Software Overview

Year: \_2019\_ Semester: \_Spring\_ Team: \_\_17\_\_ Project : \_Face Tracking Drone\_\_\_\_\_\_

Creation Date: \_\_\_\_\_Jan 23, 2019\_\_\_\_\_\_\_\_\_\_ Last Modified: Jan 24, 2019

Author: Yi Qiao\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Email: qiao22@purdue.edu\_\_\_\_\_\_\_\_\_\_\_\_

Assignment Evaluation:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item** | **Score (0-5)** | **Weight** | **Poins** | **Notes** |
| **Assignment-Specific Items** | | | | |
| **Software Overview** |  | x2 |  |  |
| **Description of Algorithms** | 4.5 | x2 | 9 |  |
| **Description of Data Structures** |  | x2 |  |  |
| **Program Flowcharts** |  | x3 |  |  |
| **State Machine Diagrams** |  | x3 |  |  |
| **Writing-Specific Items** | | | | |
| **Spelling and Grammar** |  | x2 |  |  |
| **Formatting and Citations** |  | x1 |  |  |
| **Figures and Graphs** |  | x2 |  |  |
| **Technical Writing Style** |  | x3 |  |  |
| **Total Score** | 99 | | |  |

5: Excellent 4: Good 3: Acceptable 2: Poor 1: Very Poor 0: Not attempted

General Comments:

Excellent Report. There are a few typos in the report. Please correct them.

You should specify how you are calculating the desired movement of the drone in details.

1.0 Software Overview

The autonomous ~~facing~~ face tracking drone requires two different firmware components for its base station and on-drone controller. With the firmware functioning properly, the drone should be able to track ones face, move with the movement of the person’s head and record high-quality videos.

Base station:

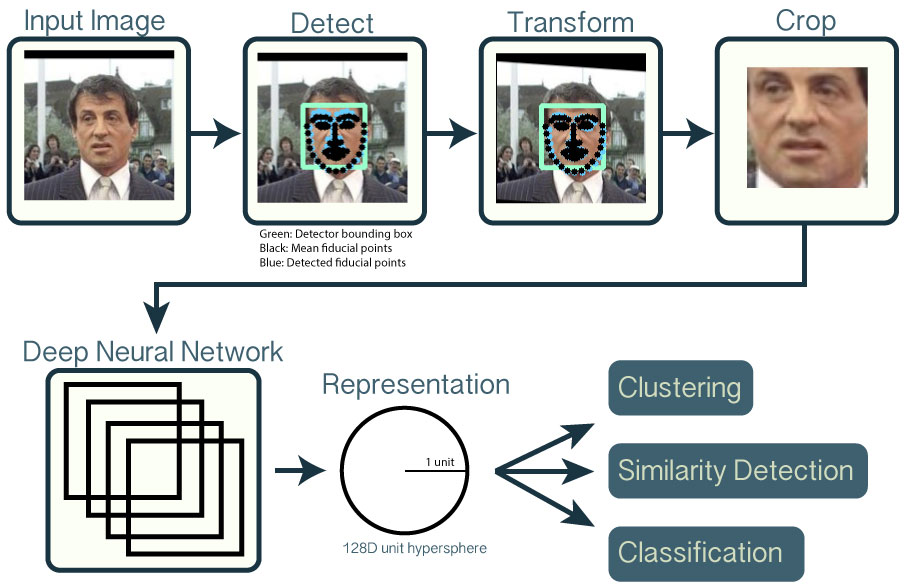
The base station is responsible for analyzing the video stream sent back from the video transmitter on the drone. First, it will initialize all components needed in the process. Then it displays the video stream in real time with bonding boxes on all the faces in the frame to the user. After the user makes the choice of which face will he/she track, the face-tracking process will start. It will take frames from the video stream and tracking the face with DNN, estimating the pose of the person’s head. After obtaining the relative position and pose in the frame, the desired movements will be computed and sent to the On-drone controller via RF/WIFI.

On-drone controller:

The On-drone controller have two function modes, which can be described in a state diagram followed in Appendix 2. The firmware will analyze the signal received from the radio receiver and switch mode between direct-control mode and face-tracking mode. With a onboard multiplexer, the direct-control mode can be directly realized through a GPIO pin and let the mux pass all the signal to the flight controller.

With face-tracking mode enabled, the mux will switch to the on-drone controller. The controller will request movement data from the base station through RF/WIFI and encode the movement command into PPM signals, and passing to the flight controller.

2.0 Description of Algorithms



Our face-tracking algorithm will be based on a open-source project from CMU called open-face[1]. It is able to do the following:

1. Detect faces with a pre-trained models from [dlib](http://blog.dlib.net/2014/02/dlib-186-released-make-your-own-object.html) or [OpenCV](http://docs.opencv.org/master/tutorial_py_face_detection.html).
2. Transform the face for the neural network. This repository uses dlib's [real-time pose estimation](http://blog.dlib.net/2014/08/real-time-face-pose-estimation.html) with OpenCV's [affine transformation](http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/warp_affine/warp_affine.html) to try to make the eyes and bottom lip appear in the same location on each image.
3. Use a deep neural network to represent (or embed) the face on a 128-dimensional unit hypersphere. The embedding is a generic representation for anybody's face. Unlike other face representations, this embedding has the nice property that a larger distance between two face embeddings means that the faces are likely not of the same person. This property makes clustering, similarity detection, and classification tasks easier than other face recognition techniques where the Euclidean distance between features is not meaningful.

