



Tecnológico de Monterrey

FINAL PROJECT

A DRONE THAT FOLLOWS A PERSON AND LANDS
WHEN HE/SHE CLOSES THE EYES

ROBOTIC VISION

Juan Manuel Guerrero Hernández A01207672

Edward Alan Gómez Vargas A01204727

León Andrés Rodríguez Guzmán A01700687

Gerardo Alcántara Landeros A01206304

30/11/2018

Description

The main objective of the final project was to apply Robotic Vision in a field outside our usual workspace. We decided to use a Drone, provided by our professor, to acquire images while it was flying and to process them and make the Drone act a certain way. We used Python to create our code, and applied preloaded libraries from OpenCV and from Bebop (the manufacturer of the Drone) in order to generate a complete project in which we could apply what we learned during clases, but also contribute with our creativity. Our project was focused on the creation of a reactive agent with three main behaviours. The first one being, by making use of image processing, if the drone camera found a person, it would follow him/her and move until the face of the person was aligned with the center of the camera. The second behaviour was presented when the drone was already in front of the person at a certain distance (defined also with the help of image processing), if he/she would close his/her eyes, the drone would land and disconnect. Finally, if the drone did not recognize any person, it would starts to rotate on its axis until it found a face. An example of this can be seen in the figure 1.1.



Figure 1.1: A drone is following a team member during a demo.

The following instructions contemplate the installation of OpenCV (a tool which allowed us to use face recognition), and the drone running libraries, specifically for the Bebop Parrot. Therefore, the steps have been separated in the OpenCV installation with a demo, and the running of the Bebop Drone code.

Environment Setup

OpenCV Installation

To install OpenCV, Anaconda should be installed, since it contains all the libraries needed. It can be downloaded from the official page. The Python 3.7 version should be selected, in figure 1.2 it's the option to the left.

- Anaconda for Windows: <https://www.anaconda.com/download/#windows>
- Anaconda for Mac: <https://www.anaconda.com/download/#macos>

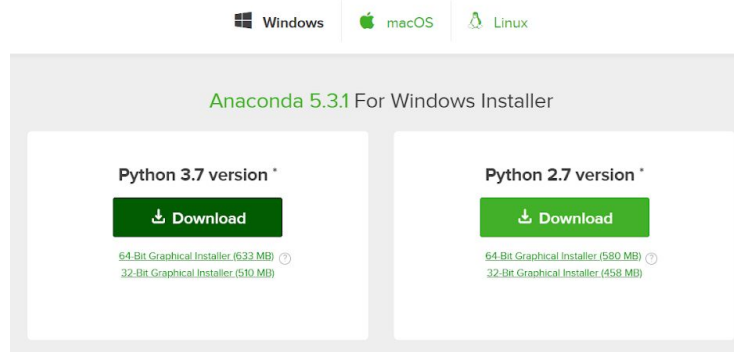


Figure 1.2: Options presented in Anaconda official page.

After you have installed Anaconda with **all the options by default**, open the Anaconda-Navigator app, which should be in the Applications folder or in the LaunchPad, the logo is shown in the figure 1.3.



Figure 1.3: Anaconda-Navigator app.

When you open it, you should see all the options available to work with Anaconda, as in the figure 1.4 is shown.

