# Búsqueda de patrones y flujo óptico

# Análisis de imágenes.

- **Análisis de imágenes**: procesamiento “inteligente” de las imágenes orientado a la extracción de información de tipo cualitativo (qué hay en las imágenes) o cuantitativo (posiciones, tamaños, distancias, tonos, etc.).
- **Objetivos del análisis**:
  - **Detección de objetos**: encontrar en la imagen las instancias de cierto tipo o clase de objetos.
  - **Reconocimiento de objetos**: distinguir la identidad específica de un objeto que se conoce que pertenece a cierta clase.
  - **Segmentación**: separar los objetos de interés del fondo.
  - **Seguimiento y correspondencia**: encontrar la equivalencia de puntos entre dos imágenes (por ejemplo, imágenes en una secuencia de vídeo o en un par estéreo).
  - **Reconstrucción 3D**: extraer información 3D de la escena, posiciones, ángulos, velocidades, etc.

# Búsqueda de patrones.

- La **búsqueda de patrones** es una técnica de análisis que se puede aplicar en detección de objetos, reconocimiento, seguimiento y correspondencia.
- **Idea de la técnica**: dada una imagen (un **patrón** o **modelo**) encontrar sus apariciones dentro de otra imagen mayor.
- No se buscan sólo las apariciones “exactas”, sino permitiendo cierto grado de variación respecto al patrón.
- **Ejemplo**. Buscar el patrón: 

  en la imagen dada.


**Resultado**: nº de aparicio-nes, localización de cada una y “verosimilitud”

# Búsqueda de patrones.

- El método más sencillo de búsqueda de patrones es el ***template matching*** (**comparación de plantillas**).
- **Template matching**: sea **A** una imagen (de tamaño WxH), y sea **P** un patrón (de wxh), el resultado es una imagen **M** (de tamaño (W-w+1)x(H-h+1)), donde cada píxel **M(x,y)** indica la "verosimilitud" (probabilidad) de que el rectángulo [x,y] – [x+w-1, y+h-1] de **A** contenga el patrón **P**.
- La imagen **M** se define usando alguna función de diferencia (o similitud) entre dos trozos de imagen.

  **M**(x,y):= d({A(x,y),...,A(x+w-1,y+h-1)}, {P(0,0),..., P(w-1,h-1)})
- **Ejemplo**. Suma de diferencias al cuadrado:

$$M(x, y) := \sum_{a=0..w-1} \sum_{b=0..h-1} (P(a, b) - A(x+a, y+b))^2$$

Es parecido a una **convolución** (pasar una máscara por toda la imagen)

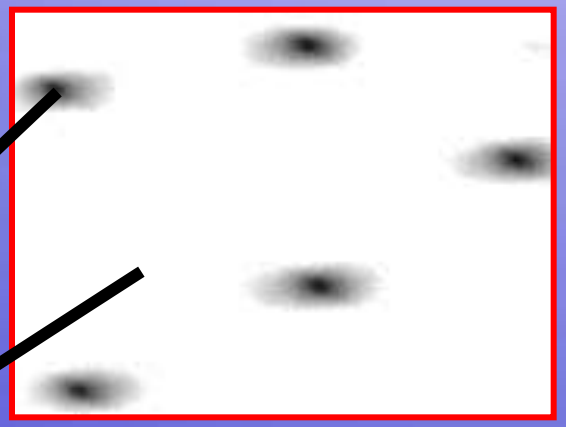
# Búsqueda de patrones.

- **Ejemplo**. *Template matching* con suma de diferencias al cuadrado.

Imagen de entrada **A** (239x156)

**P** - patrón a buscar (68x37)

Mapa de matching **M**



$6{,}58 \cdot 10^6$

$125{,}3 \cdot 10^6$

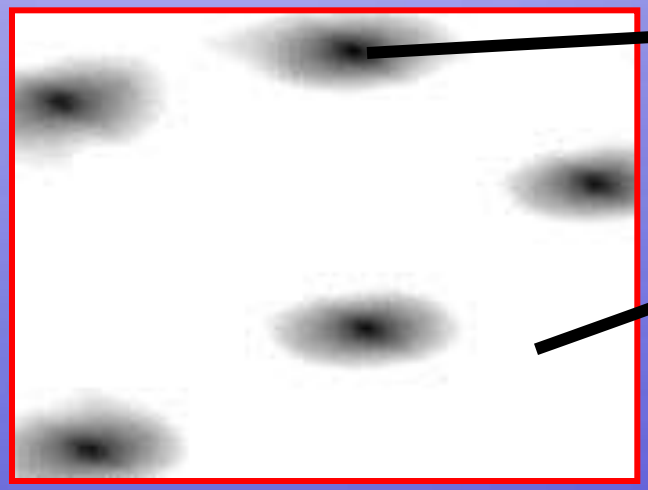
Mapa superpuesto

# Búsqueda de patrones.

- Los **valores bajos** (color oscuro) indican alta probabilidad de que el patrón se encuentre en esa posición (esquina superior izquierda).
- Los **valores altos** (color blanco) indican probabilidad baja.
- ¿Cuánto es alto o bajo? → Normalizar el resultado.
- **Normalización**: dividir el resultado por:

$$\text{sqrt}\left( \sum_{a=0..w-1} \sum_{b=0..h-1} P(a,b)^2 \cdot \sum_{a=0..w-1} \sum_{b=0..h-1} A(x+a, y+b)^2 \right)$$

- **Ejemplo**. Diferencias al cuadrado normalizadas.



