Título

Índice general

Capítulo 1

Introducción

1.1. Motivación

1.2. OpenCV

OpenCV es una librería de código abierto escrita en C y C++ destinada a la visión artificial y al tratamiento de imágenes. Se trata de una librería multiplataforma con versiones para GNU/Linux, Windows, Mac OS y Android y actualmente cuenta con interfaces para Python, Java y MATLAB/OCTAVE. Las últimas versiones incluyen soporte para GPU usando CUDA. Desarrollada originalmente por Intel, su primera versión alfa se publicó en el año 2000 en el *IEEE Conference on Computer Vision and Pattern Recognition*. OpenCV nació inicialmente como un proyecto para avanzar en las aplicaciones de uso intenso de la CPU y dando gran importancia a las aplicaciones en tiempo real. Hoy en día cuenta con más 2500 algoritmos optimizados que abarcan todo tipo de campos relacionados con la visión artificial. Estos algortimos pueden ser usados para tareas como: detectar y reconocer caras y gestos, identificar objetos, detección de características 2D y 3D, estimación de movimiento, seguimiento del movimiento, visión estéro y calibración de la cámara, eliminar los ojos rojos de las fotografías realizadas con flash...
OpenCV es ampliamente utilizada por todo tipo de empresas (desde grandes empresas como Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota a pequeñas empresas), grupos de investigación y organismos gubernamentales y en sectores de todo tipo como: inspección de los productos en las fábricas, seguridad, usos médicos, robótica...

Ejemplos

```
import cv2
def load (name):
    img = cv2.imread(name)
    cv2.imshow('Example1',img)
    cv2.waitKey(0)

if __name__ == '__main__':
    import sys
    try: name = sys.argv[1]
    except:
        print("Error, introduce nombre del archivo")
        sys.exit()
    load(name)
```

ejemplos/ejemplo1.py

```
import cv2
def load (name):
    capture =  cv2.VideoCapture(name)
    while (True):
        ret, frame = capture.read()
        if (not ret): break
        cv2.imshow("Example2",frame)
        c = cv2.waitKey(33)
        if (c==27): break


if __name__ == '__main__':
    import sys
    try: name = sys.argv[1]
    except:
        print("Error, introduce nombre del archivo")
        sys.exit()
    load(name)
```

ejemplos/ejemplo2.py

```
import cv2
global g_slider_position

def nothing(*arg):
    pass

def onTrackbarSlide(pos):
    g_capture.set(cv2.cv.CV_CAP_PROP_POS_FRAMES, pos)

def load (name):
```

```
    g_slider_position=0
    global g_capture
    g_capture =  cv2.VideoCapture(name)
    frames = int(g_capture.get(cv2.cv.CV_CAP_PROP_FRAME_COUNT))
    if (frames != 0):
        cv2.namedWindow("Example3")
        cv2.createTrackbar("Position","Example3",
   g_slider_position,frames,onTrackbarSlide)
    while (True):
        ret, frame = g_capture.read()
        if (not ret): break
        cv2.imshow("Example3",frame)
        g_slider_position = int(g_capture.get(cv2.cv.
   CV_CAP_PROP_POS_FRAMES))

        cv2.setTrackbarPos("Position","Example3",
   g_slider_position)
        c = cv2.waitKey(33)
        if (c==27): break


if __name__ == '__main__':
    import sys
    try: name = sys.argv[1]
    except:
        print("Error, introduce nombre del archivo")
        sys.exit()
    load(name)
```

ejemplos/ejemplo3.py

```
import cv2
def load (name):

    img = cv2.imread(name)
    cv2.imshow('Example4-in',img)

    out = cv2.GaussianBlur(img,(9,9), 0)
    cv2.imshow("Example4-out", out)
    cv2.waitKey(0)

if __name__ == '__main__':
    import sys
    try: name = sys.argv[1]
    except:
        print("Error, introduce nombre del archivo")
        sys.exit()
    load(name)
```

