Trabajo Práctico de Laboratorio N° 9

Medición de Objetos

Visión por computadora

Ingeniería Electrónica

Profesores:

- Ing. Araguás Gastón
- Ing. Redolfi Javier

Curso: 6R3
Año: 2020

Alumnos:

- Arias Ignacio -Legajo 63080
- Bayley Tomás -Legajo 67503
- Mongi Martin -Legajo 67798

Introducción

En este proyecto determinaremos las medidas de diferentes objetos a partir de una imagen que tiene cierta perspectiva. Para hacerlo se necesita rectificar la imagen y conocer por lo menos una medida en dimensiones reales para usar como patrón de medición.

Enunciado

La siguiente imagen es una foto con diferentes objetos en un mismo plano. Sabiendo que el largo del sobre de jugo es de 91mm.

A. Escribir un programa que en forma automática usando solo las medidas conocidas sea capaz de medir algunas de las cosas siguientes: ancho y alto de la tarjeta y el sobre, y radio de la moneda.

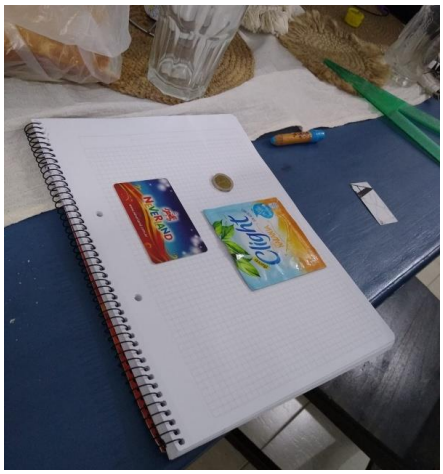

Fig. 1 – Imagen Real

Rectificación

Para lograr la rectificación de una imagen se debe conocer las esquinas de la superficie donde se encuentran los objetos (en nuestro caso, la hoja del cuaderno) para pasarlos como parámetros para generar la matriz de la transformación perspectiva que le aplicaremos a estas cuatro esquinas junto a las esquinas destino que en este caso serán las esquinas de la imagen completa.

En este caso se realizó de manera manual la ubicación de las esquinas mediante la función “click” que responde a los eventos del mouse y luego de haber presionado sobre las cuatro esquinas, se registró las coordenadas devueltas por la función como “c1, c2, c3 y c4”

Las esquinas de la imagen se obtienen utilizando la función “image.shape” que devuelve las dimensiones de “image”.

La función que realiza la transformada es la siguiente:

```
def TransformacionProyectiva(image):
  (h,w) = image.shape[:2]
  pts1 = np.float32([[c1], [c2], [c3], [c4]])
  pts2 = np.float32([[0, 0], [w, 0], [0, h], [w, h]])
  M = cv2.getPerspectiveTransform(pts1, pts2)
  dst = cv2.warpPerspective(image, M, (w,h))
  dst = cv2.flip(dst,0)
  return dst
```

Recibe la imagen original por parámetro y luego realiza la transformación con la matriz M generada con los puntos anteriormente explicados y devuelve la imagen guardada en la variable “dst” luego de haberle realizado un espejado tanto en el eje “x” como en el “y”.

Medición

Una vez obtenida la imagen rectificada podemos pasar a la parte de la medición de cualquier distancia entre dos puntos seleccionados que se encuentren sobre la hoja del cuaderno.

Es necesario conocer alguna medida en dimensiones reales para poder lograrlo por lo que anteriormente se realizó la calibración del programa midiendo el largo del sobre de jugo con una regla y luego se midió la misma distancia en pixeles en la imagen rectificada. La relación obtenida fue que 91mm reales equivalen a 244 pixeles.

