Visión por Computadora I

Ing. Andrés F. Brumovsky

(abrumov@fi.uba.ar)

Laboratorio de Sistemas Embebidos -FIUBA

FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

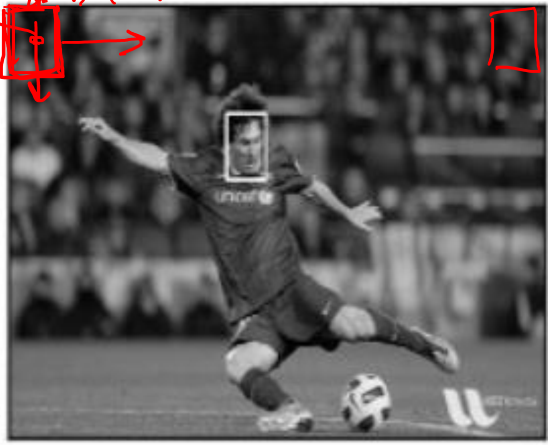
TEMPLATE MATCHING

Template

template(w, h)

max

g(i,j)



Resultado

Punto detectado

- La manera más directa de encontrar una característica en una imagen es si conocemos (exactamente) de antemano lo que estamos buscando.
- A esa porción de imagen que buscamos lo llamamos plantilla o "template"
- Existen a grandes rasgos dos métodos para implementarlo:
  1. Filtrado espacial lineal (cross-correlación)

     $$g(i,j) = \sum_{(k,l)} I(i+k, j+l).T(k,l)$$

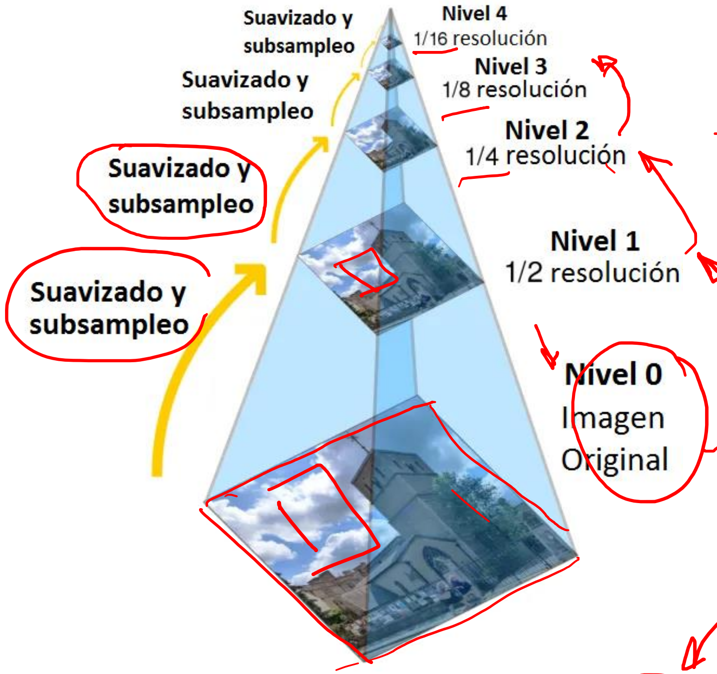
  2. Suma de las diferencias absolutas

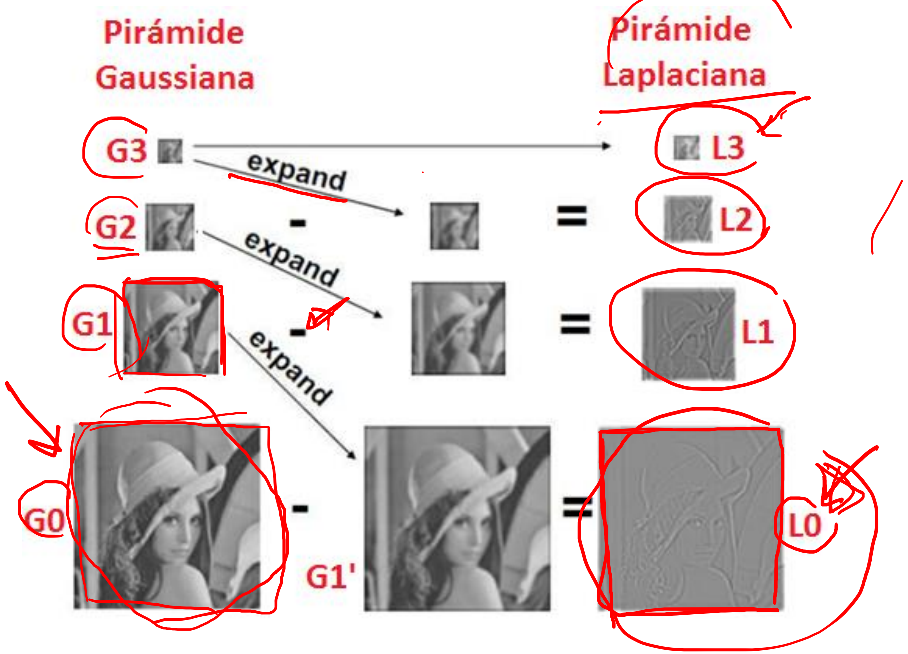
     $$g(i,j) = \sum_{(k,l)} |I(i+k, j+l) - T(k,l)|$$

