Acceso a cada píxel

Solución desafíos previos

Acceso individual

El acceso individual a cada píxel es posible, sin embargo es costoso computacionalmente y rara vez es viable para el funcionamiento en tiempo real. Dadas estas limitaciones se recomienda siempre el uso de operaciones matriciales.

```
# Lectura de imagen. El parámetro 0 indica que es en escala de grises
entrada = cv2.imread("Lena.png",0)
# De la imagen leida se extrae el alto y el ancho
alto, ancho = entrada.shape
# Se crea una imagen vacia para almacenar el resultado
salida = np.zeros((alto,ancho), np.uint8)
# Permite crear un ciclo infinito. Similar a LOOP de Arduino
while(True):

    for i in range(0, alto): #Se recorre la imagen de 0 al alto
        for j in range(0, ancho): #Se recorre la imagen de 0 al ancho
            #Se lee la información del píxel en la posición i,j
            f = entrada.item(i,j)
            # Se resta 30 al valor original
            g = (f-30)
            #Se asigna el valor calculado, g, a la imagen nueva
            salida.itemset((i,j),g)

    #Se muestra la imagen de entrada en una ventana llamada original
    cv2.imshow("Original",entrada)
    cv2.imshow("Resultado",salida)

    #Se espera a que se presione la tecla Q para finalizar
    ch = 0xFF & cv2.waitKey()
    if ch == ord('q'):

        #Se almacena la imagen y se rompe el ciclo
        cv2.imwrite("Resultado.jpg",salida)
        break

#Se destruyden las ventanas creadas
cv2.destroyAllWindows()
```

Acceso individual

El acceso individual a cada píxel sólo debería ser usado en prototipado de algoritmos, los algoritmos de acceso matricial están optimizados y siempre se recomienda su uso

1. Acceso individual Solución de desafíos

El promedio de cada canal da como resultado la imagen en escala de grises, es decir:

- ImgGris = (R + G + B)/3

```
12 entrada = cv2.imread("Lena.png")
13
14 alto, ancho, canales = entrada.shape
15
16 salida = np.zeros((alto,ancho), np.uint8)
17
18 while(True):
19     for i in range(0, alto):
20         for j in range(0, ancho):
21
22             b0 = entrada.item(i,j,0)
23             g0 = entrada.item(i,j,1)
24             r0 = entrada.item(i,j,2)
25
26             promedio = (b0+g0+r0)/3
27             salida.itemset((i,j),promedio)
28
29     cv2.imshow("Original",entrada)
30     cv2.imshow("Resultado",salida)
31
32     ch = 0xFF & cv2.waitKey()
33     if ch == ord('q'):
34
35         cv2.imwrite("Resultado.jpg",salida)
36         break
37
38 cv2.destroyAllWindows()
```

1. Acceso individual Solución de desafíos

Una alternativa al promedio, altamente usada, es la suma ponderada de cada canal, es decir:

- ImgGris = (0.2126)R + (0.7152)G + (0.0722)B

Esta alternativa nace del hecho de que el ojo humano no percibe con la misma potencia cada canal.

2. Acceso individual Solución de desafíos

Se encuentra el mínimo iniciando en un valor alto que se actualiza cada vez que encuentra un valor más bajo.

Se encuentra el máximo iniciando en un valor bajo que se actualiza cada vez que encuentra un valor más alto

```
import numpy as np
import cv2

entrada = cv2.imread("LCCity.jpg",0)
alto, ancho = entrada.shape
salida = np.zeros((alto,ancho), np.uint8)

while(True):

    minimo = 300
    maximo = 0
    for i in range(0, alto):
        for j in range(0, ancho):

            f = entrada.item(i,j)
            if( f< minimo):
                minimo = f
            if(f > maximo):
                maximo = f

    for i in range(0, alto):
        for j in range(0, ancho):

            f = entrada.item(i,j)
            g = int((f-minimo)*(255.0/(maximo-minimo)))
            salida.itemset((i,j),g)

    cv2.imshow("Original",entrada)
    cv2.imshow("Resultado",salida)

    ch = 0xFF & cv2.waitKey()
    if ch == ord('q'):

        cv2.imwrite("Clase2Desafio2.jpg",salida)
        break

cv2.destroyAllWindows()
```

3. Acceso individual Solución de desafíos

Se realiza la operación por cada canal.