# Búsqueda de patrones.

- Se pueden usar también otras medidas de distancia.
- **Ejemplo**. Producto escalar de patrones "centrados".

$$M(x, y) := \sum_{a=0..w-1} \sum_{b=0..h-1} (P'(a, b) \cdot A'(x+a, y+b))$$

Esto es lo que se llama la **correlación**

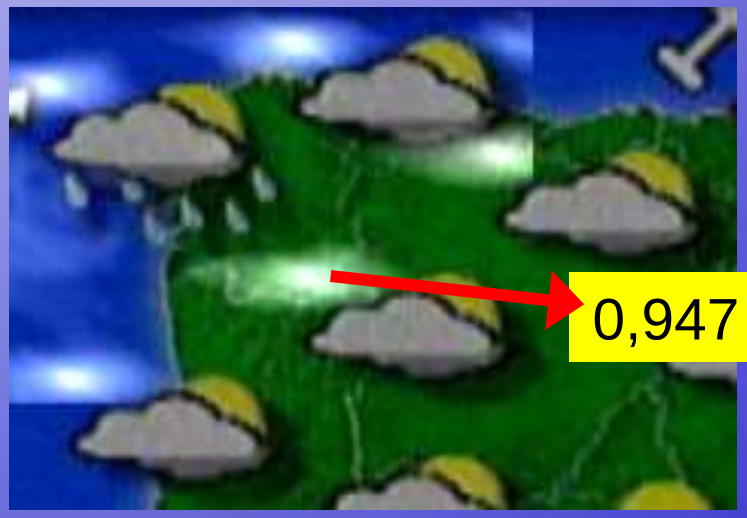
donde P'(a,b):= P(a,b) - Media(P). Lo mismo para A'.

- El valor (normalizado) está entre -1 y +1. Cuanto mayor (más próximo a +1) más probabilidad.

Patrón, **P**

Imagen de entrada, **A**

Mapa de matching, **M**



# Búsqueda de patrones.

- Una de las principales aplicaciones del *template matching* es la detección de objetos.
- **Proceso de detección de objetos** usando búsqueda de patrones.

1) Conseguir un patrón, **P**, representativo de la clase de objetos a buscar.
2) Aplicar el *template matching* a la imagen, obteniendo **M**.
3) Buscar los máximos (o mínimos) locales de **M**.
   - 3.1) Buscar el máximo global, (**Ix**, **Iy**)= $\text{argmax}_{\forall x,\, y}$ **M**(x, y).
   - 3.2) Si **M**(**Ix**, **Iy**) es menor que cierto umbral, acabar.
   - 3.3) Añadir la posición (**Ix**, **Iy**) a una lista de localizaciones resultantes del proceso.
   - 3.4) Poner a cero en **M** el rectángulo [Ix-w,Iy-h]–[Ix+w,Iy+h].
   - 3.5) Volver al paso 3.1.

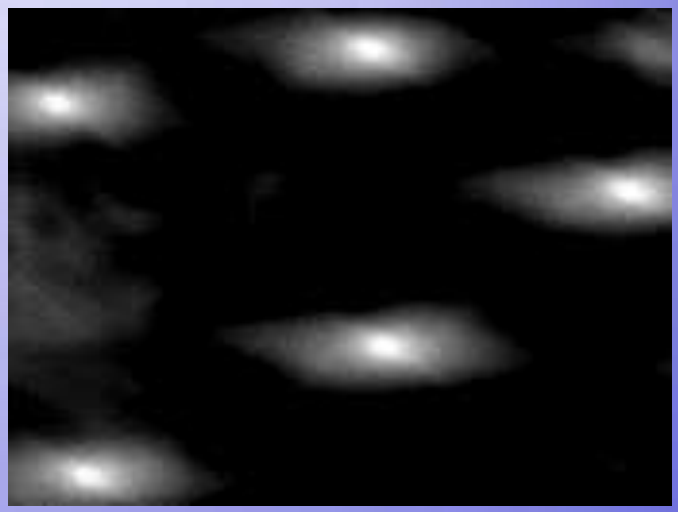
# Búsqueda de patrones.

- **Ejemplo 1**. Detección de objetos con *template matching*.

Patrón, **P**

Imagen de entrada, **A**

Mapa de matching, **M**



**Resultados**:
Posición (97, 87) con: 0.947
Posición (93, 10) con: 0.941
Posición (161, 47) con: 0.939
Posición (12, 24) con: 0.906
Posición (20, 121) con: 0.899
Posición (165, 9) con: 0.332

# Búsqueda de patrones.

- Pero, normalmente, el problema no es tan sencillo. Las clases de objetos presentan mayor variabilidad, y pueden haber variaciones de tamaño y rotación.
- El umbral debe bajarse, produciendo **falsos positivos**.
- **Ejemplo 2**. Detección de caras humanas.

Patrón, **P**
(29x27)

