AUTONOMOUS FACE DETECTION AND HUMAN TRACKING USING AR DRONE QUADROTOR

Mevlana Gemici, ECE/CS Junior/ Undergraduate, mcg74@cornell.edu
Yuandong Zhuang, MEng/CS, Graduate, yz527@cornell.edu
Keywords: face detection, iPhone game, vision, hand glove, awesome
Robot: Parrot AD Drone Quadrotor

Abstract:

Though there are many applications of personal robots that are developed to do useful tasks autonomously and without any human interaction, taking instructions from a human as the tasks get more and more advanced will become crucial. Therefore developing systems that will allow a robot to navigate towards and track a human as he moves in the environment will allow us to implement more advanced robotics tasks in the future.

Using computer vision algorithms and learning algorithms in combination allowed us to implement such a system on a flying robot platform, AR Drone. These algorithms ranged from learned feature detection (face/body) and color based detection (skin color) to image histogram tracking. Combining these different algorithms using a modified version of Kalman filter allowed our system to be quite robust in terms of tracking a moving human (1m/s) in moderately complex environments with 70% success rate in 30 trials.

I. Introduction

Increasing the interaction between a robot and a human is crucial for accomplishing some advanced autonomous tasks for a robot platform since these tasks could involve helping a human or following his/her commands. An air robot like the AR Drone Quadrotor was our choice of such a platform to test our tracking algorithms because of its ability to navigate in most common environments(normal room setting) with ease and without much need for obstacle avoidance at human height level.

Figure 1: Quadrotor

Although detecting a human body or face in a complex environment from a low resolution camera image without any depth information can be a quite a challenge, it is important to be able to implement reliable systems on such robots since many mobile platforms lack the hardware capabilities (high resolution cameras, Kinect sensors etc.) due to mobility or power concerns.

In order to develop a robust system, we divided the task of detection and tracking into different stages of operation. After the initial takeoff, our AR Drone is to search its field of view for any human faces using the face detection algorithms that we implemented to determine if there is a person in close range. If it is unable to find a person in close range, it switches to the body detection algorithm to look for humans in far distance and get close enough with constant speed and switch back to face tracking algorithms. Face detection is,

therefore, the main body of our approach since face detection was more important for us in terms of interactions with a human than the body.

Since we realized that it was almost impossible to get good results with a single detection algorithm for face tracking with a low resolution camera image, we implemented many different vision and learning algorithms and used them as independent sensors with different reliability for a Kalman filter that merged them into a single robust tracking algorithm with an overall reliability indicator. Using the output the Kalman filter in our PID controllers for the drone allowed the robot to adjust its altitude to the tracked person's face level, go towards him, keep an optimal distance (backward motion as well) and keep his face in its field of view by turning towards him as he moves in the environment.

II. Related Work

There are some of face detection algorithm developed based on similar methods: Skin color based face detection, Artificial Neural Network approach and feature classifiers. Although most of them works great in static pictures without much blurring, it is hard to make them robust individually in low resolution video streams with low frame rates where the environment and the tracked object/face moves significantly. As mentioned before, merging these algorithms using a Kalman filter was our approach for solving these hardware shortcomings which could work well with the algorithms that we didn't use in our implementation as well(like the Artificial Neural Network approach) since the output of each algorithm was considered a separate sensor in the Kalman filter,

III. Robot

As mentioned before, we are using the AD Drone Quadrotor robot in our project. It has two camera sensors (frontal and bottom camera) and an ultrasound altimeter at the bottom to determine the robot's height from the ground. The sensor data that we deal with comes from the front camera which has 320x240 pixels resolution three color image feed(stored as an multidimensional array representing colors in each dimension) to find and predict target's moves and the altimeter output(an integer value) to adjust the height of the robot to the height of the person that is followed. We haven't yet found a good use for the bottom camera image since it is unlikely that the robot will ever be right on top of a person's head and still be able to detect a face or skin from that perspective, so it was disabled.

The data is transmitted by a wireless connection between a computer and the Quadrotor which is the limiting factor in the amount of data that we get from the cameras (the camera resolution is actually 640x480) which has serious effects on our algorithms for face detection. Our algorithms require a fair amount of lighting in the environment to perform well since we are using vision as our main source of information.