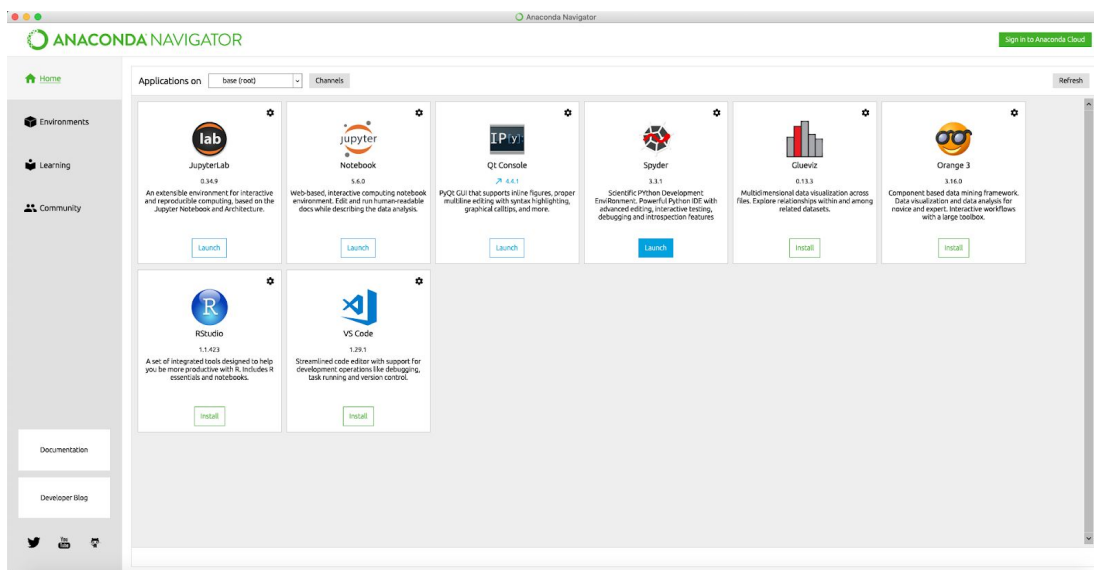
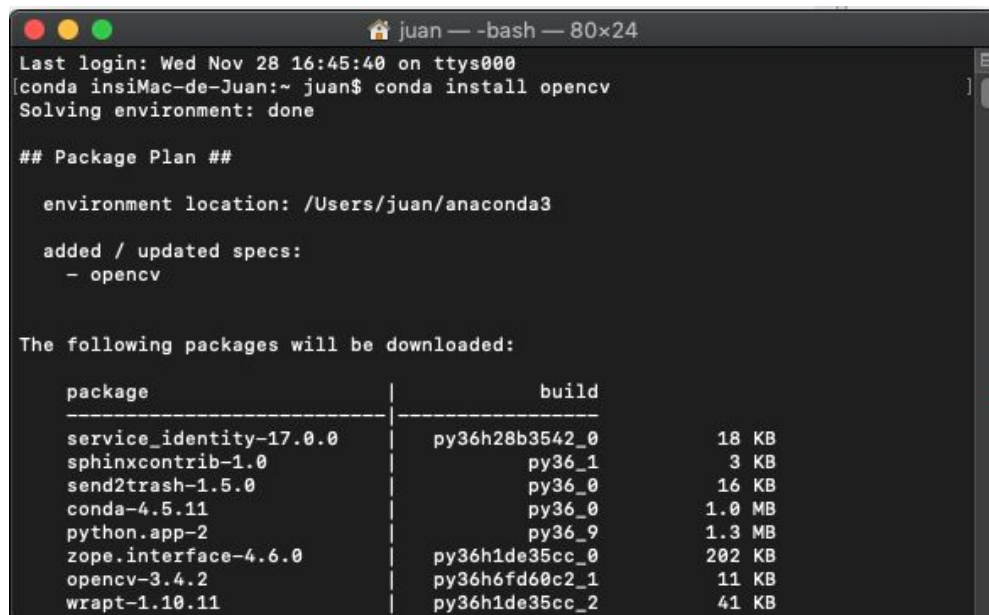


Figure 1.4: Options available to work with Anaconda.

For the current project, please install and launch the Spyder app to run a test demo, this will demonstrate that OpenCV has been installed correctly.

Once you have opened the application, a Terminal should be opened. While for Mac the regular Terminal should be opened, for Windows the Anaconda prompt should be opened. The Anaconda prompt can be found by clicking the Windows start button and typing "Anaconda prompt." After the respective prompt is opened, the following command should be typed:

