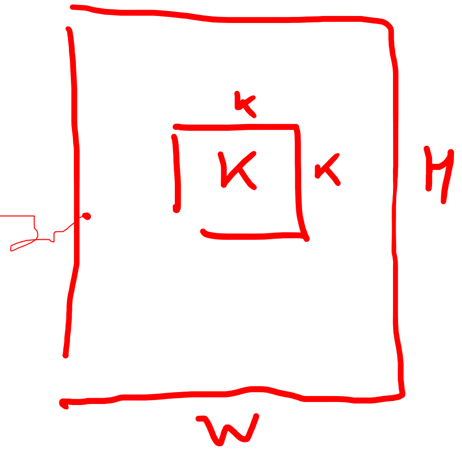
Visión por Computadora I



Ing. Maxim Dorogov

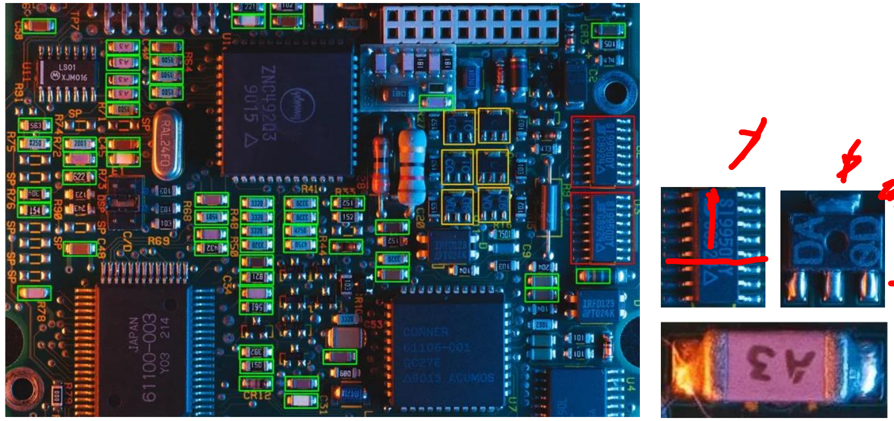
(mdorogov@fi.uba.ar)

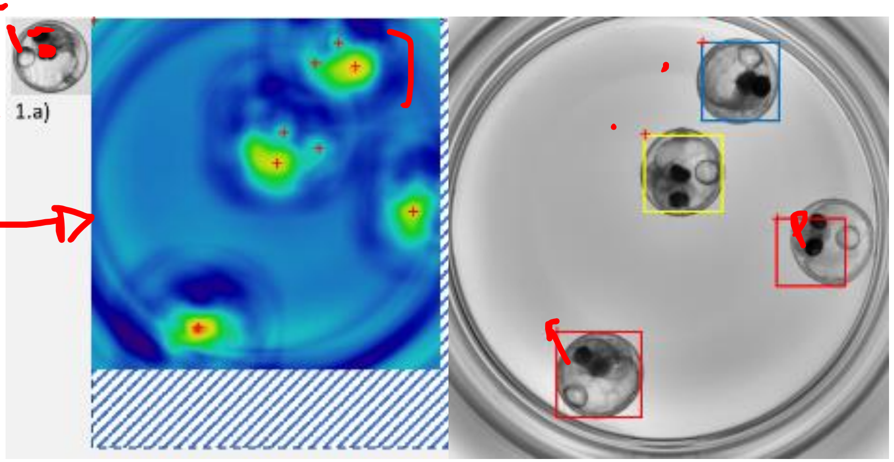
Laboratorio de Sistemas Embebidos -FIUBA

FACULTAD DE INGENIERIA
Universidad de Buenos Aires

TEMPLATE MATCHING

El objetivo es encontrar, en la imagen, las regiones que maximizan una métrica de similitud con un *template*.



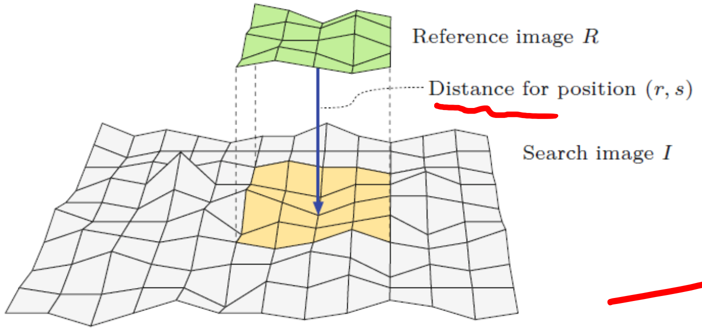


Aplicaciones:

- Tracking sobre una secuencia de imágenes o videos
- Correspondencia de características en visión estéreo
- Detección de logotipos, texto en etiquetas, u otros elementos que no presenten variaciones considerables en su apariencia respecto del *template*.
- Búsqueda de patrones específicos en una escena
- Múltiples detecciones a partir de un único *template*.

