UNIVERSIDAD POLITECNICA DE VALENCIA

Interacción con OpenCV:

Seguimiento de un objeto por color

Apellidos, nombre	Agustí Melchor, Manuel[1] (magusti@disca.upv.es) Armenteros Gutiérrez, Jose (joargu@ei.upv.es)
Departamento	[1]Dpto. De Ing. De Sistemas y Computadores
Centro	Universidad Politécnica de Valencia

UNIVERSIDAD POLITECNICA DE VALENCIA

1 Resumen de las ideas clave

En este artículo vamos a presentar uno de los ejemplos que acompañan a la librería OpenCV [1, 2] encaminada al desarrollo de aplicaciones que utilizan técnicas de Visión por Computador (VxC). El ejemplo será primero examinado para, después, proponer ampliaciones sobre él. De este modo se podrá disponer de un bloque útil para la realización de aplicaciones multimedia interactivas mediante el uso de la VxC

Para ello tomaremos como partida el ejemplo (*camshiftdemo*) que acompaña a las librerías OpenCV y que se ha probado sobre la versión 1.1 y 2.1 de la misma. Para su desarrollo nos centraremos en la plataforma GNU/Linux, aunque todo lo expuesto es transportable a otras en que se haya instalado esta librería y se disponga de acceso a una cámara USB.

2 Introducción

Este artículo mostrará cómo realizar un ejercicio de interacción entre el usuario y el computador utilizando este segundo una cámara para tener acceso a la información visual de su entorno (la escena). Dentro del campo de la interacción persona-ordenador (IPO), a este tipo de sistemas se les suele llamar interfaces perceptuales.

Partiendo de un ejemplo que acompaña a la librería OpenCV (*camshiftdemo*) se expondrá su secuencia de funcionamiento, para dar pie a cómo realizar sobre ese mecanismo una interacción entre el escenario real (el del humano) y el virtual (el de la máquina).

3 Objetivos

Una vez que el lector haya leído este documento y explorado el código que se referencia, será capaz de:

- Seguir la ejecución del ejemplo *camshiftdemo*.
- Explicar la adaptación propuesta para los casos particulares expuestos sobre la base del ejemplo *camshiftdemo*.
- Proponer nuevos ejemplos de interacción.

4 El ejemplo *camshiftdemo* en OpenCV

Dentro de los ejemplos que acompañan a la librería, *camshiftdemo*[1] implementa un algoritmo de seguimiento de objetos en la escena conocido como CAMSHIFT [5]. Este algoritmo calcula la posición (centro de gravedad), orientación y área en la imagen que ocupa el objeto de interés. Este algoritmo se propuso inicialmente en diferentes aplicaciones de realidad virtual y entornos inmersivos como elemento de interfaz natural

Figura 1: Ejemplo de resultado de aplicación de CAMSHIFT (Imagen obtenida de [5]).

El algoritmo, a partir de un color dado, seleccionará de la imagen los puntos con el color indicado (como se muestra en la Fig. 1 Derecha), definiendo así un "objeto"como el definido por el conjunto de puntos de la imagen al color dado. El "objeto" (color) será monitorizado para dar sus coordenadas de posición, orientación y extensión en la escena (Fig. 1 izquierda). En el recuadro del listado 1 se resume la parte central de camshiftdemo para seguir un objeto caracterizador por un color en la escena.

(1) Seleccionar el área de interés, con ello obtener el color a seguir.
(2) Para cada imagen RGB obtenida de la cámara
    (2.1) Convertir a HSV.
    (2.2) Segmentar la componente *Hue* con los parámetros *Vmin*, *Vmax*, *Smin*.
    (2.3) Etiquetar los puntos que están alrededor del color indicado por el usuario.
    (2.4) Calcular X,Y (centro de masas), Z (tamaño del área) y ángulo del eje mayor con la horizontal.
    (2.5) Dibujar un a elipse centrada en X,Y con diámetro el ancho del área de puntos de ese color.
fPara

Listado 1: Algoritmo básico ejecutado por camShiftDemo.

