Práctica de Técnicas de Inteligencia Artificial. Primera práctica: Visión

Otto Colomina, Patricia Compañ, Francisco Escolano y Diego Viejo
Departamento de Ciencia de la Computación e Inteligencia Artificial
Universidad de Alicante

2 de marzo de 2010

1. Descripción de la práctica

La práctica de visión artificial para el curso 2009/10 consistirá en desarrollar un sistema de vídeo-vigilancia. El objetivo de este sistema será realizar un control de las personas que aparecen delante de la cámara de vídeo. Estos sistemas pueden ser utilizados en sistemas de control de paso o control de acceso.

Para desarrollar la práctica, el alumno dispondrá de una sucesión de imágenes consecutivas que han sido capturadas una cámara de vídeo. El objetivo será mostrar una segmentación de las caras de la persona o personas que aparezcan en la secuencia de vídeo.

1.1. Primera fase

En esta primera parte de la práctica de visión, el alumno se centrará en desarrollar una función de JavaVis que procese una imagen y obtenga la zona o zonas donde aparezca la cara de una persona. La segmentación se realizará por el color de la piel. Para segmentar imágenes utilizando color, el método más efectivo consiste en transformarlas previamente del modelo de color RGB original al modelo HSB que permite una segmentación por color mucho más robusta. Sin embargo, si solamente realizamos una segmentación por color se producirán numerosos falsos positivos, esto es, detectaremos cualquier cosa que tenga el mismo color que la piel aunque no se corresponda con la cara de una persona. Por ejemplo sus brazos o cualquier otro objeto de la escena con un color similar. El procesamiento de la imagen se llevará a cabo en los siguientes pasos:

1. Binarizado de la imagen: la imagen RGB original se transformará a una binaria en la que los píxeles que se correspondan con el color de la piel de una persona deben estar a 1 y el resto a 0. Para ello se emplearán la funciones FRGBToColor y FSegmentHSB. La primera convierte la imagen en formato

RGB en formato HSB. La segunda toma la imagen HSB y realiza un binarizado dejando a 1, en la imagen resultante, aquellos píxeles que estén dentro de unos rangos dados en cada una de las bandas HSB.

2. Reducción del ruido. Sobre la imagen binaria obtenida en el paso anterior, se aplicará una técnica basada en morfología matemática para reducir el ruido de la imagen. La operación a realizar se conoce como *cierre morfológico* y consiste en aplicar una dilatación y a continuación una erosión sobre la imagen. Con ello conseguiremos cerrar pequeños huecos dentro de zonas contiguas y similares de la imagen. En este caso, utilizaremos la función FClousure de JavaVis. Dicha función necesita tener definido el elemento estructurante que se utilizará para las operaciones morfológicas. Podemos encontrar un fichero con un elemento estructurante estándar en el directorio images de JavaVis. Su nombre es ee.txt. Para obtener el nombre del fichero, declararemos un parámetro de entrada a la función llamado **ee**.

3. Segmentación de las zonas candidatas. Para separar las distintas zonas de la imagen binaria resultante al aplicar los pasos anteriores se utilizará la función FBlobs de JavaVis. Esta función se encarga de buscar las zonas conectadas de la imagen, es decir, aquellas zonas con valor *true*. Esta función tiene definido un parámetro de salida llamado **blobs** que contiene la información sobre dichas zonas conexas (en adelante blobs).

4. Supresión de falsos positivos. Como se ha comentado anteriormente, la mayoría de los blobs encontrados en el paso anterior no se corresponderán con la cara de una persona. Solamente han sido segmentados por el color y con esta única característica no es suficiente para distinguir entre una cara y otro objeto del mismo color. Una heurística sencilla consiste en buscar ciertos atributos morfológicos que nos indiquen si lo que estamos segmentando puede ser una cara o no. Así, en primer lugar descartaremos los blobs que no tengan el tamaño suficiente, es decir, que no tengan un número mínimo de puntos. En segundo lugar, habrá que descartar aquellos blobs cuya relación de ancho-alto no sea la de una cara típica. Finalmente, también conviene eliminar aquellos blobs que, aún habiendo superado las dos comprobaciones anteriores, no tienen un número suficiente de puntos a *true* en relación con su tamaño.

El alumno tendrá que implementar todo este proceso dentro de una función JavaVis llamada FDetectaCaras. Dicha función deberá tener declarados todos los parámetros de entrada necesarios para ajustar su funcionamiento. Al final del proceso, esta función devolverá una imagen de tipo geométrico con recuadros que marquen las zonas de la imagen de entrada donde se detectaron caras. Además, la función debe incluir un parámetro de salida llamado **blobs** donde se devuelva un ArrayList con solamente los blobs correspondientes a las caras detectadas. Esta información se utilizará en fases posteriores de la práctica.

1.2. Segunda Fase

En una segunda fase, el alumno desarrollará un método que permita analizar la secuencia completa de imágenes. En este caso también se realizará una segmentación de la imagen, pero esta vez se utilizará información de movimiento. Por movimiento vamos a considerar la diferencia en valor absoluto de la intensidad de color de un píxel con el mismo píxel de la imagen siguiente en la secuencia. La segmentación por movimiento nos permitirá reducir el área de búsqueda de las caras en una imagen. El proceso a seguir es el siguiente:

1. Detectar movimiento. Por cada par de imágenes se generará una imagen binaria cuyos píxeles con valor 1 se corresponderán con aquellos cuyo valor de movimiento, calculado como se acaba de explicar, supere un cierto umbral. Para llegar a este resultado, previamente habrá que convertir las imágenes a formato *byte* con la función FColorToGray de JavaVis. También es posible utilizar las funciones FBinarize, para binarizar la imagen resultante de restar las imágenes de entrada ya convertidas, y FClousure, para eliminar huecos en el resultado.