There's a built API and platform in Linux that was provided to us by the Robot Learning Lab, through which we can connect to our robot and collect data from its sensors in real time, run our algorithms and send control commands to its motors. To achieve our goal: following a person automatically within a certain distance, we focus on detecting and predicting the location of a person face using the sensor data, and control the Quadrotor using the PID controllers that we implemented for yaw, pitch and height.

IV. Approach and Results

-Initial Approach

In order to allow the Quadrotor to track a person autonomously, the robot needs to determine the location of the human relative to its own location. Since we only have one sensor that gives us useful information about the environment (frontal camera), localizing the humans face in the frontal image stream corresponds to localizing relative position in the environment. In our initial approach, we didn't realize the need for having different algorithms for localizing a face in the image since we didn't know

the limitations of possible algorithms and their accuracies in low resolution image streams, nor did we realize the importance of having a body detection algorithm to realize a robust system.

Thus, we first implemented a system that had only one vision algorithm for human detection: Haar-Cascade classifier, a feature based algorithm that trains a classifier using many images of a specific object. In our case, we used the Haar-Cascade classifier to determine the human faces in an image. We considered that if the image resolution was good enough for detecting a face within a range of 10 meters with correct classification rate greater than 70% percent, we wouldn't need any other algorithms. Since this algorithm finds the center of the face in the image as well its size, we decided to use the size of the face as an indicator of how far the person is from the robot, giving us some depth information. Therefore the output of this algorithm had all we need to localize the human and set the PID control for pitch, height and the yaw of the robot.

Testing and Results

Our tests proved us wrong in many of our assumptions at this point. First we never realized the impact of the low resolution of the image that we were getting on the feature based classification that we were using. At around 5 meters away from the robot, the human face became so small(in pixels) that it was almost impossible to detect it using the classifier with precision. It was possible to decrease the window size of the training images to recognize smaller faces. However this led to a huge increase in wrong classifications which we could not deal with easily and it made the system too unreliable. Also, even if the right face was detected in the image, there was the possibility of detecting other wrong objects as faces at the same which we had no way of determining and pruning out.

The impact of using a feature based algorithm on the performance was quite large as well, a cost that we weren't willing to accept for such low correctness (hit) rate. The frame rate that we were able to process dropped from 13 fps to 2.3 fps after applying the algorithm. Following is the summary of the mean results after 10 trials where the human was in the image within a range of 2 meters from the robot.

Duration(second) of each trial	20
Total # of frames without algorithm	267
Total # of frames using algorithm	46
Total # frames with some faces detected	34
Total # Frames detected the correct face	25
Total # frames with multiple faces detected	9
Rate of face detection	73.9%
Rate of correct face detection	54.3%

Applying the position and size information to the PID controllers and testing the algorithm on the robot also showed how unreliable this system is since there were many cases of misclassifications and the robot tried to track random objects in the environment, though it sometimes detected the correct face and tried to track it correctly for a short time until the next misclassification.

-Second Approach

After testing our first algorithm, we realized the shortcomings of the feature-based classification with a low resolution camera. Especially, when the tracked human was farther than 2 meters, the Haar Cascade classifier's performance was close to zero. To improve our range of detection without a lot of cost to the performance, we decided to implement a skin color based detector that would help us make an estimate of where the person was in the frame when there wasn't any detection (or multiple detections) from the feature based classifier.

Implementation:

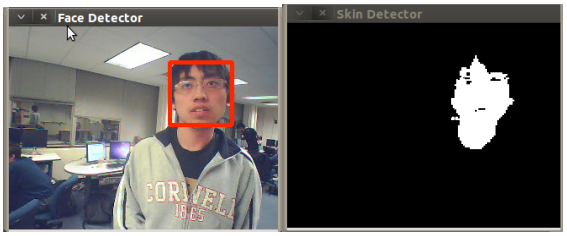
In order implement the skin detector we changed the RGB color space into YcbCr color space which separate the luma signal into Y channel and get rid of this vairable. This was done so that we would be able to detect the skin even under different illuminations.

*Y=128+0.299*r+0.587*g+0.114*b;*
*Cb =128 -0.169*r - 0.332*g + 0.500*b;*
*Cr =128+ 0.500*r - 0.419*g - 0.081*b;*

Most of the skin color pixels satisfy the filter:

Cb>85 & Cb<135 & Cr>135 & Cr<180

After classifying the image into skin and non-skin pixels, we filtered out the noise in the image and selected the greatest regions that had the shape of a face in terms of roundness and proportions.