[1]En la instalación de la versión 2.1 de OpenCV sobre GNU/Linux aparece en /usr/share/doc/opencv-doc/examples/c/c/camshiftdemo.c.gz.

El ejemplo incluido en OpenCV permite seleccionar, mediante el ratón, un área rectangular sobre la imagen obtenida de la cámara, “en vivo”, De este área se seleccionará el color predominante. La imagen, originalmente definida utilizando coordenadas de color en el espacio RGB (con componentes *Red*, *Green*, *Blue*), es convertida a HSV (con componentes *Hue*, *Saturation*, *Value*), donde el interfaz gráfico ofrece la posibilidad de ajustar el comportamiento del algoritmo: su margen de variación respecto a lo que puede variar el color buscado se realiza mediante tres controles (*Vmin, Vmax y Smin*) que darán una cierta tolerancia a los cambios de iluminación en la escena.

Si lo que se quiere seguir es la cara del usuario (como en la Fig. 1) u otro objeto (por ejemplo, una esfera de color), habrá que procurar que, en la escena, ese no sea un color que aparezca también. En caso contrario se obtendrán los valores correspondientes al “objeto” formado por lo que se quiere seguir más la parte de la escena que coincide en color con nuestro “objeto” de interés.

Visualmente se puede ajustar el intervalo (o rango de valores) que la aplicación ha de considerar como que definen el “color” a localizar en la escena: usando la tecla 'b' se intercambia el contenido que se visualiza en la ventana principal de la aplicación: se puede ver la imagen de la cámara (variable *image*) ó los puntos de la imagen (variable *backproject)* que están alrededor del valor de color definido.

Este color, el predominante en el área definida por la selección del ratón sobre la imagen de la cámara, se puede visualizar mediante el histograma de la componente *Hue* que se muestra al pulsar la tecla 'h'.

El protocolo algoritmo por la versión original del *camshiftdemo* establece un protocolo en dos partes.

- La primera parte se realiza en base a dos variables que indican en qué punto de este proceso de “aprendizaje” o determinación del color está la aplicación.
    - Cuando el usuario empieza a definir un área (detectadp en ña función *on_mouse*, dentro del procesado del evento *CV_EVENT_LBUTTONDOWN*), se asigna el valor -1 a la variable *track_object. En el programa principal se detecta esta situación y se ejecutan las instrucciones correspondientes al cálculo sobre el área que en el momento presente se esté señalando, conforme esta cambia el cálculo se actualiza de forma dinámica.* → s'ha seleccionat un àrea, hay que recalcular el color predominante. Al terminar de definir el área (evento *CV_EVENT_LBUTTONUP) se cambia el valor de la variable track_object a 1 y se utiliza el valor de color decidido. Mientras no se pulsa la tecla* 'c' (*que establece la variable track_object* a *0) se mostrará el seguimiento del “objeto” en pantalla.*
    - El área de la que extraer el color se inicia al pulsar y se actualiza mientras se lleva el botón izquierdo pulsado. Al soltar el botón, se habrá guardado en *selection* la información relativa al área que ha seleccionado el usuario y se indicará que ha acabado la operación (*select_object = 0*).
- La segunda es la correspondiente a la salida de la aplicación. Originalmente, se muestra únicamente el área de puntos del color elegido mediante el dibujo de una elipse sobre impresa a la imagen que

obtiene la cámara. De esta forma se muestra la extensión y la orientación que toma el objeto formado por el conjunto de puntos (de la imagen) que tienen ese determinado color con la instrucción

```
cvEllipseBox( image, track_box, CV_RGB(255,0,0), 3, CV_AA, 0 );
```

que utiliza como segundo parámetro un tipo (definido en *cxtypes.h*) como

```
typedef struct CvBox2D
{
 CvPoint2D32f center;        /* Center of the box.  */
 CvSize2D32f  size;          /* Box width and length.*/
 float angle;                /* Angle between the horizontal axis */
                             /*  and the first side (i.e. length) in degrees */
}CvBox2D;
```

que le proporciona toda la información necesaria para decidir los parámetros de la elipse circunscrita a ese rectángulo. (véase Fig. 2).

