Universidad Autónoma de San Luis Potosí
Facultad de ingeniería
Tratamiento de Imágenes
Practica 12
Nombre Práctica: Seguimiento con Drones
Nombre del Alumno: Manuel Ramírez Galván

Fecha: 29/04/2025

Procedimiento

12.1. Realizar un programa en Python con el cual se puedan identificar objetos y que al mover estos objetos la cámara del dron identifique hacia donde tiene que desplazarse sin hacer el desplazamiento.

```python
#Librerias
from djitellopy import tello
import numpy as np
import cv2

#Conneccion al drone
drone = tello.Tello()
drone.connect()

#Mustra la bateria en consola
print(drone.get_battery())
drone.streamon()

#variables
frameWidth = 640
frameHeight = 480
cap =
drone.get_frame_read().frame
cap.set(3, frameWidth)
cap.set(4, frameHeight)

deadZone=100
global imgContour
```

Imagen 1.- Librerías y Conexión del Dron

```python
#Funcion para trackbar
def empty(a):
    pass

#Creacion de Trackbar
cv2.namedWindow("HSV")
cv2.resizeWindow("HSV",640,240)
cv2.createTrackbar("HUE Min","HSV",19,300,empty)
cv2.createTrackbar("HUE Max","HSV",35,300,empty)
cv2.createTrackbar("SAT Min","HSV",107,255,empty)
cv2.createTrackbar("SAT Max","HSV",255,255,empty)
cv2.createTrackbar("VALUE Min","HSV",89,255,empty)
cv2.createTrackbar("VALUE Max","HSV",255,255,empty)

cv2.namedWindow("Parameters")
cv2.resizeWindow("Parameters",640,240)
cv2.createTrackbar("Threshold1","Parameters",166,255,empty
cv2.createTrackbar("Threshold2","Parameters",171,255,empty
cv2.createTrackbar("Area","Parameters",3750,30000,empty)
```

Imagen 2.- Creación del Trackbar

```python
#Verificacion de imagenes
def stackImages(scale,imgArray):
    rows = len(imgArray)
    cols = len(imgArray[0])
    rowsAvailable = isinstance(imgArray[0], list)
    width = imgArray[0][0].shape[1]
    height = imgArray[0][0].shape[0]
    if rowsAvailable:
        for x in range ( 0, rows):
            for y in range(0, cols):
                if imgArray[x][y].shape[:2] == imgArray[0][0].shape [:2]:
                    imgArray[x][y] = cv2.resize(imgArray[x][y], (0, 0), None, scale, scale)
                else:
                    imgArray[x][y] = cv2.resize(imgArray[x][y], (imgArray[0][0].shape[1], imgArray[0]
[0].shape[0]), None, scale, scale)
                if len(imgArray[x][y].shape) == 2: imgArray[x][y]= cv2.cvtColor( imgArray[x][y],
cv2.COLOR_GRAY2BGR)
        imageBlank = np.zeros((height, width, 3), np.uint8)
        hor = [imageBlank]*rows
        hor_con = [imageBlank]*rows
        for x in range(0, rows):
            hor[x] = np.hstack(imgArray[x])
        ver = np.vstack(hor)
    else:
        for x in range(0, rows):
            if imgArray[x].shape[:2] == imgArray[0].shape[:2]:
                imgArray[x] = cv2.resize(imgArray[x], (0, 0), None, scale, scale)
            else:
                imgArray[x] = cv2.resize(imgArray[x], (imgArray[0].shape[1], imgArray[0].shape[0]),
None,scale, scale)
            if len(imgArray[x].shape) == 2: imgArray[x] = cv2.cvtColor(imgArray[x], cv2.COLOR_GRAY2BGR)
        hor= np.hstack(imgArray)
        ver = hor
    return ver
```