- `conda install opencv`



```

Last login: Wed Nov 28 16:45:40 on ttys000
[conda insiMac-de-Juan:~ juan$ conda install opencv
Solving environment: done

## Package Plan ##

  environment location: /Users/juan/anaconda3

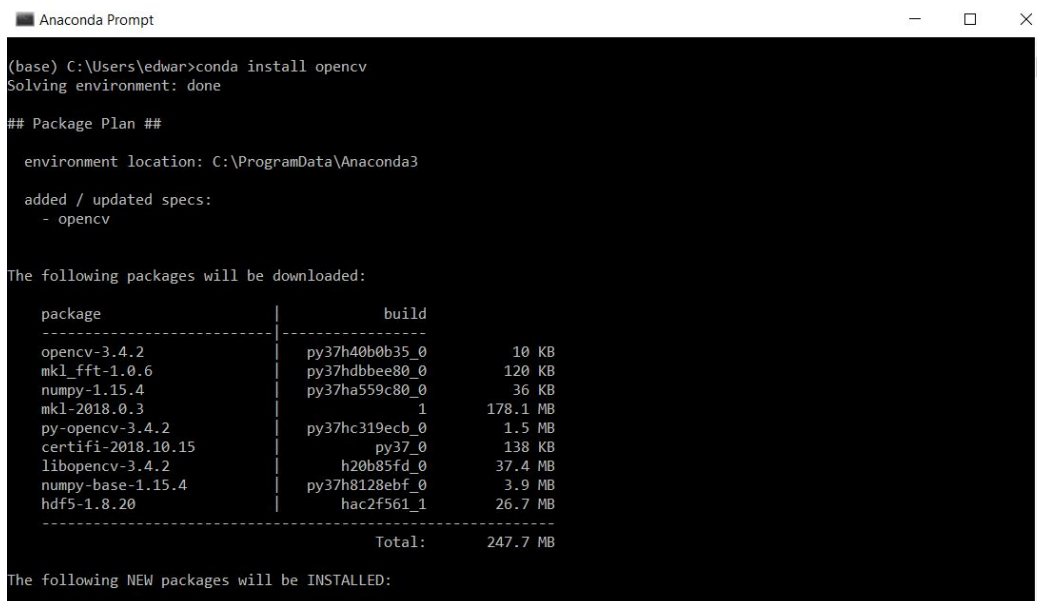
  added / updated specs:
    - opencv

The following packages will be downloaded:

package | build | size
-----|-----|-----
service_identity-17.0.0 | py36h28b3542_0 | 18 KB
sphinxcontrib-1.0 | py36_1 | 3 KB
send2trash-1.5.0 | py36_0 | 16 KB
conda-4.5.11 | py36_0 | 1.0 MB
python.app-2 | py36_9 | 1.3 MB
zope.interface-4.6.0 | py36h1de35cc_0 | 202 KB
opencv-3.4.2 | py36h6fd60c2_1 | 11 KB
wrapt-1.10.11 | py36h1de35cc_2 | 41 KB

```

Figure 1.5.1: Mac Terminal with the installation of OpenCV.



```

Anaconda Prompt
(base) C:\Users\edwar>conda install opencv
Solving environment: done

## Package Plan ##

  environment location: C:\ProgramData\Anaconda3

  added / updated specs:
    - opencv

The following packages will be downloaded:

package | build | size
-----|-----|-----
opencv-3.4.2 | py37h40b0b35_0 | 10 KB
mkl_fft-1.0.6 | py37hdbbbee80_0 | 120 KB
numpy-1.15.4 | py37ha559c80_0 | 36 KB
mkl-2018.0.3 | _1 | 178.1 MB
py-opencv-3.4.2 | py37hc319ecb_0 | 1.5 MB
certifi-2018.10.15 | py37_0 | 138 KB
libopencv-3.4.2 | h20b85fd_0 | 37.4 MB
numpy-base-1.15.4 | py37h8128ebf_0 | 3.9 MB
hdf5-1.8.20 | hac2f561_1 | 26.7 MB
-----|-----|-----
Total: | 247.7 MB

The following NEW packages will be INSTALLED:

```

Figure 1.5.2: Anaconda prompt opened in Windows with the installation of OpenCV.

When the installation finishes, please copy the sample code of Github and paste it in the Spyder app as the figure 1.6 shown. **The code is in this link:** https://github.com/JuanGro/Tutorial-OpenCV/blob/master/Figures/draw_line.py