Testing and Results

The impact of the skin detector on the speed of the overall algorithm (feature-based (Haar) detection+skin color based detection) was quite small (less than 0.1 fps) in addition to the initial approach. However, we found that, this algorithm was quite bad in terms of classification performance by itself especially in complex backgrounds. Even though its classification rate was 90%, the wrong classification rate (falsely detected frame/ total detect frame) was more than 70%. This was due to the fact that the color in a complex background would also include many objects with similar YcbCr values. In our regular test setting of moderate complexity (Upson Lab 317) it detected many yellow or brown objects (door, table etc.) as skin and it was quite difficult to determine these misclassifications using shape features. Even though we tried to use this classifier only as a means to help the Haar Cascade feature classifier to choose between detected faces in the same frame, this proved to be of little value in terms of decision making as well in >1m range since the amount of misclassification was too large at that range. We have decided to give up using a skin color based detector as a possible classifier at this point.

-Final Approach

Our experiences with testing the Haar-cascade feature based face detection algorithm on the robot so far, made us realize the importance of choosing the correct face among all the faces that are detected in the image since this would increase the correct detection rate from %50 to around %75-80 which would make a lot of difference for a flying platform that needs a continuous information about the location of the face in order to stabilize itself on the right track towards the person. In our initial approach of choosing the face randomly among all the faces that are detected in the classifier, caused a lot of trouble for the robot since a wrong choice would lead it out from the right path, possibly causing the person to move out of the field of view of the camera, the only sensor that we have to detect the person again.

Therefore, in order to increase the correct detection rate from the feature based classifier, we decided to choose the most likely face among all the faces that are detected in a frame by looking at the previous frame that there was valid detection and simply choosing the closest face to the previously detected face. This was done according to this distance formula where size of the face was also considered important:

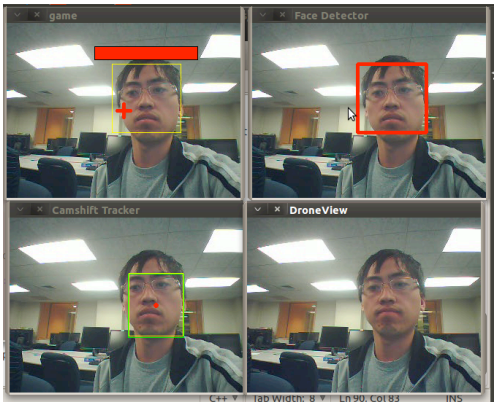
$D=sqrt((x\_prev-x\_current)^2 +(y\_prev-y\_current)^2+(sqrt(size\_prev)-sqrt(size\_current))^2)$

As it will be mentioned in the results, this improved the correct classification rate significantly (almost equal to the face detection rate which was greater

than 70%) which allowed us to use another method to improve the robustness of our system.

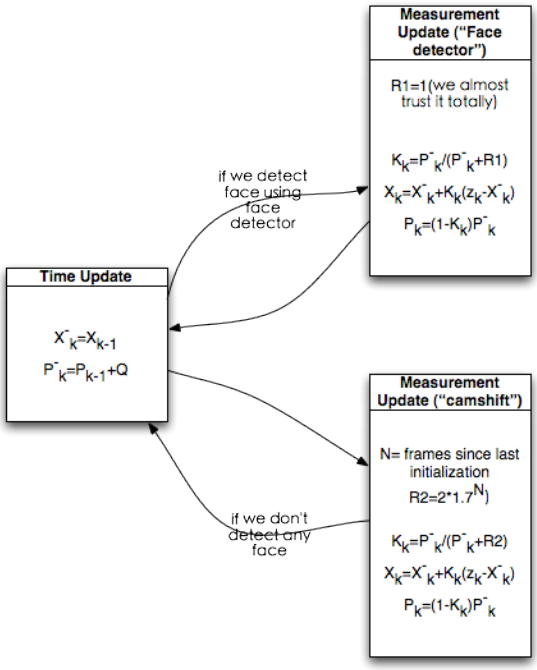
Image Histogram Based Tracking- Continuous Adaptive Mean Shift (Camshift)

