

Cinvestav
Tamaulipas



Reconocimiento de objetos en fotografías

Ponente: Dr. Wilfrido Gómez Flores
Investigador CINVESTAV Tamaulipas

wgomez@tamps.cinvestav.mx

Ciudad Victoria, Tamaulipas, 3 de junio de 2015

Introducción

Reconocimiento de objetos

- El reconocimiento de objetos es la tarea de **encontrar** e **identificar** automáticamente objetos en una imagen.
- El ser humano no tiene ninguna dificultad para reconocer objetos, aún si éstos sufren variaciones.
- ¿Cómo hacer que una máquina lo haga?



Reconocimiento de objetos

- Actualmente no existe un sistema artificial “reconoce-todo”.



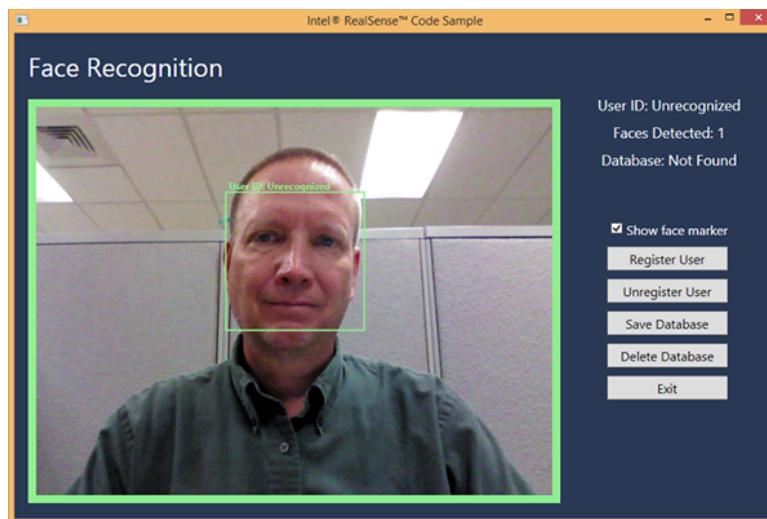
Reconocimiento de objetos

- Un sistema de reconocimiento de objetos está diseñado para detectar y clasificar objetos específicos en una imagen.
- Aplicaciones en la industria, seguridad, medicina, milicia, robótica, etc.