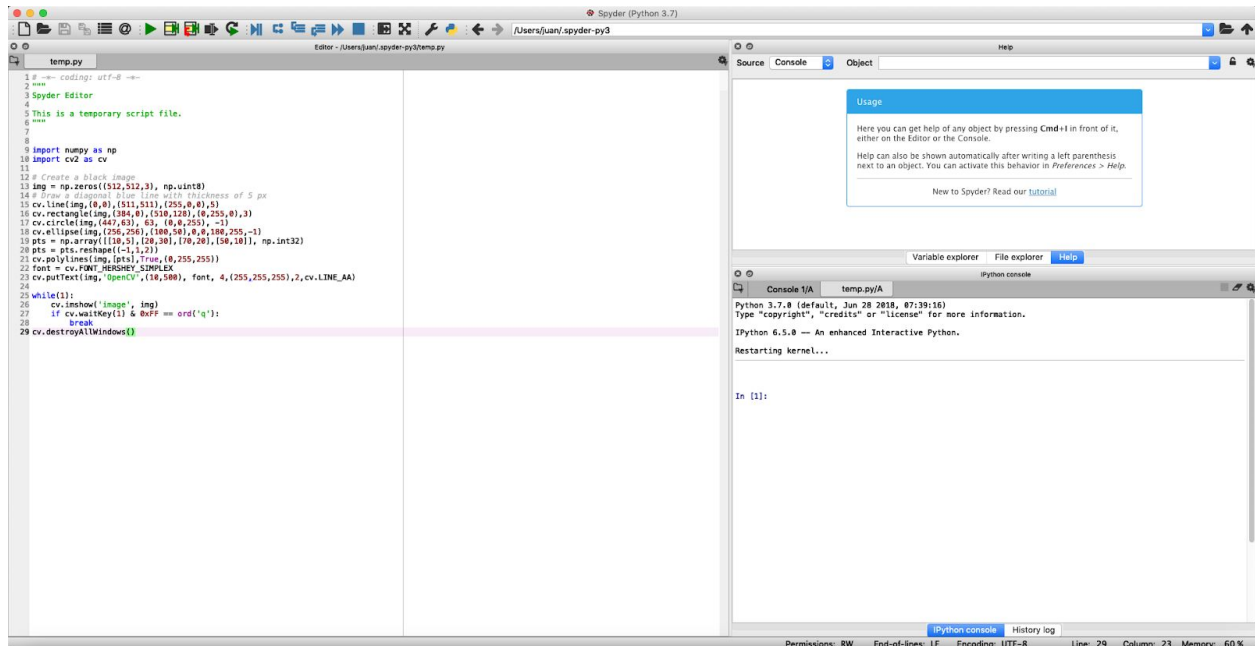


Figure 1.6: Spyder app with the sample code.

Run the file (F5) and the result should be something similar as the figure 1.7.

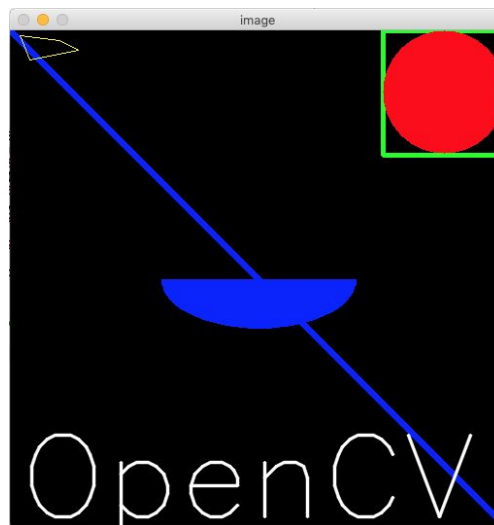


Figure 1.7: Result of the file execution.

Executing this code is mainly for testing purposes. If this code doesn't execute, it means OpenCV is not installed correctly. Shall this be the case, the command to install it should be typed again in the terminal, or Anaconda should be reinstalled; during class, some of the

class members had trouble running the OpenCV code, but after reinstalling Anaconda, this problem did not persist.