Figura 2: Ejemplo de salida de camshiftdemo [6].

5 Interacción natural mediante el color

Veamos ahora que se puede instruir a la máquina para que, ajustándose a las condiciones de la escena, realice el trabajo que tiene encomendado. Básicamente, cambiar la forma de implementar los pasos 1 y 2.5 del listado 1.

Para que este algoritmo se pueda utilizar como un medio de interacción entre el usuario y la máquina es necesario que se establezca un protocolo de actuación igual de flexible, en tanto que permita cambiar los parámetros de funcionamiento de la aplicación, pero más guiado para que no tenga que ser conocedor de los detalles el usuario final. Con estas ideas se propone:

- Que la determinación del objeto a seguir se convierta en una “presentación” del objeto a seguir (en realidad un color) a la aplicación. Esto consiste en que la aplicación pueda calibrar o determinar los parámetros de su actividad en función de una serie de elementos cuya correspondencia de los valores a determinar pueda ser establecida. En este caso el intervalo de color a monitorizar en la escena.
  Para ello se propone que la aplicación tenga una opción de configuración que sea pedir al usuario que situé el objeto en una predeterminada posición de la escena y transcurrido un tiempo, tomar ese área como el que define el usuario en el ejemplo original. Así se dispone de una posición inicial (X, Y, Z y ángulo).

- Ajustar o convertir los valores obtenidos a los esperados por la aplicación final. El caso más genérico sería el de generar sintéticamente los eventos que desencadenaría cualquier elemento hardware, de clase posición 3D, de interfaz.
  Por minimizar el código, se ha pensado que se utilice directamente la información para presentar objetos en pantalla con los que realizar una actividad lúdica interactiva.

6 ¡Atrápalo! con OpenCV

En un trabajo [4] de la asignatura Integración de Medios Digitales (IMD[2]), tomando como base estas ideas se planteó hacer un sencillo juego que permitiera "alcanzar" objetos del mundo virtual a través del movimiento del usuario en la escena que procesa el computador. Para ello se habrá de mostrar delante la cámara un objeto y seleccionarlo. Se decidió que esta identificación del objeto se haría a partir del color por la capacidad de ajustar a condiciones de luz cambiantes y rapidez de decisión del ejemplo explorado.

Basado en el ya comentado ejemplo de la función *cvCamShift* (*camshitfdemo.c*), la variación inicial consistió en dibujar un círculo que representa el objeto a tocar y que la salida de *cvCamShift* mostrara la posición del objeto que mueve el usuario. Cuando la distancia entre los dos objetos fuera casi 0, se considerará alcanzado el objeto y se creará uno nuevo (se redibujará) en otra posición. Creando así un pequeño juego de perseguir indefinidamente un objeto como se muestra la Fig. 3. El objeto rojo es movido por la acción del usuario con la capucha azul del bolígrafo. El objetivo es tocar el círculo verde.

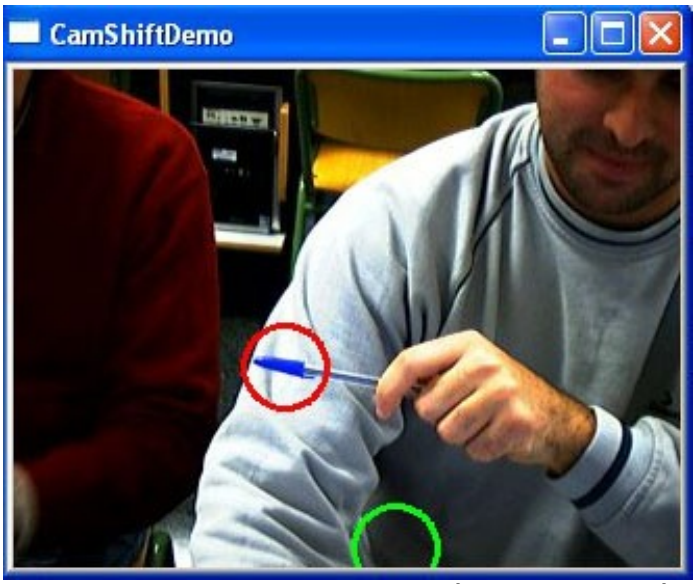

Figura 3: Planteamiento de la modificación sobre el código original.