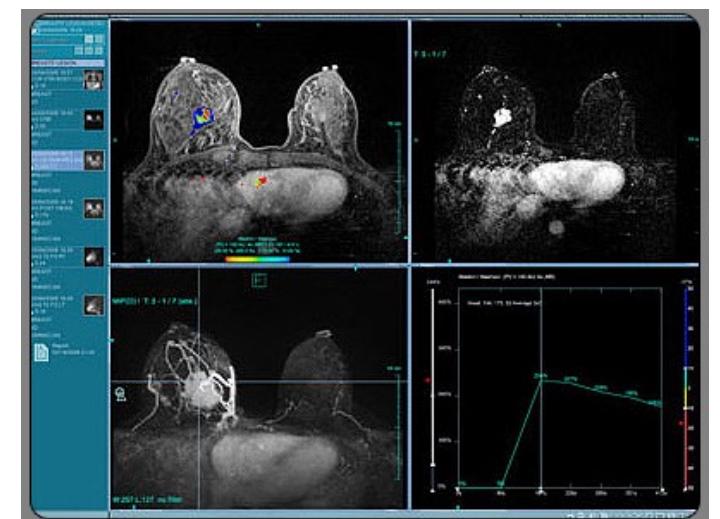
Reconocimiento de placas



Identificación de personas

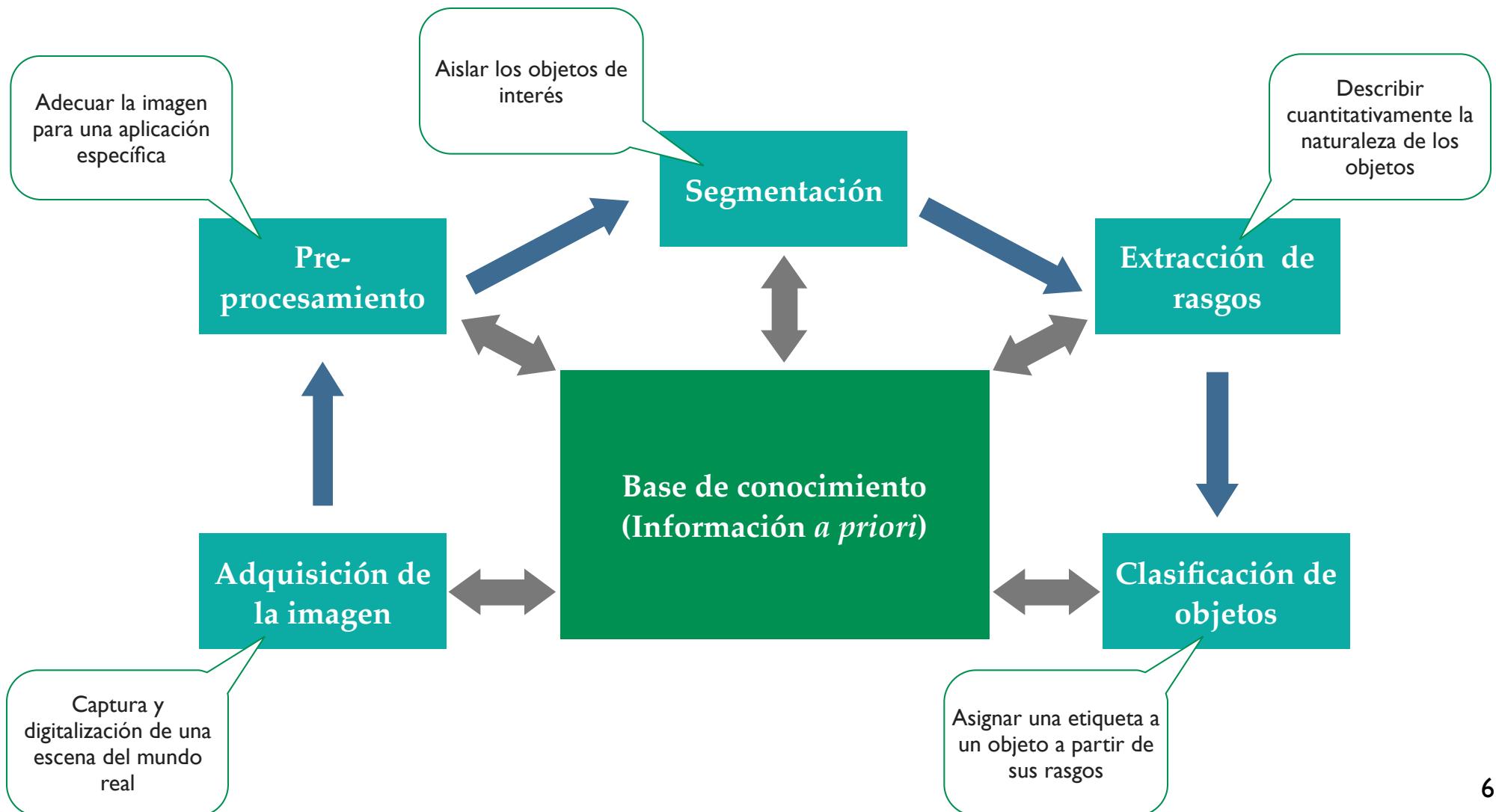


Diagnóstico asistido por computadora



Reconocimiento de objetos

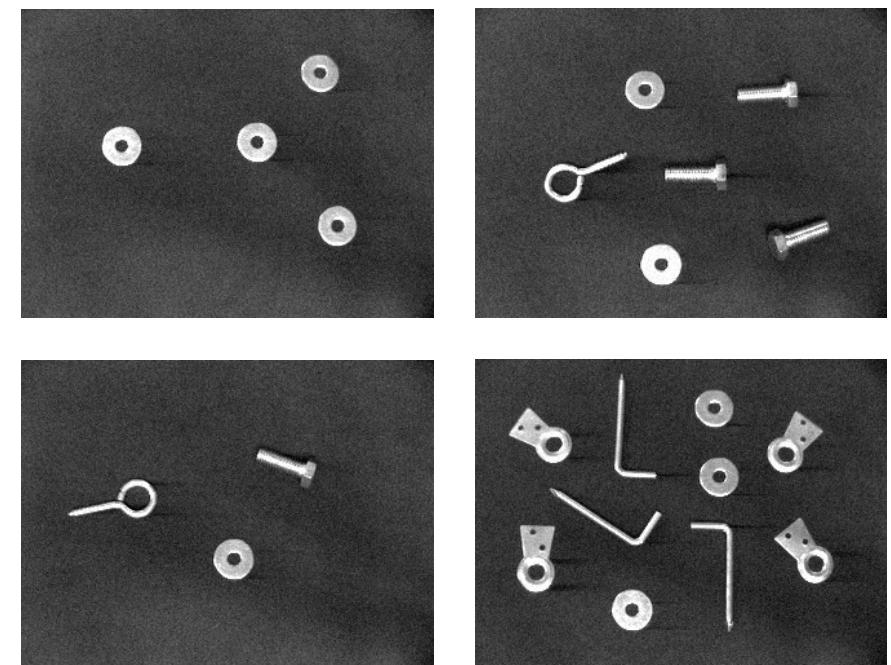
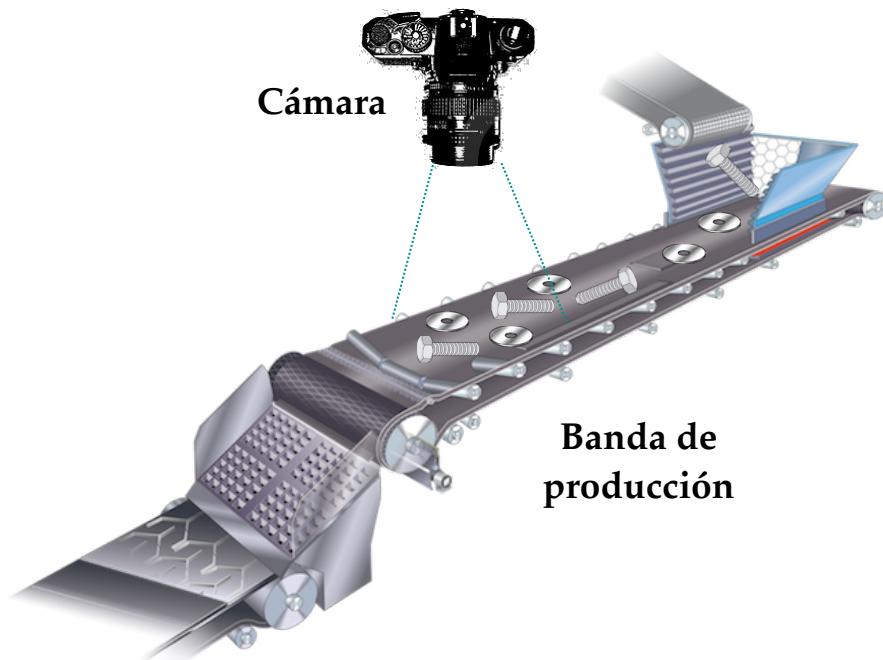
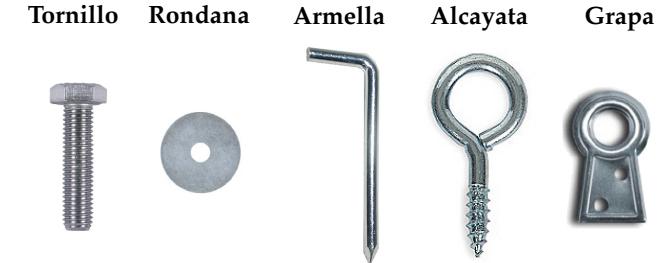
- Un sistema de reconocimiento de objetos generalmente está constituido por las siguientes etapas básicas:



Desarrollo del sistema de reconocimiento de objetos

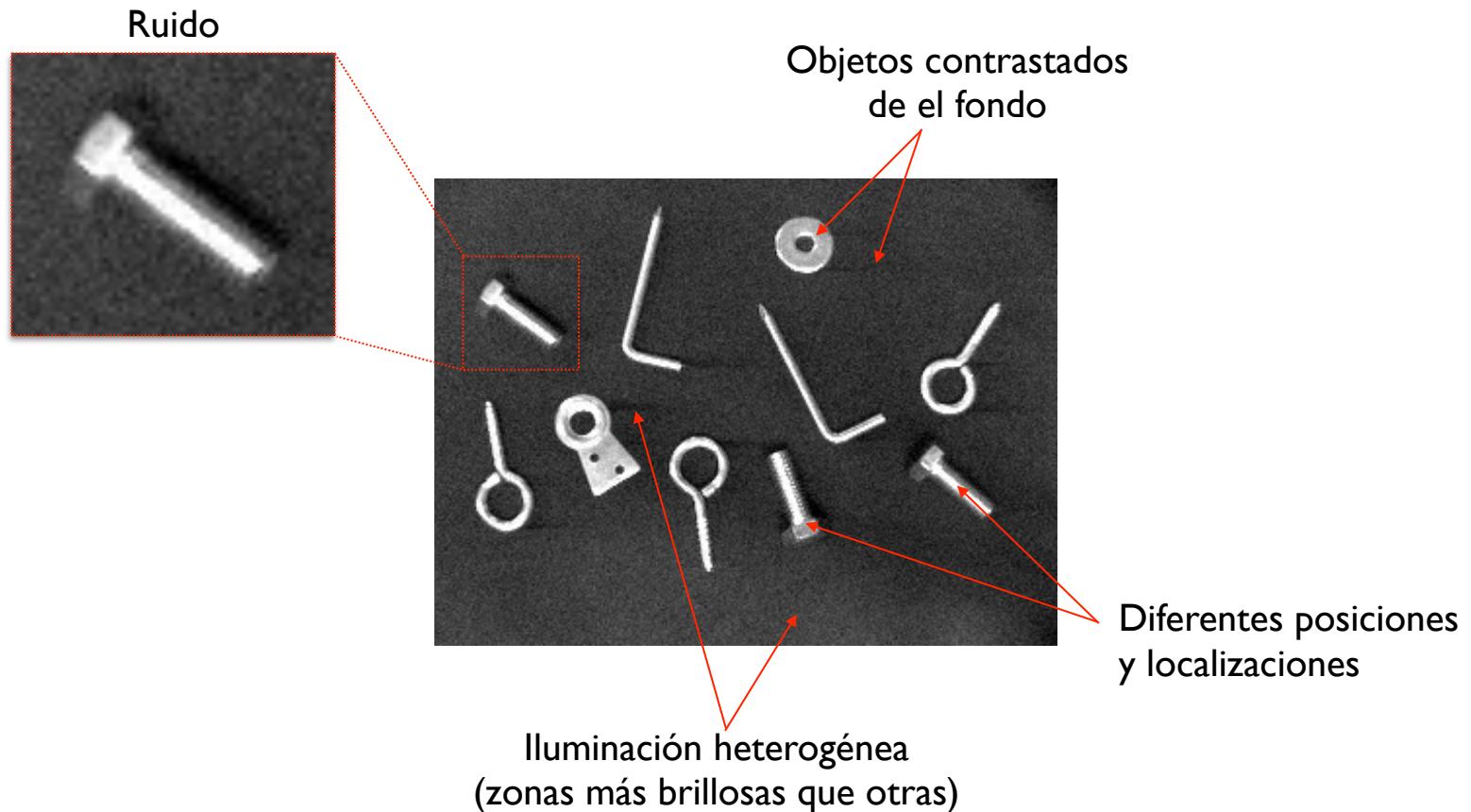
Base de conocimiento

- **Problema:** reconocer automáticamente cinco clases de piezas de ferretería que pasan por una banda de producción.
- Una cámara convencional proporciona fotografías de 320×240 píxeles y 8-bits de profundidad (i.e., 256 niveles de gris).