2. Segmentar el movimiento. Siguiendo un proceso de segmentación similar al de la primera fase, se utilizará la función FBlobs para extraer las distintas partes de la imagen donde se haya detectado el movimiento.

3. Recortar la imagen. Utilizando la información de los blobs extraídos por movimiento, se generará una imagen por cada blob con las dimensiones de este y con la información de color del área cubierta por el blob en la imagen original (la segunda de las dos imágenes en cada par). De este modo se tendrán imágenes de menor tamaño que llamaremos imágenes de movimiento.

4. Detectar caras. Utilizando la función FDetectaCaras desarrollada en la fase 1 se obtendrán los blobs correspondientes a las caras que se encuentren en las imágenes recortadas en el paso anterior. Utilizando estos blobs y dichas imágenes de movimiento se obtendrán las imágenes de las caras que aparezcan en ellas.

5. Almacenar de los resultados. Las imágenes de las caras que se vayan extrayendo se almacenarán de la siguiente manera. Las caras de un mismo individuo se almacenarán en un ArrayList de Java. Para estimar cuando una cara pertenece a un mismo individuo se realizará la búsqueda del blob asociado más cercano de los extraídos en la imagen anterior. Se supondrá que éste corresponde al mismo individuo y por lo tanto se almacenará en el mismo ArrayList. Se considerará que un individuo que vuelve a entrar en escena es un nuevo individuo, aunque haya pasado antes. Del mismo modo, se considerará un nuevo individuo si en algún momento de la secuencia deja de detectarse su cara. Finalmente, se utilizará un parámetro de salida de la fun-

ción llamado **caras** de tipo ArrayList y que contendrá los ArrayList con las caras de cada uno de los individuos que aparecieron delante de la cámara.

El alumno deberá implementar este proceso dentro de una función de JavaVis llamada FPracVision. Dicha función deberá tener declarados todos los parámetros de entrada necesarios para ajustar su funcionamiento, no así los parámetros de la función FDetectaCaras, que ya habrán sido ajustados en la primera fase y, por lo tanto, podrán ser establecidos directamente mediante una línea de código. La función FPracVision deberá sobreescribir el método processSeq para poder manejar una secuencia de imágenes. Finalmente, aparte del parámetro de salida **caras** la función processSeq devolverá una secuencia de imágenes geométricas con los recuadros de las caras encontradas entre cada par de imágenes. Cada imagen geométrica irá seguida de la imagen original.

2. Parte experimental

Aparte del proceso de implementación de las funciones FDetectaCaras y FPracVision, el alumno deberá desarrollar un proceso de ajuste de los distintos umbrales, parámetros y rangos utilizados en dichas funciones. El resultado de la parte experimental permitirá encontrar un conjunto óptimo de valores que aseguren el mejor resultado al aplicar todo el proceso. Además, la parte experimental se completará con pruebas de rendimiento que muestren el tiempo necesario para realizar todo el procesamiento.

3. Documentación

La práctica irá acompañada de la correspondiente documentación. En dicho documento, el alumno explicará y justificará los detalles de diseño de la función implementada, flujo de datos, resultados parciales y resultado final. Además, en la documentación tendrá que aparecer una detalla descripción del método utilizado para realizar la parte experimental así como los resultados obtenidos durante la misma y una breve discusión en la que se analice el funcionamiento de la función.

4. Parte optativa

Como parte optativa de la práctica se deja al alumno la posibilidad de mejorar cualquier aspecto del proceso de detección de caras como, por ejemplo, mejorar la segmentación del color de la piel, hacer una detección más robusta de las caras, mejorar el seguimiento de las caras frame a frame, etc. Aparte de su implementación, para que la parte optativa se puntúe correctamente deberá estar acompañada de la correspondiente documentación con un apartado específico donde se indiquen las mejoras realizadas, y por la correspondiente experimentación donde

se demuestre que la mejora propuesta efectivamente mejora el funcionamiento del sistema.

5. Normas de entrega y evaluación de la práctica

La práctica es individual. Cualquier indicio de plagio supondrá el suspenso de la parte de prácticas y por consiguiente de la asignatura hasta Julio. Se usarán sistemas informáticos de detección de copia.

La práctica se valorará de la siguiente manera:

- 30 % por código y funcionamiento. Se probarán varios ejemplos y se valorará el funcionamiento final. Si la práctica no tiene un mínimo de funcionamiento, se suspenderá por completo sin tener en cuenta los apartados siguientes. En cuanto al código, deberá estar perfectamente estructurado, sin código comentado, con indentación y haciendo un uso adecuado del lenguaje. El código deberá estar debidamente comentado (incluso para la generación de javadoc). Se puntuará también el respeto por los nombres de las clases y parámetros indicados en este enunciado.
- 20 % por la experimentación.
- 30 % por documentación. La documentación se podrá entregar en cualquier formato. Será responsabilidad del alumno asegurar que la documentación pueda ser leída en cualquier sistema operativo.
- 20 % por la parte optativa.

Fechas de entrega de la práctica:

- Para la primera parte de la práctica no se impondrá un plazo máximo de entrega ya que ha de utilizarse en la segunda parte. De manera orientativa, el alumno debería tener terminada la primera parte en un plazo de dos semanas para asegurarse de tener tiempo para la segunda parte.
- El plazo de entrega de la **segunda fase** de la práctica finaliza a las 24 horas del lunes 19 de Abril del 2010.
- La entrega se realizará a través del sitio Moodle de la asignatura