ejemplos/ejemplo4.py

```
import cv2

def doPyrDown(im):
    h, w = im.shape[:2]
    assert (w%2 == 0 and h%2 == 0)
    out= cv2.pyrDown(im)
    return out


if __name__ == '__main__':
    import sys
    try: name = sys.argv[1]
    except:
        print("Error, introduce nombre del archivo")
        sys.exit()

    img = cv2.imread(name)
    out = doPyrDown(img)
    cv2.imshow('Example5-in',img)
    cv2.imshow("Example5-out", out)
    cv2.waitKey(0)
```

ejemplos/ejemplo5.py

```
import cv2
import numpy as np

def doCanny(im, lowThresh, highThresh, aperture):
    if (len(im.shape) != 2):
        aux =  cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
        out = cv2.Canny(aux, lowThresh, highThresh, aperture)
    else:
        out = cv2.Canny(im, lowThresh, highThresh, aperture)
    return out

def doPyrDown(im):
    h, w = im.shape[:2]
    assert (w%2 == 0 and h%2 == 0)
    out= cv2.pyrDown(im)
    return out

if __name__ == '__main__':
    import sys
    try: name = sys.argv[1]
    except:
```

```
        print("Error, introduce nombre del archivo")
        sys.exit()

    img = cv2.imread(name)
    img1 = doPyrDown(img)
    img2 = doPyrDown(img1)
    out = doCanny(img2, 10, 100, 3)
    cv2.imshow('Example6-in',img)
    cv2.imshow("Example6-out", out)
    cv2.waitKey(0)
```

ejemplos/ejemplo7.py

```
import cv2

if __name__ == '__main__':

    capture =  cv2.VideoCapture(0)

    while (True):
        ret, frame = capture.read()
        if (not ret): break
        cv2.imshow("Example9",frame)
        c = cv2.waitKey(33)
        if (c==27): break
```

ejemplos/ejemplo9.py

```
import cv2
global r
def call(pos):
    global r
    r = pos
    return
def main ():
    global r
    r = 3
    cv2.namedWindow("Blur")
    capture =  cv2.VideoCapture(0)
    cv2.createTrackbar("Radio","Blur",r,30,call)
    while (True):
        ret, frame = capture.read()
        if (not ret): break
        if ( r!=0):
            out = cv2.blur(frame, (r,r))
            cv2.imshow("Blur",out)
        else:
            cv2.imshow("Blur",frame)
        c = cv2.waitKey(33)
```

```
            if (c==27): break
    cv2.destroyAllWindows()
main()
```

ejemplos/BlurCam.py

1.3. Android

Android es un sistema operativo desarrollado por Google y orientado a móviles y tablets. Está basado en linux y es de código abierto. Actualmente se estima que en torno un 80 % de los dispositívos móviles usan el sistema operativo android. Su núcleo está programado en C, pero la aplicaciones y toda la interfaz de usuario se programan en Java. Este sistema operativo está estructurado en 4 capas:

Núcleo linux
: Se encarga de las funcionalidades básicas del sistema, como manejo de procesos, de la memoria y de los dispositivos como la cámara, la pantalla, etc. Asimismo funciona como una capa de ebstracción entre el hardware y el resto del software.

Librerías y Runtime de Android
: Por encima del núcleo, hay una serie de librerías usadas por componentes del sistema, entre ellas destacan Surface Manager, Media Framework, SQLite, WebKit, SGL y Open GL
El runtime de Android proporciona un componente clave,la máquina virtual Dalvik. Cada aplicación Android corre su propio proceso, con su propia instancia de esta máquina virtual. Además, el runtime de android proporciona librerías básicas que proporcionan la mayor parte de las funciones del lenguaje de programación Java.

Framework de aplicaciones
: Esta capa proporciona a las aplicaciones muchos servicios en forma de clases de Java

Aplicaciones
: En esta capa se encuentran tanto las aplicaciones base como las que instalemos y desarrollemos ...

Estructura de una aplicación Android

Las aplicaciones android están por los llamados componentes de aplicación. Hay cuatro tipo de componentes:

Activities
: Representa una única pantalla de la aplicación con su interfaz de usuario.

Services

Son componentes que se ejecutan en segundo plano y que no constan de interfaz gráfica.