Restricciones

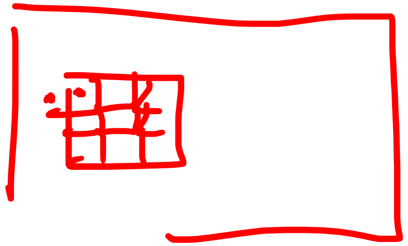
- Solo es invariante a desplazamientos
- Requiere procesamiento adicional para ser invariante a escala
- La calidad de detección depende de la métrica de similitud elegida



TEMPLATE MATCHING

255 | 255 | 255

Métricas de detección:



- Suma de diferencias al cuadrado:

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2$$

- Suma de diferencias normalizada:

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

- Coeficiente de correlación:

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))$$

- Correlación normalizada:

0.9 0.1

0.0001

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

…y muchas mas! (Ver opencv metrics)

Al normalizar podemos plantear umbrales de detección entre 0 y 1 para encontrar múltiples objetos dado un nivel de "confianza" determinado.

TEMPLATE MATCHING

Por que la correlación sirve como métrica de detección?

Podemos desarrollar la sumatoria de diferencias como:

cte

$$\begin{aligned} d_E^2(r,s) &= \sum_{(i,j)\in R} \big(I(r+i,s+j) - R(i,j)\big)^2 \\ &= \underbrace{\sum_{(i,j)\in R} I^2(r+i,s+j)}_{A(r,s)} + \underbrace{\sum_{(i,j)\in R} R^2(i,j)}_{B} - 2\cdot \underbrace{\sum_{(i,j)\in R} I(r+i,s+j)\cdot R(i,j)}_{C(r,s)} . \end{aligned}$$

Minimizar la suma de diferencias al cuadrado es equivalente a buscar el máximo de la correlación entre la imagen y el *template* ya que B es un valor cte y A, a priori, no aporta información a la búsqueda.

Esto permite operar en el dominio de frecuencia sabiendo que*:*

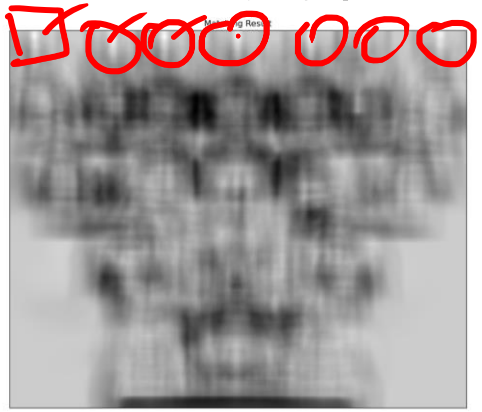
$$Corr(I,R) \Leftrightarrow f(I)\cdot f(R)^*$$

TEMPLATE MATCHING

El resultado de una operación de *template matching* es una nueva imagen:

image

Ejemplo utilizando la métrica de diferencias al cuadrado (arriba) y correlación (abajo y derecha).

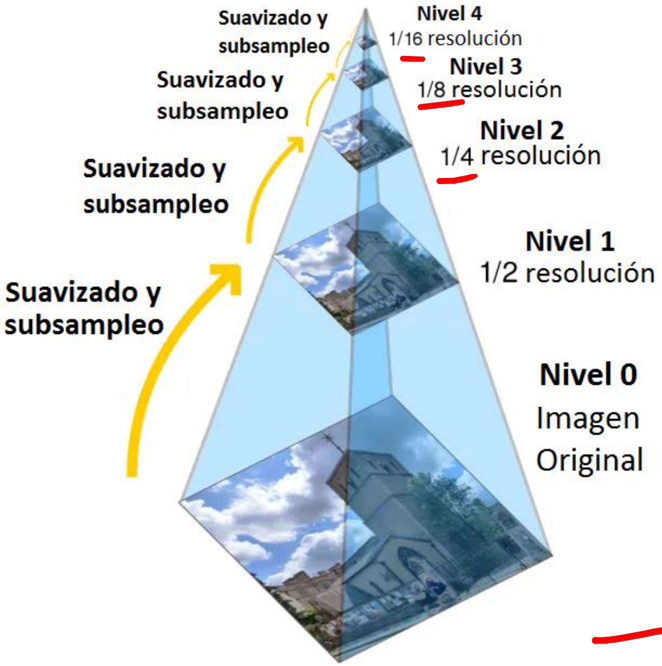


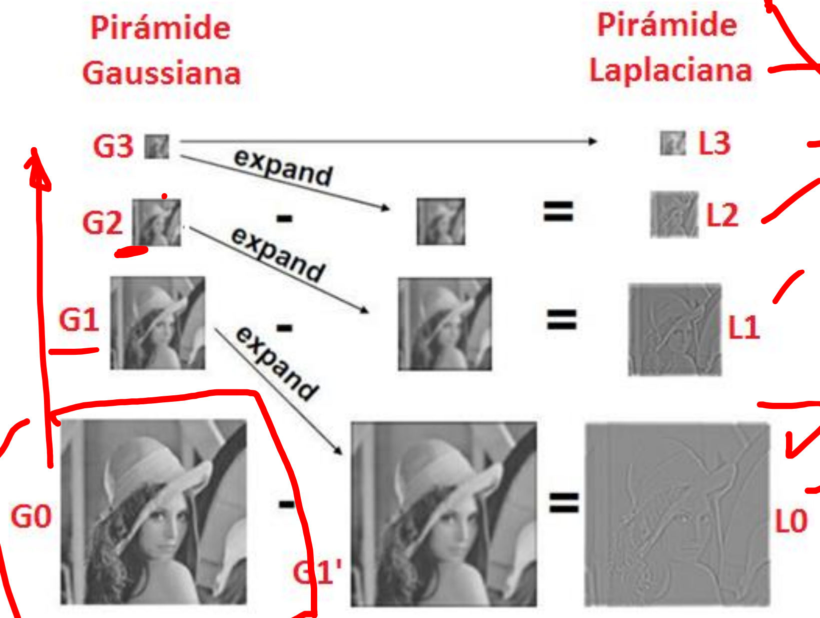
Una vez obtenida la imagen resultado se debe post procesar para encontrar los máximos o mínimos de intensidad.

