Procesado de imagen y visión por computador
Curso 2018/2019
Programación de aplicaciones. Eventos

1. Manejo de eventos con OpenCV

1.1. Eventos del teclado

Con OpenCV es posible capturar los eventos del teclado con distintas funciones propias. En concreto, la función *cv2.waitKey(delay)→ retval* detiene la ejecución del programa durante el tiempo especificado en *delay* o hasta que el usuario pulsa una tecla (si *delay* es cero, se detiene indefinidamente hasta que se pulse una tecla). Además, si el usuario pulsó una tecla, devuelve el código de ésta, por lo que puede ser empleada en instrucciones de comparación dentro del código, y ejecutar acciones distintas en función de la tecla pulsada. Es importante saber que esta función sólo funciona si hay una ventana activa de OpenCV.

A la hora de comparar el código devuelto, es necesario tener en cuenta que la función devuelve un valor *int* (32 bits). Para poder comparar u obtener las teclas pulsadas tenemos las funciones de Python: *ord()* que devuelve el entero correspondiente a un carácter y *chr()* que devuelve el carácter asociado a un entero (8 bits):

```
>>> ord('q')
113
>>> chr(113)
'q'
>>>
```

El uso más común el valor devuelto es incluirlo dentro de estructuras *if* o *switch* para, en función de la tecla pulsada, realizar una acción u otra, como se verá a continuación.

Hay que tener en cuenta que las teclas especiales del teclado modificarán el código de salida de la función. Los 16 bits más significativos representan la combinación de teclas modificadoras pulsadas, tales como *Ctrl*, *Atl*, *May*, etc. Los 8 menos significativos contienen el carácter ASCII en cuestión. Por ello muchas veces veremos en los códigos de Python algo como:

```
if cv2.waitKey(20) & 0xFF == ord('q'):
        break
```

En este ejemplo comprobamos cómo solo se comparan los 8 bits menos significativos para obtener la tecla pulsada.

1.2. Eventos del ratón

El uso más común de los eventos del ratón en procesado de imagen es la de poder marcar coordenadas dentro de una imagen de forma rápida. Para ello, necesitamos la capacidad de capturar la posición del ratón cuando el usuario pulsó alguno de sus botones, y en ocasiones también cual fue el botón pulsado (botón izquierdo, botón derecho, rueda central, …). En OpenCV, para poder realizar esto, necesitamos relacionar la ventana de la imagen donde el usuario debe pulsar, con la función que atenderá los eventos que ésta genere. Es necesario tener en cuenta que cada ventana puede tener su propia función, es decir, si el usuario pulsa con el ratón en una ventana, se ejecutará una función concreta, y si lo hace sobre otra, se podrá ejecutar otra función.

Para relacionar una ventana con la función que se ejecutará cuando el usuario actúe con el ratón sobre dicha ventana, emplearemos la siguiente función:

- `cv2.setMouseCallback(ventana, funcion, datos)`
  - `"ventana": Nombre de la ventana de OpenCV.`
  - `"funcion": Nombre de la función que atiende el evento`
  - `"datos": Opcional.`

Aparte de la llamada a dicha función (siempre después de haber creado dicha ventana, por ejemplo, con *imshow*), también es necesario definir la función que atenderá al evento del ratón. Dicha función tiene la siguiente forma:

- `def funcion(evento, x, y, flags, datos):`
  - `"evento": Tipo de evento: click, dbl click, mouse over,...`
  - `"x, y": Cordenadas del puntero del ratón`
  - `"flags": Teclas modificadoras (Ctrl, Alt, May, ...)`
  - `"datos": Opcional.`

Dichas funciones suelen recibir el nombre de *callback*, por lo que a partir de ahora las denominaremos así. En general, será necesario chequear en primer lugar el tipo de evento (por ejemplo, detectar si se ha pulsado el botón derecho o el izquierdo del ratón), para realizar la acción oportuna, o incluso no hacer nada. La lista completa de eventos disponibles puede consultarse en varias web, por ejemplo: http://opencvexamples.blogspot.com/2014/01/detect-mouse-clicks-and-moves-on-image.html.

