

Prácticos

Araguás, Gastón

Redolfi, Javier

3 de abril de 2020

Práctico 1 - Función en python

- A. Crear una función **adivina** que permita adivinar un número secreto generado en forma aleatoria, según las siguientes consignas:
- El número secreto debe estar entre 0 y 100, y debe ser generado dentro de la función.
 - La función **adivina** debe recibir un parámetro que indique la cantidad de intentos permitidos.
- B. Luego escribir un programa **adivinador.py** que:
- pida al usuario que adivine el número secreto, es decir que ingrese un número entre 0 y 100,
 - use la función **adivina** anterior para verificar si adivinó (todo en el mismo archivo),
 - si el usuario adivinó el número secreto antes de superar la cantidad permitida de intentos, imprima un mensaje con el número de intentos en los que adivinó.
 - En caso de que esta cantidad de intentos sea superada el programa debe terminar con un mensaje.
- C. Finalmente ejecutar el programa **adivinador.py** desde la consola.

Ayuda: código para generar un número aleatorio

```
import random

...
numero = random.randint(0, 100)
...
```

Práctico 2 - Segmentando una imagen

- A. Escribir un programa en python que lea una imagen y realice un umbralizado binario, guardando el resultado en otra imagen.
- NOTA: No usar ninguna función de las OpenCV, excepto para leer y guardar la imagen.



Figura 1: Binarizado de imagen.

Ayuda:

- Se puede usar como base el siguiente template:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import cv2

img = cv2.imread( 'hoja.png' , 0)

# Agregar código aquí
# Para resolverlo podemos usar dos for anidados

cv2.imwrite( 'resultado.png' , img)
```

Práctico 3 - Propiedades de video

Considerando el siguiente programa

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import sys
import cv2

if (len(sys.argv) > 1):
    filename = sys.argv[1]
else:
    print('Pass a filename as first argument')
    sys.exit(0)

cap = cv2.VideoCapture(filename)

fourcc = cv2.VideoWriter_fourcc('X', 'V', 'I', 'D')
framesize = (640, 480)
out = cv2.VideoWriter('output.avi', fourcc, 20.0, framesize)

delay = 33
while(cap.isOpened()):
    ret, frame = cap.read()
    if ret is True:
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        out.write(gray)
        cv2.imshow('Image gray', gray)
        if cv2.waitKey(delay) & 0xFF == ord('q'):
            break
    else:
        break

cap.release()
out.release()
cv2.destroyAllWindows()
```

- A. ¿Cómo obtener el frame rate o fps usando las OpenCV? Usarlo para no tener que *hardcodear* el **delay** del **waitKey**.
- B. ¿Cómo obtener el ancho y alto de las imágenes capturadas usando las OpenCV? Usarlo para no tener que *hardcodear* el **frameSize** del video generado.

Práctico 4 - Manipulación de imágenes

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import cv2
import numpy as np

drawing = False # true if mouse is pressed
mode = True # if True, draw rectangle. Press 'm' to toggle to curve
```

```

ix, iy = -1, -1

def draw_circle(event, x, y, flags, param):
    global ix, iy, drawing, mode
    if event == cv2.EVENT_LBUTTONDOWN:
        drawing = True
        ix, iy = x, y
    elif event == cv2.EVENT_MOUSEMOVE:
        if drawing is True:
            if mode is True:
                cv2.rectangle(img, (ix, iy), (x, y), (0, 255, 0), -1)
            else:
                cv2.circle(img, (x, y), 5, (0, 0, 255), -1)
    elif event == cv2.EVENT_LBUTTONUP:
        drawing = False
        if mode is True:
            cv2.rectangle(img, (ix, iy), (x, y), (0, 255, 0), -1)
        else:
            cv2.circle(img, (x, y), 5, (0, 0, 255), -1)