[2]La URL de esta asignatura es <http://web-sisop.disca.upv.es/~imd>

Para que el programa supiera cuándo ambos círculos entraban en contacto se puede plantear que se detecte la intersección de ambas áreas. Este cálculo es relativamente complejo en tanto que cada círculo se posiciona en el espacio de la ventana y su centro es la coordenada (0,0) con lo que tiene regiones de áreas positivas y negativas.

También se puede optar, que es la opción recomendada y utilizada en aplicaciones reales de carácter lúdico, por una fórmula mucho más sencilla y eficiente: el cálculo de la distancia Euclídea, que obtiene la distancia entre dos puntos, en este caso los centros de las circunferencias, como muestra la ecuación 1.

$$d(X,Y) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Ecuación 1: Distancia euclídea entre dos puntos de un espacio bidimensional.

A partir de aquí se va modificando el programa original aportando nuevas funcionalidades:

- Se prosigue pretendiendo eliminar la interacción con el cursor, *camshiftdemo* lo usa para elegir el objeto a rastrear. Para ello se dibuja un rectángulo en la pantalla y, pasado cierto tiempo, el programa marcará lo que haya dentro de dicho rectángulo. Véase Fig. 4.

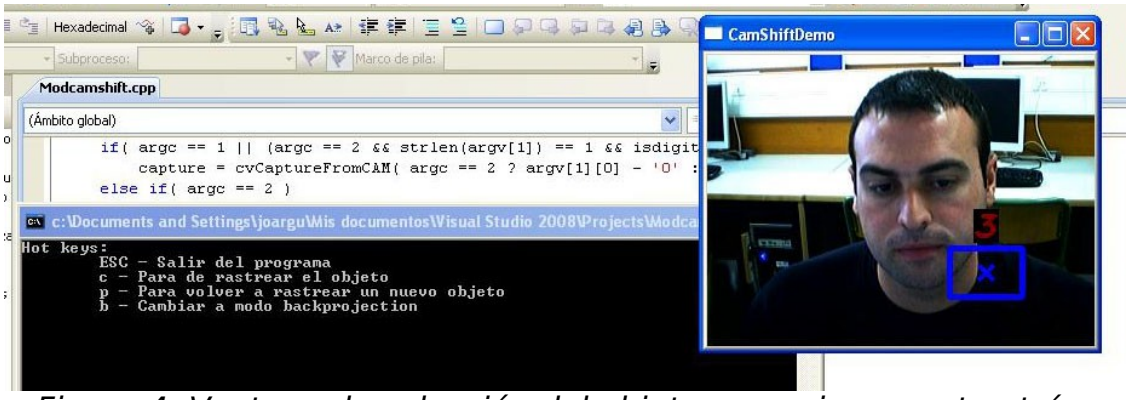

Figura 4: Ventana de selección del objeto a seguir y cuenta atrás.

- Para mostrar el tiempo que queda para “enseñarle” el objeto a la aplicación, se escribirá un dígito cuyo valor va decreciendo. Para ello, en lugar de bloquear el programa durante un cierto periodo de tiempo (con las funciones *sleep* o *wait*, por ejemplo), se tomará un número de cuadros antes de cambiar el valor. Así se podría ajustar incluso durante ese periodo.
- Se crea un contador interno para que el juego finalice tras haber tocado el círculo objetivo “x” veces.
- A fin de facilitar al usuario la interfaz con el programa se sustituyó la función de dibujo de la elipse de seguimiento del objeto por una función de círculo de radio fijo. Así, aunque hubieran ligeras variaciones sobre el

tamaño del objeto a seguir serían transparentes al “jugador” y el centro siempre queda claramente definido con lo que aumenta la jugabilidad.

- Se invierte la imagen que se procesa, para evitar el efecto espejo y que sea más sencillo alcanzar el objeto a seguir
- Si el objeto que se intenta rastrear es demasiado grande, bien sea por la distancia que tenemos a la cámara o debido a que el color que buscamos capturar se confunde con el fondo, se para de rastrear y se devuelve un mensaje (Fig. 5) pidiendo al usuario que se reinicie la captura del objeto.