Content providers

Broadcast receivers

Capítulo 2

Estabilización de imagen

2.1. Algoritmo

Se trata de un algoritmo básico de estabilización de imagen que se basa en la suposición de que el único movimiento de la cámara es el que queremos eliminar; es decir, excepto por ligeros movimientos no deseados, la cámara está estática. La idea del algoritmo es muy sencilla: Tomamos dos frames consecutivos y buscamos una serie de puntos característicos mediante el algoritmo propuesto por Shi y Tomashi [2], usando la función: goodFeaturesToTrack; después vemos a donde se han movido esos puntos en el siguiente frame mediante el algoritmo de flujo óptico de Lucas-Kanade. Finalmente calculamos la homografía que lleva los puntos originales a donde hemos calculado que se han movido y le aplicamos la inversa de esa transformación al segundo frame para colocarlo donde debería estar.

2.1.1. Encontrar puntos característicos

EXPLICACIÓN

2.1.2. Algoritmo de Lucas-Kanade

EXPLICACIÓN

2.1.3. Codigo

```
#!/usr/bin/env python

import numpy as np
import cv2
```

```
if __name__ == '__main__':

    import sys
    try: name = sys.argv[1]
    except:
        print("Error, introduce nombre del archivo de entrada")
        sys.exit()
    cap = cv2.VideoCapture(name)

    # params for ShiTomasi corner detection
    feature_params = dict( maxCorners = 10000,
                           qualityLevel = 0.001,
                           minDistance = 50,
                           blockSize = 30 )

    # Parameters for lucas kanade optical flow
    lk_params = dict( winSize  = (30,30),
                      maxLevel = 4,
                      criteria = (cv2.TERM_CRITERIA_EPS | cv2.
   TERM_CRITERIA_COUNT, 10, 0.03))

    # Create some random colors
    color = np.random.randint(0,255,(10000,3))

    draw = False

    # Take first frame and find corners in it
    ret, old_frame = cap.read()
    old_gray = cv2.cvtColor(old_frame, cv2.COLOR_BGR2GRAY)
    p0 = cv2.goodFeaturesToTrack(old_gray, mask = None, **
   feature_params)

    # Create a mask image for drawing purposes
    mask = np.zeros_like(old_frame)

    cv2.imshow('original',old_frame)
    cv2.imshow('stabilized',old_frame)

    while(1):
        ret,frame = cap.read()
        cv2.imshow('original',frame)
        frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        # calculate optical flow
        p1, st, err = cv2.calcOpticalFlowPyrLK(old_gray,
   frame_gray, p0, None, **lk_params)

        # Select good points
        good_new = p1[st==1]
        good_old = p0[st==1]
```

```
        h, w = old_frame.shape[:2]

        H, status = cv2.findHomography(p0, p1, cv2.RANSAC, 0)

        overlay = cv2.warpPerspective(frame, H, (w, h), flags=
    cv2.INTER_LINEAR + cv2.WARP_INVERSE_MAP)


        # draw the tracks
        old_frame = overlay.copy()
        if draw:
            for i,(new,old) in enumerate(zip(good_new,good_old))
    :
                a,b = new.ravel()
                c,d = old.ravel()
                cv2.circle(overlay,(a,b),5,color[i].tolist(),-1)


        cv2.imshow('stabilized',overlay)
        k = cv2.waitKey(30) & 0xff
        if k == 27:
            break
        if k == ord('d'):
            draw = not draw

        # Now update the previous frame and previous points

        old_gray = cv2.cvtColor(old_frame, cv2.COLOR_BGR2GRAY)
        p0 = cv2.goodFeaturesToTrack(old_gray, mask = None, **
    feature_params)

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

estabilizacion.py

2.2. Uso del acelerometro para estabilizar

Se trata de intentar utilizar los datos extra de los que disponemos, es decir, los que nos proporciona el acelerómetro del móvil para intentar estabilizar la imagen utilizando estos datos. Para ello en primer lugar hacemos un programa para android que se encarga de recoger estos datos, grabando el vídeo a la vez que registra los datos del acelerómetro y los guarda en un archivo junto con el momento exacto en el que se han registrado los datos. Luego, un segundo programa en el ordenador proceaso los arhivos para inten-

tar estabilizar la imagen: recoge los datos de aceleración en cada instante de media y aproxima el movimiento en dichos instantes. Luego calcula mediante interpolación cuanto se ha movido la camára en el instante del fotograma y recoloca el fotograma de acuerdo con lo obtenido.

2.2.1. Programa para Android

2.2.2. Programa de procesado de los datos