`np.where()`, `cv.minMaxLoc()`

A tener en cuenta:

- Utilizar alguna métrica para filtrar solapamiento entre detecciones muy cercanas (IoU, NMS, etc…)
- Hacer un análisis exploratorio y elegir la métrica de apropiada.
- Pre-procesar el template y/o las imágenes para reducir la información a correlacionar (canny, pirámides, etc…)
- La correlación puede dar un falso positivo si se aplica a áreas de mucha intensidad ej: zonas blancas.
- La suma de diferencias puede llevar a resultados erróneos si la imagen tiene zonas de baja intensidad.





PIRÁMIDES

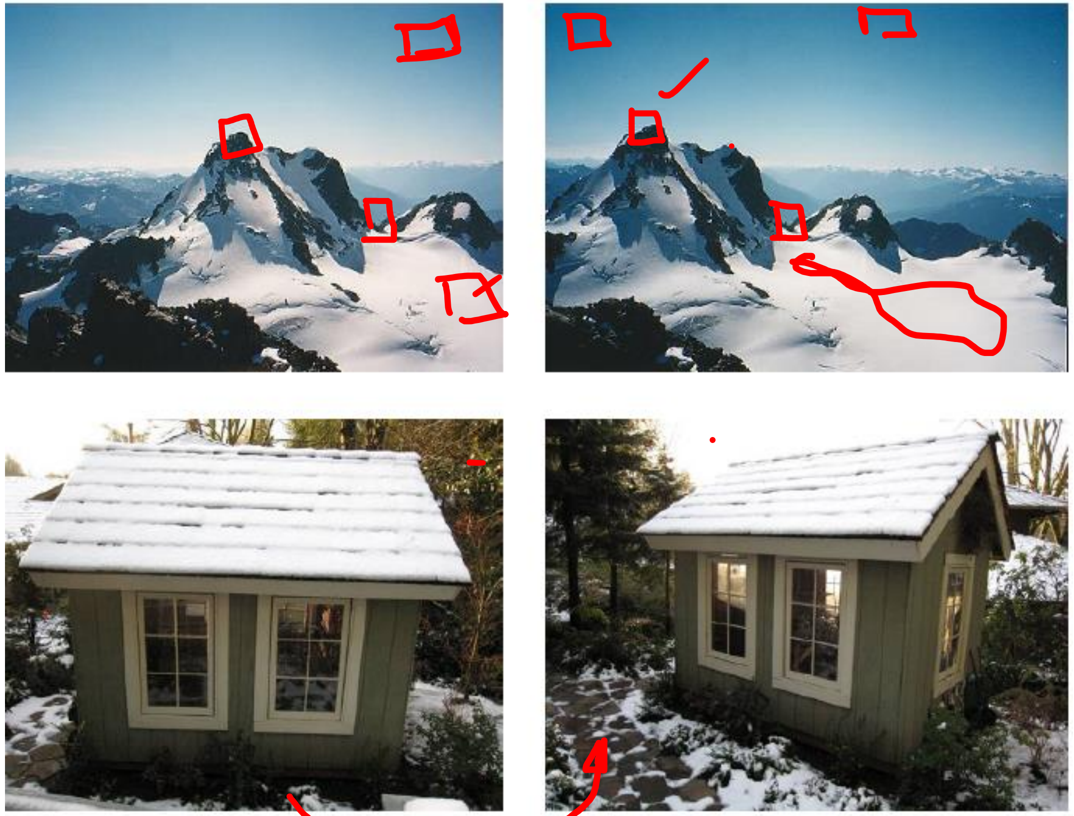
- ¿Qué pasa si queremos encontrar la cara de Messi en distintas fotos en la web, con diferentes escalas?
  - *"Los sistemas visuales biológicos funcionan también con una jerarquía de escalas (Marr, 1982)"*
- Las pirámides se refieren a representar una misma imagen en *múltiples resoluciones*.
- Cada 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 Gaussiana (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* $G1'$*)*

