WINNING THE DRONE RACE

FINAL PROJECT
ADVANCED MECHATRONICS

MITRA VARUN ANAND
SAMANTH

UNDER
Prof VIKRAM KAPILA
INSPIRATION
DEVELOPMENT

• An Augmented Reality based Trainer module to practice accurate control of the drone using Raspberry Pi.
• An obstacle avoiding mechanism using Ping sensors and Arduino Uno to navigate the drone in closed spaces.
• Quick take off mechanism using EZ-Builder and OpenCV.
• Autonomous control of the drone with NodeJS
• Color following drone with EZ Builder.
AUGMENTED REALITY TRAINING MODULE

• Using OpenCV and Raspberry Pi to create a training module for practicing accurate movements.
• Raspberry pi camera continuously tries to track a red marker on top of the drone.
• When the radius of the AR circle matches with radius of circle on top of the drone, the user gets 1 point.
CREATION OF VIRTUAL TRACK

- We used ‘addweighted’ function of openCV to achieve
- `cv2.addWeighted(src1, alpha, src2, beta, gamma[, dst[, dtype]]) → dst`\[\]

**Parameters:**
- `src1` – first input array.
- `alpha` – weight of the first array elements.
- `src2` – second input array of the same size and channel number as `src1`.
- `beta` – weight of the second array elements.
- `dst` – output array that has the same size and number of channels as the input arrays.
- `gamma` – scalar added to each sum.
- `dtype` – optional depth of the output array; when both input arrays have the same depth, `dtype` can be set to -1, which will be equivalent to `src1.depth()`.

\[
dst = src1*alpha + src2*beta + gamma;
\]
CONTINUOUS TRACKING OF DRONE
COLLISION AVOIDANCE

- Telnet to 192.168.1.1
- Copy the AR Drone node.js file (converted from official AR Drone api)
- Connect the circuit as shown.
- All set to go!
PRINCIPLE

FRONT sensor

Trigger sends ultrasonic wave (Pulse = HIGH)

Echo wave received from obstacle (Pulse = HIGH)

Time delay between Trigger and Echo calculated

Hence Distance(cm) is calculated

Is the distance greater than 70cm?

TRUE

FALSE

Print 'GO FORWARD'

Print 'STOP'

<table>
<thead>
<tr>
<th>Serial Print</th>
<th>Client API Command</th>
<th>Drone Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>client.land()</td>
<td>LAND</td>
</tr>
<tr>
<td>R</td>
<td>client.right(speed)</td>
<td>RIGHT, maintains forward heading</td>
</tr>
<tr>
<td>L</td>
<td>client.left(speed)</td>
<td>LEFT, maintains forward heading</td>
</tr>
<tr>
<td>F</td>
<td>client.front(speed)</td>
<td>Goes FORWARD</td>
</tr>
<tr>
<td>P</td>
<td>client.right(speed)</td>
<td>RIGHT (Front blocked)</td>
</tr>
<tr>
<td>Q</td>
<td>client.left(speed)</td>
<td>LEFT (Front blocked)</td>
</tr>
<tr>
<td>S</td>
<td>client.stop()</td>
<td>STOP</td>
</tr>
</tbody>
</table>
CODE

#define FRONT_TRIG 13 //FRONT
#define FRONT_ECHO 12 //FRONT
#define LEFT_TRIG 11 //LEFT
#define LEFT_ECHO 10 //LEFT
#define RIGHT_TRIG 9 //RIGHT
#define RIGHT_ECHO 8 //RIGHT
#define TOP_TRIG 7 //TOP
#define TOP_ECHO 6 //TOP

void setup() {
  Serial.begin (9600);
pinMode(FRONT_TRIG, OUTPUT);
pinMode(FRONT_ECHO, INPUT);
pinMode(LEFT_TRIG, OUTPUT);
pinMode(LEFT_ECHO, INPUT);
pinMode(RIGHT_TRIG, OUTPUT);
pinMode(RIGHT_ECHO, INPUT);
pinMode(TOP_TRIG, OUTPUT);
pinMode(TOP_ECHO, INPUT);
}

