cannot be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License;
Waivable Compulsory License Schemes. In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme can be waived, the Licensor waives the exclusive right to collect such royalties for any exercise by You of the rights granted under this License; and,
Voluntary License Schemes. The Licensor waives the right to collect royalties, whether individually or, in the event that the Licensor is a member of a collecting society that administers voluntary licensing schemes, via that society, from any exercise by You of the rights granted under this License.
The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats. Subject to Section 8(f), all rights not expressly granted by Licensor are hereby reserved.

4. Restrictions.

The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

You may Distribute or Publicly Perform the Work only under the terms of this License. You must include a copy of, or the Uniform Resource Identifier (URI) for, this License with every copy of the Work You Distribute or Publicly Perform. You may not offer or impose any terms on the Work that restrict the terms of this License or the ability of the recipient of the Work to exercise the rights granted to that recipient under the terms of the License. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties with every copy of the Work You Distribute or Publicly Perform. When You Distribute or Publicly Perform the Work, You may not impose any effective technological measures on the Work that restrict the ability of a recipient of the Work from You to exercise the rights granted to that recipient under the terms of the License. This Section 4(a) applies to the Work as incorporated in a Collection, but this does not require the Collection apart from the Work itself to be made subject to the terms of this License. If You create a Collection, upon notice from any Licensor You must, to the extent practicable, remove from the Collection any credit as required by Section 4(c), as requested. If You create an Adaptation, upon notice from any Licensor You must, to the extent practicable, remove from the Adaptation any credit as required by Section 4(c), as requested.
You may Distribute or Publicly Perform an Adaptation only under the terms of: (i) this License; (ii) a later version of this License with the same License Elements as this License; (iii) a Creative Commons jurisdiction license (either this or a later license version) that contains the same License Elements as this License (e.g., Attribution-ShareAlike 3.0 US)); (iv) a Creative Commons Compatible License. If you license the Adaptation under one of the licenses mentioned in (iv), you must comply with the terms of that license. If you license the Adaptation under the terms of any of the licenses mentioned in (i), (ii) or (iii) (the "Applicable License"), you must comply with the terms of the Applicable License generally and the following provisions: (I) You must include a copy of, or the URI for, the Applicable License with every copy of each Adaptation You Distribute or Publicly Perform; (II) You may not offer or impose any terms on the Adaptation that restrict the terms of the Applicable License or the ability of the recipient of the Adaptation to exercise the rights granted to that recipient under the terms of the Applicable License; (III) You must keep intact all notices that refer to the Applicable License and to the disclaimer of warranties with every copy of the Work as included in the Adaptation You Distribute or Publicly Perform; (IV) when You Distribute or Publicly Perform the Adaptation, You may not impose any effective technological measures on the Adaptation that restrict the ability of a recipient of the Adaptation from You to exercise the rights granted to that recipient under the terms of the Applicable License. This Section 4(b) applies to the Adaptation as incorporated in a Collection, but this does not require the Collection apart from the Adaptation itself to be made subject to the terms of the Applicable License.
If You Distribute, or Publicly Perform the Work or any Adaptations or Collections, You must, unless a request has been made pursuant to Section 4(a), keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or if the Original Author and/or Licensor designate another party or parties (e.g., a sponsor institute, publishing entity, journal) for attribution ("Attribution Parties") in Licensor's copyright notice, terms of service or by other reasonable means, the name of such party or parties; (ii) the title of the Work if supplied; (iii) to the extent reasonably practicable, the URI, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and (iv) , consistent with Ssection 3(b), in the case of an Adaptation, a credit identifying the use of the Work in the Adaptation (e.g., "French translation of the Work by Original Author," or "Screenplay based on original Work by Original Author"). The credit required by this Section 4(c) may be implemented in any reasonable manner; provided, however, that in the case of a Adaptation or Collection, at a minimum such credit will appear, if a credit for all contributing authors of the Adaptation or Collection appears, then as part of these credits and in a manner at least as prominent as the credits for the other contributing authors. For the avoidance of doubt, You may only use the credit required by this Section for the purpose of attribution in the manner set out above and, by exercising Your rights under this License, You may not implicitly or explicitly assert or imply any connection with, sponsorship or endorsement by the Original Author, Licensor and/or Attribution Parties, as appropriate, of You or Your use of the Work, without the separate, express prior written permission of the Original Author, Licensor and/or Attribution Parties.
Except as otherwise agreed in writing by the Licensor or as may be otherwise permitted by applicable law, if You Reproduce, Distribute or Publicly Perform the Work either by itself or as part of any Adaptations or Collections, You must not distort, mutilate, modify or take other derogatory action in relation to the Work which would be prejudicial to the Original Author's honor or reputation. Licensor agrees that in those jurisdictions (e.g. Japan), in which any exercise of the right granted in Section 3(b) of this License (the right to make Adaptations) would be deemed to be a distortion, mutilation, modification or other derogatory action prejudicial to the Original Author's honor and reputation, the Licensor will waive or not assert, as appropriate, this Section, to the fullest extent permitted by the applicable national law, to enable You to reasonably exercise Your right under Section 3(b) of this License (right to make Adaptations) but not otherwise.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability.

EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Adaptations or Collections from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.

Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

Each time You Distribute or Publicly Perform the Work or a Collection, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.

Each time You Distribute or Publicly Perform an Adaptation, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.

If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.

This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

The rights granted under, and the subject matter referenced, in this License were drafted utilizing the terminology of the Berne Convention for the Protection of Literary and Artistic Works (as amended on September 28, 1979), the Rome Convention of 1961, the WIPO Copyright Treaty of 1996, the WIPO Performances and Phonograms Treaty of 1996 and the Universal Copyright Convention (as revised on July 24, 1971). These rights and subject matter take effect in the relevant jurisdiction in which the License terms are sought to be enforced according to the corresponding provisions of the implementation of those treaty provisions in the applicable national law. If the standard suite of rights granted under applicable copyright law includes additional rights not granted under this License, such additional rights are deemed to be included in the License; this License is not intended to restrict the license of any rights under applicable law.