

Selección eficiente de candidatos a cara mediante detección de ojos

Autores:

Beatriz Pontes Balanza

Manuel Resinas Arias de Reyna

Procesamiento de Imágenes Digitales 5º Ingeniería Informática

Curso 2003/2004

Índice



- Introducción
- Operador *average*
- El algoritmo paso a paso
- El verificador de caras
- Conclusiones
- Recursos utilizados

Introducción (I)

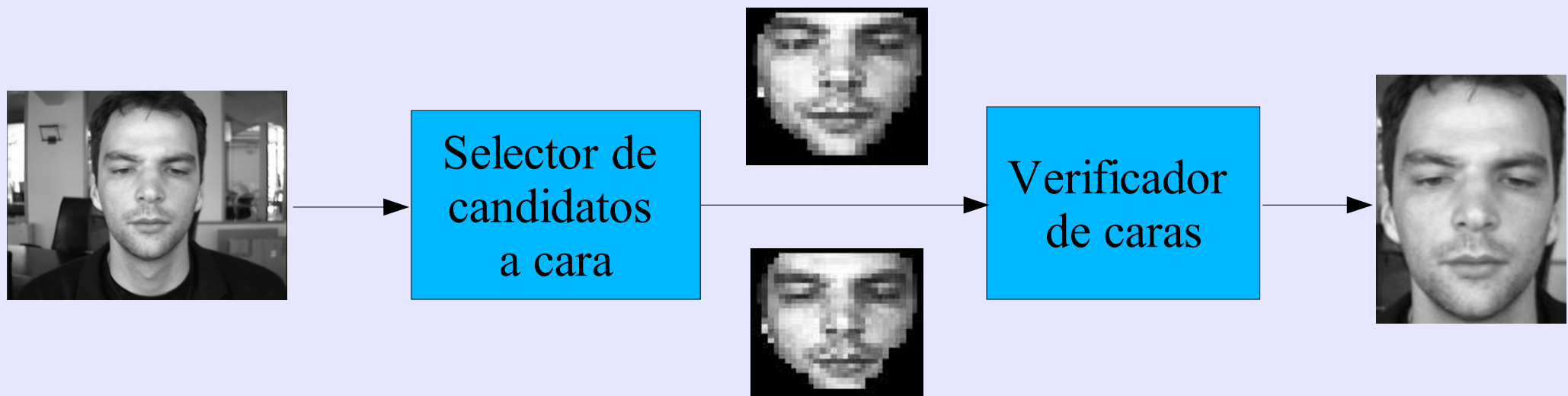


- En general, hay dos tipos de algoritmos para la detección de caras:
 - Basados en la configuración geométrica de rasgos faciales (ojos, cejas, nariz, boca...)
 - Basados en la comparación con plantillas de caras.
- En ambos casos, es necesaria la detección correcta de los ojos dentro de una imagen

Introducción (II)



- Por otro lado, la mayoría de los algoritmos de detección de caras están basados en dos fases:



Introducción (III)

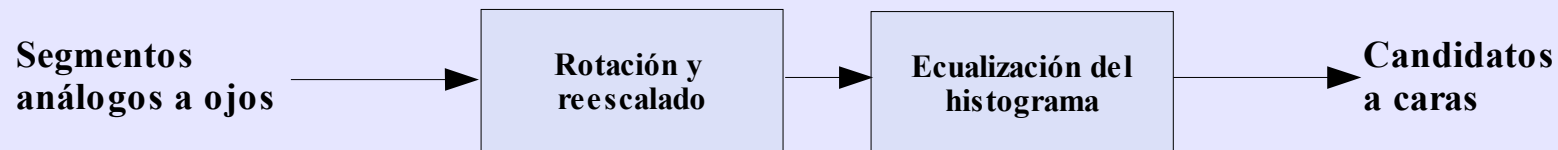
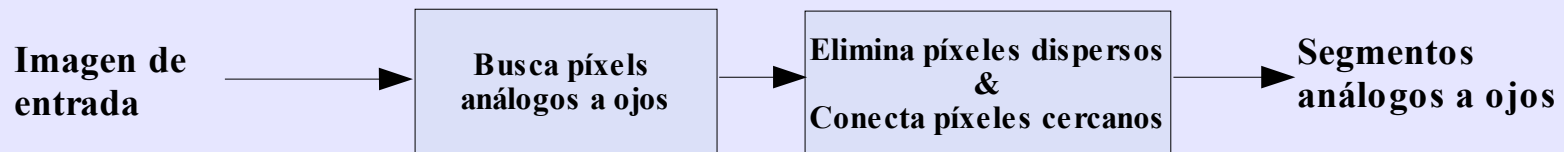
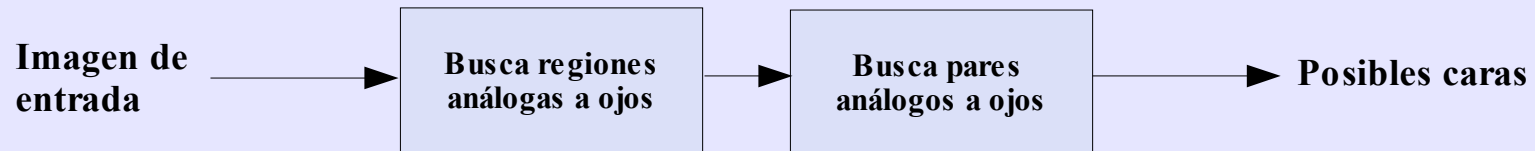