With the improved Haar-Cascade feature classifier giving a correct classification of >70%, we decided to implement an algorithm that will fill in the blanks between each valid face detection and allow us to predict the future position of the face for a short period of time (time enough to move with the Quadrotor and try to keep the person in the image frame). The algorithm that we used for this task was the Camshift algorithm of OpenCV, where we took advantage of the fact that each subsequent frame was related to each other, due to their continuous video nature. The main idea of this algorithm is using color histograms of each frame to keep following a region in the continuous stream of images. Given a region of space in the image, the algorithm compares the new images with the initial image to determine the best match in the new image in terms of color histograms and sets this new region to be the region to be tracked in the next frame. Even though in time (error exponentially increases), the region to be tracked becomes completely wrong, it was possible for us to refresh the region to be tracked each time there is a valid face detection from the Haar cascade feature classifier. This resulted in a tremendous increase in the correct face detection rate when there is a face to be detected in the image.



Kalman filter

We implemented a modified version of Kalman filter to combine the data we get from the feature-based classifier and the Camshift algorithm to produce a reliability indicator so that the Quadrotor doesn't try to follow random objects in the environment. Following diagram shows the general implementation of the Kalman filter that has two different sensors with different reliabilities.



In prediction stage, we selected the process noise covariance Q=20 for each dimension which is quite reasonable since usually the average changes in pixel between frames in 5 pixels for each dimension.

In measurement update stage, we select the update sensor depending on whether we get valid detection from the Haar-cascade classifier. When we do get a valid detection, we use the first measurement update which has a very small error covariance(R=1) since we trust the feature detection algorithm.

When we do not get valid face detection, we update using the data from the Camshift algorithm which

has an exponentially increasing error covariance with respect to time.

$R = 2*(1.7^N)$

This means that the Kalman gain of the filter will decrease in time even though we do the measurement update using the Camshift data.

On the graph to the right, we can see at the bottom of the image in black, the error covariance of overall algorithm which slowly increases in time as we do not make any valid face detections from the Haar-cascade classifier (therefore the robot has only the Camshift data). Every time there is valid face detection from the classifier, the error covariance decreases sharply.

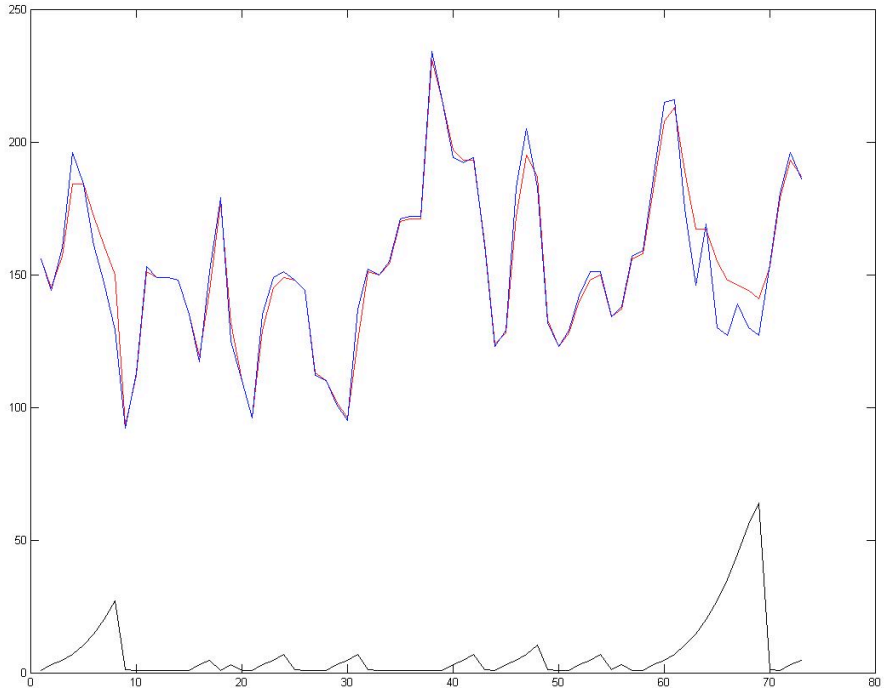
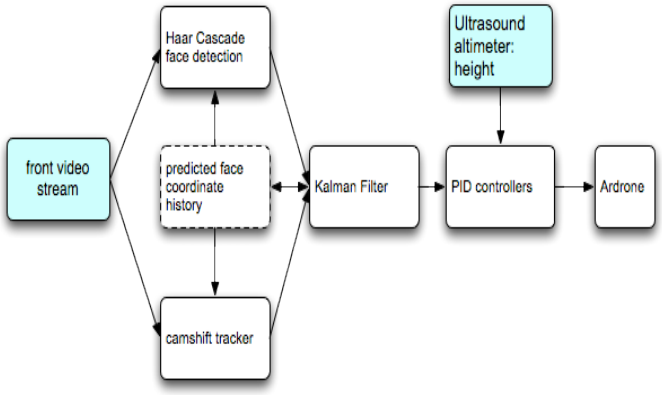