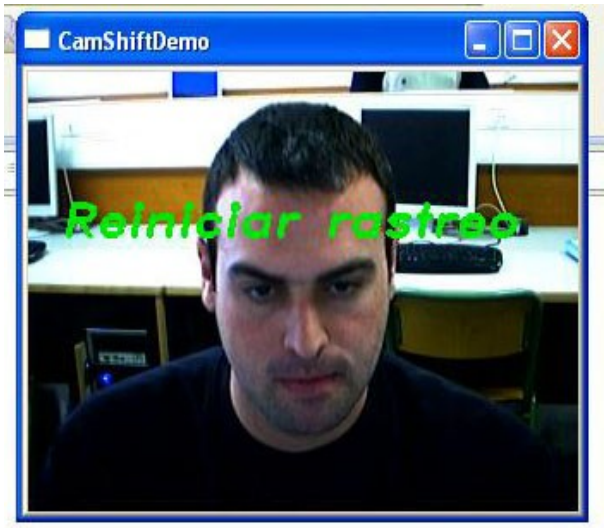

Figura 5: El programa sugiere reiniciar el proceso si la probabilidad de encontrar el objeto es baja.

7 Conclusión

Este artículo ha explorado un ejemplo de interacción entre el usuario y el computador mediante el uso de una cámara para que el segundo pueda tener acceso a la información de su entorno exterior que cabe dentro de los denominados interfaces perceptuales.

Sobre el ejemplo que acompaña a la librería OpenCV (*camshiftdemo*) se ha expuesto su secuencia de funcionamiento, la adaptación básica a una aplicación multimedia interactiva y la necesidad de introducir otros aspectos en el funcionamiento que acompañen al usuario en su uso

Al realizar este trabajo se ha mostrado como el hecho de reutilizar piezas existentes es posible para el prototipado rápido.

Trabajar con librerías multiplataforma y sin ninguna atadura al sistema operativo permite portar esta aplicación a cualquier sistema y entorno de desarrollo para la creación del ejecutable. La más sencilla es la línea de órdenes que para este caso es de la forma:

```
$ gcc -I/usr/local/include/opencv  -L/usr/local/lib -lcxcore -lcv -lhighgui -lcvaux -lml  camshiftdemo.c -o camshiftdemo)
```

O, en GNU/Linux con *pkg-config* instalado:

```
$ gcc `pkg-config opencv --cflags` `pkg-config opencv --libs` camshiftdemo.c -o camshiftdemo)
```

El uso de la visión por computador promete, en el futuro cercano, ofrecernos alternativas naturales a los modos de interacción computacional a que estamos acostumbrados. Esperamos que ayudará a salvar barreras y crear un manejo más intuitivo de las aplicaciones y dispositivos.

Ahora, apreciadlo lector, es tu turno. Te animo a ponerte manos a la obra sobre el ejemplo y construir tu propio juego o tu sistema interactivo personal. Después, compártelo con todos.

8 Bibliografía y referencias

[1] “Open Computer Vision Library”. Disponible en <http://sourceforge.net/projects/opencvlibrary/>. Última consulta mayo de 2011.

[2] “OpenCVWiki” , Disponible en http://opencv.w.illowgarage.com/wiki/Welcome.

[3] Vadim Pisarevsky, “Introduction to OpenCV”, Intel Corporation, Software and Solutions Group.

[4] Jose Armenteros Gutierrez. Salvapantallas activo con cámara web (Camshift). Trabajo de la asignatura IMD <http://web-sisop.disca.upv.es/~imd>, curso 2008/2009.

[5] Gary R. Bradski. Computer Vision Face Tracking For Use in a Perceptual User Interface (1998). Disponible en <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.14.7673>.

[6] Robin Hewitt . Seeing With OpenCV - A Five-Part Series. Part 3 (March 2007), Follow That Face! Disponible en <http://www.cognotics.com/opencv/servo_2007_series/index.html>. Última consulta enero de 2011.