With open-face, we are aiming at estimating the position and pose of a selected person’s face in real-time, with frame rate higher than 30 fps. After the position and pose is detected, the movement will be determined with the following factors:

1. Size of the face: When the program is initializing, the target face size will be determined as well. The size of face will be compared with the target size, and determine the pitch value of the drone. If the face is too large, the drone is supposed to move backward, with pitch being negative, or, face is too small, the drone is supposed to move forward, with pitch begin positive.
2. Position of the face: With a target location set in the frame, by comparing the real and target position, the roll value and height will be determined.
3. Pose of the face: After analyzing the pose of the target person’s head, the yaw value and the height will be determined.

While the drone is moving to the desired location, the gimbal is also moving in a way that compensate the movement of the drone and always point to the target person’s face. While the gimbal will always try to point the camera at the target, the drone will try to make movements on a larger scale.

3.0 Description of Data Structures

As said in 2.0, data packages will be sent from the base station to the on-board controller, containing the desired movement information. While nrf24l01 is able to transmit data at 2Mbps in-air, and our target operation speed is 30Hz-50Hz, the maximum package size will be around 5kB, which is more than enough for our purpose. Our drone is using around 8-10 channels, each channel can be represent with a 4 byte int value, composing 32 bytes-40 bytes per package in total. The layout will roughly be like this:

{

“yaw”: “0xFFFFFFFF”,

“row”: “0xFFFFFFFF”,

“pitch”: “0xFFFFFFFF”,

“throttle”: “0xFFFFFFFF”,

“gimble\_yaw”: “0xFFFFFFFF”,

“gimble\_row”: “0xFFFFFFFF”,

“gimble\_pitch”: “0xFFFFFFFF”,

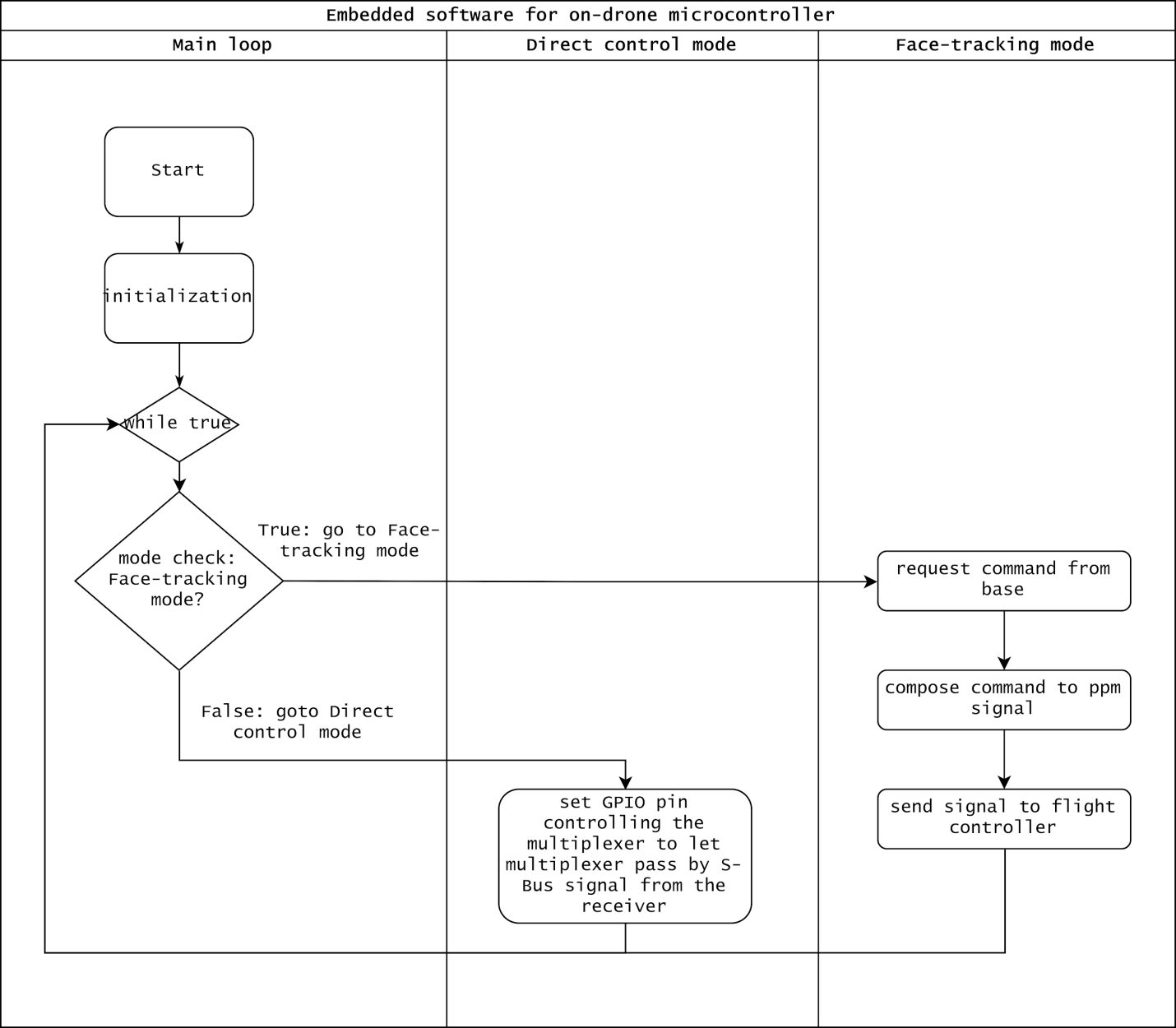
}

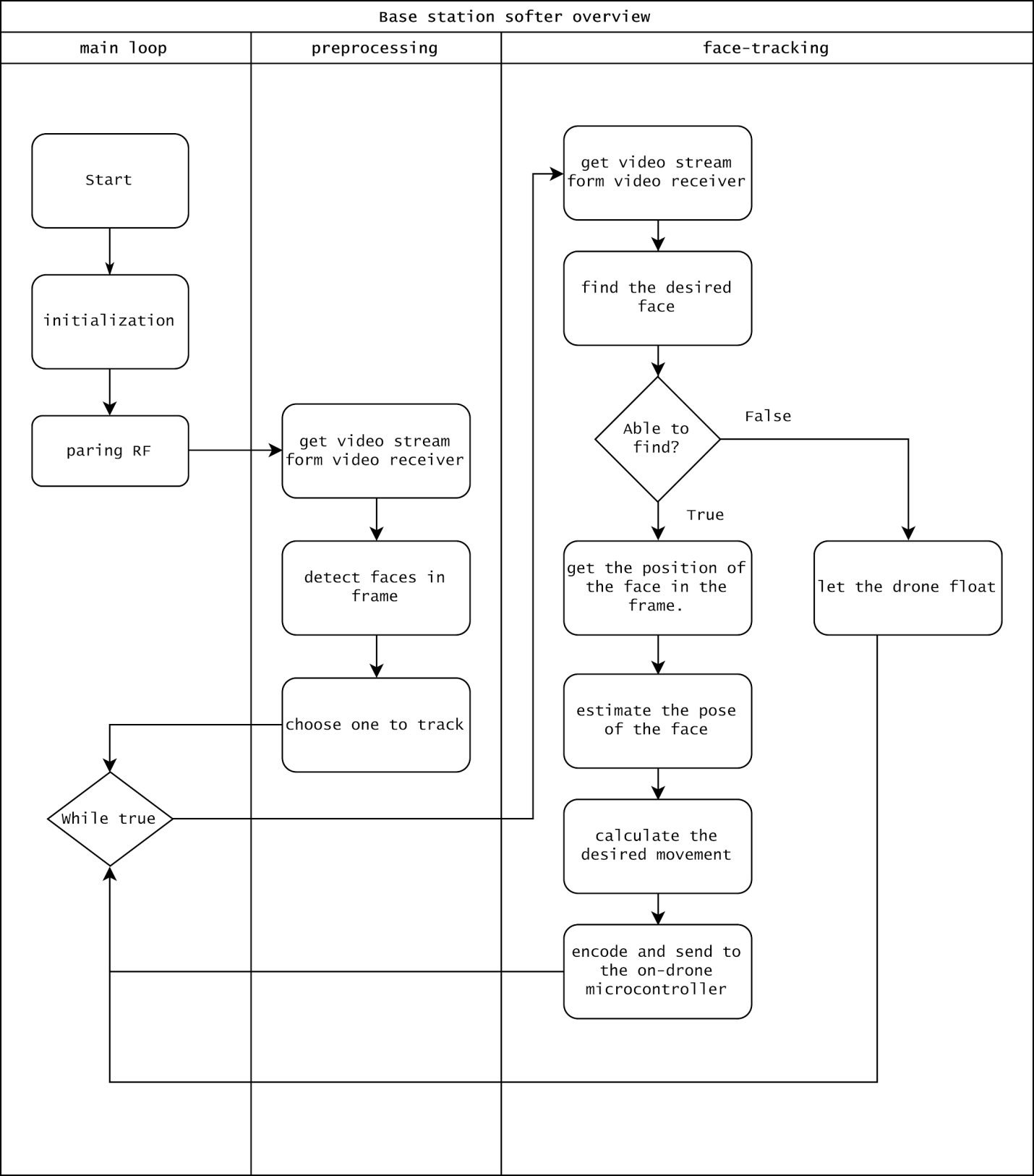
but in a more robust and concise manner.

4.0 Sources Cited:

[1] OpenFace. [Online]. Available: https://cmusatyalab.github.io/openface/. [Accessed: 25-Jan-2019].

Appendix 1: Program Flowcharts





Appendix 2: State Machine Diagrams