Si bien esto puede mejorar el resultado de la imagen puede realzar canales que por el contexto de la imagen eran bajos.

```
entrada = cv2.imread("LCCity.jpg")
alto, ancho, canales = entrada.shape
salida = np.zeros((alto,ancho,3), np.uint8)

while(True):

    minimos = [300,300,300]
    maximos = [0,0,0]
    for i in range(0, alto):
        for j in range(0, ancho):
            for h in range(0, canales):
                pixel = entrada.item(i,j,h)
                if( pixel < minimos[h]):
                    minimos[h] = pixel
                if( pixel > maximos[h]):
                    maximos[h] = pixel

    print(maximos)
    print(minimos)

    for i in range(0, alto):
        for j in range(0, ancho):
            for h in range(0, canales):
                pixel = entrada.item(i,j,h)
                g = int((pixel-minimos[h])*(255.0/(maximos[h]-minimos[h])))
                salida.itemset((i,j,h),g)

    cv2.imshow("Original",entrada)
    cv2.imshow("Resultado",salida)

    ch = 0xFF & cv2.waitKey()
    if ch == ord('q'):

        cv2.imwrite("Clase2Desafio3.jpg",salida)
        break

cv2.destroyAllWindows()
```

4. Acceso individual Solución de desafíos

Se realiza la operación con base en la escala de grises

```
entrada = cv2.imread("LCCity.jpg")
grises = cv2.imread("LCCity.jpg",0)
alto, ancho, canales = entrada.shape
salida = np.zeros((alto,ancho,3), np.uint8)

while(True):

    minimo = 300
    maximo = 0
    for i in range(0, alto):
        for j in range(0, ancho):

            f = grises.item(i,j)
            if( f< minimo):
                minimo = f
            if(f > maximo):
                maximo = f

    print(maximo)
    print(minimo)

    for i in range(0, alto):
        for j in range(0, ancho):
            for h in range(0, canales):
                pixel = entrada.item(i,j,h)
                g = int((pixel-minimo)*(255.0/(maximo-minimo)))
                salida.itemset((i,j,h),g)

    cv2.imshow("Original",entrada)
    cv2.imshow("Resultado",salida)

    ch = 0xFF & cv2.waitKey()
    if ch == ord('q'):

        cv2.imwrite("Clase2Desafio4.jpg",salida)
        break

cv2.destroyAllWindows()
```

Acceso matricial

Operaciones aplicadas por imagen y no por píxel

Acceso matricial

Las librerías de procesamiento de imágenes cuentan operaciones optimizadas para realizarse sobre toda la imagen.

En este caso las operaciones se miran por imágenes no por píxel

Comparación promedio de imágenes

```
entrada0 = cv2.imread("Lena.png")
entrada1 = cv2.imread("Github.jpg")

while(True):

    salida = cv2.addWeighted(entrada0,0.5,entrada1,0.5,0)
    cv2.imshow("Resultado",salida)

    ch = 0xFF & cv2.waitKey()
    if ch == ord('q'):

        cv2.imwrite("Resultado.jpg",salida)
        break
```

```
entrada0 = cv2.imread("Lena.png")
entrada1 = cv2.imread("Github.jpg")

alto, ancho, canales = entrada0.shape
salida = np.zeros((alto,ancho,3), np.uint8)

while(True):
    for i in range(0, alto):
        for j in range(0, ancho):

            b0 = entrada0.item(i,j,0)
            g0 = entrada0.item(i,j,1)
            r0 = entrada0.item(i,j,2)

            b1 = entrada1.item(i,j,0)
            g1 = entrada1.item(i,j,1)
            r1 = entrada1.item(i,j,2)

            salida.itemset((i,j,0),0.5*b0+0.5*b1)
            salida.itemset((i,j,1),0.5*g0+0.5*g1)
            salida.itemset((i,j,2),0.5*r0+0.5*r1)

    cv2.imshow("Resultado",salida)

    ch = 0xFF & cv2.waitKey()
    if ch == ord('q'):

        cv2.imwrite("Resultado.jpg",salida)
        break
```