```
#!/usr/bin/env python

import numpy as np
import cv2
import cv2.cv as cv


def get_array(filename):
    acc_file = open( filename, "r" )
    array = []
    first = True
    old_date = 0.0
    for line in acc_file:
        if first:
            aux_date = line.split(":")
            t0 = ((int(aux_date[0])*60 + int(aux_date[1]))*60 +
   float(aux_date[2]))
            first = False
        else:
            aux = line.split()
            date_str = aux[2]
            aux_date = date_str.split(":")
            #date = [int(aux_date[0]), int(aux_date[1]), float(
   aux_date[2])]
            #get the time in seconds
            date = ((int(aux_date[0])*60 + int(aux_date[1]))*60
   + float(aux_date[2])) - t0
            fline = [float(aux[0]), float(aux[1]), date]
            #esto es un poco cutre habria que cambiarlo
            if date != old_date:
                array.append( fline )
                old_date = date
    acc_file.close()
    return array

def calculate_movement(filename):
    array = get_array(filename)
    t0 = 0.0
```

```
    x0 = 0.0
    vx0 = 0.0
    y0 = 0.0
    vy0 = 0.0
    dx = [(0.0,0.0)]
    dy = [(0.0,0.0)]
    for line in array:
        ax = line[0]
        ay = line[1]
        t1 = line[2]
        vx1 = vx0 + ax*(t1-t0)
        vy1 = vy0 + ay*(t1-t0)
        x1 = x0 + (vx0 + vx1)*(t1-t0)/2
        y1 = y0 + (vy0 + vy1)*(t1-t0)/2
        dx.append((t1, x1))
        dy.append((t1, y1))
        x0 = x1
        y0 = y1
        t0 = t1
    return dx, dy


if __name__ == '__main__':

    import sys
    try:
        name = sys.argv[1]
        accname = sys.argv[2]
    except:
        print("Error, introduce nombre del archivo de entrada")
        sys.exit()
    cap = cv2.VideoCapture(name)
    x, y = calculate_movement(accname)
    print x,y
    fps = cap.get(cv2.cv.CV_CAP_PROP_FPS)
    size = (int(cap.get( cv.CV_CAP_PROP_FRAME_WIDTH)),int(cap.
   get(cv.CV_CAP_PROP_FRAME_HEIGHT)))
    print fps
    fps = 30
    t = 0.0

    # Create some random colors
    color = np.random.randint(0,255,(10000,3))

    # Take first frame and find corners in it
    ret, old_frame = cap.read()
    old_gray = cv2.cvtColor(old_frame, cv2.COLOR_BGR2GRAY)
```

```
# Create a mask image for drawing purposes
mask = np.zeros_like(old_frame)

cv2.imshow('original',old_frame)
cv2.imshow('stabilized',old_frame)

while(1):
    ret,frame = cap.read()
    cv2.imshow('original',frame)
    frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)


    h, w = old_frame.shape[:2]

    #calculamos la traslacion
    t = t + 1.0/fps
    dx = np.interp(t,[d[0] for d in x], [d[1] for d in x])
    dy = np.interp(t,[d[0] for d in y], [d[1] for d in y])

    dx = dx*(138000.0)
    dy = dy*(138000.0)

    M = np.array([[1, 0, dx],[0, 1, dy]])


    overlay = cv2.warpAffine(frame, M, (w,h), flags=cv2.
INTER_LINEAR + cv2.WARP_INVERSE_MAP)


    old_frame = overlay.copy()


    cv2.imshow('stabilized',overlay)
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

    # Now update the previous frame and previous points

    old_gray = cv2.cvtColor(old_frame, cv2.COLOR_BGR2GRAY)

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

estabilizacion_acelerometro.py

Bibliografía

[1] Gary Bradski and Adrian Kaehler. *Learning OpenCV*. O'Reilly, 2008.

[2] Jianbo Shi and Carlo Tomasi. Good features to track. In Springer, editor, *9th IEEE Conference on Computer Vision and Pattern Recognition*, 1994.