Computer Vision Technology

Follow me Project

Roberto Chavez Jr
RESEARCH INTERNSHIP, UNIVERSITY OF CALIFORNIA, SANTA CRUZ

This research was done under the supervision of Joel Kubby with the financial support from NASA Ames Research Center within a total of 24 weeks, from January 2th to June 1st of 2016.

First release, June 2016
# Contents

1 Introduction ................................................................................. 5
  1.1 Overview ........................................................................... 5
  1.2 Technical Discussion Outline ............................................. 5
    1.2.1 Feature Detection .......................................................... 5
    1.2.2 Face Detection .............................................................. 5
    1.2.3 Obstacle Detection ......................................................... 6
    1.2.4 Integrating Computer Vision onto the Drone .................. 6
    1.2.5 Problem Statement (In progress: Currently working on the machine learning capabilities) .................................................. 6
    1.2.6 Equipment Required for the Project ............................... 7

2 Feature Detection ........................................................................ 9
  2.1 OpenCV for Feature Detection (Object Tracking) ............... 9
    2.1.1 The code .................................................................... 10
    2.1.2 Analyze ..................................................................... 10
    2.1.3 Commands used for the Feature .................................. 10
    2.1.4 Conclusion ................................................................. 11

3 Face Detection .............................................................................. 13
  3.1 OpenCV for Face Detection ................................................ 13
    3.1.1 The code .................................................................... 14
    3.1.2 Analyze ..................................................................... 15
    3.1.3 Conclusion ................................................................. 16
4 Obstacle Detection ............................................. 17
  4.1 OpenCV for Obstacle Detection .......................... 17
    4.1.1 The code .............................................. 18
    4.1.2 Analyze .................................................. 19
    4.1.3 Conclusion .............................................. 19

5 Conclusion and Reference ................................. 21
  5.1 Conclusion .................................................. 21
  5.2 Reference .................................................. 21
1. Introduction

1.1 Overview
When I applied for the research opportunity, the title of the project was *Follow Me* which is a condensed field in computer vision technology.

My research project is implementing computer vision technology onto a drone for facial recognition and Object detection [1]. The Drone was built last year from a senior design project to manually fly the machine [coudn’t find reference]. This year will be to autonomously fly the machine with having a camera to detect moving object to alter its direction. The technology used for Computer Vision Technology onto the drone is an open source framework called OpenCV enabling me to program in C, C++, and also in python. The purpose behind the project is to give the drone intelligence to not harm other people within their environment...thus preventing regulations towards drones.

1.2 Technical Discussion Outline
The major points I plan on addressing on the final project is an overview what computer vision is and how it’s application goes with the Drone.

1.2.1 Feature Detection
It’s a feature [1] in OpenCV to have a “still” video camera detecting any moving objects. The way we see an object has moved is based on frame comparison from a recorded video. We first detect an object by setting an image gray assuming no objects are moving. Any object that is moving like a ball for example, bouncing against a wall will convert the outer layer of the ball into white pixels to gather the absolute difference with the next frame captured. We take the absolute difference with the pixel gap to decide either the ball has moved or the camera didn’t stay “still”.

1.2.2 Face Detection
This feature requires more processing power in OpenCV to detect someone’s face [2] and recognizing who the person is. For example, if Obama’s face was used as a picture [3][4], the program would
process the amount of pixel data to tell the user the picture is President Barrack Obama. The way the design work is to initially detect the darkest section of an image which is usually the person’s eyes. The feature also needs to have neural net capabilities to detect who is in the picture. This is a condense field related to machine learning.

1.2.3 **Obstacle Detection**

This is built on feature detection in order have a classification, in other words, give intelligence to the computer depending on the algorithm such as neural networking. Neural Network gathers input values from the Feature Detection in order to classify what is displayed on the image. If I took a picture of a stop sign [5], feature detection classifies the outlines of an image from gathering input values on a single pixel. Obstacle Detection [5][6] gathers the input values from Feature Detection, cluster them into one gigantic input value which is the entire picture instead of one pixel, and make a decision if the picture is a stop sign.