     2

- Si la imagen es de tamaño (WxH) y la plantilla de tamaño (wxh) la imagen de salida será de tamaño

g(i,j) → $(W - w + 1, H - h + 1)$





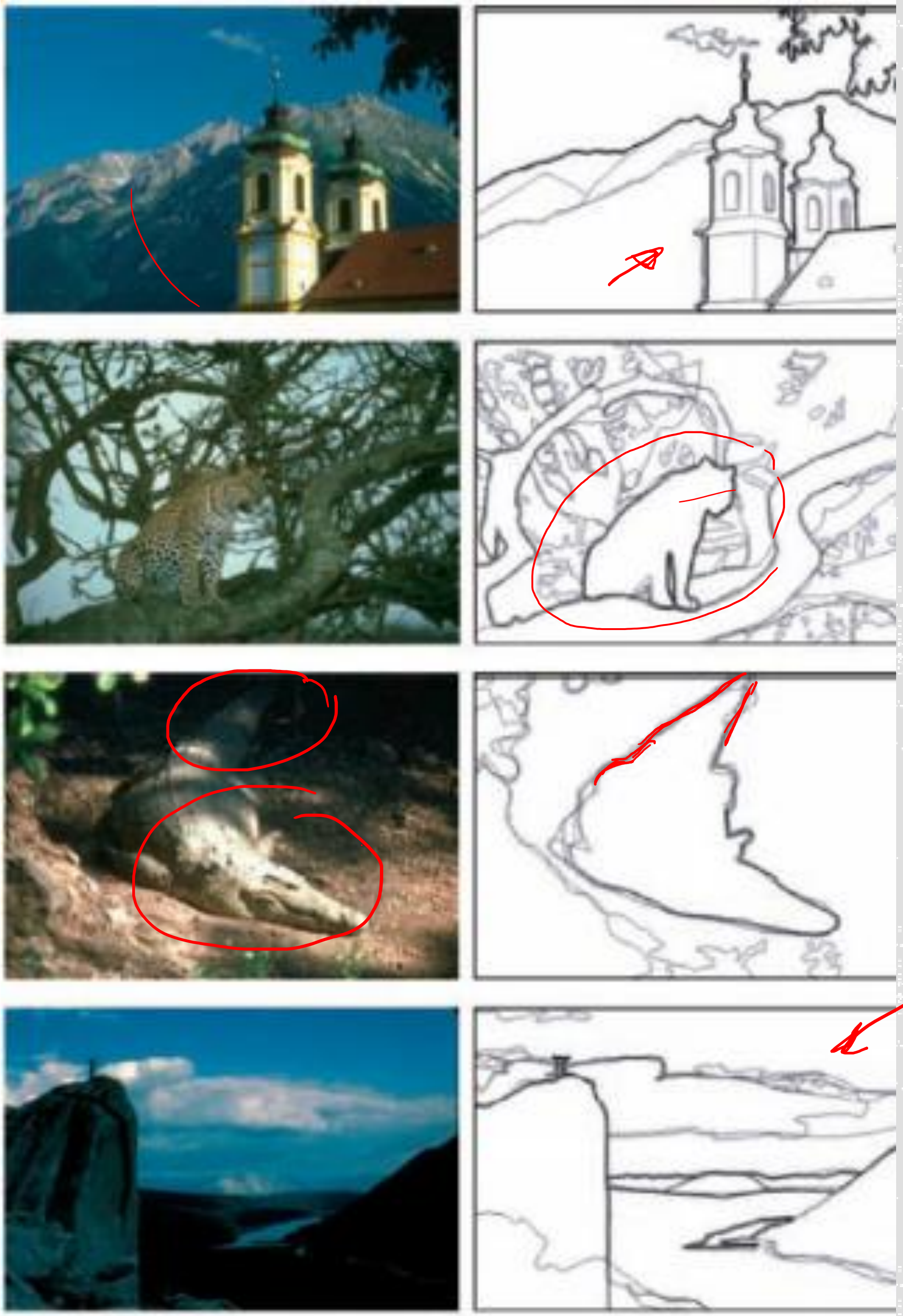
PIRÁMIDES

- ¿Qué pasa si queremos encontrar la cara de Messi en distintas fotos en la web, con diferentes escalas?
  - *“Los sistemas visuales biológicos funcionan también con una jerarquía de escalas (Marr, 1982)”*
- Las pirámides se refieren a representar una misma imagen en ***múltiples resoluciones***.
- Cada nivel implica dos pasos:
  1. **Suavizado** (reduce efectos de aliasing que se producirían de reducir directo)
  2. **Submuestreo** (a la mitad de la resolución)
- Los dos tipos más comunes de pirámides son:
  1. Pirámide Gausiana (Síntesis de textura)
  2. Pirámide Laplaciana (Compresión de imágenes)

Las pirámides Laplacianas se forman a partir de las pirámides Gaussianas (en realidad son una aproximación por diferencia de Gaussianas). Son como imágenes de bordes (la mayoría de los elementos son cero)

$\boldsymbol{G0} = \boldsymbol{L0} + \boldsymbol{G1'} \rightarrow$ *En lugar de guardar G0 guardamos L0 y G1 (con la que reconstruimos G1')*

- Aplicaciones:
  1. Búsqueda de objetos en distintas resoluciones
  2. Aceleración de procesamiento (encontrando objetos en las resoluciones más bajas y procesando en las altas
  3. Características que pueden pasar desapercibidas en una resolución se pueden hallar en alguna otra
  4. Operaciones de fusión de imágenes manteniendo los detalles



BORDES

- Los puntos de interés son útiles para encontrar la ubicación de objetos en una imagen, sin embargo los bordes o contornos pueden contener importantes asociaciones semánticas, además pueden corresponder inclusive a objetos parcialmente ocluidos.
- Puntos correspondientes a bordes aislados pueden agruparse en contornos de curvas más grandes o a segmentos de líneas rectas.
- Dada una imagen cómo encontramos los bordes?
  1. A través del gradiente (es preferible filtrar previamente para no ser susceptible a ruido)

$$J(x) = \nabla I(x) = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right)(x) \rightarrow J_\sigma(x) = \nabla[G_\sigma(x) * I(x)] \rightarrow [\nabla G_\sigma](x) * I(x)$$

