Introducción a las OpenCV

Redolfi, Javier

OpenCV

Historia de las OpenCV

- Nacen en 1999 en Intel
- Luego continúa su desarrollo en WillowGarage
- En la actualidad es mantenida por la comunidad y cualquier persona puede aportar código a la librería
- Tiene licencia de código abierto BSD

Características de las OpenCV

OpenCV

- Librería para procesamiento de imágenes y visión por computadora
- Escrita en C++, pero con interfaz para:
  - **Python**
  - Java
  - MATLAB/OCTAVE
  - C#
  - Perl
  - Ch
  - Haskell
  - Ruby
- Soporta múltiples sistemas operativos:
  - **Linux**
  - Windows
  - macOS
  - FreeBSD
  - NetBSD
  - OpenBSD
  - Android

Características de las OpenCV

Aceleración de Hardware

- Soporta IPP (Intel Performance Primitives)
- CUDA
- OpenCL

Características de las OpenCV

Módulos Principales

- core. Core Functionality
- imgproc. Image processing
- imgcodecs. Image file reading and writing
- videoio. Media I/O
- highgui. High-level GUI
- video. Video Analysis
- calib3d. Camera Calibration and 3D Reconstruction
- features2d. 2D Features Framework
- objdetect. Object Detection
- ml. Machine Learning
- flann. Clustering and Search in Multi-Dimensional Spaces
- photo. Computational Photography
- stitching. Images stitching

Instalación en Linux

Debian

- Podemos instalarla desde el repositorio oficial de Debian
- Si queremos la última versión, podemos bajar el código fuente y compilarlo

Instalación desde el repositorio oficial

```
apt update && apt install libopencv-dev python3-opencv
```

OpenCV-Python

- Python es más lento que C/C++
- Pero puede ser fácilmente extendido con código en C/C++
- Tenemos código computacionalmente intensivo escrito en C++, el cual puede ser usado como un módulo de Python a través de un wrapper
- De esta manera nuestro código es tan rápido como el código en C++, pero es más simple y rápido para codificar
- Además OpenCV usa numpy arrays, que son arreglos de una librería de Python para cálculo numérico llamada numpy, los cuales están muy optimizados

Empezando con imágenes

Como leer una imagen

- Para esto se usa la función **`cv2.imread(filename[, flags])`**
- **filename** es el nombre relativo del archivo o el path completo
- Las **flags** son opcionales, pero la más usada es la que indica el modo de lectura:
  - **`cv2.IMREAD_COLOR`** Abre la imagen a color. Si tiene transparencias se descarta. Es la flag por defecto.
  - **`cv2.IMREAD_GRAYSCALE`** Abre la imagen en escala de grises.
  - **`cv2.IMREAD_UNCHANGED`** Abre la imagen incluyendo el canal alpha.

```
import cv2

# Load an color image in grayscale
img = cv2.imread('messi5.jpg', cv2.IMREAD_COLOR)
```

Empezando con imágenes

Como mostrar una imagen

```
cv2.imshow('image', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- Usamos la función **`cv2.imshow(winname, img)`**
- **`winname`** es el nombre de la ventana, string
- **`img`** es la imagen que vamos a mostrar
- **`cv2.waitKey([, delay])`**
  - Espera durante **`delay`** ms a que se presione una tecla
  - Si **`delay = 0`** espera indefinidamente
- **`cv2.destroyAllWindows()`** destruye todas las ventanas
- **`cv2.destroyWindow(winname)`** destruye la ventana con nombre **`winname`**

Escribiendo una imagen

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

- **`cv2.imwrite(filename, img[, params])`**
  - **filename** es el nombre relativo del archivo o el path completo en donde queremos guardar la imagen
  - **img** es la imagen que queremos guardar
  - Los **params** son opcionales y dependen del tipo de imagen a guardar, por ejemplo **`png`**, **`jpg`**

Escribiendo una imagen

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

- **`cv2.imwrite(filename, img[, params])`**
  - **filename** es el nombre relativo del archivo o el path completo en donde queremos guardar la imagen
  - **img** es la imagen que queremos guardar
  - Los **params** son opcionales y dependen del tipo de imagen a guardar, por ejemplo **`png`**, **`jpg`**

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


import cv2

img = cv2.imread('homer.jpg', 0)
cv2.imshow('image', img)
k = cv2.waitKey(0)

if k == ord('s'):    # wait for 's' key to save and exit
    print('Save image as gray scale.')
    cv2.imwrite('homer_gray.png', img)
else:
    print('Not save image.')

cv2.destroyAllWindows()
```