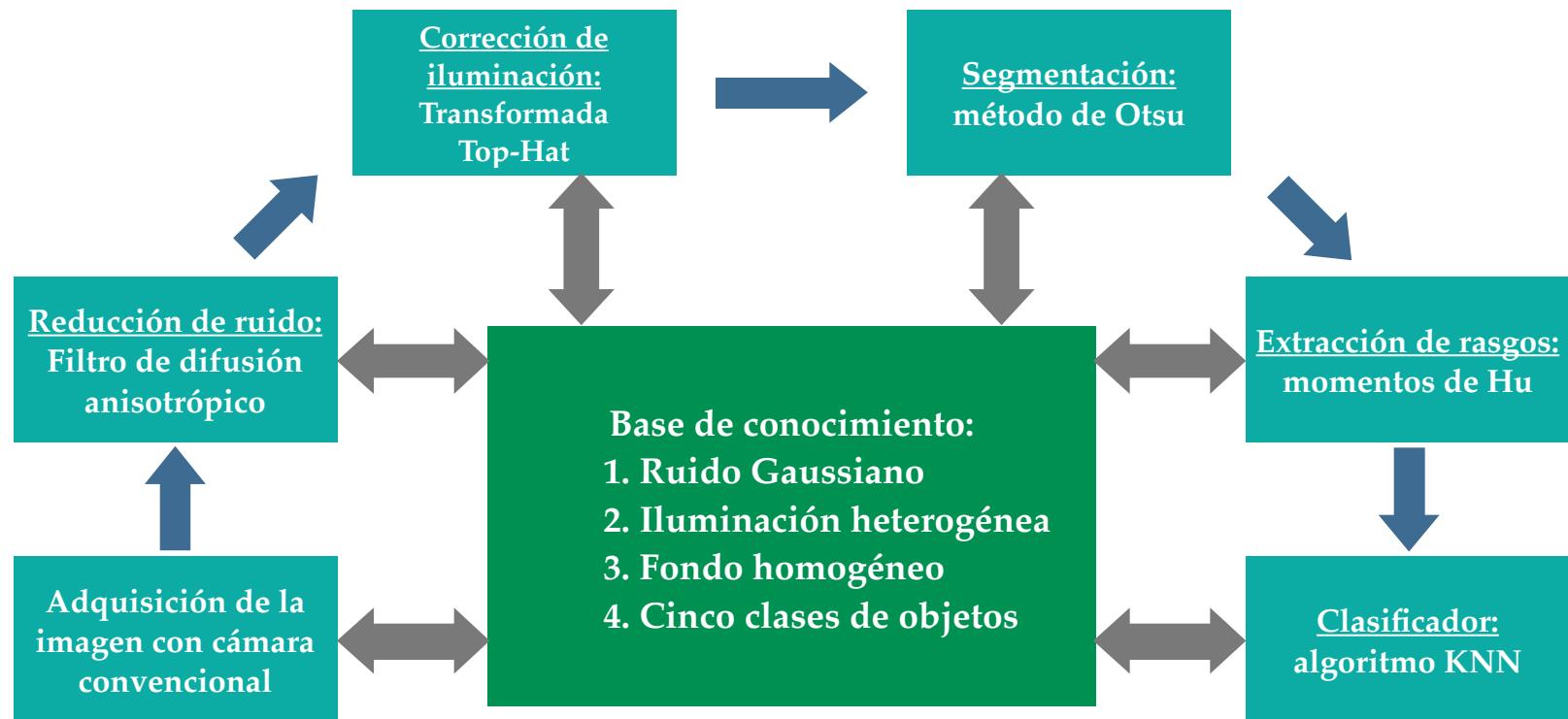
Base de conocimiento

- Características notables de las imágenes:



Base de conocimiento

- Propuesta de solución basado en las etapas básicas de un sistema de reconocimiento de objetos.

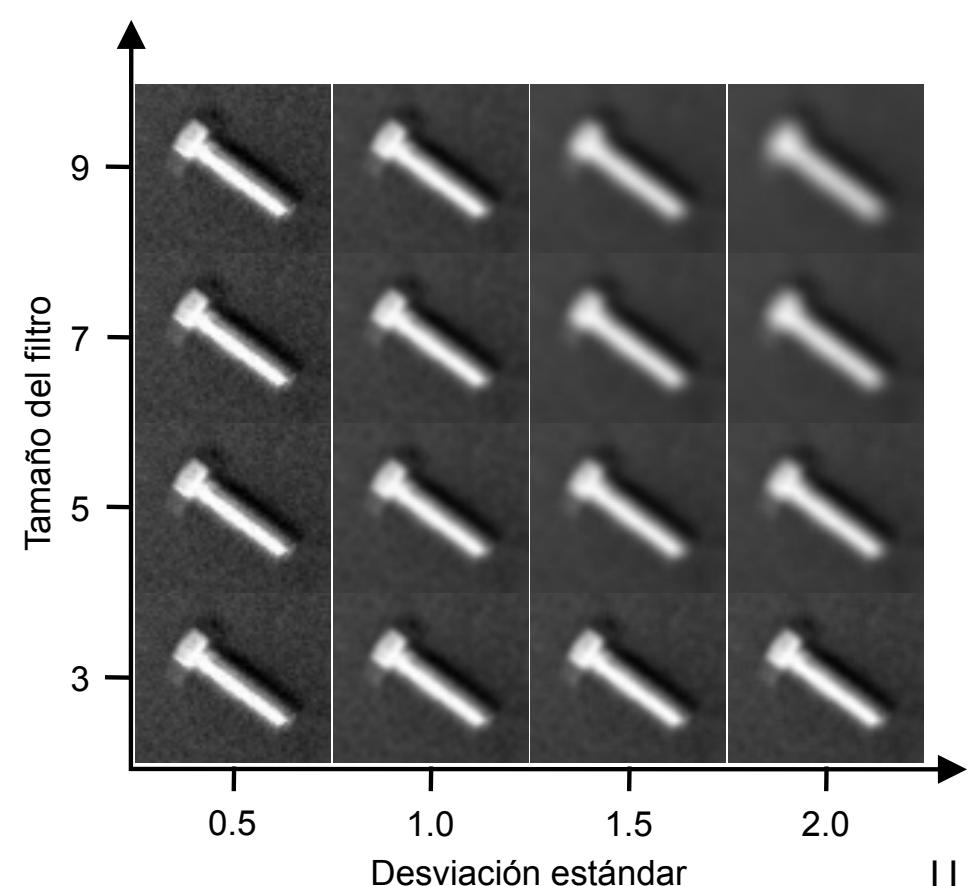
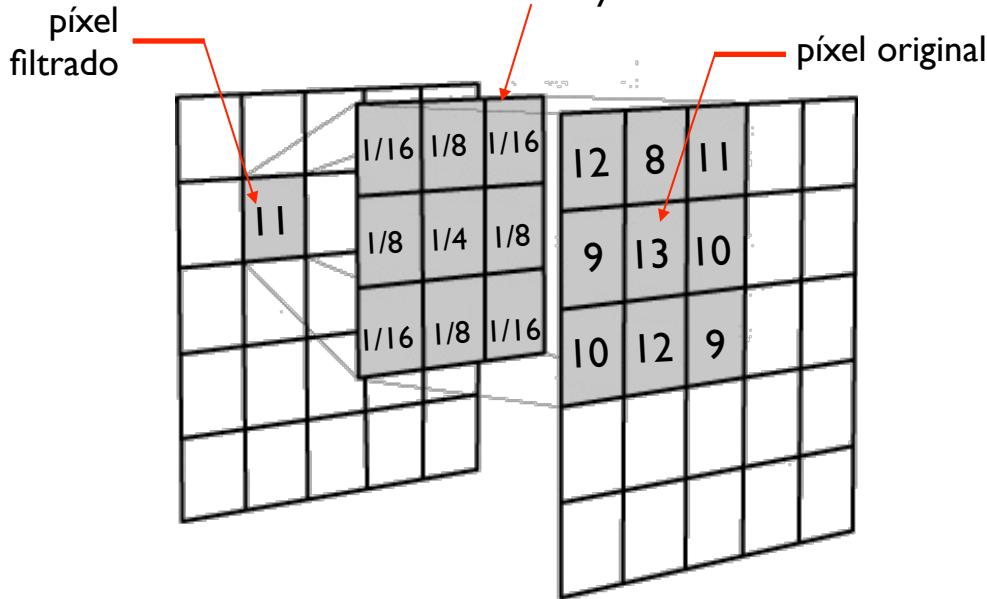