- El algoritmo que hemos implementado realiza la selección de candidatos a cara mediante la detección de ojos.
 - Resulta muy útil como primera fase de la detección de una cara.
 - Está basado en el algoritmo publicado por Jianxin Wu y Zhi-Hua Zhou: *Efficient face candidates selector for face detection*
- Además, también hemos implementado un verificador de caras muy simple.

Introducción (II)



- Las fases del algoritmo son:

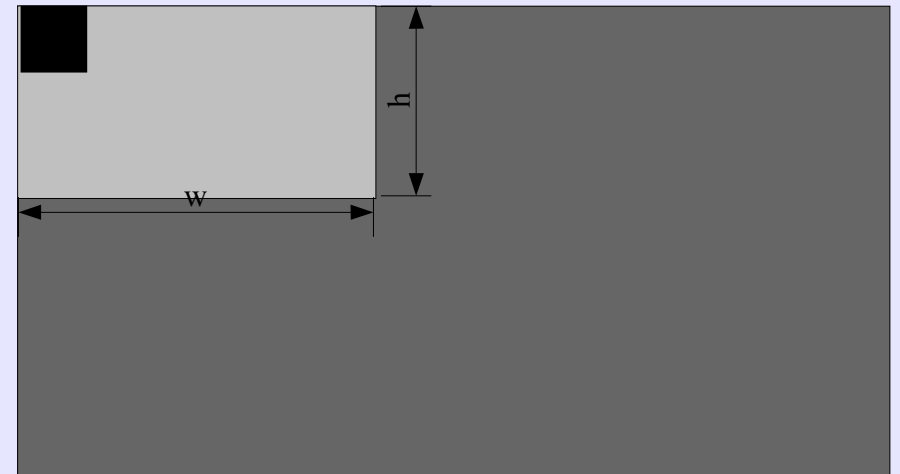


Operador *average*



- Para varios pasos del algoritmo se usa el operador *average*
- Consiste en asignar al pixel el valor de la media de un rectángulo de $w * h$:

$$avg(P, x, y, h, w) = \frac{\sum_{i=x}^{x+h-1} \sum_{j=y}^{y+w-1} P(i, j)}{h \cdot w}$$



Operador *average* (II)



- Se puede aplicar una optimización al operador *average*:
 - Descomposición por filas y columnas
 - Fácil implementación
- Se consigue mejorar el tiempo de ejecución sustancialmente

$$avg(P, x, y+1, 1, w) = avg(P, x, y, 1, w) + \frac{P(x, y+w) - P(x, y)}{w}$$

$$avg(P, x+1, y, h, 1) = avg(P, x, y, h, 1) + \frac{P(x+h, y) - P(x, y)}{h}$$

Algoritmo de búsqueda de candidatos



- 1. Pasar a escala de grises
- 2. Ocho vecinos
- 3. Borra componentes conexas grandes
- 4. Elimina píxeles dispersos
- 5. Agranda mejores candidatos a ojos
- 6. Elimina falsos candidatos a ojos
- 7. Detecta posibles pares de ojos
- 8. Prepara la imagen para el verificador de caras



Consideraciones iniciales

- Establecemos un tamaño de ojos que será el que busquemos por la imagen
 - En nuestro caso, el tamaño inicial es de 40 x 20
- En general, los ojos son más oscuros que otras partes de la cara
 - Por tanto, encontraremos las regiones análogas a ojos buscando trozos en la imagen original que sean aproximadamente del tamaño del ojo y más oscuras que sus vecinas

1. Pasar a escala de grises



- El algoritmo en el que nos hemos basado trabaja con imágenes en escala de grises.
 - Por tanto, es necesario una conversión previa de la imagen a escala de grises (si no lo está ya)



2. Ocho vecinos



- Comenzamos buscando las regiones análogas a ojos
 - Para eso, consideraremos que un píxel pertenece a una de estas regiones (píxel análogo a ojo) si es más oscuro que la mayoría de sus vecinos (en concreto, seis)
 - El “es más oscuro” lo medimos mediante el operador *average*



3. Borra componentes conexas grandes



- Si un grupo de píxeles análogos a ojos es demasiado grande para ser un ojo, se elimina
 - Esto se hace buscando las componentes conexas de la imagen del paso anterior y eliminando aquellas cuyo rectángulo exterior tiene una anchura o una altura mayor que dos veces la del ojo



4. Elimina píxeles dispersos



- Casi todos los píxeles entre las pestañas son análogos a ojos. Por tanto si hay píxeles análogos a ojos aislados, se eliminan
 - El saber si están aislados se hace acudiendo otra vez al operador *average*
 - En concreto, se eliminará si:

$$avg(P', x - \lfloor h_e/2 \rfloor, y - \lfloor w_e/2 \rfloor, h_e, w_e) \leq 0.2$$