Figure 2: Kalman Filter Input (blue), Output (red) and Error Covariance (black) for x direction with respect to time

It is this error covariance that makes us choose between trusting the output of the Kalman filter to go towards a target and inaction. Therefore, only when the error covariance is below a certain threshold (we determined it to be 28 - the sum of the errors in all directions) we set the PID control of the Quadrotor to move towards a predicted position of the target.

On the upper side of the graph, we can also see the input and output positions for the x direction of the Kalman filter. As we can see the output follows the input almost perfectly when there is valid face detection (where the error covariance becomes very small) and at the same time getting rid of the sharp changes (errors) from the Camshift data when there is no valid detection from the feature classifier.

Final High Level Design:



Body Detector:

After we finalized our face detection algorithms, we decided to include the body detector that was implemented by Yuandong for his MEng project that involves the quadrotor which used the Haar-cascade as a body classifier. This allowed us to increase the range of 2.5 meters that we get from our face detection algorithms to 6.5 meters which made a quite an impact on the overall robustness of the system. This was separated from the Kalman filter and Camshift algorithms, and simply involved going towards a body in the image with constant low speed until it gets close enough for face detection. At this point this algorithm was disabled. The error rate of this feature classifier was smaller due to the size of the body when the person was within 7 meters.

Results:

We tested our algorithms from our final design both offline and on the robot with good results. The tests were conducted with different levels of difficulty with moderately hard background settings (lab with many objects that could cause misclassifications).

In the first test set, the drone was able to successfully move towards a person that was standing still and keep a constant distance for 20 seconds with right height level 9 times out of 10. This didn't include the body detector and the drone started from 3 meters range.

The second test set was conducted when the person was moving with a slow speed 0.4m/s) in the environment. The drone was able to keep a proper distance and keep its direction towards the person 8 out of 10 cases. Initial start conditions were the same as the first test set.

In the third test set, the person moved in a higher speed (around 0.8 m/s), the drone successfully followed the person 6 times out of 10, though it sometimes came close to crashing to the person if he made a sudden stop. Even though the person moved in a fast manner he was always in the field of view of the robot and within 3 meters of range.

The final correct face detection rate was 82% from a range of 2.5 meters.

Conclusion:

Designing a robotic system to interact with humans could be quite challenging since there are many issues like navigation in the environment, sensor shortcomings and hardware issues due to mobility concerns. Confronted with this kind of a situation, we developed several learning and vision algorithms that complemented each other and improved the overall performance of the task we wanted to accomplish.

Acknowledgments

We thank Professor Saxena for his guidance and ideas for the Kalman filter that allowed us to combine our algorithms in a meaningful manner. We also appreciate all the help that we got from our TAs, especially Cooper Bills and Colin Ponce. Also the implementation of the Camshift algorithm that we used was a slightly modified version of Cooper Bills's own algorithm which he provided.

Reference

Ian R. Fasel and Javier R. Movellan "A Comparison of Face Detection Algorithms"
Paul Viola, Michael J Jones "Robust Real-Time Face Detection " International Journal of Computer Vision 57, pp. 137-154, Netherlands, 2004

Sanjay Kr. Singh, D. S. Chauhan, Mayank Vatsa, Richa Singh "A Robust Skin Color Based Face Detection Algorithm" *Tamkang Journal of Science and Engineering, Vol. 6, No. 4, pp. 227-234 (2003)*

Son Lam Phung, Abdesselam Bouzerdoum, and Douglas Chai "A Novel Skin Color Model in YCbCr Color Space and Its Application to Human Face Detection"