Patrón
ampliado

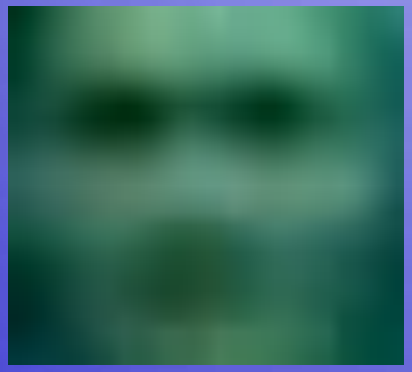


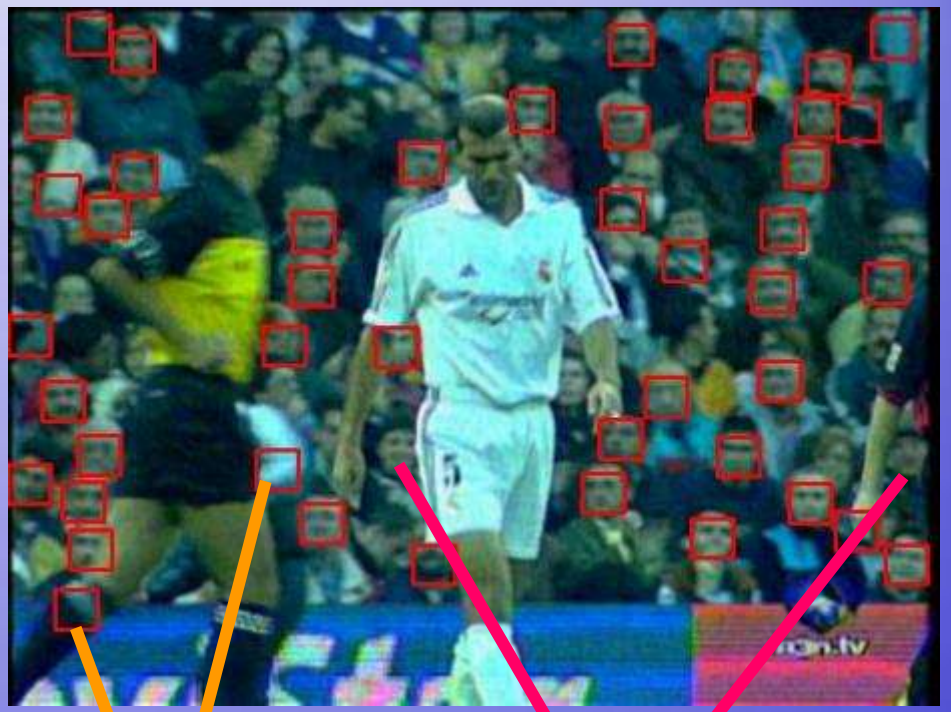
Imagen de entrada, **A** (640x480)

# Búsqueda de patrones.

- **Ejemplo 2**. Detección de caras humanas con *template matching*.

Mapa de matching, **M**

Resultados de la detección




- **Función**: producto vectorial.
- **Umbral** usado: 0,5

# Búsqueda de patrones.

- Obviamente, la técnica es muy sensible a cambios de **escala**, **rotación** o **deformaciones 3D** de los objetos.
- **Ejemplo 1**. Cambio de escala.

Imagen de entrada, **A**

63% 82% 100% 116% 143%

Mapa de matching, **M**

0,523 0,724 0,947 0,761 0,640

- **Ejemplo 2**. Cambio de rotación.

Imagen de entrada, **A**

20º 10º 0º 15º 25º

Mapa de matching, **M**

0,574 0,756 0,947 0,664 0,507

# Búsqueda de patrones.

- **Soluciones**:
  - Utilizar varios patrones, con distintos tamaños y rotaciones.
  - Hacer una **búsqueda multiescala**. Aplicar el proceso escalando la imagen a: 50%, 75%, 100%, 125%, ...
  - Usar alguna técnica de **atención selectiva**. Por ejemplo, usar color o bordes para centrar la atención en ciertas partes de la imagen.

- Otra aplicación interesante del *template matching* es la **correspondencia**: dado un par de imágenes de una misma escena, encontrar los puntos equivalentes de ambas.
- **Idea**: el patrón se extrae de una imagen y se aplica en la otra. El máximo (o mínimo) *matching* indica la equivalencia de puntos.
- **Ejemplo**: composición panorámica.

# Búsqueda de patrones.

- **Problema**: dadas dos imágenes de sitios adyacentes, obtener una composición panorámica de forma automática.

Imagen **A** (izquierda)

Imagen **B** (derecha)

- Como vimos en el tema 4, se usa una transf. geométrica.
- ¿Cómo obtener los parámetros de la transf.? → Encontrar **puntos equivalentes** entre ambas imágenes.

# Búsqueda de patrones.

- **Proceso** de composición panorámica:

1) Escoger dos trozos de la imagen **A** que se espera que aparezcan en **B**. ¿Qué trozos?

   1.1) Deben ser trozos en el solapamiento entre **A** y **B**. Si **A** es la imagen izquierda, un trozo de la derecha.

   1.2) El trozo debe tener elementos claramente definidos.

Imagen **A** (izquierda)

Este patrón no es muy bueno, no es nada significativo... descartarlo

Este patrón es OK

Este patrón también parece OK, pero... ¿y la barandilla?

# Búsqueda de patrones.

2) Para cada patrón escogido, buscarlo en la imagen B.

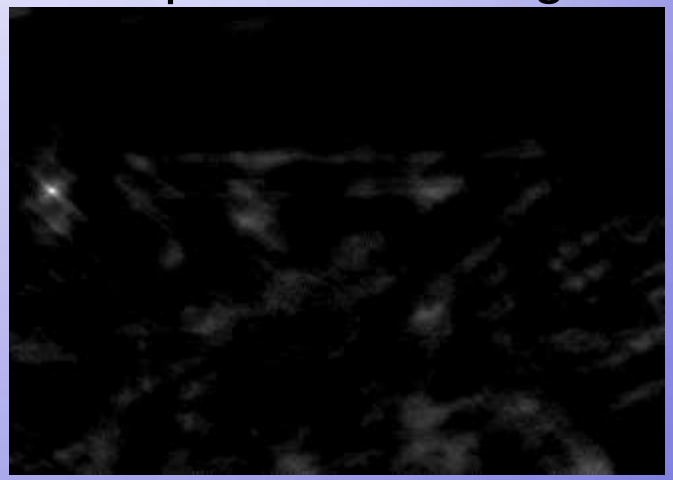
2.1) Aplicar *template matching*.

2.2) Quedarse con el máximo.