Reducción de ruido

- Ruido es cualquier componente indeseada que deteriora la calidad de la imagen.
- Comúnmente el ruido se reduce mediante filtros lineales suavizantes.
- Compromiso entre reducción de ruido y deformación de la imagen.

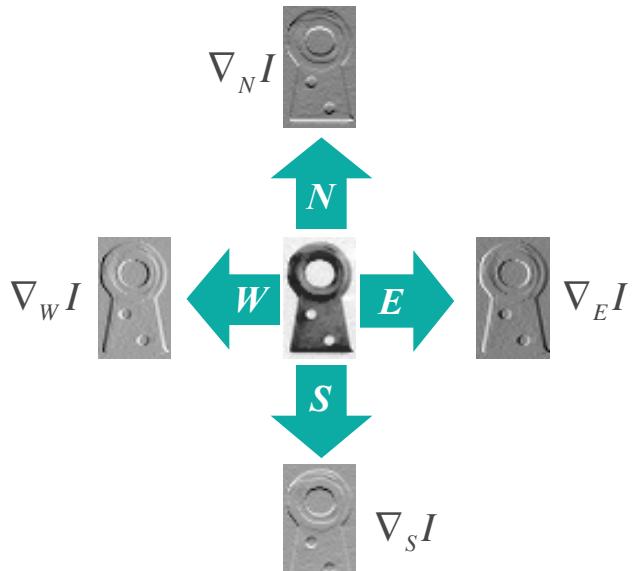
Un filtro lineal realiza una suma de productos

Filtro Gaussiano
tamaño 3×3 y $\sigma = 1$



Reducción de ruido

- Para preservar los detalles de la forma original de los objetos y reducir el ruido eficientemente se aplica el **filtro de difusión anisotrópico** (ADF).
- Reduce el ruido en las regiones homogéneas y no en los bordes.

Teoría	Explicación
<p>★ El filtrado ADF se expresa de manera iterativa como:</p> $I_{i,j}^{t+1} = I_{i,j}^t + \frac{1}{4} [g_N \nabla_N I + g_S \nabla_S I + g_W \nabla_W I + g_E \nabla_E I]_{i,j}^t$ <p>donde:</p> $\nabla_N I = I_{i-1,j} - I_{i,j}$ $\nabla_S I = I_{i+1,j} - I_{i,j}$ $\nabla_E I = I_{i,j+1} - I_{i,j}$ $\nabla_W I = I_{i,j-1} - I_{i,j}$ <p>y</p> $g_N = g(\nabla_N I)$ $g_S = g(\nabla_S I)$ $g_E = g(\nabla_E I)$ $g_W = g(\nabla_W I)$ <p>y la función de difusión:</p> $g(\nabla I) = \frac{1}{1 + (\ \nabla I\ /\kappa)^2}$	<p>★ El símbolo ∇ indica gradiente de la imagen I en alguna de las direcciones N, S, W, ó E.</p> <p>★ Se calcula mediante la resta entre el píxel $I_{i,j}$ y cuatro vecinos alrededor de él.</p> 

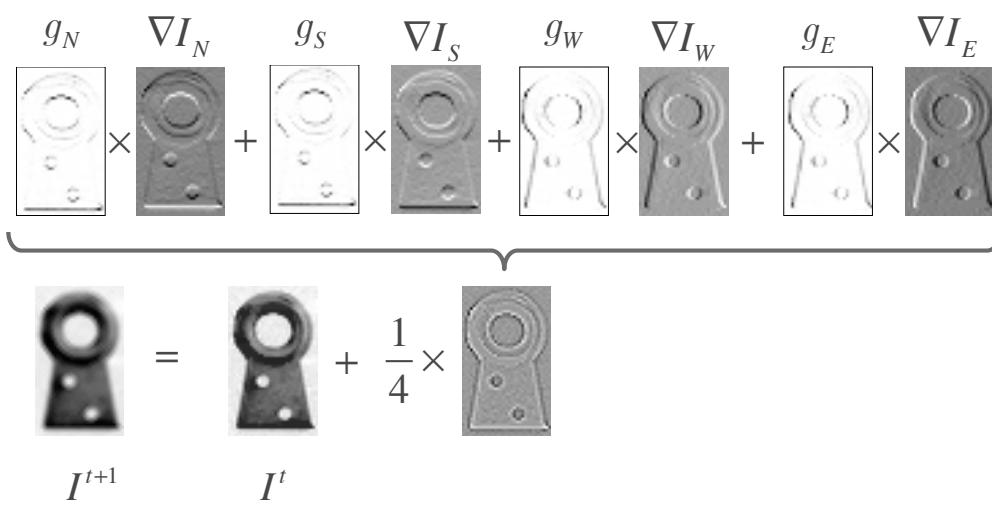
Reducción de ruido

- Para preservar los detalles de la forma original de los objetos y reducir el ruido eficientemente se aplica el **filtro de difusión anisotrópico** (ADF).
- Reduce el ruido en las regiones homogéneas y no en los bordes.