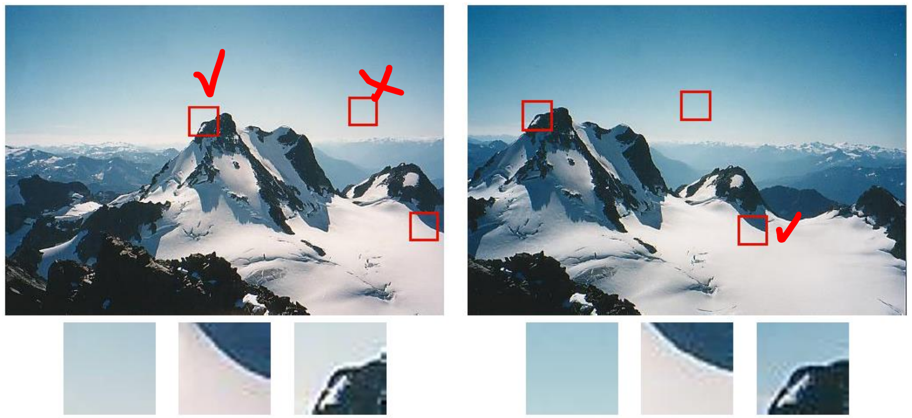
- Aplicaciones:
  1. Búsqueda de objetos en distintas resoluciones
  2. Aceleración de procesamiento (encontrando objetos en las resoluciones más bajas y procesando en las altas
  3. Características que pueden pasar desapercibidas en una resolución se pueden hallar en alguna otra
  4. Operaciones de fusión de imágenes
  5. Eulerian Magnification: http://people.csail.mit.edu/mrub/evm/

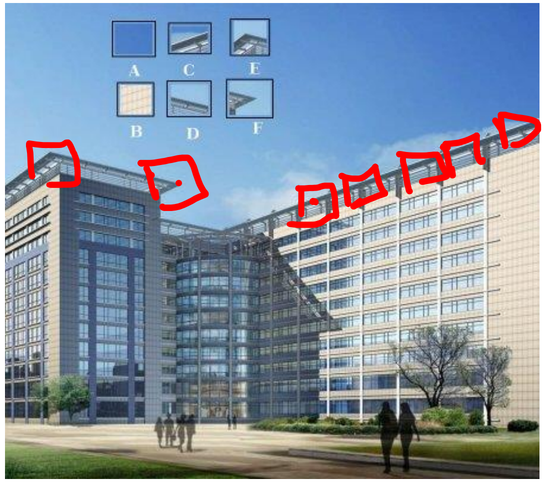
CARACTERÍSTICAS LOCALES

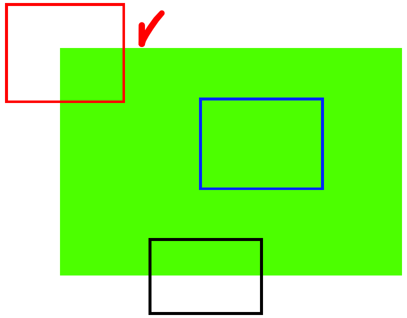


- La detección de características (en general, para su posterior coincidencia) son un componente esencial en muchas aplicaciones de visión:
  1. Alineación de imágenes
  2. Estimación de POSE/SLAM
  3. Construcción de modelos 3D
  4. Generación de vistas intermedias

- Podemos dividir el proceso de detección/coincidencia de características en 4 etapas:
  1. Detección (extracción) x, y.
  2. Descripción (conversión en descriptores estables, invariantes)
  3. Correspondencia (búsqueda de coincidencias)
  4. Seguimiento (A través de una secuencia de imágenes o video)







CARACTERÍSTICAS LOCALES

- Parches
  - Bordes
  - Esquinas

La correspondencia de parches se puede escribir como:

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

$\boldsymbol{I_0}$, $\boldsymbol{I_1}$: *Imagen y parche*
$\boldsymbol{w}(\boldsymbol{x_i})$: *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.

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

DETECTOR DE ESQUINAS DE HARRIS (I)

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

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

$\cancel{x_i} + \Delta u - \cancel{x_i}$

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

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

$\Delta u = \Delta v + \Delta u$

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

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

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

- 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

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

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

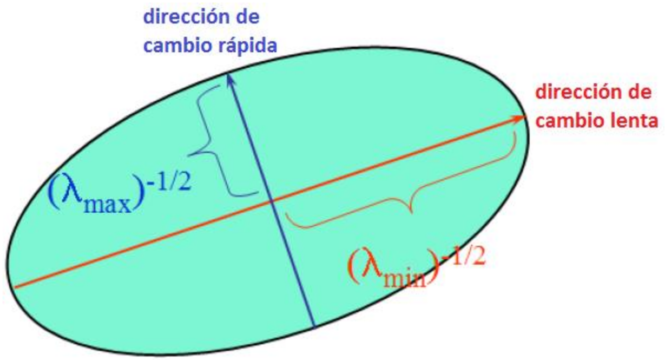
DETECTOR DE ESQUINAS DE HARRIS (II)

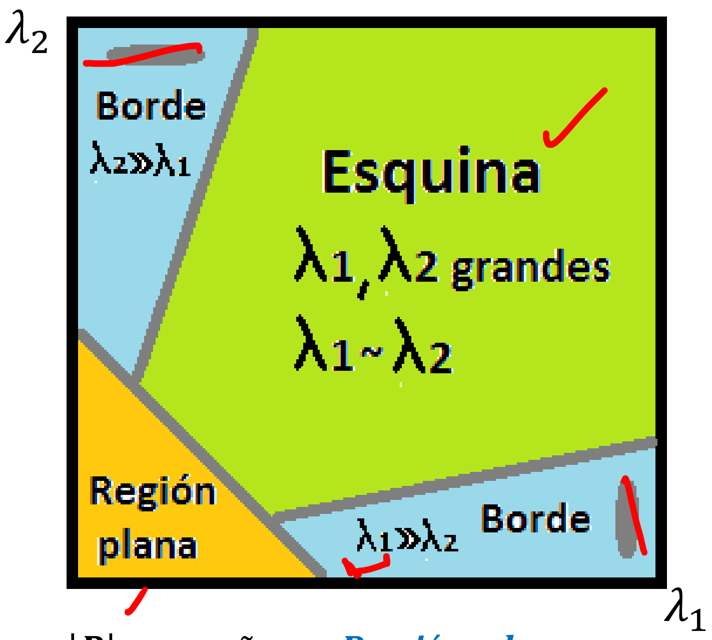
- La mejor manera de visualizar la acción de la matriz de autocorrelación es realizando un análisis de autovalores.
- Luego, para encontrar una puntuación que indique las características hay distintas aproximaciones:
  1. Harris/Stephens (1988)