5. Agranda mejores candidatos a ojos



- Las pestañas, la pupila y el lagrimal son zonas más oscuras del ojo que el resto.
 - Por tanto, puede que las partes claras no se hayan marcado como píxeles análogos a ojos.
 - Lo hacemos aplicando:

$$avg(P', x - \lfloor h_e/4 \rfloor, y - \lfloor w_e/4 \rfloor, \lfloor h_e/2 \rfloor, \lfloor w_e/2 \rfloor) > 0.35$$



6. Elimina falsos candidatos a ojos



- Si los grupos de píxeles son demasiados pequeños o demasiado grandes para ser ojos, se eliminan
 - Para esto, volvemos a buscar las componentes conexas y sus rectángulos exteriores y la eliminamos si cumplen:

$$h/h_e < 0.5 \text{ ó } h/h_e > 2.0$$

$$h/w > 0.8$$

$$w/w_e < 0.5 \text{ ó } w/w_e > 2.0$$



7. Detecta posibles pares de ojos



- Finalmente, marcaremos los pares de regiones candidatas a ojos como aquellas cuya distancia sea como la de dos ojos humanos
 - Concretamente, si (x_i, y_i) y (x_j, y_j) son los centros de dos regiones análogas a ojos, deben cumplir:

$$d_{ij} < 2.5 \cdot w_e$$

$$|x_i - x_j| < h_e$$

$$d_{ij} > 1.5 \cdot w_e$$



8. Prepara la imagen para el verificador de caras

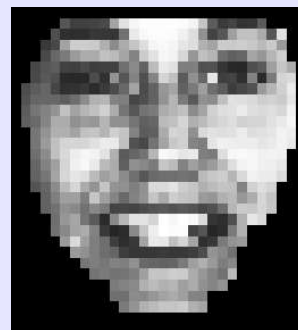


- El verificador necesita que las imágenes cumplan una serie de propiedades. Para eso:
 - **Rotamos** la imagen para corregir ligeras inclinaciones de la posible cara (se usan los ojos como guía)
 - **Reescalamos y recortamos** la cara al tamaño especificado por el verificador (en nuestro caso, 30 x 30).
 - **Aplicamos una máscara** para eliminar píxeles cercanos a los bordes
 - **Ecualizamos el histograma** para suavizar cambios de iluminación en la imagen original

Candidatos a caras



- Tras aplicar el paso anterior, a nuestro verificador de caras le llegan dos candidatos:



Pero...

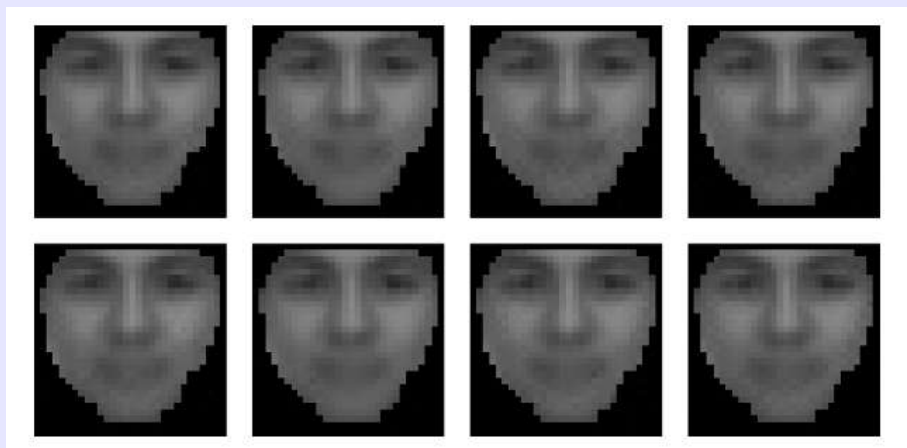


- ¿y si no encuentra ojos que pasa?
 - Simplemente los busca de otro tamaño.
 - El algoritmo supone que no había ojos del tamaño que buscaba y va probando otros tamaños multiplicando por una escala de 1.2 y 0.8 alternativamente hasta 8 veces.

El verificador



- Para completar el proceso hemos implementado también un sencillo verificador de caras basado en concordancia de plantillas.
 - Disponemos de 8 plantillas de caras que compararemos con los candidatos para decidirnos por uno
 - La comparación se hace por el mayor valor de correlación



Conclusiones



- Es un algoritmo eficiente que, en sólo varias pasadas por la imagen, reconoce los ojos de forma bastante aceptable.
- También es bastante robusto: sólo depende de que
 - los ojos son zonas más oscuras que sus vecinos
 - las regiones análogas a ojos deben tener el tamaño de ojos a una determinada escala

Problemas



- No obstante, el algoritmo también presenta algunos inconvenientes:
 - le afectan los brillos en los ojos
 - rotación de la cara demasiado pronunciada
 - a veces confunde los ojos con las cejas o con la boca

La aplicación