img = np.zeros((512, 512, 3), np.uint8)
cv2.namedWindow('image')
cv2.setMouseCallback('image', draw_circle)
while(1):
    cv2.imshow('image', img)
    k = cv2.waitKey(1) & 0xFF
    if k == ord('m'):
        mode = not mode
    elif k == 27:
        break
cv2.destroyAllWindows()

```

- A. Usando como base el programa anterior, escribir un programa que permita seleccionar un rectángulo de una imagen, luego
- con la letra “g” guarda la imagen dentro del rectángulo en el disco,
 - con la letra “r” restaura la imagen original y permite realizar nuevamente la selección,
 - con la “q” finaliza.

Práctico 5 - Rotación + Traslación (o Transformación Euclidiana)

- A. Crear un método nuevo que aplique una transformación euclidiana, recibiendo los siguientes parámetros:
- Parámetros
 - **angle:** Ángulo
 - **tx:** traslación en x

-
- **ty**: traslación en y

Recordar que la transformación euclidiana tiene la siguiente forma:

$$\begin{bmatrix} \cos(\text{angle}) & \sin(\text{angle}) & tx \\ -\sin(\text{angle}) & \cos(\text{angle}) & ty \end{bmatrix}$$

- B. Usando como base el programa anterior, escriba un programa que permita seleccionar un rectángulo de una imagen y
- con la letra “e” aplique una transformación euclidiana a la imagen dentro del rectángulo y la guarde en el disco y sale.

Práctico 6 - Transformación de similaridad

- A. Agregar al método anterior un parámetro para permitir también el escalado de la imagen.
- Parámetros
 - **angle**: Ángulo
 - **tx**: traslación en x
 - **ty**: traslación en y
 - **s**: escala

Recordar que la transformación de similaridad tiene la siguiente forma:

$$\begin{bmatrix} s \cdot \cos(\text{angle}) & s \cdot \sin(\text{angle}) & tx \\ -s \cdot \sin(\text{angle}) & s \cdot \cos(\text{angle}) & ty \end{bmatrix}$$

- B. Usando como base el programa anterior, escriba un programa que permita seleccionar un rectángulo de una imagen y
- con la letra “s” aplique una transformación de similaridad a la imagen dentro del rectángulo y la guarde en el disco.

Práctico 7 - Transformación afín - Incrustando imágenes

Teniendo en cuenta que:

- una transformación afín se representa con una matriz de 2×3 (tiene 6 grados de libertad) y

-
- puede ser recuperada con **3** puntos no colineales.
- A. Crear un método que compute la transformación afín entre los 3 puntos seleccionados y las esquinas de la segunda imagen.
- B. Usando como base el programa anterior, escriba un programa que
- con la letra “a” permita seleccionar con el mouse 3 puntos no colineales en una imagen e incruste entre estos puntos seleccionados una segunda imagen.

Ayuda

- `cv2.getAffineTransform`
- `cv2.warpAffine`
- Generar una máscara para insertar una imagen en otra

Práctico 8 - Rectificando imágenes

Teniendo en cuenta que:

- una homografía se representa con una matriz de 3×3 (pero tiene sólo **8** grados de libertad) y
 - puede ser recuperada con **4** puntos no colineales.
- A. Crear un método que compute la homografía entre los 4 puntos seleccionados y las esquinas de la segunda imagen de $m \times n$ píxeles.
- B. Usando como base el programa anterior, escriba un programa que
- con la letra “h” permita seleccionar con el mouse 4 puntos no colineales en una imagen y transforme (rectifique) la selección en una nueva imagen rectangular.

Ayuda

- `cv2.getPerspectiveTransform`
- `cv2.warpPerspective`

Práctico 9 - Medición de objetos

- A. Capturar una imagen de un plano sobre la cual se asiente una figura de tamaño conocido. Luego usando esta figura como patrón, medir la distancia real entre dos puntos cualquiera que se encuentren en la imagen.

Práctico 10 - Calibración de una cámara

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import numpy as np
import cv2
import glob

criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)

objp = np.zeros((5 * 7, 3), np.float32)
objp[:, :2] = np.mgrid[0:7, 0:5].T.reshape(-1, 2)

objpoints = []
imgpoints = []

images = glob.glob('tmp/*.png')

for fname in images:
    img = cv2.imread(fname)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    ret, corners = cv2.findChessboardCorners(gray, (7, 5), None)

    if ret is True:
        objpoints.append(objp)

        corners2 = cv2.cornerSubPix(gray, corners, (11, 11), (-1, -1),
                                     criteria)
        imgpoints.append(corners2)

        img = cv2.drawChessboardCorners(img, (7, 5), corners2, ret)
        cv2.imshow('img', img)
        cv2.waitKey(300)

cv2.destroyAllWindows()

ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, gray.shape[::-1],
                                                    None, None)
```