void loop() {
  long duration, FRONT, RIGHT, LEFT, TOP; // Duration used to calculate distance of an object from each sensor
  digitalWrite(TOP_TRIG, LOW); // LOW triggers to ensure no interference from incoming signals, before triggering HIGH
  delayMicroseconds(2);
digitalWrite(TOP_TRIG, HIGH); // Send out ultrasonic wave
  delayMicroseconds(10);
  // Delay allows for ample time to receive the echo signal from object
  digitalWrite(TOP_TRIG, LOW);
  duration = pulseIn(TOP_ECHO, HIGH); // Calculates time taken to receive signal from reflected signal, pulse is HIGH when signal is received
  TOP = (duration/2) / 29.1; // Calculates distances using the time calculated above and the speed of sound (340m/s)
digitalWrite(TOP_TRIG, LOW);
  delayMicroseconds(2);
digitalWrite(TOP_TRIG, HIGH);
  delayMicroseconds(10);
digitalWrite(TOP_TRIG, LOW);
  duration = pulseIn(TOP_ECHO, HIGH);
  FRONT = (duration/2) / 29.1;
digitalWrite(RIGHT_TRIG, LOW);
  delayMicroseconds(2);
digitalWrite(RIGHT_TRIG, HIGH);
  delayMicroseconds(10);
digitalWrite(RIGHT_TRIG, LOW);
  duration = pulseIn(RIGHT_ECHO, HIGH);
  RIGHT = (duration/2) / 29.1;
digitalWrite(LEFT_TRIG, LOW);
  delayMicroseconds(2);
digitalWrite(LEFT_TRIG, HIGH);
  delayMicroseconds(10);
digitalWrite(LEFT_TRIG, LOW);
  duration = pulseIn(LEFT_ECHO, HIGH);
  LEFT = (duration/2) / 29.1;
}

Time Swag
duration = pulseIn(RIGHT_ECHO, HIGH);
RIGHT = (duration/2) / 29.1;
digitalWrite(LEFT_TRIG, LOW);
delayMicroseconds(2);
digitalWrite(LEFT_TRIG, HIGH);
delayMicroseconds(10);
digitalWrite(LEFT_TRIG, LOW);
duration = pulseIn(LEFT_ECHO, HIGH);
LEFT = (duration/2) / 29.1;

if (TOP < 60) {
    Serial.println("\n");
}
else{
    if (RIGHT < 90 && (FRONT > 80)) {
        Serial.println("L\n");
    }
    if (LEFT < 90 && (FRONT > 80)) {
        Serial.println("R\n");
    }
    if (FRONT > 90) {
        Serial.println("F\n");
    }
    if (FRONT >= 7 && FRONT <= 50) {
        Serial.println("D\n");
    }
    else{
        if (LEFT < 90 && (FRONT >= 7 && FRONT <= 59)) {
            Serial.println("F\n");
        }
        if (LEFT > RIGHT && (FRONT >= 7 && FRONT <= 59)) {
            Serial.println("D\n");
        }
    }
}
VIDEO
QUICK FIRE TAKEOFF

• Humans are slower to react to a green signal, delaying the take off once the race starts.