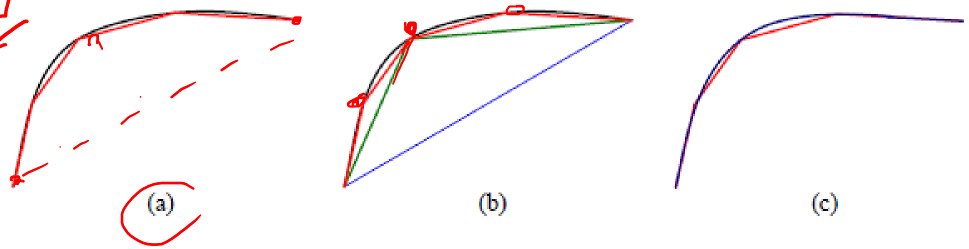
  2. A través del LoG (si solo queremos devolver bordes finos, de un pixel de ancho, en los cambios de intensidad)

$$S_\sigma(x) = \nabla J_\sigma(x) = [\nabla^2 G_\sigma](x) * I(x)$$

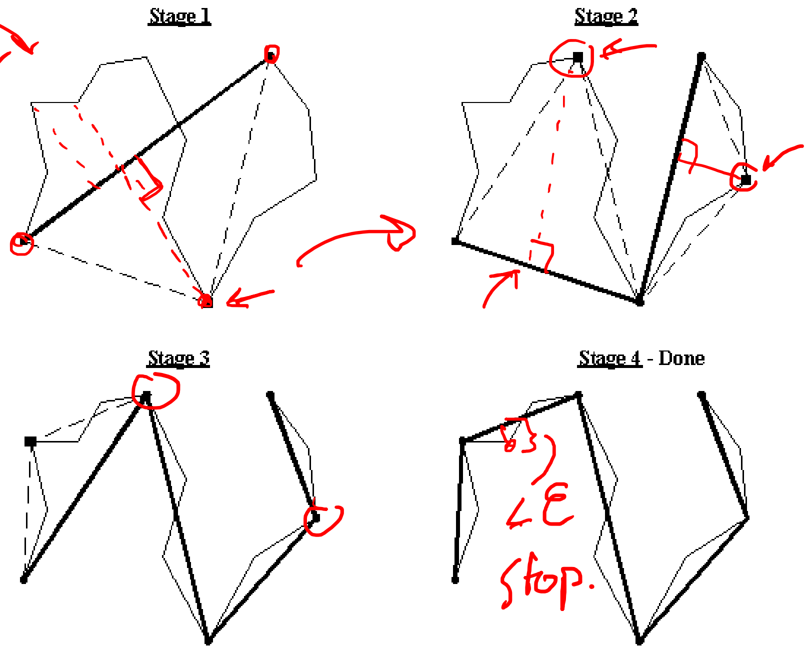
- Los bordes pueden "enlazarse" de varias maneras, ya vimos un método con el algoritmo de Canny.
- Las figuras de la izquierda muestran detecciones de borde "human powered", es decir, donde los ubicarían humanos.

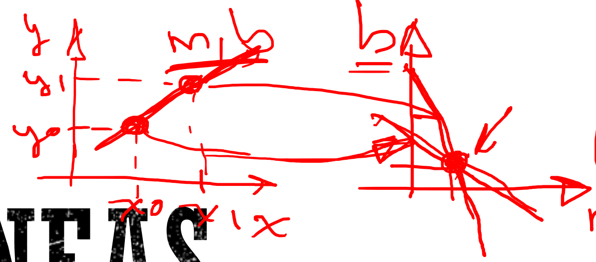
LÍNEAS

- Los bordes y curvas son útiles para describir contornos de objetos naturales. Sin embargo los humanos fabricamos objetos con infinidad de líneas rectas.
- Detectar estas líneas es útil para infinidad de aplicaciones.
- **Aproximación sucesiva de curvas por polilíneas:**
  - Una de las propuestas más sencillas (Ramer, 1972 – Douglas y Peucker, 1973) recursivamente divide la curva al punto más lejano de la línea que une los dos extremos.
  - Una vez hecha la simplificación se puede utilizar para aproximar la curva original o si se desea una representación más suave, hacer una interpolación por splines.
- **Transformada de Hough (Hough, 1962):**
  - Las polilíneas pueden ser exitosas al tratar de extraer líneas de una imagen, pero en el mundo real muchas veces esas líneas están "rotas" (es decir, son discontínuas) y estar formadas por tramos colineales.
  - Esta es una técnica de "votación" para posiciones factibles.
  - Puede detectar cualquier forma, mientras sea matemáticamente parametrizable.

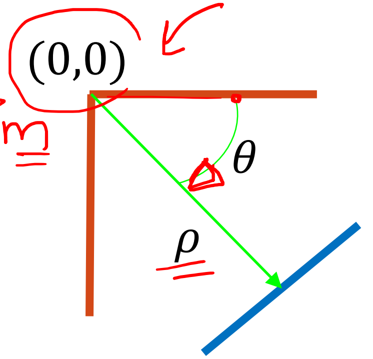


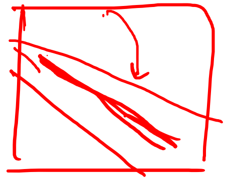
Douglas y Peucker: Cada punto agregado corresponde al vértice más alejado del segmento (los candidatos deben superar un umbral especificado)





HOUGH - LÍNEAS





$-180º < \theta < 180º$

$\theta = 0º \rightarrow$ *líneas verticales*

$\theta = 90º \rightarrow$ *líneas horizontales*

$\theta < 0º \rightarrow$ *pasa por arriba del origen*

$\theta > 0º \rightarrow$ *pasa por debajo del origen*

- Una línea: $y = m.x + b$ puede representarse en forma paramétrica como:

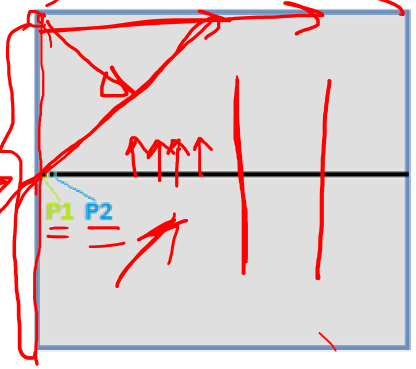
$b = -xm + y$

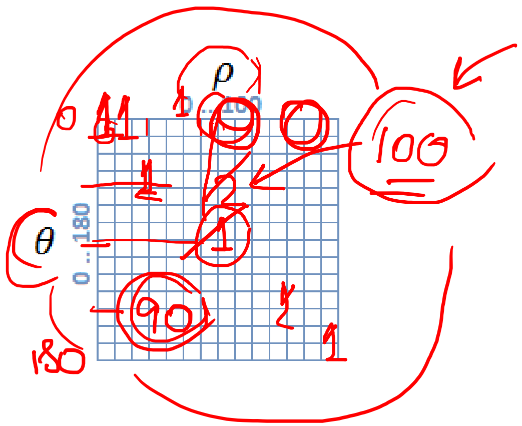
$$\rho = x.\cos(\theta) + y.\sin(\theta)$$