1.2.4 **Integrating Computer Vision onto the Drone**

All the concepts we mentioned earlier for computer vision is important when implementing each feature from coding in OpenCV. I do not have enough info provided to integrate each feature mentioned earlier into the drone. This report will contain information in OpenCV that will be implemented to the drone.

1.2.5 **Problem Statement (In progress: Currently working on the machine learning capabilities)**

There are multiple steps required to have a drone with Computer Vision Intelligence. Once the camera is implemented into the drone, it will video record continuously when the drone is ready to fly. Within every frame, we will have an image that will be translated into binary sequences to decide if there’s a face is in the image [5][6] (Face Detection [4]). Once the drone detects a face, the drone will be in a state to continuously follow the persons face. If the location of the face has changed from the previously location of the camera, the drone will need to have feedback control to center the face back to the camera (haven’t finished), it will require a mapping of it’s location through a Lidar (laser) sensor that will let it be able to map out a 2-D map of it’s surrounding.

Once we successfully track something on screen, you can translate the flight commands to follow any object within each frame per second recorded from the Zed video camera. We plan on using multiple sensors for the drone such as the communication protocol (mavlink) from the Zed camera to the microcontroller embedded in the drone to transition to a state, altering it’s direction. I started off with a Mavlink message to relay on the inertial measurement data embedded like the gyroscope. I would take the drone and rotate the mechanism to see the values change from 90 degrees’ position to 75 degrees on one axis for accurate inputs. If I ever wish to hone my skills even further for feedback control system on the drone, Elkaim’s and/or Mirca’s lab that knows ardupilot and mavlink.

GPS is the coordinates for global positioning. You can also control the copter with GPS by using mavlink to see its current position. There are many possibilities for you to get creative and innovative. One huge factor is that you can use multiple methods since flying indoors would be a huge bonus other then the fact gyroscopes are more accurate outdoors (tradeoff).
1.2.6 Equipment Required for the Project

There are three equipment’s that will be the main focus for the project. First is the Zed camera with high definition 3-D Camera for depth sensing. The product cost up to 449 which is a bit pricy. The camera will be connected to the second equipment which is the central computer. The central computer, model is unknown at the moment, will be the bridge between the control feedback of the drone flight controls to the OpenCV program application. The last equipment is the Lidar (Laser) sensor which is to map out the location where the drone is at a limited range. The map will be displayed in a ‘2-D reference view.”
2. Feature Detection

2.1 OpenCV for Feature Detection (Object Tracking)

If you read up to this point, I’ve only gave out the higher level abstraction for Feature Detection [1]. Now I plan on conveying the coding process. Feature Detection is utilized to find the outer layer of a small object. To find the outer layer by invoking the command

```cpp
cv::absdiff(frame1, frame2, output)
```

Figure 2.1: threshold comparison [1]

The command converts the image in gray to make it easier to find a moving object which is displayed in white colors (threshold image). If you look at Figure 2.1 [1], we have a gray background with two white circles crossing over in the middle. This is an image of a ball captured from two different frame shots, stack the two images into one to indicate if the ball has moved. This is a method used to calculate the absolute difference in computer vision by collecting the objects outer
layer which is also known as the threshold image. If the ball was staying still for two frames, we would have one white circle instead of two. This is still high level abstraction behind the coding process.

### 2.1.1 The code

```cpp
while(1){
    while(capture.get(CV_CAP_PROP_POS_FRAMES)<
        capture.get(CV_CAP_PROP_FRAME_COUNT)-1){
        capture.read(frame1);
        cv::cvtColor(frame1,grayImage1,COLOR_BGR2GRAY);
        capture.read(frame2);
        cv::cvtColor(frame2,grayImage2,COLOR_BGR2GRAY);
        cv::absdiff(grayImage1,grayImage2,differenceImage);
        cv::threshold(differenceImage,thresholdImage,SENSITIVITY_VALUE,255,THRESH_BINARY);
        if(debugMode==true){
            cv::imshow("Difference Image",differenceImage);
            cv::imshow("Threshold Image", thresholdImage);
        }else{
            cv::destroyWindow("Difference Image");
            cv::destroyWindow("Threshold Image");
        }
        cv::blur(thresholdImage,thresholdImage,cv::Size(BLUR_SIZE,BLUR_SIZE));
        cv::threshold(thresholdImage,thresholdImage,SENSITIVITY_VALUE,255,THRESH_BINARY);
        if(debugMode==true){
            imshow("Final Threshold Image",thresholdImage);
        }
    }
}
```