Imagen 3.- Código Ejercicio 3

```python
#Busqueda de contornos
def getContours(img,imgContour):

    contours, hierarchy = cv2.findContours(img, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
    for cnt in contours:
        area = cv2.contourArea(cnt)
        areaMin = cv2.getTrackbarPos("Area", "Parameters")
        if area > areaMin:
            cv2.drawContours(imgContour, cnt, -1, (255, 0, 255), 7)
            peri = cv2.arcLength(cnt, True)
            approx = cv2.approxPolyDP(cnt, 0.02 * peri, True)
            print(len(approx))
            x , y , w, h = cv2.boundingRect(approx)

            cx = int(x + (w / 2))
            cy = int(y + (h / 2))

            if (cx <int(frameWidth/2)-deadZone):
                cv2.putText(imgContour, " GO LEFT " , (20, 50), cv2.FONT_HERSHEY_COMPLEX,1,(0, 0, 255),
3)
                cv2.rectangle(imgContour,(0,int(frameHeight/2-deadZone)),(int(frameWidth/2)-
deadZone,int(frameHeight/2)+deadZone),(0,0,255),cv2.FILLED)
            elif (cx > int(frameWidth / 2) + deadZone):
                cv2.putText(imgContour, " GO RIGHT ", (20, 50), cv2.FONT_HERSHEY_COMPLEX,1,(0, 0, 255),
3)
                cv2.rectangle(imgContour,(int(frameWidth/2+deadZone),int(frameHeight/2-deadZone)),
(frameWidth,int(frameHeight/2)+deadZone),(0,0,255),cv2.FILLED)
            elif (cy < int(frameHeight / 2) - deadZone):
                cv2.putText(imgContour, " GO UP ", (20, 50), cv2.FONT_HERSHEY_COMPLEX,1,(0, 0, 255), 3)
                cv2.rectangle(imgContour,(int(frameWidth/2-deadZone),0),
(int(frameWidth/2+deadZone),int(frameHeight/2)-deadZone),(0,0,255),cv2.FILLED)
            elif (cy > int(frameHeight / 2) + deadZone):
                cv2.putText(imgContour, " GO DOWN ", (20, 50), cv2.FONT_HERSHEY_COMPLEX, 1,(0, 0, 255),
3)
                cv2.rectangle(imgContour,(int(frameWidth/2-deadZone),int(frameHeight/2)+deadZone),
(int(frameWidth/2+deadZone),frameHeight),(0,0,255),cv2.FILLED)

            cv2.line(imgContour, (int(frameWidth/2),int(frameHeight/2)), (cx,cy),
                     (0, 0, 255), 3)
            cv2.rectangle(imgContour, (x , y ), (x + w , y + h ), (0, 255, 0), 5)

            cv2.putText(imgContour, "Points: " + str(len(approx)), (x + w + 20, y + 20),
cv2.FONT_HERSHEY_COMPLEX, .7,
                        (0, 255, 0), 2)
            cv2.putText(imgContour, "Area: " + str(int(area)), (x + w + 20, y + 45),
cv2.FONT_HERSHEY_COMPLEX, 0.7,
                        (0, 255, 0), 2)
            cv2.putText(imgContour, " " + str(int(x))+ " "+str(int(y)), (x - 20, y- 45),
cv2.FONT_HERSHEY_COMPLEX, 0.7,
                        (0, 255, 0), 2)
```

Imagen 3.- Búsqueda de Contornos

```python
#Funcion para dibujar cuadrantes
def display(img):
    cv2.line(img,(int(frameWidth/2)-deadZone,0),(int(frameWidth/2)-deadZone,frameHeight),(255,255,0),3)
    cv2.line(img,(int(frameWidth/2)+deadZone,0),(int(frameWidth/2)+deadZone,frameHeight),(255,255,0),3)

    cv2.circle(img,(int(frameWidth/2),int(frameHeight/2)),5,(0,0,255),5)
    cv2.line(img, (0,int(frameHeight / 2) - deadZone), (frameWidth,int(frameHeight / 2) - deadZone),
(255, 255, 0), 3)
    cv2.line(img, (0, int(frameHeight / 2) + deadZone), (frameWidth, int(frameHeight / 2) + deadZone),
(255, 255, 0), 3)
```

Imagen 4.- Dibujar Cuadrantes

```
while True:
    _, img = cap.read()
    if img is None:
        break
    imgContour = img.copy()
    imgHsv = cv2.cvtColor(img,cv2.COLOR_BGR2HSV)

    h_min = cv2.getTrackbarPos("HUE Min","HSV")
    h_max = cv2.getTrackbarPos("HUE Max", "HSV")
    s_min = cv2.getTrackbarPos("SAT Min", "HSV")
    s_max = cv2.getTrackbarPos("SAT Max", "HSV")
    v_min = cv2.getTrackbarPos("VALUE Min", "HSV")
    v_max = cv2.getTrackbarPos("VALUE Max", "HSV")

    lower = np.array([h_min,s_min,v_min])
    upper = np.array([h_max,s_max,v_max])
    mask = cv2.inRange(imgHsv,lower,upper)
    result = cv2.bitwise_and(img,img, mask = mask)
    mask = cv2.cvtColor(mask, cv2.COLOR_GRAY2BGR)

    imgBlur = cv2.GaussianBlur(result, (7, 7), 1)
    imgGray = cv2.cvtColor(imgBlur, cv2.COLOR_BGR2GRAY)
    threshold1 = cv2.getTrackbarPos("Threshold1",
"Parameters")
    threshold2 = cv2.getTrackbarPos("Threshold2",
"Parameters")
    imgCanny = cv2.Canny(imgGray, threshold1, threshold2)
    kernel = np.ones((5, 5))
    imgDil = cv2.dilate(imgCanny, kernel, iterations=1)
    getContours(imgDil, imgContour)
    display(imgContour)

    stack = stackImages(0.7,([img,result],
[imgDil,imgContour]))
    cv2.cvtColor(stack, cv2.COLOR_BGR2RGB)

    cv2.imshow('Horizontal Stacking', stack)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        #drone.land()
        break
cap.release()
```