G.R. Bradski, "*Computer video face tracking for use in a perceptual user interface"*, Intel Technology Journal, Q2 1998

OpenCV reference *http://opencv.willowgarage.com/wiki/*

Greg Welc, Gary Bisho "An Introduction to the Kalman Filter"

MENG PROJECT IMPLEMENTATION OF AUTONOMOUS FACE DETECTION AND HUMAN TRACKING USING AR DRONE QUADROTOR

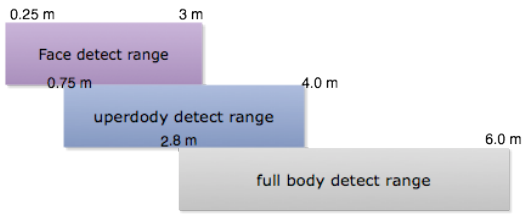
Yuandong Zhuang, MEng/CS, Graduate, yz527@cornell.edu
Keywords: face detection, iPhone game, vision
Robot: Parrot AD Drone Quadrotor

Body detect

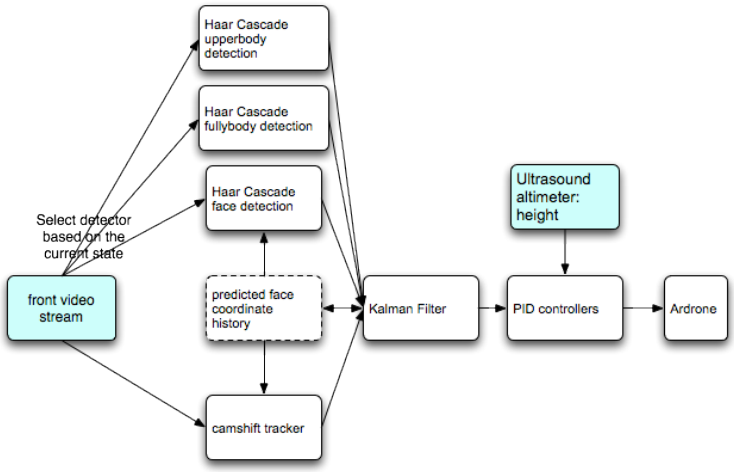
Previously, we use only the face detector to localize the human. The face detector works well in the distance between 25cm to 3m away from the camera, but it will be hard for quadrotor to recognize the face beyond this distance. To improve the performance, I improve the previous algorithm by integrating the body detector and upperbody detector based on the Haar-Cascade classifier.

In the cases that drone fail to follow the person (cannot detect the face) for a certain frames (10 frames threshold is set in the algorithm), he quadrotor will try to detect the upperbody of a human. If fails, it will try run the fullbody detector.

In the offline test, the detection distance of upperbody detector is between 75cm to 4m. And the fullbody detector works in the distance between 2.8m to 6.5m. The affective distance are showed in the following diagram:



To reduce the noise, a similar kalman filter is applied to the data produced from two detectors as he face detector. But unlike face detector, we don't apply camshift since it works much worse in both situations since the color varies a lot in histogram of that selected region which violates the precondition of Camshift. Also the size information becomes less important in this case since we know that the robot is quite far away from human. The PID controller use the center of the body as the target direction and control the drone move to the human in a constant speed(0.5 m/s). The new final high level design diagram changes into the following:



On flight testing, the new algorithm can detect and follow the human as far as 6.5m compare with 2m in previous algorithm.

Implement the algorithm in iOS

The code we wrote previously is for linux platform. To implement the game feature in iOS, I start code using the code base provided by Parrot S.A developer kid. The code is written in object-C, it provide the basic GUI, control interface and video live feed from quadrotor implemented using Open-GL ES.

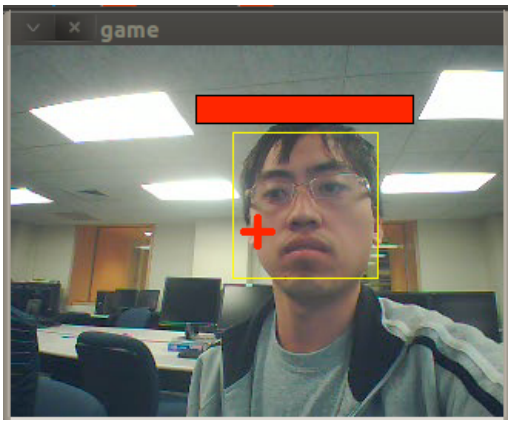
iOS doesn't support native OpenCV. The first step is to compile static OpenCV library and integrate it into the code. Several online resource explained how to implement it (attached in the reference) and I also met some minor issue caused by the different version of OpenCV and iOS.

