UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO

FACULTAD DE
INGENIERÍA

Feature Extraction en multicóptero con Raspberry Pi

Práctica Profesional Supervisada

Presentado por

Klara Soreil

en Diciembre 2022
para la carrera de Ingeniería en Mecatrónica

Supervisor: Dr. Ing. Rodrigo Gonzalez

Índice

1 Introducción

2 Entorno de trabajo

2.1 Configuración del entorno de trabajo

A continuacion se describe los principales pasos para configurar la Raspberry Pi B+ para implementar feature extraction con PiCamera y OpenCV.

```
sudo apt-get update && sudo apt-get upgrade
```

Es necesario verificar que la versión de python es 3 o superior.

```
python -V
```

Instalamos la versión 3 de pip

```
sudo apt-get install python3-pip
```

Instalación de un entorno virtual

```
sudo pip install virtualenv virtualenvwrapper
vim ~/.bashrc
```

Agregar las siguientes líneas al final del archivo :

```
export WORKON_HOME=$HOME/.virtualenvs
export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
source /usr/local/bin/virtualenvwrapper.s
```

Guardar el archivo y correr el siguiente comando para cargar los cambios :

```
source ~/.bashrc
```

Ahora creamos el entorno virtual en la carpeta 'src'

```
mkvirtualenv env -p python3
```

Usando PiCamera, hay que instalar el módulo picamera, y a continuacion el modulo para OpenCV en Python.

```
pip install "picamera[array]"
pip install opencv-python
```

2.2 Utilización del entorno de trabajo

Para empezar al encender la placa Raspberry :

Dejamos los usuario y contraseña por defecto.

- usuario : pi
- contraseña : raspberry

Acordarse que es necesario activar el entorno virtual.

```
cd src
workon env
```

Para intercambiar archivos con la Raspberry

- Mandar un archivo

```
scp filename.py pi@ipaddress:src/
```

En nuestro caso, reemplazar ipaddress por 192.168.23.198.

- Descargar una imagen

```
scp pi@ipaddress:src/image.jpg ./capture.jpg
```

3 Utilización básica de la PiCamera con OpenCV

Por razones practicas, se hizo pruebas de funcionamiento del codigo de Feature Extraction en la mayor parte con una camera USB. La puesta en marcha es un poco distinta con la camera de la Raspberry Pi. A continuacion son unos comandos básicos con algunos propios a la PiCamera.

- Initializar la cámara

```
cam = PiCamera()
```

- Empezar y terminar el video

```
cam.start_preview()
cam.stop_preview()
```

- Mejorar la calidad de la imagen

```
cam.resolution = (1280,720) #640,480
cam.contrast = 20
```

- Capturar una foto

```
cam.capture(rawCapture, format="bgr")
img = rawCapture.array
```

- Mostrar la imagen

```
cv2.imshow("Window Name", img)
```

4 Feature Extraction

4.1 Principio de funcionamiento

4.1.1 Keypoints y descriptores

Los puntos clave (keypoints) se utilizan para identificar las regiones clave de un objeto que se utilizan como base para posteriormente emparejarlo e identificarlo en una nueva imagen. El punto clave es independiente de las condiciones en la cual se toma la imagen (iluminación, ángulo, escala, fondo...). Un punto clave se calcula considerando un área de determinadas intensidades de píxeles a su alrededor. Se calculan utilizando varios algoritmos diferentes (SIFT, SURF, FAST, etc).

Después de detectar los puntos claves de la imagen, se proceda a definir los descriptores que se usan para comparar los distintos puntos claves. Vienen con formato de vectores y son independiente de la posición de los keypoints.

4.2 Métodos de extracción

4.2.1 SIFT (Scale-Invariant Feature Transform)

El algoritmo SIFT significa Transformación de Características Invariante en Escala y se usa para extraer los puntos clave incluso si la escala cambia. SIFT resuelve la rotación de la imagen, las transformaciones afines, la intensidad y el cambio de punto de vista en la coincidencia de características.