• Our mechanism makes use of color detection to immediately start the take off process, better yet, give an initial push to kick-start the race.
CODE
<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>gyro_offset_th_p</td>
<td>4.00000000e-00</td>
</tr>
<tr>
<td>gyro_offset_th_r</td>
<td>5.00000000e-01</td>
</tr>
<tr>
<td>pwm_ref_gyros</td>
<td>500</td>
</tr>
<tr>
<td>osctun_value</td>
<td>62</td>
</tr>
<tr>
<td>osctun_load</td>
<td>TRUE</td>
</tr>
<tr>
<td>altitude_max</td>
<td>5000</td>
</tr>
<tr>
<td>altitude_min</td>
<td>50</td>
</tr>
<tr>
<td>outdoor</td>
<td>FALSE</td>
</tr>
<tr>
<td>flight_without_shell</td>
<td>TRUE</td>
</tr>
<tr>
<td>autonomous_flight</td>
<td>FALSE</td>
</tr>
<tr>
<td>control_level</td>
<td>0</td>
</tr>
<tr>
<td>euler_angle_max</td>
<td>2.50000000e-01</td>
</tr>
<tr>
<td>control1_pitch_tilt</td>
<td>3.40000000e-01</td>
</tr>
<tr>
<td>control3_yaw</td>
<td>1.00000000e+03</td>
</tr>
<tr>
<td>control_yaw</td>
<td>1.00000000e+00</td>
</tr>
<tr>
<td>manual_trim</td>
<td>FALSE</td>
</tr>
<tr>
<td>indoor_euler_angle_max</td>
<td>2.50000000e-01</td>
</tr>
<tr>
<td>indoor_control1_roll</td>
<td>1.00000000e+03</td>
</tr>
<tr>
<td>indoor_control1_yaw</td>
<td>3.00000000e+00</td>
</tr>
<tr>
<td>outdoor_euler_angle_max</td>
<td>3.40000000e-01</td>
</tr>
<tr>
<td>outdoor_control1_roll</td>
<td>1.00000000e+03</td>
</tr>
<tr>
<td>outdoor_control1_yaw</td>
<td>3.40000000e-01</td>
</tr>
<tr>
<td>[network]</td>
<td></td>
</tr>
<tr>
<td>ssid_single_player</td>
<td>ardrone2_106282</td>
</tr>
<tr>
<td>ssid_multi_player</td>
<td>ardrone2_106282</td>
</tr>
<tr>
<td>wifi_mode</td>
<td>0</td>
</tr>
<tr>
<td>ownr_mac</td>
<td>00:00:00:00:00:00:00:00:00</td>
</tr>
<tr>
<td>[pic]</td>
<td></td>
</tr>
<tr>
<td>ultrasound_freq</td>
<td>0</td>
</tr>
<tr>
<td>ultrasound_watchdog</td>
<td>3</td>
</tr>
<tr>
<td>pic_version</td>
<td>184877090</td>
</tr>
<tr>
<td>[video]</td>
<td></td>
</tr>
<tr>
<td>camfra_fps</td>
<td>30</td>
</tr>
<tr>
<td>camfra_buffers</td>
<td>2</td>
</tr>
<tr>
<td>num_trackers</td>
<td>12</td>
</tr>
<tr>
<td>video_on_usb</td>
<td>TRUE</td>
</tr>
<tr>
<td>video_file_index</td>
<td>1</td>
</tr>
<tr>
<td>codec_fps</td>
<td>30</td>
</tr>
</tbody>
</table>
enemy_colors = 1
enemy_without_shell = 0
groundstripe_colors = 16
detect_type = 3
detections_select_h = 0
detections_select_v_hsync = 0
detections_select_v = 0
d[syslog]
output = 7
max_size = 102400
nb_files = 5
d[custom]
application_desc = Default application configuration
profile_desc = Default profile configuration
session_desc = Default session configuration
d[userbox]
d[gps]
latitude = 5.00000000000000e+02
longitude = 5.00000000000000e+02
altitude = 0.00000000000000e+00
accuracy = 0.00000000000000e+00
d[flightplan]
default_validation_radius = 3.0000000000e+00
default_validation_time = 1.0000000000e+00
max_distance_from_takeoff = 1000
gps_ip = 2
video_step_delay = 10
low_battery_go_home = FALSE
automatic_heading = TRUE
com_lost_action_delay = 0
altitude_go_home = 0.0000000000e+00
mavlink_is_roll_left = x
mavlink_is_roll_right = x
mavlink_is_pitch_front = y
mavlink_is_pitch_back = y
mavlink_is_yaw_left = 4
```
4

129  mavlink_js_pitch_back = y-
130  mavlink_js_yaw_left = 4
131  mavlink_js_yaw_right = 5
132  mavlink_js_go_up = 0
133  mavlink_js_go_down = 1
134  mavlink_js_inc_gains = 6
135  mavlink_js_dec_gains = 7
136  mavlink_js_select = 8
137  mavlink_js_start = 9
[reserve]
from(ardrone_library.js) load:

arDrone.createServer([]ip))

143  Takeoff()
144  Sleep()
145  rollright()
146  rollleft()
147  Land

148  Mat image, resized, gray;
149  cout""opening camera...""<<endl;
150  if (!Camera.open()) (cerr<<"Error opening the
151  " Camera""<<endl;return -1;
152  //set capture properties
153  sleep();
154  Camera.set (CV_CAP_PROP_FRAME_WIDTH, 1000);
155  Camera.set (CV_CAP_PROP_FRAME_HEIGHT, 960);
156  Camera.set (CV_CAP_PROP_BRIGHTNESS, 50);
157  Camera.set (CV_CAP_PROP_CONTRAST, 50);
158  namedWindow ("Camera Video", CV_WINDOW_AUTOSIZE );
159  Camera.grab();
160  Camera.retrieve ( image);
161  if (inRange(processed, Scalar(160, 20, 20), Scalar(180, 255, 255), processed));
162  drawContours(image, contours, -1, Scalar(255,0,0), 3);
163  Takeoff()
164  Sleep(1500)
165  rollright(1000)
166  Land
167  else()
168  CloseControl()
```
PROCESS

• Connect to ARDrone network through WiFi.
• Open the Script and load on EZBuilder.
• Click connect.
• Whenever the drone detects the color, it will take-off.
Autonomous Control of Drone

• Install node.js.
• Code:
  arDrone = require('drone');
  client = arDrone.createClient();

  client.takeoff();

  client
    .after(2000, function() {
      this.up(1);
      /*})
    .after(2000, function() {
      this.animate('turnaround',500);
    })
CODE 2

.after(5000, function() {
    this.front(1.0);
})
.after(2000, function() {
    this.clockwise(0.5);
})
.after(5000, function() {
    this.back(0.8);
}))*/
.after(200, function() {
    this.land();
});
NEEDS IMPROVEMENT/FIXING

• Standalone system incorporating all the features to control with a same controller.
• Fix Green light issue.
• Mods to counter-act forces caused by additional components for stability.
• FPV glasses-stream to make it more similar to the actual race experience.
• A full fledged gaming app incorporating the training module and using custom made Augmented Reality tracks to simulate racing environment.