El archivo *Apartado1.py* implementa un programa que, después de abrir una imagen y mostrarla en la pantalla, asocia a la ventana "*Imagen*" la callback *EventoRaton*, función que está implementada en el mismo archivo. Finalmente, crea un bucle infinito que sólo finaliza cuando el usuario pulsa la tecla *Esc*. Dentro del bucle el programa atiende tanto a los eventos del teclado (cada vez que el usuario pulsa una tecla, se imprime por pantalla el código de la tecla pulsada), como a los del ratón (cada vez que el usuario pulsa con el ratón en la ventana, imprime por pantalla el tipo de evento y las coordenadas donde se pulsó).

Así mismo, el archivo demuestra cómo pasar variables a la callback de atención al evento. En este caso, pasamos a la callback el color deseado para pintar los puntos al hacer doble click. Si fuese

necesario pasar más de una variable a esta callback, la forma más simple sería crear una lista o tupla donde empaquetar todas las variables. Las variables que hemos usado son variables globales accesibles desde todo el programa, podríamos usar el parámetro *datos* sin sólo queremos que sean accesibles por la función de callback.

Utilizando como partida el archivo *Apartado1.py*, modificar dicho archivo para que, utilizando los eventos *cv2.LBUTTONDOWN* y *cv2.LBUTTONUP*, marcar con el ratón un rectángulo dentro de la imagen. Posteriormente, si el usuario pulsa la tecla +, pintar dicho rectángulo con una línea de más grosor y si pulsa la tecla – reducir el grosor. Para pintar el rectángulo sobre la imagen, se puede utilizar la función *rectangle()* de OpenCV.

Hay que tener en cuenta que el proceso debe hacerse en dos pasos. El primer paso sería cuando se pulsa el botón sobre la imagen (evento *LBUTTONDOWN*), lo que constituirá una de las esquinas del rectángulo. Por tanto, será necesario capturar y guardar las coordenadas del ratón en ese momento. El segundo paso sería cuando se suelta el botón (después de haberlo arrastrado). Estas coordenadas constituirán la otra esquina del rectángulo.

2. Barras de desplazamiento

Las barras de desplazamiento constituyen una manera cómoda de modificar el valor de una variable o parámetro en un programa de forma interactiva. Una barra de desplazamiento está asociada a una variable, y a medida que se desliza el cursor a lo largo de la barra, el valor de esta variable varía entre un valor mínimo y un valor máximo. Aunque la librería OpenCV sólo incluye un tipo de barra, ésta será suficiente para nuestras necesidades.

Una barra de desplazamiento en OpenCV se crea con la siguiente instrucción:

- `cv2.createTrackbar(nombre, ventana, valor, maximo, callback)`
  - `"nombre": nombre de la barra de desplazamiento`
  - `"ventana": ventana OpenCV sobre la que se acopla la barra`
  - `"valor": valor inicial de la barra`

La instrucción anterior acopla la barra de desplazamiento horizontal sobre la ventana indicada. El valor asociado a esta barra puede variar entre 0 y el valor indicado en "*maximo"*. Podemos acceder al valor de la barra de dos formas distintas:

1. Usando *cv2.getTrackbarPos()* que nos devuelve el valor entero correspondiente a la barra.
2. Si en "*callback*" especificamos el nombre de una función con un argumento, cada vez que se cambie el valor de la barra, se lanzará un evento que ejecutará esta función (igual que, por ejemplo, con los eventos del ratón vistos antes). El argumento que hayamos definido en esa función tomará el valor de la barra.

La opción 1 permite una programación más cómoda del código, ya que para obtener el valor de la barra, basta con leer su valor. Sin embargo, si queremos que la información se actualice de forma inmediata, deberíamos acceder continuamente a dicho valor, lo cual puede ralentizar la ejecución del programa. La opción 2 es más eficiente, ya que sólo se ejecuta cuando realmente se produce un cambio en el valor de la barra.

El archivo *Apartado2.py* implementa un ejemplo de uso de una barra de desplazamiento siguiendo la opción 1. El código modifica los colores de una imagen dando más o menos peso a cada una de la componentes de color. Analice el código para entender bien el funcionamiento de las barras de desplazamiento.

Crear un nuevo programa que haga la misma operación pero usando la opción 2. Sólo deberá modificarse la componente que hayamos tocado y la nueva imagen sólo se mostrará tras esa modificación.