     $$\boldsymbol{R = \det(A) - k\left(tr(A)\right)^2 = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2}$$
     $\det(A) = \lambda_1\lambda_2$
     $tr(A) = \lambda_1 + \lambda_2$
     $k = 0{,}06$
  2. Triggs (2004). Mejor respuesta a bordes 1-D (donde se sobredimensiona el autovalor más pequeño)

     $$\boldsymbol{R = \lambda_1 - k\,\lambda_2}$$
     $k = 0{,}05$
  3. Brown, Szeliski, Winder (2005). Usa la media armónica, función más suave donde $\lambda_1 \approx \lambda_2$

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





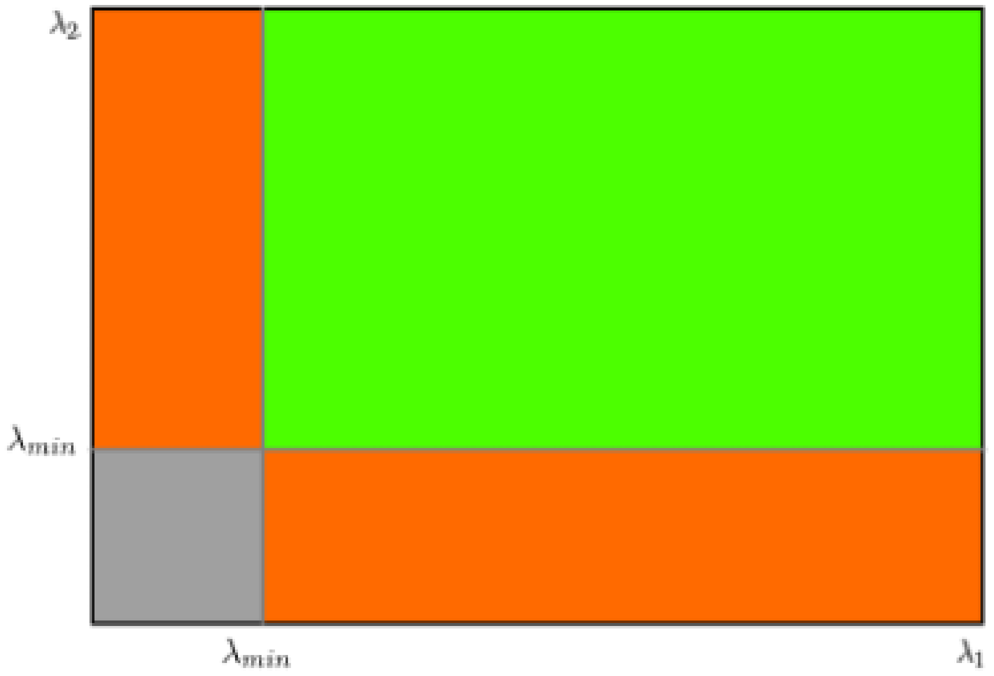
$\boldsymbol{|R|\ pequeño} \rightarrow$ ***Región plana***
$\boldsymbol{R < 0} \rightarrow \boldsymbol{\lambda_i \gg \lambda_j} \rightarrow$ ***Borde***
$\boldsymbol{R\ grande} \rightarrow \boldsymbol{\lambda_1\ y\ \lambda_2\ grandes} \rightarrow$ ***Esquina***

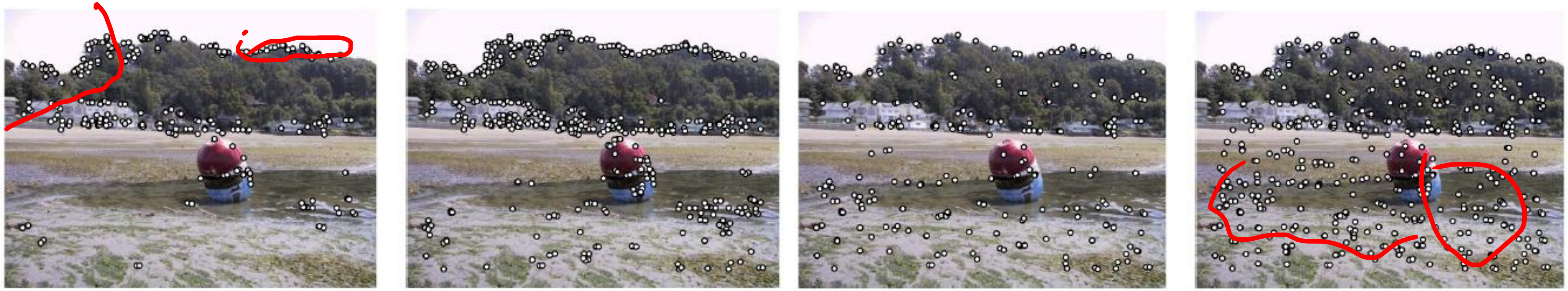
DETECTOR DE ESQUINAS – SHI-TOMASI

- 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:

$$R = min(\lambda_1, \lambda_2)$$

- 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) ✓





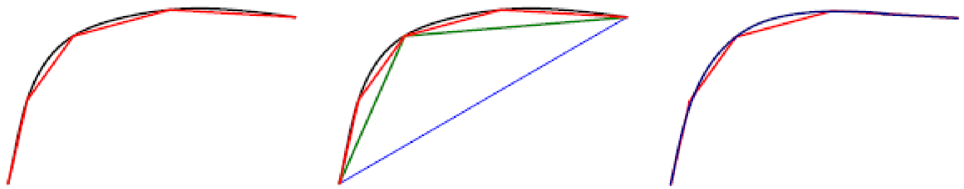
(a) Strongest 250 (b) Strongest 500 (c) ANMS 250, $r = 24$ (d) ANMS 500, $r = 16$

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**

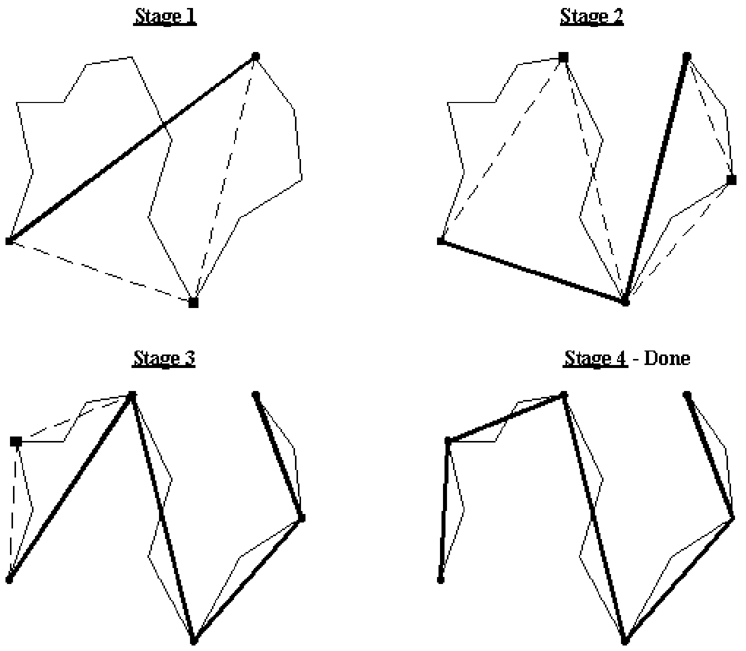
LÍNEAS

- 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.
  - Soporta oclusión parcial.

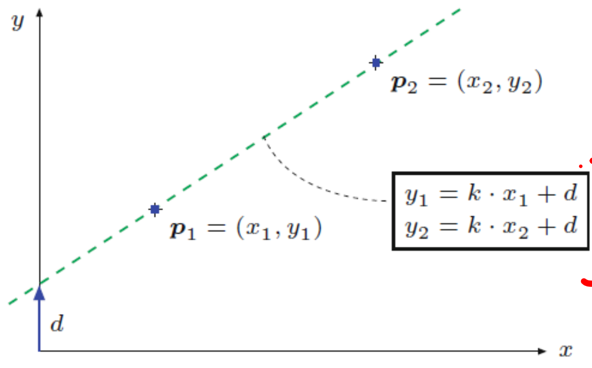


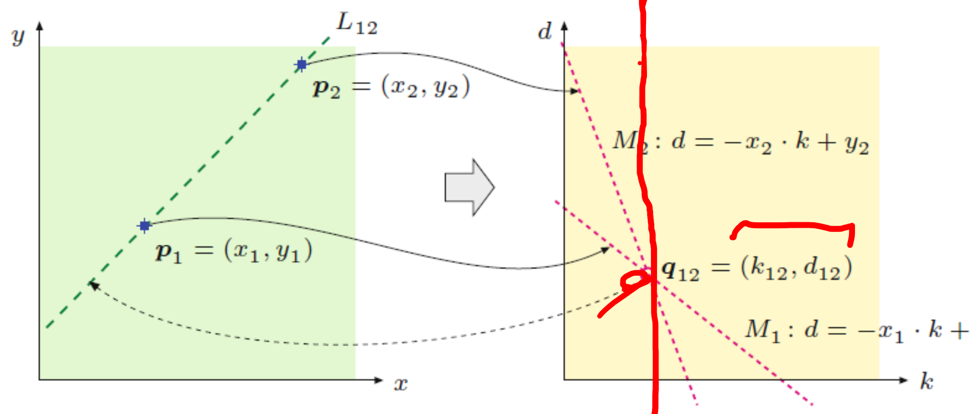
(a) (b) (c)

Douglas y Peucker: Cada punto agregado corresponde al vértice más alejado del segmento