Installation of Bebop Drone environment

To fly the drone, first you need to install two libraries, which are *zeroconf* and *pyparrot*, to get them, please type in the corresponding Terminal (regular for Mac, and Anaconda prompt for Windows) the following command:

- `pip install pyparrot`

Once it has finished, type:

- `pip install zeroconf`

To test the drone, first we will run in the **Terminal** or **Prompt** a basic program which flies in straight line. The code can be downloaded in a ZIP as the figure 1.8 shows. The repository link is: <https://github.com/JuanGro/Project-OpenCV-Blink.git>

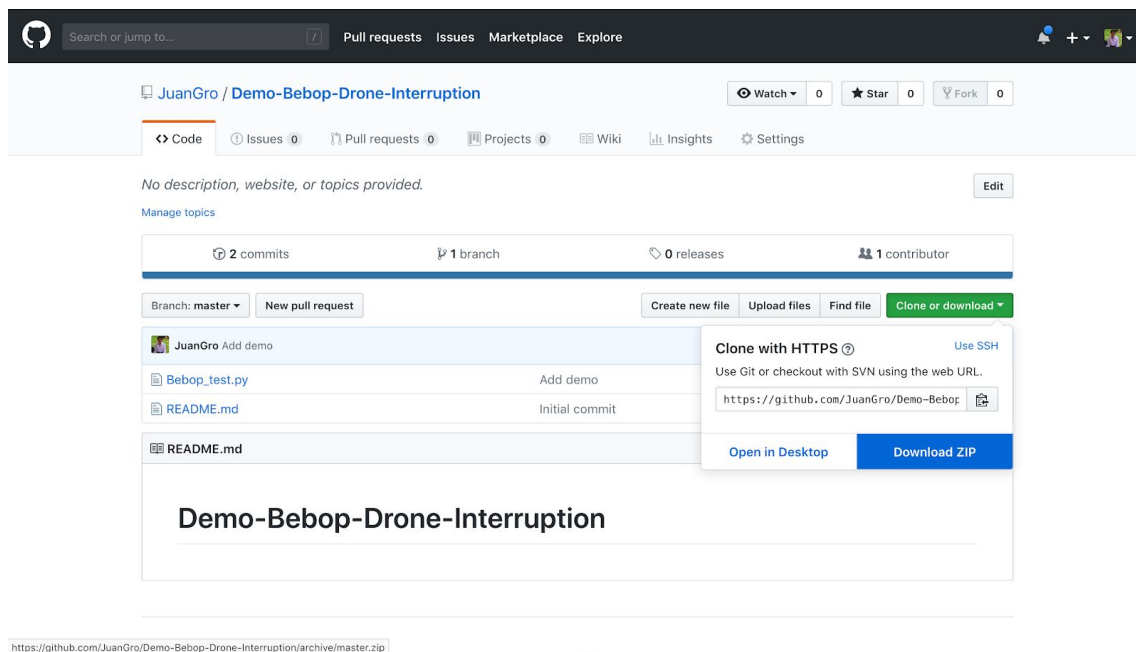


Figure 1.8: Option to download all the code as ZIP.

Decompress the zip to get all the files and code needed for the drone execution, place it where you want and access to that location from terminal. For example, if you place it in the Desktop, you should type in the **Terminal**:

- `cd Desktop/Project-OpenCV-Blink/`

Then run it with the command:

- `python3 Bebop_test.py`

In case that you need to land it as emergency or to avoid a collision, use the command:

- **Ctrl + c**

Project Download

The setup is now complete, if you wish to test our code, you can download it from the following link: <https://github.com/JuanGro/Project-OpenCV-Blink>. You can download all the files selecting the option “Download ZIP” as the figure 1.9 shows.