Patrón 1 

Imagen **B** (derecha)

Mapa de matching



Localización resultante

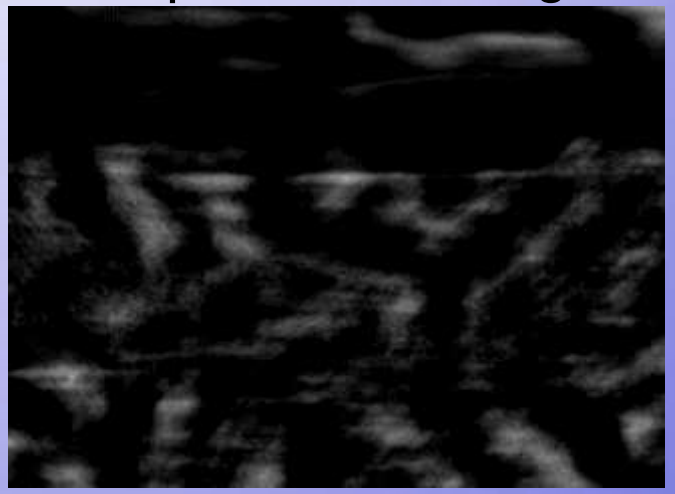
# Búsqueda de patrones.

2) Para cada patrón escogido, buscarlo en la imagen B.

Patrón 2

Imagen **B** (derecha)

Mapa de matching

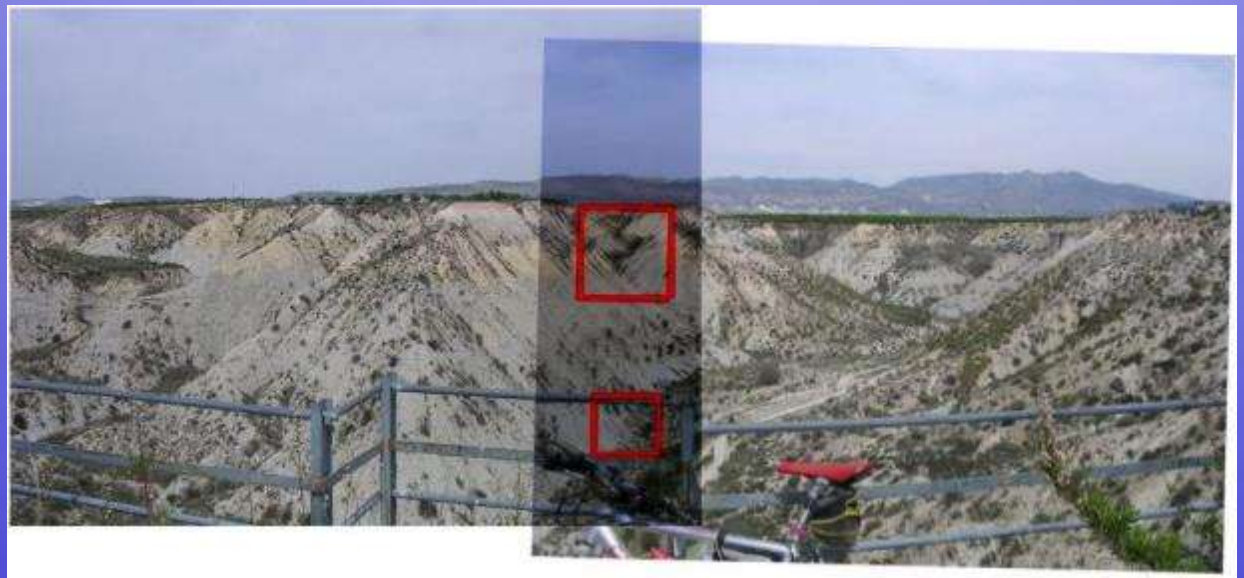


Localizaciones resultantes

Aquí la cosa no está tan clara, pero podríamos aplicar unas cuantas **heurísticas** sencillas y descartar las localizaciones inviables...

# Búsqueda de patrones.

3) Con las localizaciones equivalentes, calcular los parámetros de la transf. geométrica.
4) Aplicar la transformación y componer las dos imágenes.



# Búsqueda de patrones.

- Otra aplicación es el **seguimiento de objetos**: localizar la posición de un objeto a lo largo de una secuencia de vídeo.
- En un vídeo se espera que haya cierta "**continuidad temporal**", los elementos de la escena varían lentamente.
- **Idea**: aplicar *template matching*, usando como patrón el ROI del objeto en el instante *t*, aplicado sobre la imagen en *t*+1.
- **Ejemplo**. Seguimiento de caras. Suponemos una detección inicial.

Imagen en t = 0

Patrón de cara, $P_0$

# Búsqueda de patrones.

- **Proceso de seguimiento** usando *template matching*:

1) Detectar la posición inicial del objeto, $R_0$.
2) Repetir para cada frame t= 1...N:
   2.1) Extraer la región $P_{t-1}$ del frame t-1 usando $R_{t-1}$.
   2.2) Aplicar matching del patrón $P_{t-1}$ en la imagen del frame t.
   2.3) Seleccionar la pos. del máximo, poniendo el resultado en $R_t$.