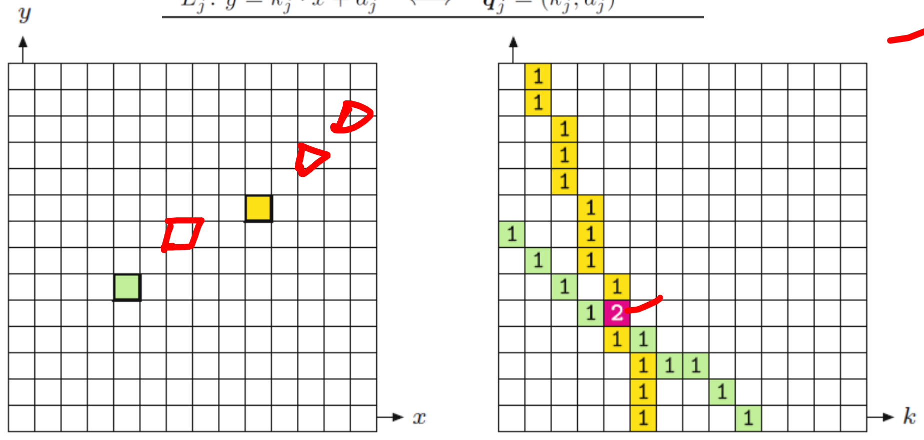
TRANSFORMA DE HOUGH





(a) x/y Image space (b) k/d Parameter space

$p_i = (x_i, y_i)$	$\longleftrightarrow$	$M_i : d = -x_i \cdot k + y_i$
$L_j : y = k_j \cdot x + d_j$	$\longleftrightarrow$	$q_j = (k_j, d_j)$



(a) Image space (b) Accumulator map

- El objetivo es encontrar los parámetros que describen a las líneas que contienen mas puntos de borde.
- Buscamos puntos de intersección en el espacio de parámetros para elegir a los mejores candidatos (d,k)
- Nos permite encontrar otras formas geométricas parametrizadas a partir de una imagen de bordes.
- Para el caso de rectas: Hough transforma puntos, desde el espacio de la imagen, a líneas en el espacio de parámetros.

HOUGH - LÍNEAS

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

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

  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)

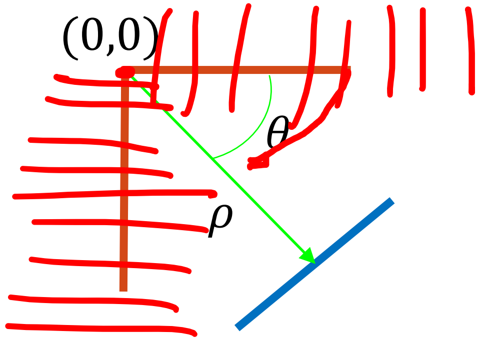
$$k^n \ (n\ dimensiones, k\ bins\ cada\ uno)$$

- ¿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.

$$\nabla I = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) = \hat{n}:(n_x, n_y)$$

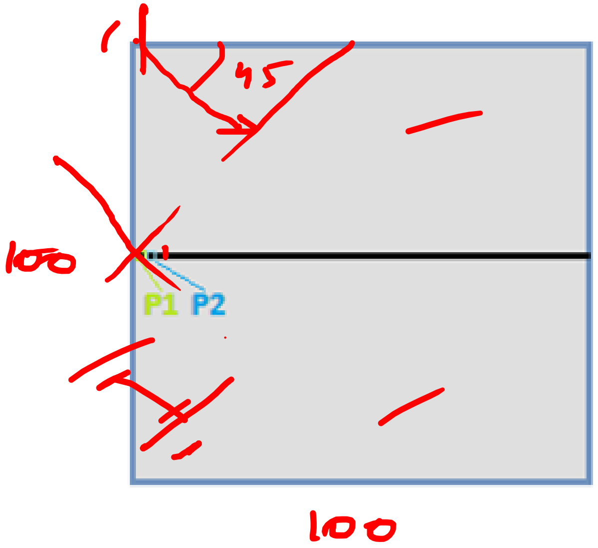
$$\theta = \tan^{-1}\left(\frac{n_y}{n_x}\right) \ ; \ \rho = x.nx + y.ny$$

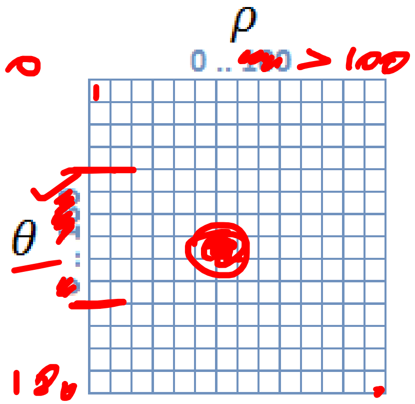
  2. Dar mayor peso a los bordes más fuertes (mayor magnitud)
  3. Cambiar la resolución de (ρ, θ) de mayor a menor iterativamente
  4. Utilizar el mismo procedimiento con círculos, cuadrados, etc.