### 2.1.2 Analyze

Above us we have ourself source code on Github [2]. In order to continuously capture a frame, we need to go through an infinite while loop when the Zed camera is turned on. Since the while (1) is being executed every 1 micro second, we won’t be able to capture a frame since we need one second. Which is why we have our self another infinite loop to only be executed every second.

```cpp
while(capture.get(CV_CAP_PROP_POS_FRAMES)<capture.get(CV_CAP_PROP_FRAME_COUNT)-1)
```

If you look at the code above us, not only we have to wait every second to capture a new frame, we also have to wait until the previous frame has completed gathering inputs for the threshold data. Gathering the threshold data takes less then 10 milliseconds, so as long the data doesn’t take more then 1 second’s process, we shouldn’t alter the code in any way.

### 2.1.3 Commands used for the Feature

- `absdiff` is the absolute difference between two frames.
- `frame1` is the threshold image of the moving object. In this case the white circle which is the ball bouncing for this example.
2.1 OpenCV for Feature Detection (Object Tracking)

- the output will display a 1 or 0
  - 1: The ball moved from an appropriate absolute difference between two frame comparisons.
  - 0 Ball was still.
- capture is used to keep an image stored.
- cvtColor is used to convert the image to grey to make the processing color analyze more quickly.
- Threshold command is used to find sensitive objects moving from the frame comparisons. If we recorded a video of a wall, there would be no threshold frame comparison due to not having small or sensitive object moving a lot to the human eye.
- imshow is used to display the threshold image to the computer processing the threshold. This command is only called when any code doesn’t need typu of debug.
- destroyWindow is used in order to terminate the program if we encountered a bug inside the program.
- blur is to blur out the recently used threshold content so we can utilize the next frame the moment we are going to iterate capturing the next frame.

Figure 2.2: multiple items for object detection [1]

2.1.4 Conclusion

If you look at the image above us, we have an example of multiple objects being detected from the camera. Since the object are not moving, there not given accurate outlines since the frame by frame comparison, nothing has changed so we have a constant value of the outlines. In the next chapter I will be discussing the topics how to apply this concepts to find a person’s face instead of an object.
3. Face Detection

3.1 OpenCV for Face Detection

In this chapter we are going to be building off the logic from the previous chapter into detecting people’s faces. What makes facial detection so powerful is that we can apply neural networking technology to identify who is the person in the picture. To start off with the main concepts for facial detection, we need to take a look at Haar Feature-based Cascade Classifiers that was proposed by Paul Viola and Michael Jones [2]. These two brilliant Computer Visionist found out this method can be used in machine learning technology to detect positive (bright) and negative (dark) in object.

Figure 3.1: Image used for Edge features [2]

The image above is one of the three methods in Haar Feature-based Cascade Classifiers to find a persons face. The concept is utilized to find a persons face by locating the most negative region of the face which are the eyes. We have millions (depending on the pixel density of the picture) of these edge algorithms to compute someones face where most of the face will be positive values.
3.1.1 The code