Comparación mejora de imagen

```python
import cv2

def Encontrar(imagen):
    minimo,maximo, ret, ret = cv2.minMaxLoc(imagen)
    delta = 255/(maximo-minimo)
    return minimo,delta

entrada = cv2.imread("Lena.png")

while(True):

    b,g,r = cv2.split(entrada)
    bMin,bDelta = Encontrar(b)
    gMin,gDelta = Encontrar(g)
    rMin,rDelta = Encontrar(r)

    b = cv2.subtract(b,bMin)
    g = cv2.subtract(g,gMin)
    r = cv2.subtract(r,rMin)

    b = cv2.multiply(b,bDelta)
    g = cv2.multiply(g,gDelta)
    r = cv2.multiply(r,rDelta)

    salida = cv2.merge((b,g,r))
    cv2.imshow("Resultado",salida)

    ch = 0xFF & cv2.waitKey()
    if ch == ord('q'):

        cv2.imwrite("Resultado.jpg",salida)
        break

cv2.destroyAllWindows()
```

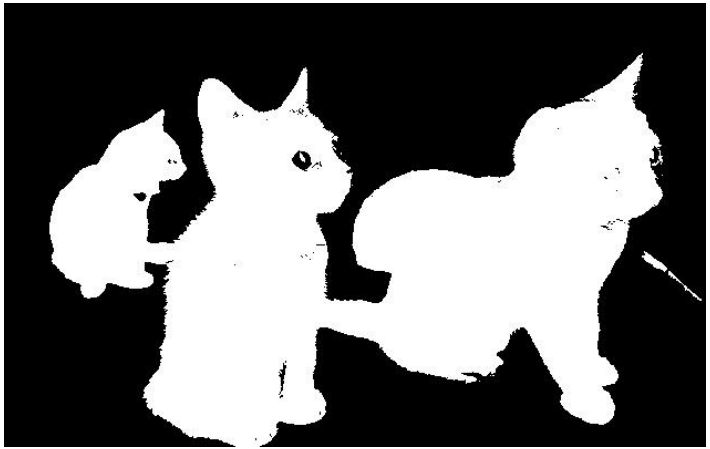
```python
minimos = [300,300,300]
maximos = [0,0,0]
for i in range(0, alto):
    for j in range(0, ancho):
        for h in range(0, canales):
            pixel = entrada.item(i,j,h)
            if( pixel < minimos[h]):
                minimos[h] = pixel
            if( pixel > maximos[h]):
                maximos[h] = pixel

print(maximos)
print(minimos)

for i in range(0, alto):
    for j in range(0, ancho):
        for h in range(0, canales):
            pixel = entrada.item(i,j,h)
            g = int((pixel-minimos[h])*(255.0/(maximos[h]-minimos[h])))
            salida.itemset((i,j,h),g)
```

Desafíos

Acceso píxel a píxel vs matricial

Desafíos

1. Cargar una imagen en escala de grises, recorrer cada píxel, si el valor del píxel actual es superior a 80 se guarda un 255 en la imagen de salida, de lo contrario se guarda un 0
2. Realizar la misma actividad usando el comando InRange



Desafíos

1. Definir un umbral inferior y un umbral superior (Ej: 0,100 | 0, 50 | 50,250), crear una imagen de salida de un canal, cargar una imagen a color, recorrer cada píxel de cada canal, si el valor del píxel se encuentra en el umbral el píxel de salida se pinta de blanco 255, en caso contrario de negro 0. Nota: el operador es tipo “y”, es decir que para el ejemplo B debe estar entre 0 y 100, a la vez que G está entre 0 y 50, a la vez que R está entre 50 y 250
2. Realizar la misma actividad usando el comando InRange

Video

Operación sobre múltiples imágenes

Imágenes en movimiento

El cerebro humano percibe movimiento de 2 formas:

- El encendido y apagado de luces o puntos de forma secuencial
- La transición rápida entre imágenes

Transición rápida entre imágenes

Los videos son transiciones de imágenes hechas a la velocidad requerida para que el ojo humano perciba el movimiento, si no se cumple esta velocidad el cerebro comienza a ver cada imagen y no un video

http://www.mediacollege.com/video/frame-rate/

Transición rápida entre imágenes

Cada una de las imágenes que compone un video es conocido como fotograma o frame y la velocidad de transición de cada fotograma es conocida como fotogramas por segundo o FPS

https://www.geeksaresexy.net/wp-content/uploads/2015/06/fpsdemo1.gif

Captura de la cámara

Para acceder a la cámara en OpenCv debe hacerse uso de un ciclo infinito que se actualice cada a una velocidad específica.

```
#Se crea un objeto tipo captura y se especifica el dispositivo
captura = cv2.VideoCapture(0)
entrada = cv2.imread("Fondo.jpg")

#Se crea un ciclo infinito
while(True):
    #Se captura el fotograma actual
    disponible, fotograma = captura.read()
    if (disponible == True):
        cv2.imshow('Captura',fotograma)

        salida = cv2.addWeighted(entrada,0.35,fotograma,0.65,0)
        cv2.imshow('Resultado',salida)

    else:
        print("Cámara no disponible")

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

# Cuando finaliza libera la cámara y destruye las ventanas
captura.release()
cv2.destroyAllWindows()
```