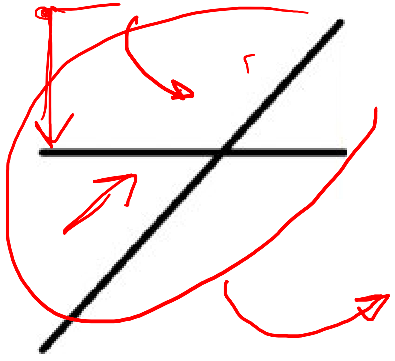
ρ: distancia perpendicular desde el origen a la línea
θ: ángulo formado por esa perpendicular y la horizontal, sentido antihorario

- ¿Por qué no se usa el modelo con parámetros m y b?. El problema viene por los valores que puede tomar m…
- Tipo de imagen de entrada → Bordes.
- Complejidad del algoritmo (memoria)
  k^n (*n dimensiones*, *k bins cada uno*)
- ¿Consumo de tiempo? → lineal con el número de elementos de borde
- Variaciones
  1. Utilizar "edgels" para no tener que iterar sobre todas las posibles orientaciones.
     $$\nabla I = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) = \hat{n}: (n_x, n_y)$$
     $$\theta = \tan^{-1}\left(\frac{n_y}{n_x}\right)\ ;\ \rho = x.nx + y.ny$$
  2. Dar mayor peso a los bordes más fuertes (mayor magnitud)
  3. Cambiar la resolución de (ρ, θ) de mayor a menor iterativamente
  4. Utilizar el mismo procedimiento con círculos, cuadrados, etc.

Ejemplo:





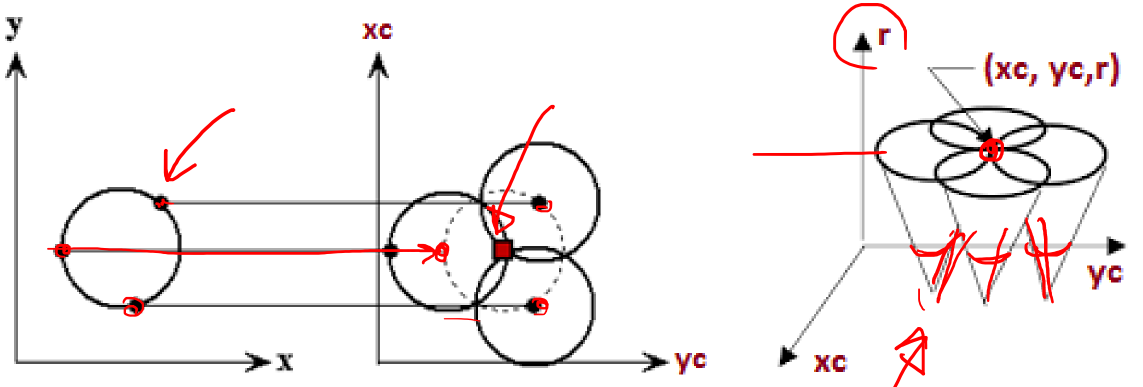


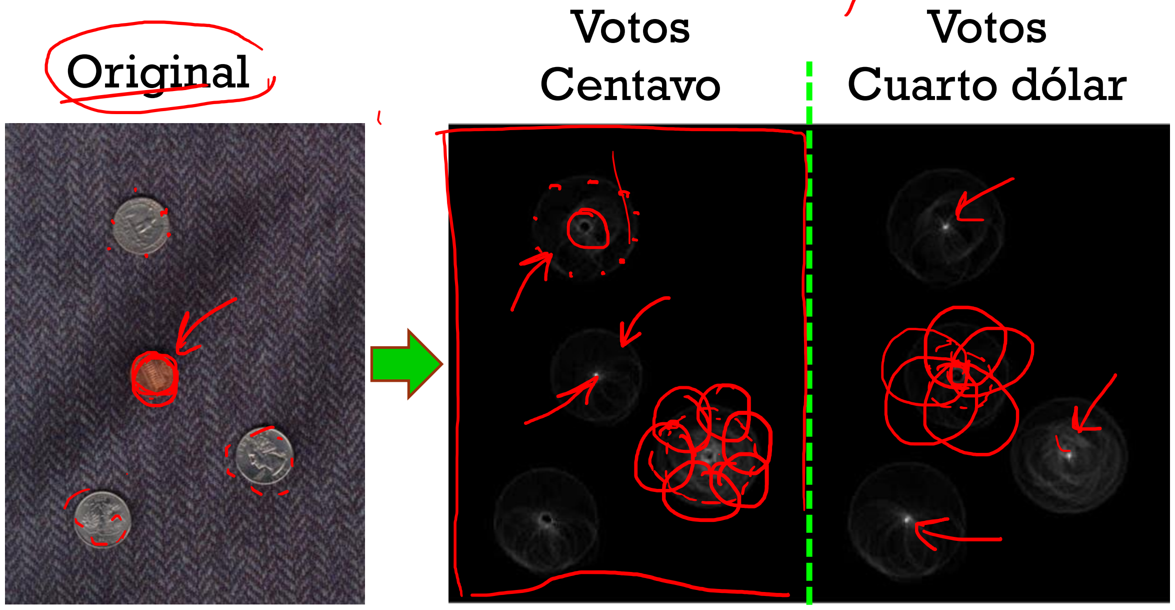
HOUGH - CÍRCULOS

- Un círculo puede parametrizarse según

$$r^2 = (x - x_C)^2 + (y - y_c)^2$$

- Si el radio es conocido (como en el caso de la búsqueda de monedas) tendremos dos parámetros (x_c, y_C) para ajustar.
- Si el radio es desconocido tendremos tres (x_c, y_C, r) y la transformada de Hough nos llevará a un espacio de tres dimensiones
- Si utilizamos el método visto anteriormente para la votación de mirar para cada elemento de borde el gradiente (lo que sería la tangente a nuestro círculo), entonces votaríamos sobre una única línea tangente a nuestro cono, reduciendo sustancialmente la cantidad de bins.