$R_0$

Imagen en t = 0

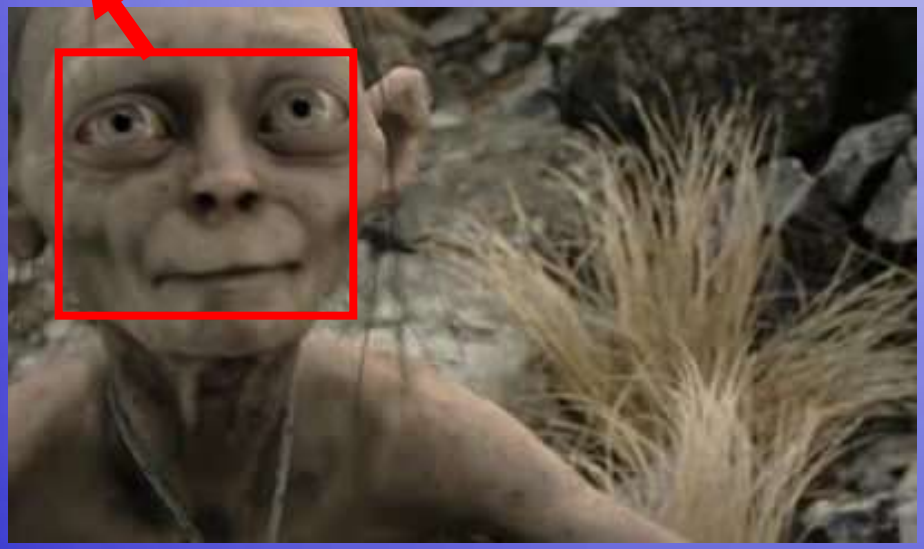


Imagen en t = 1

# Búsqueda de patrones.

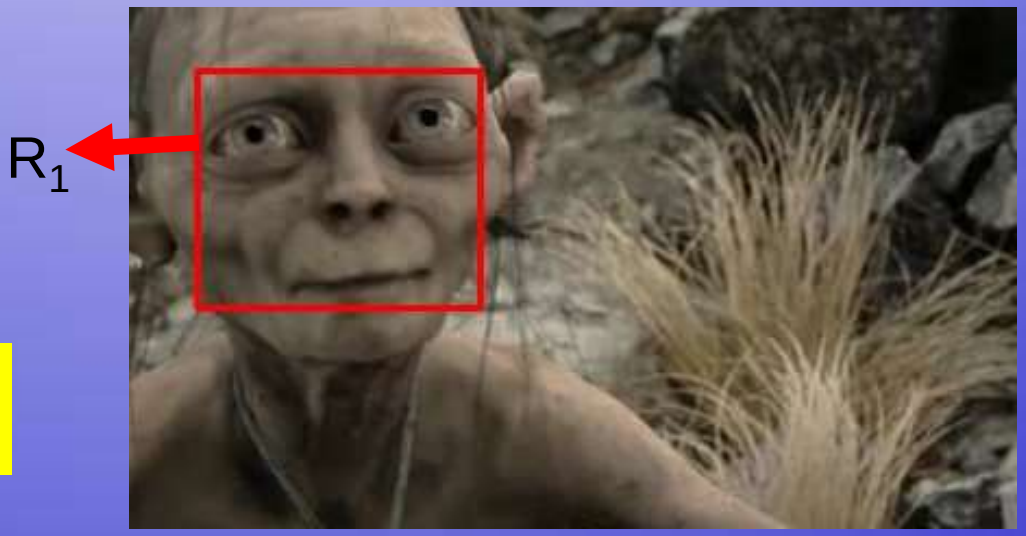
Patrón de cara,
$P_0$

Imagen en t = 1

Mapa de matching

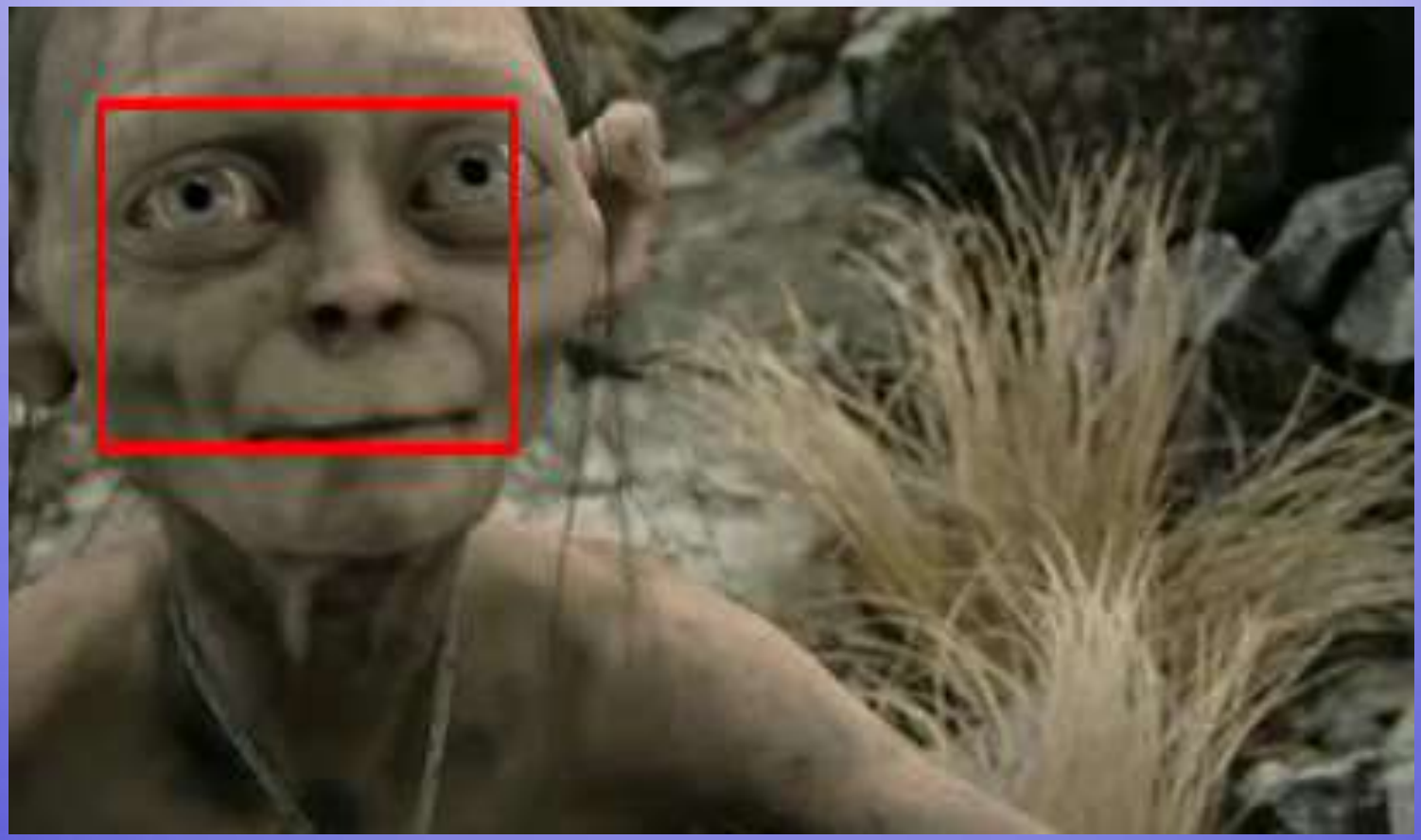
Ver que aquí el máximo es bastante destacado