Teoría	Explicación															
<p>★ El filtrado ADF se expresa de manera iterativa como:</p> $I_{i,j}^{t+1} = I_{i,j}^t + \frac{1}{4} [g_N \nabla_N I + g_S \nabla_S I + g_W \nabla_W I + g_E \nabla_E I]_{i,j}^t$ <p>donde:</p> $\begin{aligned} \nabla_N I &= I_{i-1,j} - I_{i,j} & g_N &= g(\nabla_N I) \\ \nabla_S I &= I_{i+1,j} - I_{i,j} & g_S &= g(\nabla_S I) \\ \nabla_E I &= I_{i,j+1} - I_{i,j} & y & \\ \nabla_W I &= I_{i,j-1} - I_{i,j} & g_E &= g(\nabla_E I) \\ & & g_W &= g(\nabla_W I) \end{aligned}$ <p>y la función de difusión:</p> $g(\nabla I) = \frac{1}{1 + (\ \nabla I\ /\kappa)^2}$	<p>★ La función de difusión se aplicará a cada píxel de la imagen de gradiente.</p> <table border="1" data-bbox="1077 816 1964 1289"> <thead> <tr> <th data-bbox="1119 832 1161 864">∇I</th> <th data-bbox="1267 832 1309 864">W</th> <th data-bbox="1436 832 1478 864">E</th> <th data-bbox="1605 832 1647 864">S</th> <th data-bbox="1774 832 1816 864">N</th> </tr> </thead> <tbody> <tr> <td data-bbox="1119 881 1309 1060"></td><td data-bbox="1267 881 1309 1060"></td><td data-bbox="1436 881 1478 1060"></td><td data-bbox="1605 881 1647 1060"></td><td data-bbox="1774 881 1816 1060"></td></tr> <tr> <td data-bbox="1119 1077 1309 1256"></td><td data-bbox="1267 1077 1309 1256"></td><td data-bbox="1436 1077 1478 1256"></td><td data-bbox="1605 1077 1647 1256"></td><td data-bbox="1774 1077 1816 1256"></td></tr> </tbody> </table> <p>★ El coeficiente de difusión indica si un píxel debe ser filtrado o no.</p>	∇I	W	E	S	N										
∇I	W	E	S	N												

Reducción de ruido

- Para preservar los detalles de la forma original de los objetos y reducir el ruido eficientemente se aplica el **filtro de difusión anisotrópico** (ADF).
- Reduce el ruido en las regiones homogéneas y no en los bordes.

Teoría	Explicación
<p>★ El filtrado ADF se expresa de manera iterativa como:</p> $I_{i,j}^{t+1} = I_{i,j}^t + \frac{1}{4} [g_N \nabla_N I + g_S \nabla_S I + g_W \nabla_W I + g_E \nabla_E I]_{i,j}^t$ <p>donde:</p> $\begin{aligned} \nabla_N I &= I_{i-1,j} - I_{i,j} & g_N &= g(\nabla_N I) \\ \nabla_S I &= I_{i+1,j} - I_{i,j} & g_S &= g(\nabla_S I) \\ \nabla_E I &= I_{i,j+1} - I_{i,j} & y & \\ \nabla_W I &= I_{i,j-1} - I_{i,j} & g_E &= g(\nabla_E I) \\ & & g_W &= g(\nabla_W I) \end{aligned}$ <p>y la función de difusión:</p> $g(\nabla I) = \frac{1}{1 + (\ \nabla I\ /\kappa)^2}$	<p>★ Finalmente, la imagen para la siguiente iteración ($t+1$) se actualiza:</p>  $I^{t+1} = I^t + \frac{1}{4} \times \underbrace{\left[g_N \nabla_N I + g_S \nabla_S I + g_W \nabla_W I + g_E \nabla_E I \right]}_{I^{t+1}} = I^t + \frac{1}{4} \times \left[g_N \nabla_N I + g_S \nabla_S I + g_W \nabla_W I + g_E \nabla_E I \right]$

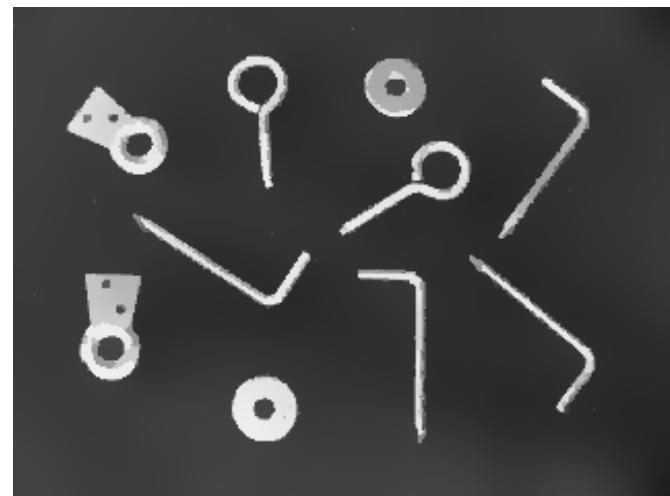
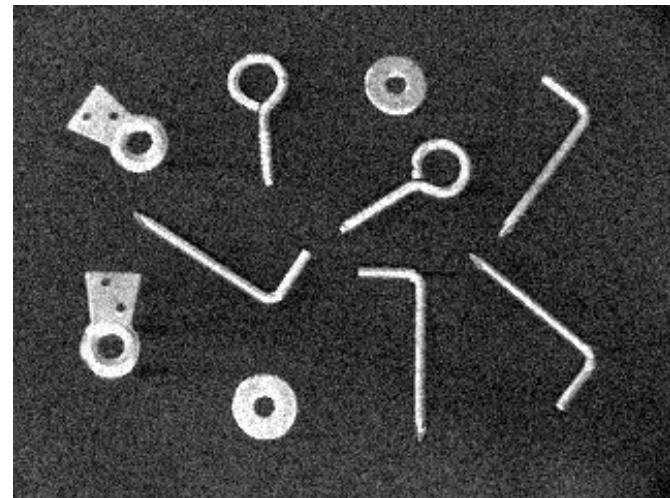
Reducción de ruido