HOUGH – PROS Y CONS

Pros

- Todos los puntos son procesados independientemente (admite oclusión)
- Bastante invariante a ruido (el ruido rara vez puede contribuir consistentemente a un mismo set de parámetros)
- Se pueden detectar varias instancias de un modelo en una única corrida

Cons

- La complejidad aumenta exponencialmente con el número de parámetros del modelo (raramente se utilice con más de tres parámetros)
- Figuras no buscadas pueden producir picos espurios en el espacios de parámetros.
- Cuantización: Es difícil elegir un buen tamaño de grilla para la segmentación (número de bins) de los parámetros.

TRANSFORMADA DE HOUGH GENERALIZADA

- Introducido por Dana H. Ballard (1981) como modificación a la transformada de Hough y utilización del principio de Template Matching
- La idea es utilizar el algoritmo de Hough para curvas no-analíticas (figuras arbitrarias)
- Algoritmo:
  - **Construcción de la tabla R**
    1. Se elije un punto de referencia arbitrario dentro del objeto (puede ser el centro de masa)
    2. Preparar una tabla con k entradas de manera de indexar la orientación del gradiente según $\emptyset_i = 1 \dots k$ con pasos de $180º/k$
    3. Para cada punto de borde (x, y) se encuentran dos parámetros:

$$r = \sqrt{(x - x_c)^2 + (y - y_c)^2}$$
$$\beta = \tan^{-1}\ (y - y_c)/(x - x_c)$$

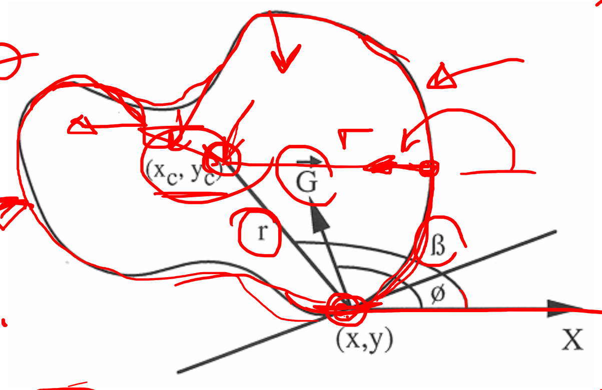
    4. Agregar estos parámetros a una tabla (tabla R) en la posición $\emptyset_i$ más cercana al gradiente para ese punto.
    5. Armamos un array 2D de Hough de dos parámetros $H(x_c, y_c)$ inicializado en cero.
  - **Detección de objetos en una imagen arbitraria**
    1. Para cada elemento de borde (x, y) calcular el gradiente
    2. Ubicar la entrada más cercana $\emptyset_i$ de ese gradiente en la tabla R y para cada uno de los n_j pares (r, β) calcular:

$$x_c = x + r.\cos\beta$$
$$y_c = y + r.\sin\beta$$

    3. Ir populando el array 2D $H(x_c, y_c)$ con estos valores y avanzar una posición.
    4. Finalmente todos los elementos de H que superen un umbral predefinido (de detección) representarán la ubicaciones de esa forma en la imagen
  - **Ampliación para rotaciones y escala variable**
    1. Se agregan los parámetros S de escala y θ de ángulo de rotación y el espacio de Hough pasa a ser $H(x_c, y_c, S, \theta)$
    2. Luego para cada elemento de borde en la imagen a analizar se calcula

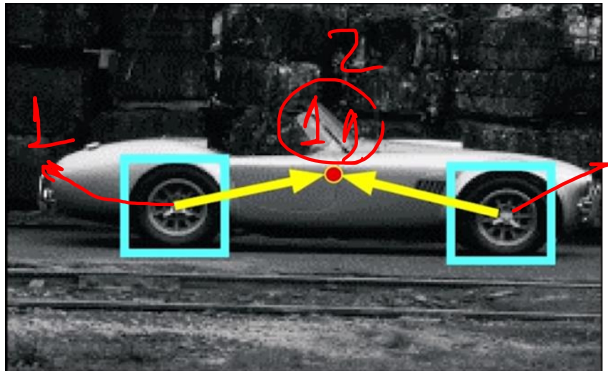
$$x_c = x + r.S.\cos(\beta + \theta)$$
$$y_c = y + r.S.\sin(\beta + \theta)$$

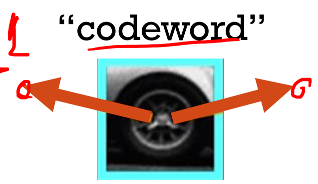
  - **Esto puede generalizarse no para elementos de borde sino patrones, parches o puntos de interés**