Then the OpenGL rendering of the video feed is converted into OpenCV data, and intergrated with the C code rewrote from the C++ implementation of our own algorithm. The PID controller interacts with the control data interface to operate the quadrotor automatically.

The processor in iPhone is not as powerful as a regular X86 processor. The frame rate drops from 2.3 fps to 0.8 fps when running Haar Cascade feature classifier for face detection. And after integrating the body detector, it drops to 0.6 fps. It's far below the necessary frame rate to control the quadrotor normally.

Game feature in Linux

The algorithm we've wrote can be implementing as an interactive virtual reality games. We are controlling the quadrotor as a weapon and try to search around to find the target person. At the same time, the target can also move to avoid be locking and attacking by the quadrotor. When the target is not attacked, he/she will regain health slowly, which makes the game more interesting. The harm done by the quadrotor is determined by how good the target is tracked. The closer the target to the center of the screen, the greater the harm will be. Sample screenshot is as follow:



A NOVEL GLOVE CONTROLLER FOR THE AR DRONE QUADROTOR

Mevlana Gemici- Justin Kuo

ECE 4760 Final Project

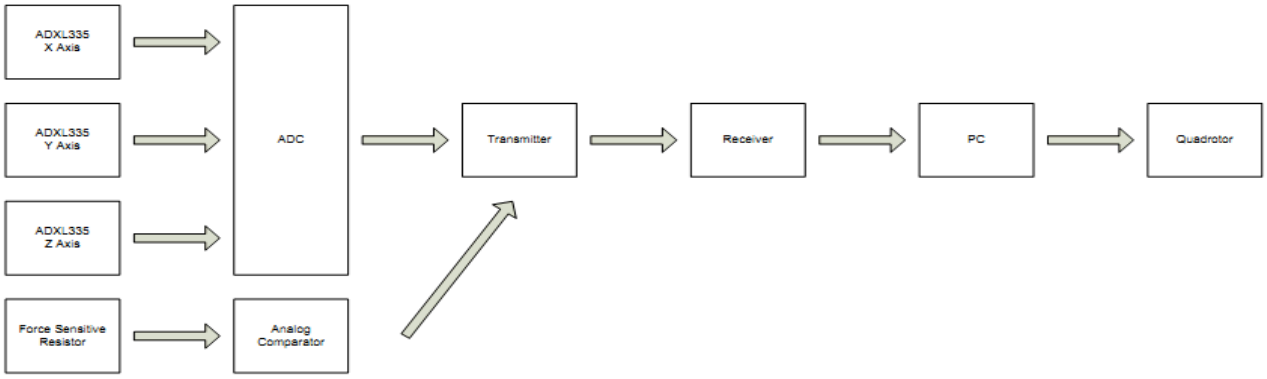
Project Description:

Our project is a novel hand held controller in which we use an accelerometer to wirelessly control the motion of a Parrot AR Drone Quadrotor.

Rationale:

The main idea of our project was building a cool glove controller for a flying platform, a Quadrotor in this case, which would give the user a different sense of power over the robot, unlike conventional controllers like the Xbox controller. Even though this is an ambitious project(especially for a budget limitation of $75) that aims to make a moderately good controller, it can open the way to further development for this type of controllers for flying platforms used in the Robot Learning Lab in Cornell.

High Level Design and Logical structure:



Summary:

We utilized the ATMega32 microcontrollers in our project to determine the hand motions of a person by processing the data from an ADXL335 accelerometer chip that detects motions in x-y-z directions. These sensor data were sent in an encoded fashion through a radio frequency (RF) link and received using another microcontroller of the same type. These wireless messages were processed by the receiver which was connected to the computer that sent the processed command signals to the Quadrotor to be controlled.

For a full report on the implementation, hardware, code and related discussions:

http://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/s2011/mcg74_jck226/mcg74_jck226/mcg74_jck226.html Videos will be uploaded on the course website soon as well.