Implementación numérica

```
function J = adf(I,kappa,tmax)
I = double(I);
[Y,X] = size(I);
N = [2:Y, Y];
S = [1, 1:Y-1];
E = [1, 1:X-1];
W = [2:X, X];
for t = 1:tmax
    [dN,dS,dE,dW] = gradients(I,N,S,E,W);
    gN = 1./(1+((abs(dN)/kappa).^2));
    gS = 1./(1+((abs(dS)/kappa).^2));
    gE = 1./(1+((abs(dE)/kappa).^2));
    gW = 1./(1+((abs(dW)/kappa).^2));
    I = I + 0.25*(gN.*dN + gS.*dS + gE.*dE + gW.*dW);
end
J = uint8(I);

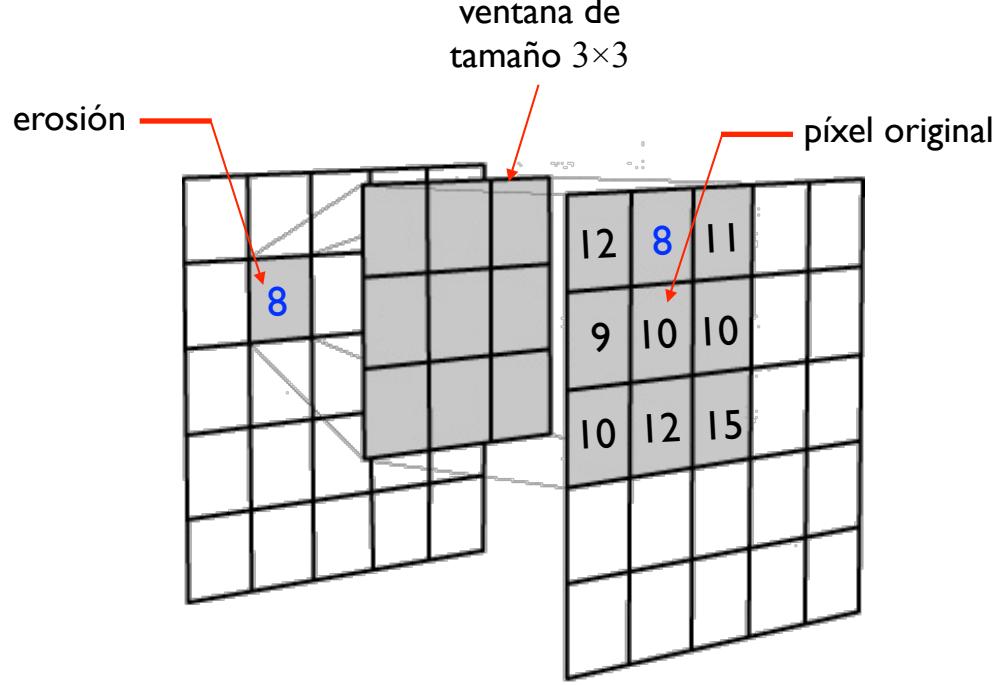
%*****%
% Calcula los gradientes
function [dN,dS,dE,dW] = gradients(I,N,S,E,W)
dE = I(:,E) - I;
dW = I(:,W) - I;
dN = I(N,:) - I;
dS = I(S,:) - I;
```

Ejemplo



Corrección de iluminación

- La iluminación heterogénea se puede corregir utilizando el **filtrado Top-Hat**.

Teoría	Explicación																																																																																										
<p>★ El filtro Top-Hat resta la imagen original, I, menos la imagen resultante después de someter I a un proceso de erosión y dilatación con una ventana, w, mayor al tamaño de los objetos en la imagen:</p> $\text{TH}_w(I) = I - \delta_w[\varepsilon_w(I)]$ <p>donde los operadores ε y δ denotan erosión y dilatación, respectivamente, y se computan como:</p> $\varepsilon_w(I) = \min_{(s,t) \in w} \{I(x-s, y-t)\}$ $\delta_w(I) = \max_{(s,t) \in w} \{I(x-s, y-t)\}$	<p>★ La erosión, ε, indica que se toma el valor mínimo de una ventana w y se coloca en el píxel central:</p>  <table border="1" data-bbox="1087 905 2080 1460"> <tr> <td></td><td></td><td></td><td></td><td></td> <td></td><td></td><td></td><td></td> </tr> <tr> <td>12</td><td>8</td><td>11</td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td>9</td><td>10</td><td>10</td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td>10</td><td>12</td><td>15</td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> </table>																																														12	8	11							9	10	10							10	12	15																								
12	8	11																																																																																									
9	10	10																																																																																									
10	12	15																																																																																									

Corrección de iluminación

- La iluminación heterogénea se puede corregir utilizando el **filtrado Top-Hat**.