El algoritmo SIFT tiene 4 pasos básicos :

- estimar un extremo del espacio de escala utilizando la diferencia de Gaussiana

(DoG)

- una localización de puntos clave donde los candidatos a punto clave son localizados y refinados eliminando los puntos de bajo contraste
- una asignación de la orientación del punto clave orientación de los puntos clave basada en el gradiente local de la imagen
- un generador de descriptores para calcular el descriptor local de la imagen para cada punto clave basado en la magnitud del gradiente de la imagen y la orientación

4.2.2 SURF (Speed-Up Robust Feature)

El algoritmo SURF es parecido a SIFT pero más rápido. SURF se aproxima a la Diferencia de Gaussiana con filtros de caja. En lugar de gaussiano promediando la imagen, se utilizan cuadrados para aproximación ya que la convolución con el cuadrado es mucho más rápida si se utiliza la imagen integral. Además, esto puede hacerse en paralelo para diferentes escalas. El SURF utiliza un detector de manchas que se basa en la matriz hessiana para encontrar los puntos de interés. Para la asignación de la orientación, utiliza respuestas de ondícula en las direcciones horizontal y vertical aplicando pesos gaussianos adecuados. Para la descripción de características, SURF también utiliza las respuestas wavelet. Se selecciona un vecindario alrededor del punto clave y se divide en subregiones y, a continuación, para cada subregión se toman las respuestas wavelet y se representan para obtener el descriptor de características SURF. El signo del laplaciano que ya se ha calculado en la detección se utiliza para los puntos de interés. El signo del Laplaciano distingue las manchas brillantes sobre fondos oscuros del caso contrario. En el caso de la coincidencia, las características se comparan sólo si tienen el mismo tipo de contraste (basado en el signo), lo que permite una coincidencia más rápida.

El limite del algoritmo SURF, tal y como el de SIFT es que tiene patente.

4.2.3 ORB (Oriented Fast and Rotated Brief)

Una alternativa de los metodos SIFT and SURF seria ORB, tanto como para la patente que en costo computacional y en rendimiento.

La técnica ORB (Oriented FAST and Rotated BRIEF) utiliza el algoritmo FAST para calcular los puntos clave. FAST son las siglas en inglés de Prueba de Características de Segmentos Acelerados. FAST calcula los puntos clave teniendo en cuenta el brillo de los píxeles alrededor de un área determinada.

ORB es una fusión del detector de puntos clave FAST y el descriptor BRIEF con muchas modificaciones para mejorar el rendimiento. En primer lugar, utiliza FAST para encontrar los puntos clave y, a continuación, aplica la medida de esquinas de Harris para encontrar los N puntos principales entre ellos. También utiliza la pirámide para producir características multiescalares.

5 Aplicación a la Raspberry Pi

A continuación es un código simple para implementar la detección de los puntos claves y descriptores con el método ORB.

```
#import modules
from time import sleep
from picamera import PiCamera
import picamera.array
import cv2
import numpy as np

# Create ORB object, we can specify the number of key points we
    desire
orb = cv2.ORB_create(20)

with picamera.PiCamera() as cam:
    # Wait for camera to warm up
    sleep(2)
    while True:
        cam.resolution = (640,480) #640,480
        cam.framerate = 24
        output = np.empty((480 * 640 * 3,), dtype=np.uint8)
        cam.capture(output, 'bgr')
        image = output.reshape((480, 640, 3))

        imgGray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        keypoints = orb.detect(imgGray, None)

        # Obtain the descriptors
        keypoints, descriptors = orb.compute(imgGray, keypoints)

        # Draw keypoints
```

```
28          image = cv2.drawKeypoints(image, keypoints, image, flags=
    cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
29          cv2.imshow("ORB Method", image)
30
31          if cv2.waitKey(1) & 0xFF == ord('q'):
32              break
33
34      cv2.destroyAllWindows()
```

6 Conclusions