Empezando con videos

Como capturar un video

- Para esto se usa la clase **`cv2.VideoCapture(device)`**
  - **`device`** puede ser un string que indique el filename **`'/dev/video0'`** o un índice entero 0
  - Si tenemos más de una cámara usamos 0, 1, 2, . . . o **`'/dev/video0'`**, **`'/dev/video1'`**, **`'/dev/video2'`**
- Nos devuelve un objeto que representa a la cámara
- Para chequear que esté correctamente abierto usamos el método **`isOpened()`**
- Para capturar una imagen usamos **`read()`**
- Para cerrar la cámara usamos **`release()`**

Empezando con videos - Como capturar un video

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

import cv2

cap = cv2.VideoCapture(0)

while(True):
    ret, frame = cap.read()

    # Our operations on the frame come here
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    cv2.imshow('frame', gray)
    if((cv2.waitKey(1) & 0xFF) == ord('q')):
        break

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

Empezando con videos - Abriendo un video desde un archivo

```
#! /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)

while(cap.isOpened()):
    ret, frame = cap.read()

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    cv2.imshow('frame', gray)
    if((cv2.waitKey(33) & 0xFF) == ord('q')):
        break

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

Práctica 3

¿Cómo obtener el frame rate o fps usando las OpenCV? Usarlo para no tener que *harcodear* el **`delay`** del **`waitKey`**.

Empezando con videos - Como guardar un video

- Usamos la clase:
  `cv2.VideoWriter([filename, fourcc, fps, frameSize])`
  - **`filename`** es el nombre del archivo de salida
  - **`fourcc`** es la codificación del video, DIVX, XVID, MJPG, X264, WMV1, WMV2.
  - **`fps`** es el número de cuadros por segundo (frames per second)
  - **`frameSize`** tupla con la resolución del video a guardar, (W, H) = (ancho, alto)
- El formato lo especificamos con **`cv2.VideoWriter_fourcc`**, esto se usa para generar un código con cuatro carácteres (four character code)

Empezando con videos - Como guardar un video

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


import cv2

cap = cv2.VideoCapture(0)
fourcc = cv2.VideoWriter_fourcc('X', 'V', 'I', 'D')
out = cv2.VideoWriter('output.avi', fourcc, 20.0, (640, 480))

while(cap.isOpened()):
    ret, frame = cap.read()
    if ret is True:
        out.write(frame)
        cv2.imshow('frame', frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break

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

Práctica 4

¿Cómo obtener el ancho y alto de las imágenes capturadas usando las OpenCV? Usarlo para no tener que *harcodear* el **`frameSize`** del video generado.

Dibujando líneas

- `img = cv.line(img, pt1, pt2, color[, thickness[, lineType]])`
- **`img`** imagen sobre la que vamos a dibujar, es la misma que la de salida
- **`pt1`** punto inicial de la línea, indicado con una tupla **`(x, y)`**
- **`pt2`** punto final de la línea, indicado con una tupla **`(x, y)`**
- **`color`** color de la línea, indicado con una tupla **`(B, G, R)`**
- **`thickness`** ancho de la línea en pixeles, entero
- **`lineType`** tipo de línea
  - **`cv.LINE_8`**
  - **`cv.LINE_4`**
  - **`cv.LINE_AA`**

Dibujando rectángulos

- `img = cv.rectangle(img, pt1, pt2, color[, thickness[, lineType]])`

Dibujando círculos

- `img = cv.circle(img, center, radius, color[, thickness[, lineType]])`
- **`center`** centro del círculo, indicado con una tupla **`(x, y)`**
- **`radius`** radio del círculo en pixeles, entero

Dibujando elipses

- Tiene dos formas
- `img = cv.ellipse(img, center, axes, angle, startAngle, endAngle, color[, thickness[, lineType]])`
- **`center`** centro del círculo, indicado con una tupla **`(x, y)`**
- **`axes`** largo de los ejes mayor y menor, indicado con una tupla **`(w, h)`**
- **`angle`** ángulo de inclinación del eje mayor
- **`startAngle`** ángulo de inicio
- **`endAngle`** ángulo de finalización
- `img = cv.ellipse(img, box, color[, thickness[, lineType]])`

Dibujando polígonos

- `img = cv.polylines(img, pts, isClosed, color[, thickness[, lineType]])`
- **`pts`** arreglo con los puntos vértices de las líneas, debe ser un arreplo de dimensiones `(npuntos, 1, 2)`, `[[[x0, y0]], [[x1, y1]], ...]`
- **`isClosed`** bool que indica si hay que unir el último punto con el primero

Agregando texto

- `img = cv.putText(img, text, org, fontFace, fontScale, color[, thickness[, lineType[, bottomLeftOrigin]]])`
- **`text`** texto a escribir
- **`org`** ubicación del punto inferior izquierdo de la imagen especificado como una tupla
- **`fontFace`** fuente
- **`fontScale`** factor para escalar la fuente, float
- **`bottomLeftOrigin`** bool que indica tomar como inicio de la imagen el borde inferior izquierdo

Ejemplo

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


import numpy as np
import cv2 as cv

blue = (255, 0, 0)
red = (0, 255, 0)
green = (0, 0, 255)

# Creamos una imagen RGB de color negro
img = np.zeros((512, 512, 3), np.uint8)

img = cv.line(img, (0, 0), (511, 511), blue, 5)

img = cv.rectangle(img, (384, 0), (510, 128), green, 3)

img = cv.circle(img, (447, 63), 63, red, -1)

img = cv.ellipse(img, (256, 256), (100, 50), 0, 0, 180, 255, -1)

pts = np.array([[10, 5], [20, 30], [70, 20], [50, 10]], np.int32)
pts = pts.reshape((-1, 1, 2))
img = cv.polylines(img, [pts], True, (0, 255, 255))

font = cv.FONT_HERSHEY_SIMPLEX
cv.putText(img, 'CV2018', (10, 500), font, 4, (255, 255, 255), 2, cv.LINE_AA)

cv.imshow('frame', img)
key = cv.waitKey(0)
print(key)
cv.destroyAllWindows()
```