Localización resultante



# Búsqueda de patrones.

- El proceso se repite para todos los frames de la secuencia.



- Se podrían añadir algunas **heurísticas** adicionales: que el salto no sea muy grande, que el valor de *matching* no baje de un umbral, etc.

# Búsqueda de patrones.

**Conclusiones**:

- **Template matching**: buscar las apariciones de un trozo de imagen en otra imagen de tamaño mayor.
- El proceso de cálculo es parecido a una convolución.
- **Ventajas**:
  - La idea es muy sencilla, aunque tiene un gran potencial.
  - Aplicación en detección, reconocimiento, seguimiento de objetos, etc.
- **Desventajas**:
  - Es muy sensible a rotaciones, escala, etc.
  - Además, en la vida real encontramos objetos 3D flexibles, lo que supone más variabilidad.
  - La aplicación de la técnica es muy costosa, O(**WHwh**). Cuando la resolución aumenta al doble, el tiempo se multiplica por 16.

# Flujo óptico.

- El **flujo óptico** es una técnica de análisis de imágenes que se aplica en secuencias de vídeo.
- En concreto, el **flujo óptico** define los vectores de movimiento de diferentes trozos de la imagen.
- **Aplicaciones**: detección de movimiento, seguimiento de objetos por partes, compresión de vídeo, composición, etc.
- **Ejemplo**. Imágenes de entrada Flujo óptico resultante

# Flujo óptico.

- Existen diversas formas de calcular el flujo óptico.
- Una forma sencilla está basada en la **técnica del *template matching***: dividir la imagen en **bloques**, para cada bloque de una imagen buscar la correspondencia en la otra.

Imagen en *t*-1

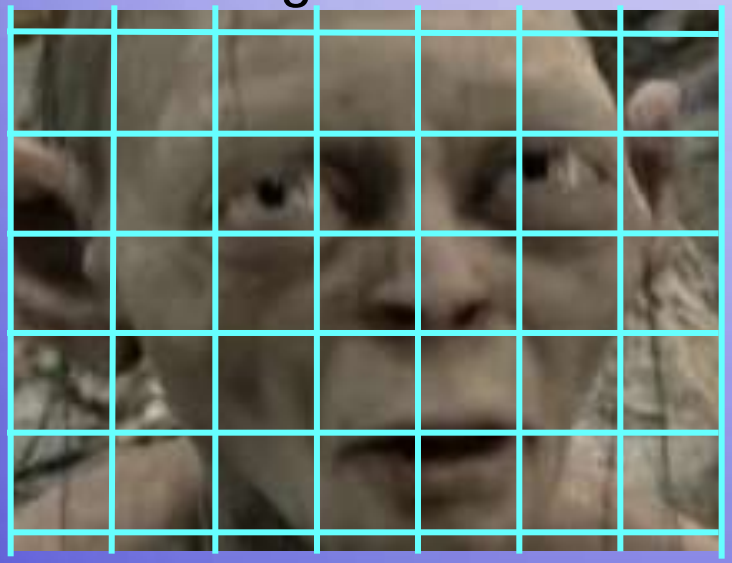


Imagen en *t*

- Buscar todos los trozos en la otra imagen sería muy costoso...
- Pero normalmente el desplazamiento será pequeño → Buscar sólo en una cierta **vecindad local**.

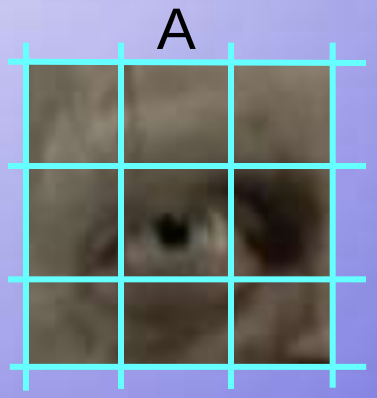
# Flujo óptico.

- **Parámetros** para el cálculo del flujo óptico:
  - **Tamaño de los bloques** a usar.
    - Ni muy pequeños ni muy grandes. Si son pequeños, contienen pocas características y el matching es poco fiable.
    - Si son grandes, perdemos resolución (menos vectores de movimiento). También hay problemas si el bloque se sale de la imagen.
  - **Radio de búsqueda**. Determina el tamaño de la zona, en la imagen *t*, donde se busca el bloque de entrada de la *t*-1.
    - Cuanto más grande, más tiempo de ejecución.
    - Si es muy pequeño y el movimiento es mayor, el resultado será impredecible.
  - **Función de matching** a emplear. Para este problema se podría usar una simple suma de diferencias. Para conseguir invarianza a cambios de iluminación, mejor usar un producto vectorial normalizado.

