

drone\facetracking_pid.py

```
1 import threading      #pid最終決定 プログラムfacetracking_pid.py
2 import socket
3 import time
4 import cv2
5 import numpy as np
6
7
8
9 host = '0.0.0.0'
10 port = 8889
11 locaddr = (host, port)
12
13
14 # Create a UDmport tello
15
16 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
17
18 tello_address = ('192.168.10.1', 8889)
19
20 sock.bind(locaddr)
21
22 def recv():
23     while True:
24         try:
25             data, server = sock.recvfrom(1518)
26             print(data.decode(encoding="utf-8"))
27         except Exception:
28             print('\nExit . . . RECV\n')
29             break
30 #recvThread create
31 recvThread = threading.Thread(target=recv)
32 recvThread.start()
33
34
35
36 sock.sendto(b'command', tello_address)
37 print('command ok')
38 time.sleep(0.5)
39 sock.sendto(b'streamon', tello_address)
40 print('stream on')
41 time.sleep(1)
42 sock.sendto(b'takeoff', tello_address)
43 time.sleep(2)
44
45 # #cmd = 'rc {} {} {} {}' rc{roll}{pitch}{throttle}{yaw}
46 rc_command = 'rc 0, 0, 20, 0'
47 sock.sendto(rc_command.encode('utf-8'), tello_address)
48 time.sleep(2)
49
50
51 LOCAL_IP = '192.168.10.1'
52 LOCAL_PORT = '11111'
53 addr = ("udp://%s:%s?overrun_nonfatal=1&fifo_size=5000000" % ('192.168.10.1', '11111'))
54
55 cap = cv2.VideoCapture(addr)
56 #cap.set(cv2.CAP_PROP_BUFFERSIZE, 2)
57
58 print('start cap')
59
60
61
62
63 pid = [0.4, 0, 0.4]
64 pError = 0
```

```

65 w,h = 480, 360
66
67
68 def findFace(img):
69     global face_center_y
70
71     face_cascade = cv2.CascadeClassifier(cv2.data.harcascades + 'haarcascade_frontalface_default.xml')
72
73
74     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
75     faces = face_cascade.detectMultiScale(gray, 1.3, 5)
76
77
78     FaceList_center = []
79     FaceList_area = []
80     face_center_y = 0
81
82
83     for (x, y, w, h) in faces:
84
85         cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
86         face_center_x = int(x + (w / 2))          #顔の中心x軸
87         face_center_y = int(y + (h) / 2)
88
89         face_area =int(w * h)          #面積
90         cv2.circle(img,(face_center_x, face_center_y), 10, (0,0,225), cv2.FILLED) #中心point
91         cv2.putText(img, f'facecenter:({face_center_x}, {face_center_y})', org=(50, 50),
fontFace=cv2.FONT_HERSHEY_SIMPLEX, fontScale=0.5, color=(204, 0, 255), thickness=1, lineType=cv2.LINE_4)
92         cv2.putText(img, f'facearea: ' + str(face_area), org=(30, 30), fontFace=cv2.FONT_HERSHEY_SIMPLEX,
fontScale=0.5, color=(204, 0, 255), thickness=1, lineType=cv2.LINE_4)
93
94         FaceList_center.append([face_center_x, face_center_y])
95         FaceList_area.append(face_area)
96
97         print(FaceList_center, FaceList_area)
98
99
100
101     if len(FaceList_area) != 0:
102         i = FaceList_area.index(max(FaceList_area))
103         return img, [FaceList_center[i], FaceList_area[i]]
104     else:
105         return img, [[0, 0], 0]
106
107
108 a = b = c = d =0          # #cmd = 'rc {a} {b} {c} {d}' rc{roll}{pitch}{throttle}{yaw}
109 def TrackFace( info, w, pid, pError): # x軸に対するPI制御
110     global b, d
111     # pError = error
112     # pid_output = d
113     area = info[1]
114     x, y = info[0]
115     b = 0 #bを初期化
116     #yaw(左右運動)算出用
117     error = x - w/2
118     d = pid[0] * error + pid[2] * (error - pError) #PID制御のうちPI制御
119     d = int(np.clip(d, -100, 100)) #RCコマンドのリミッター
120     print('pError: ' + str(pError))
121     print('error' + str(error))
122
123     #pitch(前後運動)算出用
124     if area > 6200 and area < 6800:
125         b = 0
126     elif area > 6800:
127         b = -20
128     elif area < 6200 and area != 0:
129         b = 20

```

```
130     if x == 0:
131         d = 0
132         error = 0
133
134     print(b, d)
135     return error
136
137
138 def trottle(face_center_y):
139     global c
140
141     c = 0
142     print(face_center_y)
143     if face_center_y > 180 and face_center_y < 40:
144         c = 0
145     elif 280 > face_center_y > 180 + 10 :    #frame h:360の1/2=180 顔の下降 (座標が大きくなる)
146         c = -20
147     elif 50 < face_center_y < 110 :
148         c = 20
149 #     c = int(np.clip(c, -100, 100))
150
151     print('throttle: ' + str(c))
152     return
153
154
155
156 while True:
157     for i in range(5):
158         ret, frame = cap.read()
159         if frame is None or frame.size == 0:
160             continue
161         cv2.circle(frame,(int(480), int(360)), 5, (0,225,0), cv2.FILLED) #中心point
162         img = cv2.resize(frame, (w, h))
163         img, info = findFace(img)
164         pError = TrackFace(info, w, pid, pError)
165         face_center_y = trottle(face_center_y)
166         rc_command = "rc {0} {1} {2} {3}".format(a, b, c, d)    # #cmd = 'rc {a} {b} {c} {d}' rc{roll}{pitch}
167 {throttle}{yaw}
168         sock.sendto(rc_command.encode('utf-8'),tello_address)
169         print("Sending RC command with values:", a, b, c, d)
170         cv2.putText(img, f' roll: '+ str(a) + f' pitch: '+ str(b) + f' throttle: ' + str(c) + f' yaw: ' + str(d),
171 org=(100, 100), fontFace=cv2.FONT_HERSHEY_SIMPLEX, fontScale=0.5, color=(204, 0, 255), thickness=1,
172 lineType=cv2.LINE_4)
173
174         cv2.imshow('frame', img)
175
176         if cv2.waitKey(1) & 0xFF == 27 :
177 #
178         sock.sendto(b'land', tello_address)
179         cap.release()
180         cv2.destroyAllWindows()
181         sock.close()
182         break
```