Imagen 4.- Dibujar Cuadrantes

Resultados

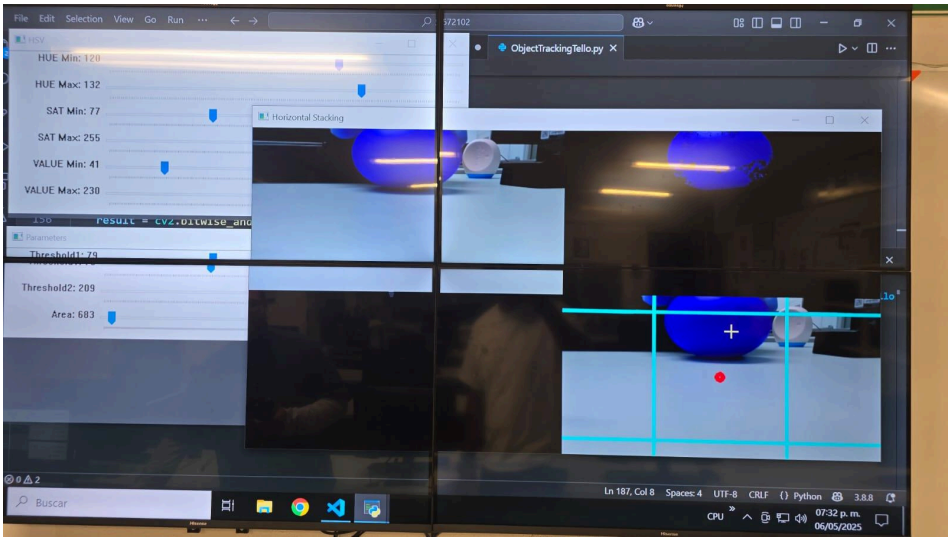

Imagen 5.- Funcionamiento y Detección con Dron

Comprensión

1. **¿Por qué es importante utilizar los trackbar a la hora de trabajar con drones?**

Permiten:

- Ajustar parámetros en tiempo real.
- Calibrar el sistema de visión del dron sin reiniciar el programa.
- Experimentar en vivo con distintos valores para mejorar la segmentación o detección de objetos

Esto es especialmente útil en drones porque:

- Las condiciones ambientales cambian constantemente.
- Se necesita precisión y respuesta rápida al procesar imágenes en vuelo.

2. **¿Para qué se utiliza el reconocimiento de objetos en drones?**

El reconocimiento de objetos permite a los drones identificar y localizar elementos específicos en su entorno visual. Esto es clave para tareas como:

- **Entrega de paquetes:** Identificar puntos de entrega, zonas seguras.
- **Agricultura de precisión:** Detectar cultivos, plagas o zonas afectadas.
- **Búsqueda y rescate:** Reconocer personas, vehículos o señales visuales.
- **Seguimiento inteligente:** Seguir a una persona u objeto sin intervención humana.
- **Inspección industrial:** Detectar torres, paneles, tuberías o daños en estructuras

3. **¿En qué otras aplicaciones se puede utilizar el tratamiento de imágenes en drones?**

- **Cartografía y topografía:** Crear mapas en 2D/3D, modelos de elevación, ortomosaicos.
- **Monitoreo ambiental:** Detección de cambios en vegetación, incendios forestales o cuerpos de agua.
- **Supervisión de obra:** Seguimiento visual del avance de construcción y detección de errores.
- **Producción audiovisual:** Aplicación de filtros, estabilización y encuadre automático.
- **Inspección con visión térmica:** Análisis de temperaturas en instalaciones eléctricas, techos o paneles solares.

Conclusiones

El seguimiento con drones permite que el dron identifique y siga objetos o personas de forma automática, facilitando tareas como seguridad, búsqueda y rescate, monitoreo y filmación aérea. Gracias al procesamiento en tiempo real y al ajuste de parámetros mediante herramientas como trackbars, se logra una mayor precisión y adaptabilidad ante cambios en el entorno.

En conjunto, el seguimiento convierte al dron en una herramienta eficiente, versátil y autónoma para diversas aplicaciones especializadas.