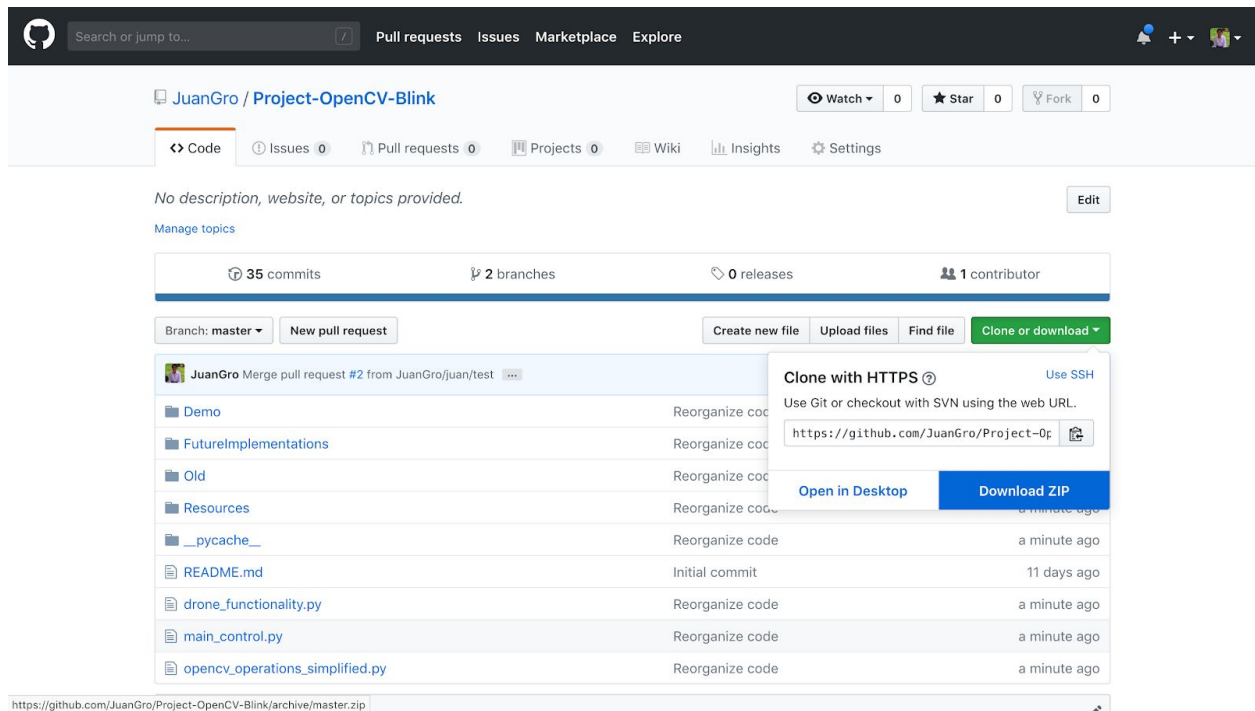


Figure 1.9: Option to download all the code as ZIP.

Decompress the zip to get all the files and code needed for the drone execution, place it where you want and access to that location from terminal.

For tutorial purposes, the code was placed in the Desktop, so the command to access it is:

- `cd Desktop/Project-OpenCV-Blink/`

However, move to the directory where the project was downloaded. Turn on the drone and connect to its wifi Network, then, execute the following command:

- `python3 main_control.py`

A graphical interface will pop-up where you can either run the code and expect the drone to execute the three behaviours defined in the description of the project, land the drone (in case of any unexpected inconvenience), or quit the program (make sure the drone has landed before quitting).

A **demo** of the drone execution can be seen in this link: <https://youtu.be/g4ly7a1cIRU>

Conclusion

During the development of the whole application, we encountered minor issues regarding several points. To begin with, the setup of the drone and the face recognition was confusing at certain points. This occurred because it was a lot of information and new topics we had to learn in a very short period of time, we had to develop individual applications regarding each of these topics, and all of them had to function separately in an optimal way to be joined in a later point. Moreover, combining all of the codes for each individual application and fitting them in a single code was challenging; we focused on making each minor application as perfect as possible, however, it is definitely different to generate a single code than to join several codes together, since it is easy to lose focus of the main application when focusing on the objective of a minor application. Another difficulty was executing the program. The reason for this was that some of the computers from the team members had some problems; one had a non-working WiFi antenna, another one had a damaged battery. Nonetheless, for each problem, we found the right with the help of teamwork and creativity.

In the end, the project was successful. We were able to understand and apply the principles of what we wanted to develop for a project and to parse the activities into minor problems, which got solved one by one, as mentioned before, with creativity and working as a team. Also, we understood the importance of robotic vision and some techniques that this subject includes, such as facial recognition, detection of objects and real time processing. We realize now the vast range of applications that robotic vision could be and is currently being used for, such as surveillance, detecting problems in crucial components for any application, and as demonstrated in this project, creating reactive agents that interact directly with humans.

References

- OpenCV library & manuals. (2018). Retrieved from <https://www.opencv.org/>
- PyParrot library. (2018). Retrieved from <https://pyparrot.readthedocs.io/en/latest/>
- Anaconda resources. (2018). Retrieved from <https://www.anaconda.com/>
- Installing Python Modules with PIP. (2018). Retrieved from <https://docs.python.org/3/installing/index.html>