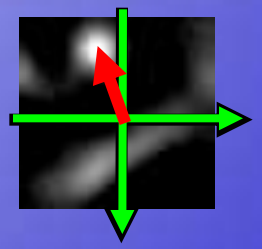
# Flujo óptico.

- **Proceso de cálculo del flujo óptico**.
  - **Parámetros de entrada**: A, B: imágenes de tamaño WxH; (w, h) tamaño de los bloques; (rx, xy) radio de búsqueda.
  - **Salida**: VelX, VelY: matrices de tamaño $\lceil W/w \rceil$ x $\lceil H/h \rceil$.

1) Para cada posición (i, j) en el rango de VelX y VelY hacer:
   1.1) Sea P el rectángulo $[i \cdot w, j \cdot h] - [(i+1)w-1, (j+1)h-1]$ de A
   1.2) Sea I el rectángulo $[i \cdot w-rx, j \cdot h-ry] - [(i+1)w-1+rx, (j+1)h-1+ry]$ de B
   1.3) Aplicar matching del patrón P en la imagen I, obteniendo el resultado en M de tamaño $(2 \cdot rx+1, 2 \cdot ry+1)$
   1.4) Buscar el máximo valor de matching: $(mx, my) = \text{argmax}_{\forall a,b} M(a,b)$
   1.5) $VelX(i, j) = mx-rx$; $VelY(i, j) = my-ry$







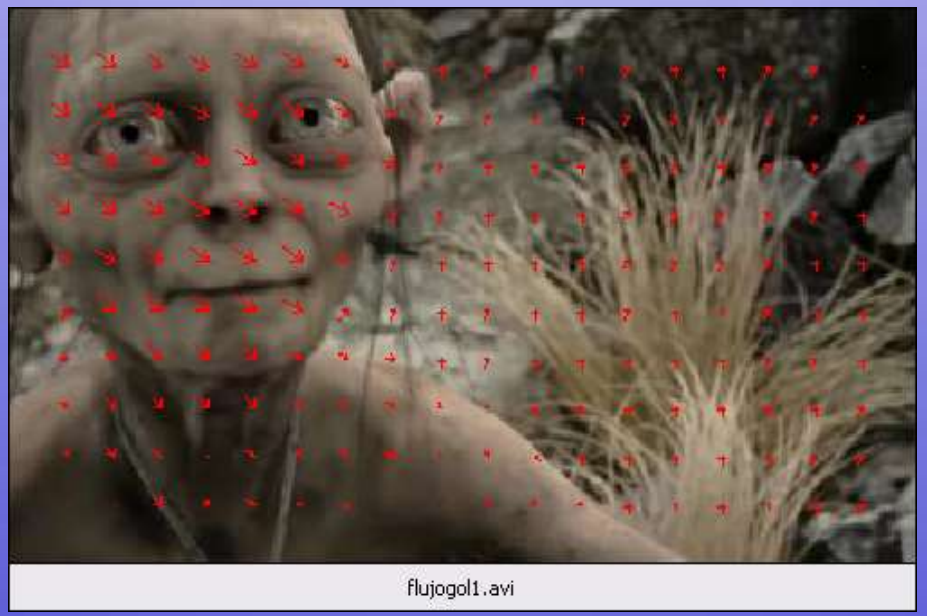

VelX,
VelY

# Flujo óptico.

- **Ejemplo**. Cálculo del flujo óptico por matching de bloques.

**Vídeo de entrada**

Resolución: 408x240




**Flujo óptico resultante**

Tamaño de bloque: 21x21
Radio de búsqueda: 21x21

# Flujo óptico.

- Otra aplicación interesante es la **composición de vídeo por barrido**.
- **Problema**: dada una secuencia de vídeo donde la cámara gira (o se desplaza lateralmente), componer una imagen con todo el campo de visión disponible.
- **Ejemplo**.

Vídeo de entrada

Panorámica resultante

# Flujo óptico.

- La **composición de vídeo** se puede ver como un proceso de **añadir tiras** de imágenes.
- El tamaño y posición de la tira añadida depende de la cantidad y dirección de movimiento detectado en las imágenes.
- **Proceso de composición por barrido**:

1) En la imagen inicial **t**= 0, seleccionar una región central (una tira) perpendicular a la dirección del movimiento.

   Inicializar con ella la imagen acumulada (Acum).

- Por ejemplo, seleccionar el rectángulo [100, 0]–[120, 320].

Imagen en *t*=0

$Acum_0$

# Flujo óptico.

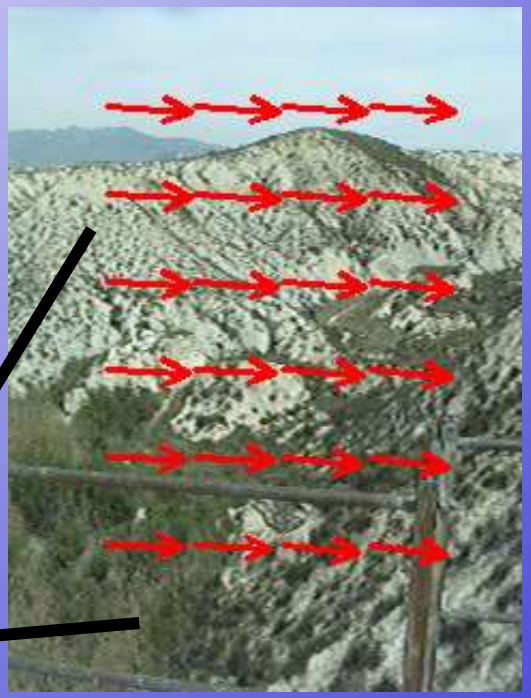
2) Usando el flujo óptico, detectar la cantidad de movimiento de cada nuevo frame, *t*, respecto al anterior, *t*-1. → VelX, VelY

- Por ejemplo, se puede tomar la media de velocidad en X e Y, vx= Media(VelX), vy= Media(VelY).