Eventos del mouse

Un evento es una acción realizada por el usuario, por ejemplo presionar algún botón del mouse, que puede ser capturada para actuar en consecuencia, por ejemplo dibujar un punto en la imagen.

EVENT_MOUSEMOVE	el puntero del mouse se movió
EVENT_LBUTTONDOWN	se apretó el botón izquierdo
EVENT_RBUTTONDOWN	se apretó el botón derecho
EVENT_MBUTTONDOWN	se apretó el botón del medio
EVENT_LBUTTONUP	se soltó el botón izquierdo
EVENT_RBUTTONUP	se soltó el botón derecho
EVENT_MBUTTONUP	se soltó el botón del medio
EVENT_LBUTTONDBLCLK	doble click con el botón izquierdo
EVENT_RBUTTONDBLCLK	doble click con el botón derecho
EVENT_MBUTTONDBLCLK	doble click con el botón del medio
EVENT_MOUSEWHEEL	scroll vertical, positivo significa hacia adelante y negativo hacia atrás
EVENT_MOUSEHWHEEL	scroll horizontal, positivo significa hacia la derecha y negativo hacia la izquierda

Dibujando puntos

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


import cv2
import numpy as np


# mouse callback
def draw_circle(event, x, y, flags, param):
    if event == cv2.EVENT_MBUTTONDBLCLK:
        print("cv2.EVENT_MBUTTONDBLCLK", event)
        cv2.circle(img, (x, y), 3, (255, 0, 0), -1)
    elif event == cv2.EVENT_LBUTTONDOWN:
        print("cv2.EVENT_LBUTTONDOWN", event)
        cv2.circle(img, (x, y), 3, (0, 255, 0), -1)
    elif event == cv2.EVENT_LBUTTONUP:
        print("cv2.EVENT_LBUTTONUP", event)
        cv2.circle(img, (x, y), 3, (0, 0, 255), -1)


""" Creamos una imagen en blanco, una ventana y
    capturamos los eventos del mouse en esa ventana"""
img = np.ones((512, 512, 3), np.uint8) * 255
cv2.namedWindow('image')
cv2.setMouseCallback('image', draw_circle)

while(1):
    cv2.imshow('image', img)
    if cv2.waitKey(20) & 0xFF == 27:
        break

cv2.destroyAllWindows()
```

Dibujando rectángulos y pintando

```python
#! /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(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)
while(1):
    cv2.imshow('image', img)
    k = cv2.waitKey(1) & 0xFF
    if k == ord('m'):
        mode = not mode
    elif k == 27:
        break
cv2.destroyAllWindows()
```

Práctica 5

Usando como base el programa anterior, crear un programa que permita seleccionar un rectángulo de una imagen, luego

- con la letra “g” lo guardamos a disco en una nueva imagen y salimos
- con la letra “r” restauramos la imagen original y volvemos a realizar la selección
- con la “q” salimos.