Make sure to include the proper OpenCV Libraries

```c
void doMosaic(IplImage* in, int x, int y,
    int width, int height, int size);

int main (int argc, char **argv)
{
    int i, c;
    IplImage *src_img = 0, *src_gray = 0;
    const char *cascade_name = 
        "/opt/local/share/opencv/haarcascades/haarcascade_frontalface_default.xml";
    CvHaarClassifierCascade *cascade = 0;
    CvMemStorage *storage = 0;
    CvSeq *faces;

    cascade = (CvHaarClassifierCascade *) cvLoad (cascade_name, 0, 0, 0);
    cvNamedWindow ("Capture", CV_WINDOW_AUTOSIZE);
    CvCapture *capture = cvCreateCameraCapture(0);
    assert(capture != NULL);

    while (1) {
        src_img = cvQueryFrame (capture);
        src_gray = cvCreateImage (cvGetSize(src_img), IPL_DEPTH_8U, 1);

        storage = cvCreateMemStorage (0);
        cvClearMemStorage (storage);
        cvCvtColor (src_img, src_gray, CV_BGR2GRAY);
        cvEqualizeHist (src_gray, src_gray);

        faces = cvHaarDetectObjects (src_gray, cascade, storage,
            1.11, 4, 0, cvSize (40, 40));
        for (i = 0; i < (faces ? faces->total : 0); i++) {
            CvRect *r = (CvRect *) cvGetSeqElem (faces, i);
            doMosaic(src_img, r->x, r->y, r->width, r->height, 20);
        }

        cvShowImage("Capture", src_img);
        cvReleaseImage(&src_gray);
        c = cvWaitKey (2);
        if (c == '\x1b')
            break;
    }

    cvReleaseCapture (&capture);
    cvDestroyWindow ("Capture");
    return 0;
}
```
3.1 OpenCV for Face Detection

3.1.2 Analyze

Just like in the previous chapter for Feature Detection, this coding was open sourced from Github [3]. The way the code works we have ourself a data structure called IplImage which is acronyms for "Intel Image Processing Library". It is an image formatting add-on feature for machine learning [4]. In other words, use the data structure to display animation effects to any object, and or the entire picture, we can detect on the screen. Once we initialize a variable to contain the data structure for IplImage, we can assign a new frame value to the variable every time we have a while loop frame iteration just like in feature detection. Feature detection didn’t require to have a IplImage data structure to capture a frame because it doesn’t require machine learning capabilities. We next need to
Chapter 3. Face Detection

worry about the next data structure thats going to handle the size of the edge data structure mentioned in the beginning of this chapter [5]. There’s a data structure called CvSize that’ll provided the proper attributes to create our own edge detection algorithm. When taking a frame, we need to capture the frame on a grey color format to retrieve accurate threshold values for the persons face we plan on recognizing. Once we recieved the threshold image data, we need to notify the computer (flag variable) we detected a face on the camera with provided arguments. OpenCV has a built in library data structure called cvHaarDetectObjects [5] which is used to return a flag whether if the computer detected the persons face through a edge detection algorithm. There is another data structure called CvRect that’ll store the coordinates of the persons face. If you know any smart phone app with facial recognizing app like snap chat, the moment a face is recognized, there is a complex outline of your face, the code provided in OpenCV outlines of a persons face with a rectangular shape so it’s not accurate.

3.1.3 Conclusion

Figure 3.2: Facial Detection expressed in rectangular coordinates [3]

If you look at the picture above me, we would expext a couple edge detections displayed in rectabgular colors [3]. Each Color is represented by a type of edge detection. For example, both respected eyes that was detected display the color green. Now that we have facial recognition, we have to implement the last feature in a situation if the head were to move away from the center of the camera so the Drone can constantly follow the persons face.
4. Obstacle Detection

4.1 OpenCV for Obstacle Detection

If you remember in Chapter 2, our job is to detect moving object. In Chapter 3 we needed to have facial recognition...Now we're combining both concepts into one project. Most of the code has been provided from both C++ class so we need to make the logic (already given from a git repository [6]).