Imagen en *t*=0

Imagen en *t*=1

VelX, VelY



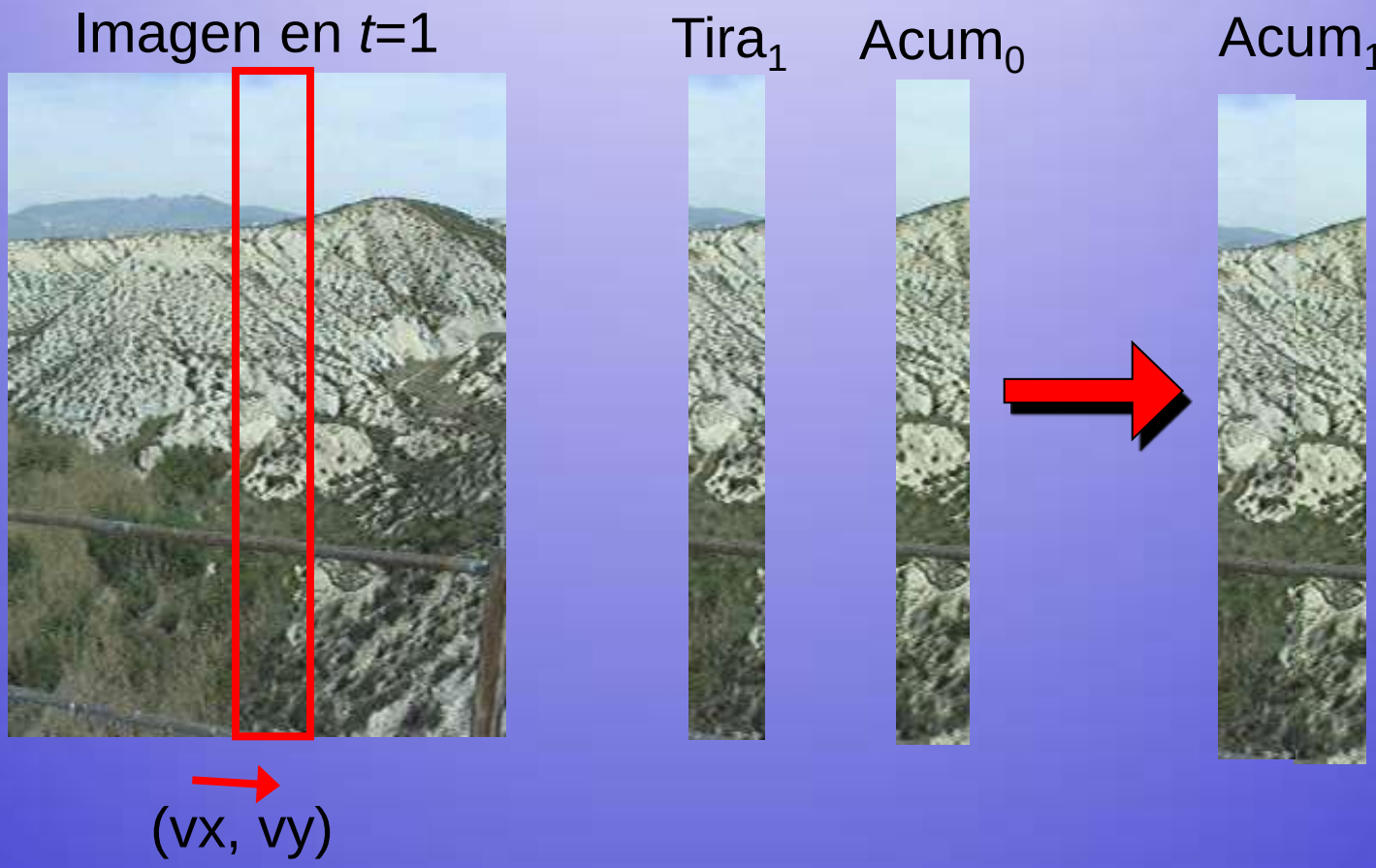
**Ojo**: descartar los bloques de los exteriores

vx= 38, vy= 4

# Flujo óptico.

3) Añadir a la imagen acumulada, Acum, la tira correspondiente en función de la velocidad calculada en el paso anterior.

- En el ejemplo, añadir el rectángulo [100, 0]–[100+vx, 320], desplazado en (-vx, -vy) píxeles respecto al último añadido.



# Flujo óptico.

- **Ejemplo**. Composición de vídeo por barrido.

- Otras **cuestiones adicionales**:
  - Compensación del brillo (y tal vez del balance de blancos).
  - Al final puede ser necesario aplicar una rotación de la imagen.
  - ¿Qué ocurre si hay movimiento en la escena?

# Flujo óptico.

**Conclusiones**:

- **Flujo óptico**: vectores de movimiento entre dos imágenes de una secuencia de vídeo.
- Es una técnica específica de vídeo.
- Además del método básico (utilizando *template matching*) existen otras muchas formas de calcularlo.
- **Ventajas**:
  - Permite comprender mejor la información contenida en un vídeo, la evolución en la escena: detectar si hay cambios en la escena, en qué posiciones, qué cantidad, etc.
- **Inconvenientes**:
  - La técnica es muy lenta. Es inviable aplicarla en tiempo real.
  - Difícil ajustar los parámetros para un funcionamiento óptimo: tamaño de bloques y radio de búsqueda.