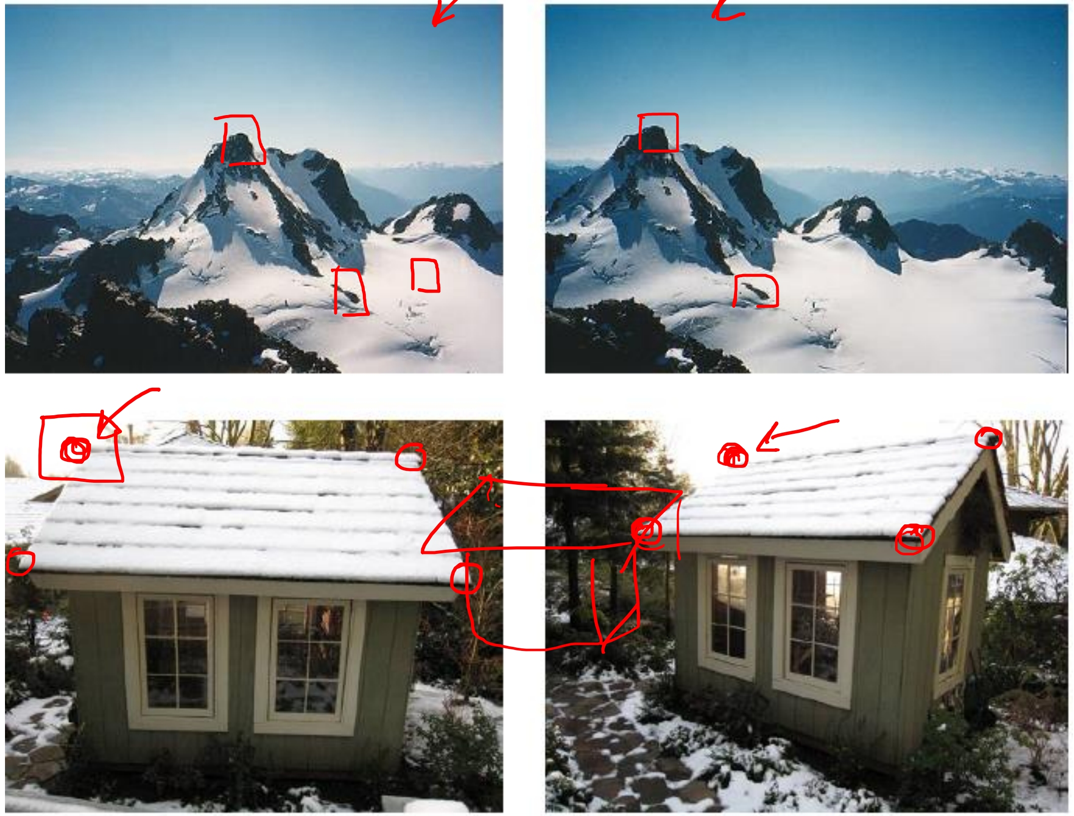
$\phi_1 = 0$	$(r,\beta)_{1_1}$	$(r,\beta)_{1_2}$	$\cdots$	$(r,\beta)_{1_{n_1}}$
$\emptyset_1 = 10$ $\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\phi_j = 90$	$(r,\beta)_{j_1}$	$(r,\beta)_{j_2}$	$\cdots$	$(r,\beta)_{j_{n_1}}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\phi_k = \pi$	$(r,\beta)_{k_1}$	$(r,\beta)_{k_2}$	$\cdots$	$(r,\beta)_{k_{n_1}}$

Generalización a patrones

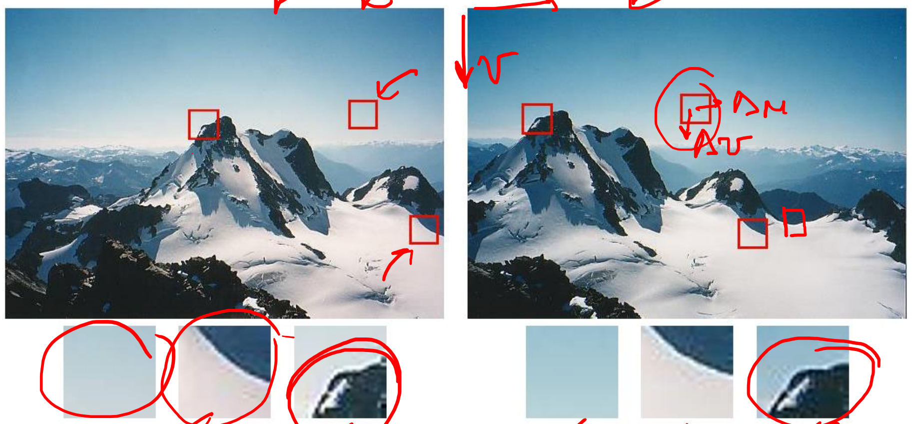


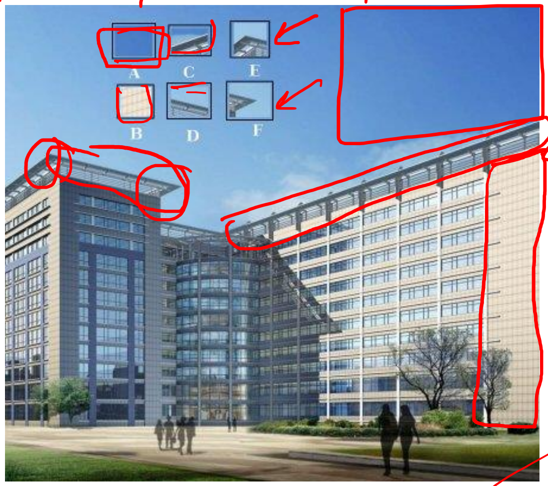


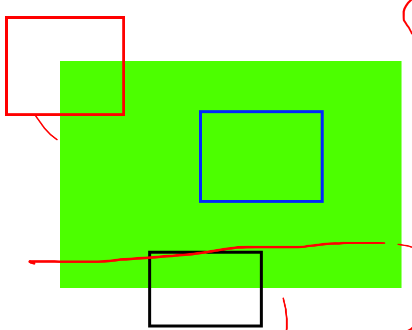
CARACTERÍSTICAS LOCALES



- La detección de características (en general, para su posterior coincidencia) son un componente esencial en muchas aplicaciones de visión:
  1. Alineación de imágenes
  2. Estimación de POSE
  3. Construcción de modelos 3D
  4. Generación de vistas intermedias
- Podemos dividir el proceso de detección/coincidencia de características en 4 etapas:
  1. Detección (extracción)
  2. Descripción (conversión en descriptores estables, invariantes)
  3. Correspondencia (búsqueda de coincidencias)
  4. Seguimiento (alternativa al punto anterior en caso de vecindario cercano → video)







CARACTERÍSTICAS LOCALES

- Parches
- Bordes
- Esquinas

La correspondencia de parches se puede escribir como:

(u, v)

$$E_{WSSD}(\boldsymbol{u}) = \sum_i w(x_i)[I_1(x_i + \boldsymbol{u}) - I_0(x_i)]^2$$

$\boldsymbol{I_0}, \boldsymbol{I_1}$: *Parches*
$\boldsymbol{w}(\boldsymbol{x_i})$: *Función ventana (cuadrada, gaussiana)*
$\boldsymbol{u} \rightarrow (u, v)$: *Vector de desplazamiento*
$\boldsymbol{i}$: *Se suma sobre todos los píxels del parche*