![Diagram](image)

Figure 4.1: Object Detection example [7]

If you look at the diagram above, we have a self-driving car that has a range both a Protective Field and Warning Field. So the person's face will need to be in the protective field when getting close to the camera implanted onto the drone. Once the drone locates the person's face, the camera needs to center its position aligned with the face. Afterwards, the feedback control systems, which by the way is material out of my range to explain, gives the drone intelligence to follow the person's face.
4.1.1 The code

Include the proper opencv libraries for this code.

```cpp
using namespace std;
using namespace cv;

void detectAndDisplay( Mat frame );

String face_cascade_name = "haarcascade_frontalface_alt.xml";
String eyes_cascade_name = "haarcascade_eye_tree_eyeglasses.xml";
CascadeClassifier face_cascade;
CascadeClassifier eyes_cascade;
String window_name = "Capture - Face detection";

int main( void )
{
  VideoCapture capture;
  Mat frame;

  if( !face_cascade.load( face_cascade_name ) )
  { printf("--(!)Error loading face cascade\n"); return -1; };
  if( !eyes_cascade.load( eyes_cascade_name ) )
  { printf("--(!)Error loading eyes cascade\n"); return -1; };

  capture.open( -1 );
  if ( ! capture.isOpened() ) { printf("--(!)Error opening video capture\n"); return -1; }

  while ( capture.read(frame) )
  {
    if( frame.empty() )
    {
      printf(" --(!) No captured frame -- Break!\n");
      break;
    }
    detectAndDisplay( frame );
    int c = waitKey(10);
    if( (char)c == 27 ) { break; } // escape
  }

  return 0;
}

void detectAndDisplay( Mat frame )
{
  std::vector<Rect> faces;
  Mat frame_gray;

  cvtColor( frame, frame_gray, COLOR_BGR2GRAY );
equalizeHist( frame_gray, frame_gray );

  face_cascade.detectMultiScale( frame_gray, faces,
```
4.1 OpenCV for Obstacle Detection

1.1, 2, 0 | CASCADE_SCALE_IMAGE, Size(30, 30) |

for ( size_t i = 0; i < faces.size(); i++ )
{
    Point center( faces[i].x + faces[i].width/2,
    faces[i].y + faces[i].height/2 );
    ellipse( frame, center, Size( faces[i].width/2,
    faces[i].height/2 ), 0, 0, 360,
    Scalar( 255, 0, 255 ), 4, 8, 0 );

    Mat faceROI = frame_gray( faces[i] );
    std::vector<Rect> eyes;
    eyes_cascade.detectMultiScale( faceROI, eyes,
    1.1, 2, 0 | CASCADE_SCALE_IMAGE, Size(30, 30) );

    for ( size_t j = 0; j < eyes.size(); j++ )
    {
        Point eye_center( faces[i].x + eyes[j].x +
        eyes[j].width/2, faces[i].y + eyes[j].y +
        eyes[j].height/2 );
        int radius = cvRound( (eyes[j].width +
        eyes[j].height)*0.25 );
        circle( frame, eye_center, radius,
        Scalar( 255, 0, 0 ), 4, 8, 0 );
    }
}

imshow( window_name, frame );

4.1.2 Analyze

This coding is open sourced from Github [6]. Inside the code, we need to have a conditional variable
in order to have the structure CvCapture which is the data structure to capture live content instead of
a recorded video. The previous two chapters had a data structure to handle recorded videos which
requires less processing power. So for the fun part of this assignment is to take the absolute difference
within two frame comparisons that are stored in an array called faces. If we have proof from the
absolute difference that the face has moved away from the camera, not implemented yet but we
would have an interrupt service routine to communicate with the Drone’s embedded microcotroller
to alter it’s flight direction to have the camera centered onto the persons face. Right now the code
can locate the face anywhere on the camera, communicating the Drone through a interrupt service
routine is the next challenge.

4.1.3 Conclusion

This is where my research has reached an end point and requires more help with other colleges in
Graduate school since the undergraduate Computer Engineering doesn’t cover classes in Computer
Vision Technology.
5. Conclusion and Reference

5.1 Conclusion
This research has been a transformative opportunity to work with Computer Vision Technology as an undergraduate. The research still requires more commitment either over the summer or for the next school year.

5.2 Reference

feature detection

facial recognition

IplImage

CvSize

Object Detection

image feature

May the force be with you, quoted by Obi Wan Kenobi