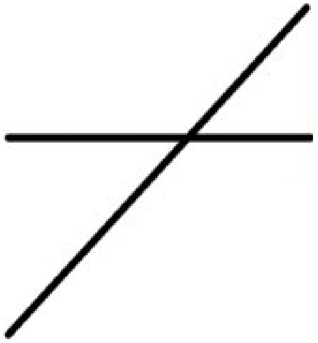


$-180º < \theta < 180º$
$\theta = 0º \rightarrow$ *líneas verticales*
$\theta = 90º \rightarrow$ *líneas horizontales*
$\theta < 0º \rightarrow$ *pasa por arriba del origen*
$\theta > 0º \rightarrow$ *pasa por debajo del origen*

Ejemplo:





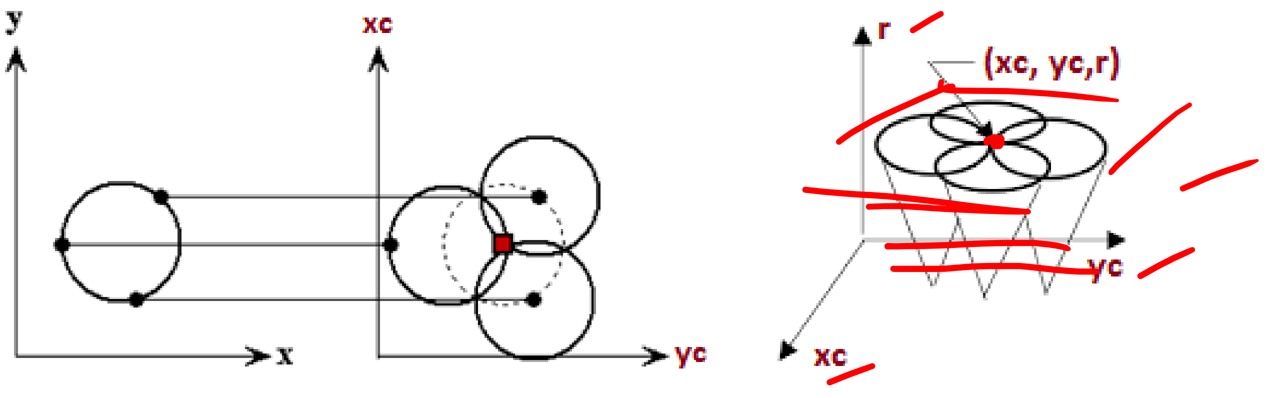


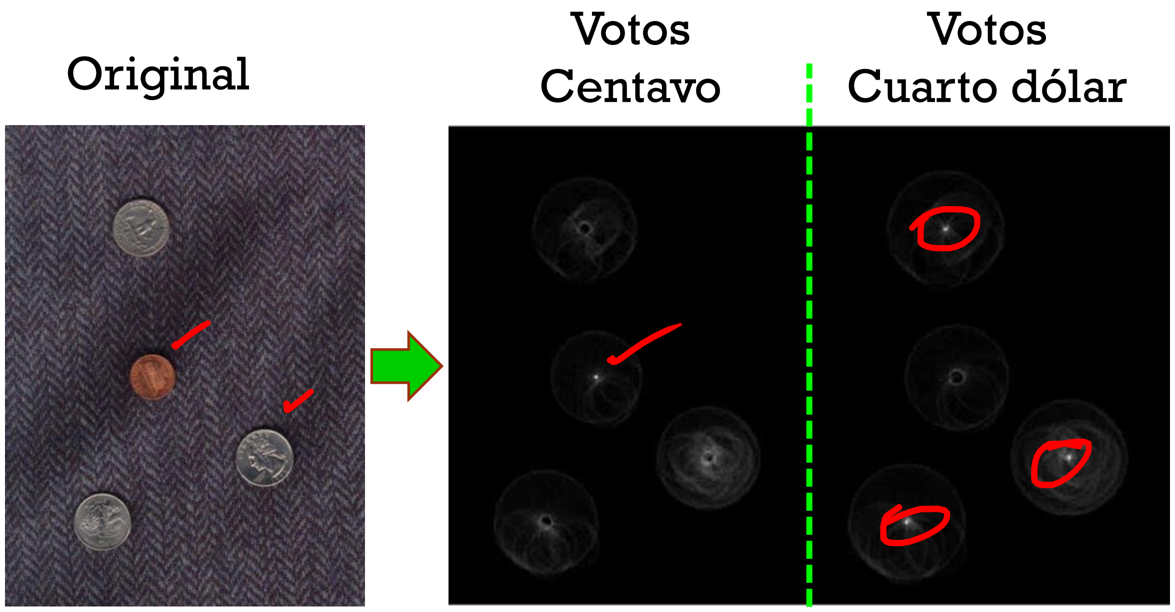
HOUGH - CÍRCULOS

- 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.





(x,y)

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.

TP3

- Encontrar el logotipo de la gaseosa dentro de las imágenes provistas en `Material_TPs/TP3/images` a partir del template `Material_TPs/TP3/template`

1. (4 puntos) Obtener una detección del logo en cada imagen sin falsos positivos
2. (4 puntos) Plantear y validar un algoritmo para múltiples detecciones en la imagen `coca_multi.png` con el mismo témplate del ítem 1
3. (2 puntos) Generalizar el algoritmo del item 2 para todas las imágenes.

Visualizar los resultados con bounding boxes en cada imagen mostrando el nivel de confianza de la detección.