Mediante la función “click” que es llamada cada vez que se registra un evento del mouse, se obtienen las coordenadas de los dos puntos con los que se calculará la distancia en la función “Medir” que se muestra a continuación:

```
def Medir():
    patronpx=244.0
    patroncm=9.1
    distanciax = c2m[0] - c1m[0]
    distanciay = c2m[1] - c1m[1]
    distanciaxcm=distanciax*patroncm/patronpx
    distanciaycm=distanciay*patroncm/patronpx
    distanciatotalcm=math.sqrt((distanciaxcm*distanciaxcm)+(distanciaycm*distanciaycm))
    print('Medida Horizontal:',abs(distanciaxcm))
    print('Medida Vertical:',abs(distanciaycm))
    print('Distancia entre los puntos:',abs(distanciatotalcm))
```

Utilizando la medida patrón se realiza una regla de tres con la distancia entre los dos clicks presionados para cada eje y luego mediante Pitágoras se obtiene la distancia real.

Por último, se imprimen las tres medidas obtenidas.

Cabe aclarar que, al iniciar el programa, se muestra en pantalla la imagen original y se imprime el mensaje “Presione m para medir”.

Una vez presionada la tecla “m”, se llama a la función que realiza la rectificación de la imagen y el programa queda a la espera de la selección de puntos para realizar la cantidad de mediciones que el usuario desee.

Se puede reiniciar el programa volviendo a la imagen original presionando la tecla “r”.

Con la tecla “q” se sale del programa terminando así su ejecución.

El código del programa es el siguiente:

```
import numpy as np
import cv2
import math

ix, iy = -1, -1
i=1
j=1
c1 = [239, 614]
c2 = [541, 460]
c3 = [36, 157]
c4 = [316, 148]

c1m = [-1, -1]
c2m = [-1, -1]
```

```
medir=False
midiendo=False
distanciax=0.0
distanciay=0.0
transformada=0

def click(event, x, y, flags, param):
   global medir,c1m,c2m,j,distanciax,distanciay,transformada

   if medir is True:
      if event == cv2.EVENT_LBUTTONDOWN:
         if j==1:
            c1m=(x,y)
            cv2.rectangle(Transformada, (x, y), (x+2, y+2), (0, 0, 255),-1)
            cv2.imshow('Paramedir', Transformada)
            j=j+1
         elif j==2:
            c2m=(x,y)
            cv2.rectangle(Transformada, (x, y), (x+2, y+2), (0, 0, 255),-1)
            cv2.imshow('Paramedir', Transformada)
            distanciax=c2m[0]-c1m[0]
            distanciay=c2m[1]-c1m[1]
            j=j+1

   return


def TransformacionProyectiva(image):
   (h,w) = image.shape[:2]
   pts1 = np.float32([[c1], [c2], [c3], [c4]])
   pts2 = np.float32([[0, 0], [w, 0], [0, h], [w, h]])
   M = cv2.getPerspectiveTransform(pts1, pts2)
   dst = cv2.warpPerspective(image, M, (w,h))
   dst = cv2.flip(dst,0)
   return dst


def Medir(image):
   patronpx=244.0
   patroncm=9.1
   distanciaxcm=distanciax*patroncm/patronpx
   distanciaycm=distanciay*patroncm/patronpx
   distanciatotalcm=math.sqrt((distanciaxcm*distanciaxcm)+(distanciaycm*distanciaycm))
   print('Medida Horizontal:',abs(distanciaxcm))
   print('Medida Vertical:',abs(distanciaycm))
   print('Distancia entre los puntos:',abs(distanciatotalcm))

   # main



cv2.namedWindow ('Paramedir')
Paramedir =cv2.imread('Paramedir.jpeg', 1)
(h, w) = Paramedir.shape[:2]
cv2.imshow('Paramedir', Paramedir)
cv2.setMouseCallback('Paramedir',click)
print('Presione m para empezar a medir')




while (1):
```

```
    k = cv2.waitKey(1) & 0xFF
    if midiendo==False:
        cv2.imshow('Paramedir', Paramedir)
    if k == ord('m'):
        Transformada = TransformacionProyectiva(Paramedir)
        cv2.imshow('Paramedir',Transformada)
        medir=True
        midiendo=True
    if j==3:
        Medicion=Medir(Transformada)
        cv2.imwrite('SalidaTp9.png',Transformada)
        j=1

    if k == ord('r'):
        Paramedir = cv2.imread('Paramedir.jpeg', 1)
        cv2.imshow('Paramedir', Paramedir)
        midiendo=False
        medir=False
    if k == ord('q'):
        break
```