- A. Utilizando el código de arriba calibrar la cámara usando las imágenes del patrón ya capturadas.
- B. Para corroborar la correcta calibración usar el resultado de calibración obtenido para eliminar la distorsión de las imágenes dadas.

Ayuda

- `cv2.undistort`

Práctica 11 - Alineación de imágenes usando SIFT

- A. Usar el siguiente código como ayuda:

```

#!/usr/bin/env python
# -*- coding: utf-8 -*-

import numpy as np
import cv2

MIN_MATCH_COUNT = 10

img1 = # Leemos la imagen 1
img2 = # Leemos la imagen 2

dscr = # Inicializamos el detector y el descriptor

kp1, des1 = # Encontramos los puntos clave y los descriptores con SIFT en la imagen 1
kp2, des2 = # Encontramos los puntos clave y los descriptores con SIFT en la imagen 2

matcher = cv2.BFMatcher(cv2.NORM_L2)

matches = matcher.knnMatch(des1, des2, k=2)

# Guardamos los buenos matches usando el test de razón de Lowe
good = []
for m, n in matches:
    if m.distance < 0.7*n.distance:
        good.append(m)

if (len(good) > MIN_MATCH_COUNT):
    src_pts = np.float32([kp1[m.queryIdx].pt for m in good]).reshape((-1, 1, 2))
    dst_pts = np.float32([kp2[m.trainIdx].pt for m in good]).reshape((-1, 1, 2))

    H, mask = cv2.findHomography(dst_pts, src_pts, cv2.RANSAC, 5.0) # Computamos la homografía con RANSAC

    wimg2 = # Aplicamos la transformación perspectiva H a la imagen 2

# Mezclamos ambas imágenes
    alpha = 0.5
    blend = np.array(wimg2 * alpha + img1 * (1 - alpha), dtype=np.uint8)

```

Ejemplo de alineación:

- Capturar dos imágenes con diferentes vistas del mismo objeto



Figura 2: Puntos claves en la imagen 1



Figura 3: Puntos claves en la imagen 2



Figura 4: Matches entre ambas imágenes

- Computar puntos de interés y descriptores en ambas imágenes
- Establecer matches entre ambos conjuntos de descriptores
- Eliminar matches usando criterio de Lowe
- Computar una homografía entre un conjunto de puntos y el otro
- Aplicar la homografía sobre una de las imágenes y guardarla en otra (mezclarla con un alpha de 50 %)

Práctica 12 - Clasificación de imágenes usando CNN

Entrenar un clasificador de dígitos manuscritos. Luego capturar una imagen con un dígito escrito por nosotros y clasificarlo usando la red entrenada.



Figura 5: Mezcla de ambas imágenes

Práctica 13 - Detección de objetos usando CNN

Recolectar imágenes de algún objeto particular y luego re-entrenar una red para detectar dicho objeto.

Evaluación

Condiciones de para la aprobación de la materia.

- Cada práctico tiene un puntaje y para aprobar hay que sumar 60 puntos o más. En la tabla se muestran los valores de cada una de los prácticos.
- Además, se debe confeccionar un informe de uno de los prácticos realizados donde se explique y justifique en forma teórica el método utilizado para realizarlo. Los prácticos elegibles para realizar el informe son del número 8 en adelante.

Número	Puntaje
Práctico 1	2
Práctico 2	2
Práctico 3	2
Práctico 4	5
Práctico 5	5
Práctico 6	5
Práctico 7	10
Práctico 8	10
Práctico 9	10
Práctico 10	10
Práctico 11	15
Práctico 12	15
Práctico 13	15
Total	121