Desafíos

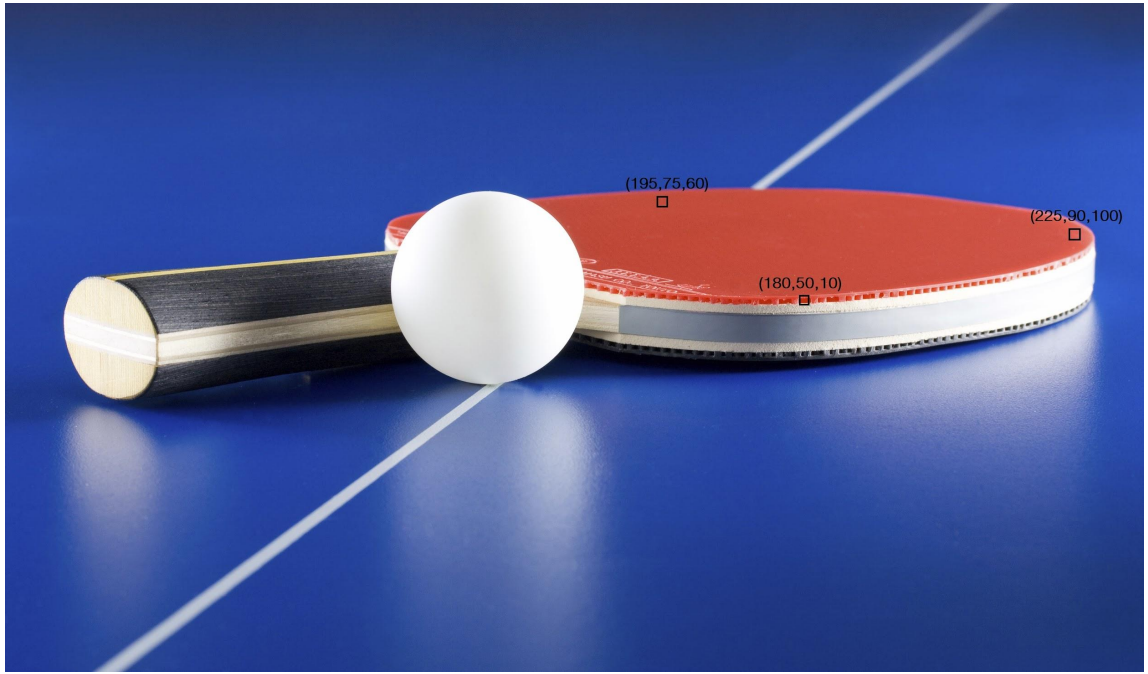
Implementación sobre fotogramas

Desafíos

1. Implementar el algoritmo de mejora de contraste y brillo fotograma capturado
   (+5 puntos)
2. Detectar objetos verdes con la función InRange
   (+10 puntos)

Recomendaciones

La función InRange no busca un color exacto, recibe un valor mínimo y máximo, y selecciona todo lo encontrado en este rango, una opción para encontrar este rango es tomar una foto y mirar puntos estratégicos.



http://www.escuriosity.com/curiosidad.php?curiosidad=Ping-Pong

Actividad Externa

Leer e implementar documentación acerca de corrección de color en múltiples escenas, esto será usado para el Chroma.

Esta actividad externa hará parte de la calificación del proyecto y podrá ser realizada de forma grupal

- Requisitos: mínimo 3 referencias y al menos 1 de las referencias debe ser de una revista indexada.

Actividad Externa

Palabras clave:

- Color - light correction
- Color - light matching
- Color - light grading

Ejemplo antes de corrección

Ejemplo después de corrección

Referencias

Usar siempre la referencia de 3.0 con el buscador de la página

http://docs.opencv.org/3.0-beta/modules/refman.html

En algunos casos puede usarse la documentación de 2.4

http://docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html