Cuando desplazamos un parche no sabemos dónde va a terminar coincidiendo, por lo que solo podremos computar cuán estable es una métrica respecto a pequeños desplazamientos (**u**) correlacionándola consigo misma.

$$E_{AC}(\Delta u) = \sum_i w(x_i)[I_0(x_i + \Delta u) - I_0(x_i)]^2$$

DETECTOR DE ESQUINAS DE HARRIS (I)

- Hagamos una expansión en series de Taylor $\left(f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots\right)$

$$I_0(x_i, \Delta_u) \cong I_0(x_i) + \nabla I_0(x_i)\Delta_{\boldsymbol{u}}$$

$$E_{AC}(\Delta u) = \sum_i w(x_i)[\cancel{I_0(x_i)} + \nabla I_0(x_i)\Delta_{\boldsymbol{u}} - \cancel{I_0(x_i)}]^2$$

$$E_{AC}(\Delta u) = \sum_i w(x_i)\left[\left(\frac{\partial I_0}{\partial x}, \frac{\partial I_0}{\partial y}\right).\Delta_{\boldsymbol{u}}\right]^2$$

Δu
Δv

$$E_{AC}(\Delta u) = \sum_i w(x_i)\left[\frac{\partial I_0}{\partial x}\Delta_u + \frac{\partial I_0}{\partial y}\Delta_v\right]^2$$

$$E_{AC}(\Delta u) = \sum_i w(x_i)\left(\left(\frac{\partial I_0}{\partial x}\right)^2 \Delta_u^2 + 2\frac{\partial I_0}{\partial x}\frac{\partial I_0}{\partial y}\Delta_u\Delta_v + \left(\frac{\partial I_0}{\partial y}\right)^2 \Delta_v^2\right)$$

$$E_{AC}(\Delta u) = \Delta_u^t.\mathrm{A}.\Delta_u$$

- El gradiente se puede calcular de diversas maneras:
  1. (Harris, Stephens, 1988) → [-2 -1 0 1 2]
  2. (Schmid, Mohr, Bauckhage, 2000 – Triggs, 2004) → Derivadas de Gaussianas con σ=1

$$A = w * \boxed{\begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}}$$

A: Matriz de autocorrelación. Buen indicador de cuáles parches se podrían "coincidir" con confiabilidad.

DETECTOR DE ESQUINAS DE HARRIS (II)

- La mejor manera de visualizar la acción de la matriz de autocorrelación es realizando un análisis de autovalores.
- Luego, para encontrar una puntuación que indique las características hay distintas aproximaciones:
  1. Harris/Stephens (1988)

$$\boldsymbol{R} = \mathbf{det}(\boldsymbol{A}) - \boldsymbol{k}\,(\boldsymbol{tr}(\boldsymbol{A}))^2 = \boldsymbol{\lambda_1 \lambda_2} - \boldsymbol{k}(\boldsymbol{\lambda_1} + \boldsymbol{\lambda_2})^2$$

$$\det(A) = \lambda_1 \lambda_2$$

$$tr(A) = \lambda_1 + \lambda_2$$

$$k = 0{,}06$$

0,04 0,06

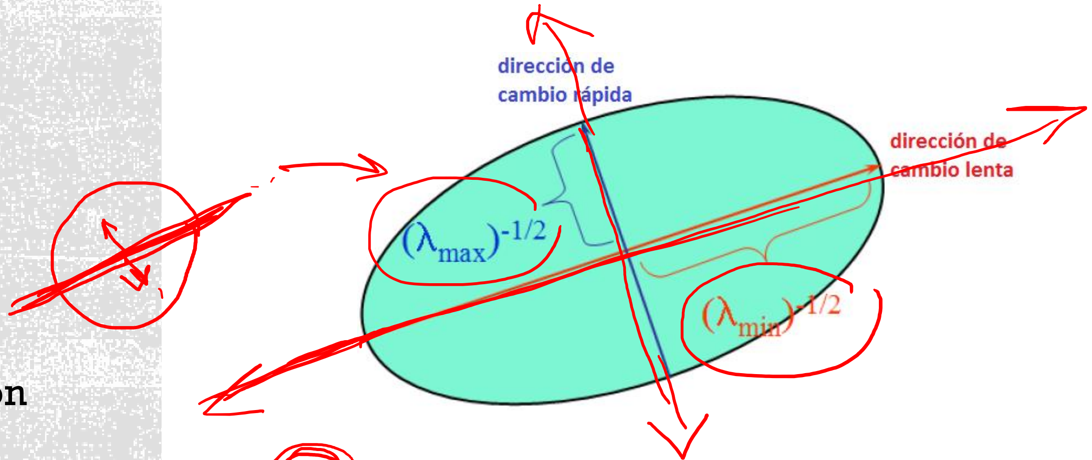
  2. Triggs (2004). Mejor respuesta a bordes 1-D (donde se sobredimensiona el autovalor más pequeño)

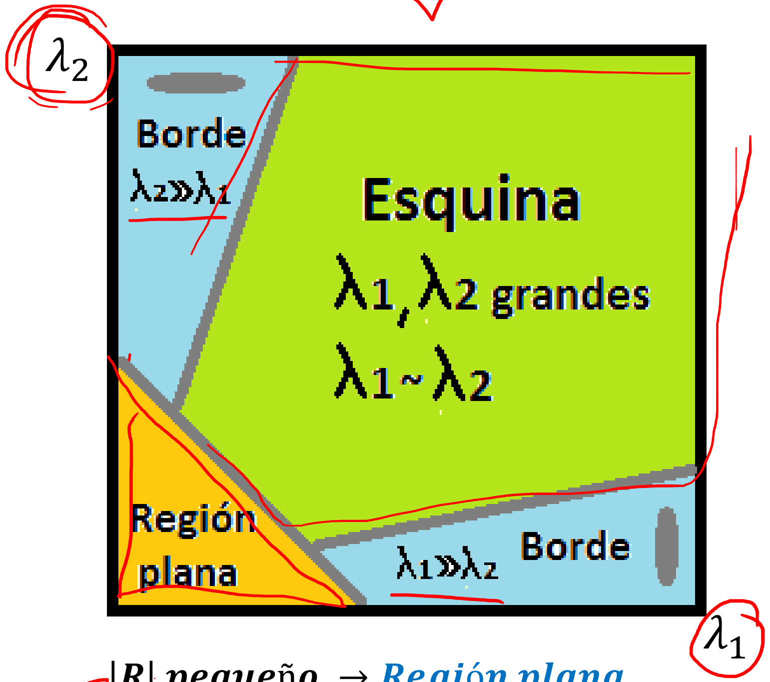
$$\boldsymbol{R} = \boldsymbol{\lambda_1} - \boldsymbol{k}\,\boldsymbol{\lambda_2}$$

$$k = 0{,}05$$

  3. Brown, Szeliski, Winder (2005). Usa la media armónica, función más suave donde $\lambda_1 \approx \lambda_2$

$$\boldsymbol{R} = \frac{\det(A)}{tr(A)} = \frac{\boldsymbol{\lambda_1 \lambda_2}}{\boldsymbol{\lambda_1} + \boldsymbol{\lambda_2}}$$





$|\boldsymbol{R}|$ ***pequeño*** → ***Región plana***

$\boldsymbol{R} < \mathbf{0}$ → $\boldsymbol{\lambda_i} \gg \boldsymbol{\lambda_j}$ → ***Borde***

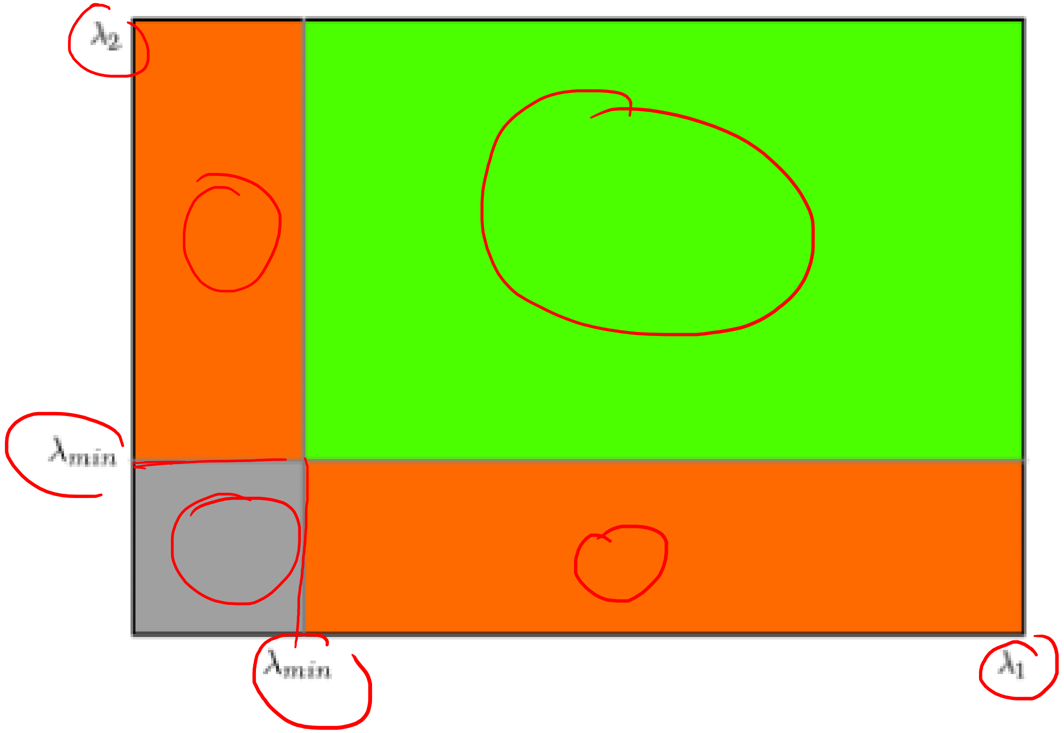
R grande → $\boldsymbol{\lambda_1}$ ***y*** $\boldsymbol{\lambda_2}$ ***grandes*** → ***Esquina***

DETECTOR DE ESQUINAS – SHI-TOMASI

- En 1994 J. Shi y C. Tomasi hicieron una modificación al detector de esquinas de Harris y lo publicaron en el paper "Good features to track", mostrando mejores resultados que el algoritmo original de Harris.
- La función de puntaje propuesta pasó a ser:

$$\boxed{R = min(\lambda_1, \lambda_2)}$$

- En este caso cuando ambos, λ_1 y λ_2, sean mayores que un λ_{min} se considerará una esquina.
- En este algoritmo:
  1. Se especifica la cantidad N de esquinas a devolver.
  2. Se especifica un "nivel de calidad" entre 0-1. Todas las esquinas por debajo de este nivel se descartan, las esquinas que sobreviven se clasifican en orden descendente.
  3. Se especifica un valor de distancia euclidiana mínima entre las esquinas detectadas (se tiran todas las esquinas cercanas en menos de esta distancia a una esquina fuerte)



(a) Strongest 250 (b) Strongest 500 (c) ANMS 250, $r = 24$ (d) ANMS 500, $r = 16$

SUPRESIÓN DE NO-MÁXIMOS ADAPTATIVA

- En general los algoritmos de detección de características buscan máximos locales en la función de puntaje, pero esto puede generar mayor densidad de características en zonas de mayor contraste.
- Frente a este problema Brown, Szeliski y Winder (2005) solo detectan características que cumplen dos condiciones:
  1. Son máximos locales
  2. Tienen un valor significativamente mayor (10%) a todos sus vecinos en un radio **r**

TP3

- Para la imagen suministrada “eyes” (por ninguna razón en especial, con heterocromía), implementar un algoritmo que:

  1. Encuentre la posición de los iris en cada par de ojos y mida su distancia en píxeles.
  2. Encuentre la posición de las pupilas en cada par de ojos y mida su distancia en píxeles.