Teoría

- El filtro Top-Hat resta la imagen original, I , menos la imagen resultante después de someter I a un proceso de erosión y dilatación con una ventana, w , mayor al tamaño de los objetos en la imagen:

$$\text{TH}_w(I) = I - \delta_w[\varepsilon_w(I)]$$

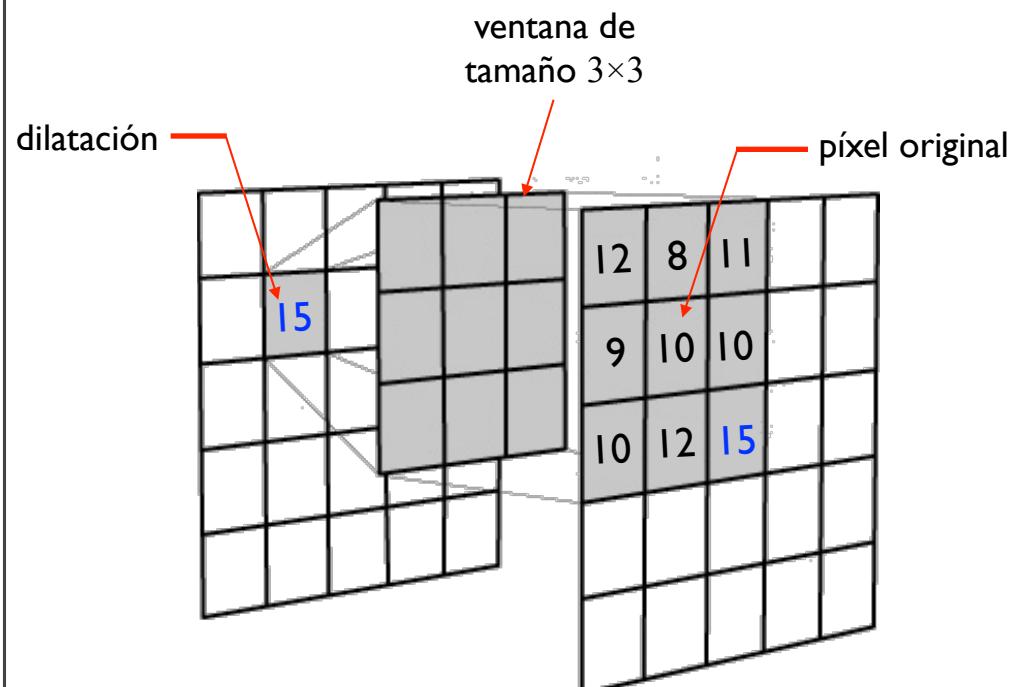
donde los operadores ε y δ denotan erosión y dilatación, respectivamente, y se computan como:

$$\varepsilon_w(I) = \min_{(s,t) \in w} \{I(x-s, y-t)\}$$

$$\delta_w(I) = \max_{(s,t) \in w} \{I(x-s, y-t)\}$$

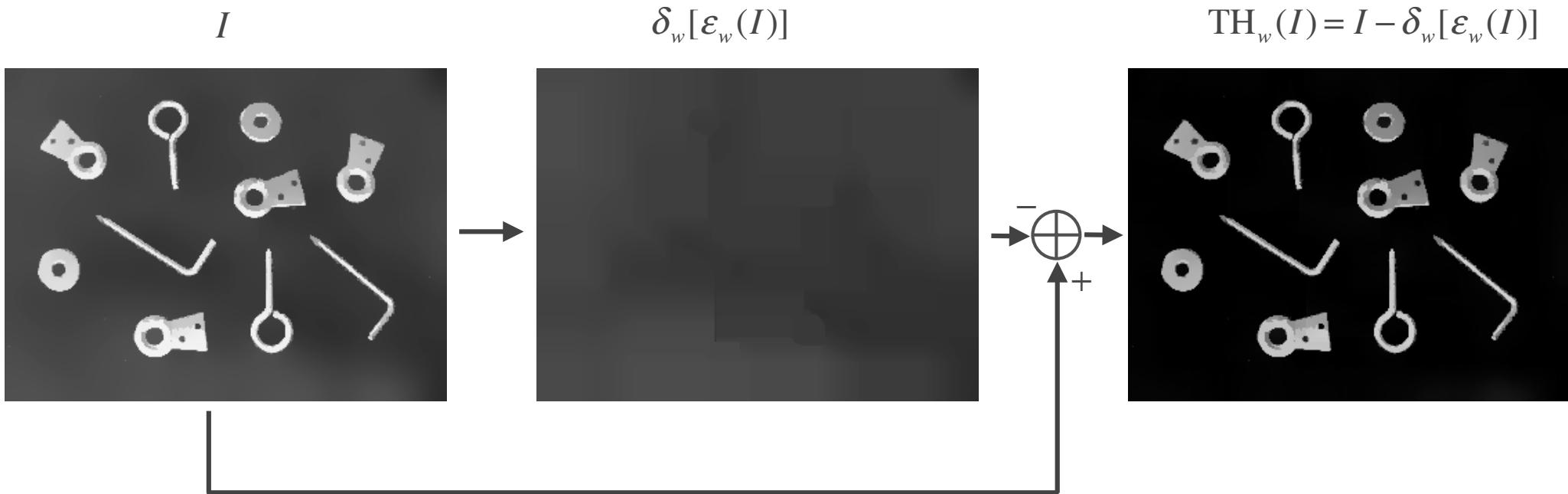
Explicación

- La dilatación, δ , indica que se toma el valor **máximo** de una ventana w y se coloca en el píxel central:



Corrección de iluminación

- **Explicación:** El filtro Top-Hat primero realiza una **erosión**, para borrar todos los objetos, seguido de una **dilatación**, para recuperar las intensidades del fondo, y el resultado lo **resta** a la imagen original, para cancelar el efecto del fondo:

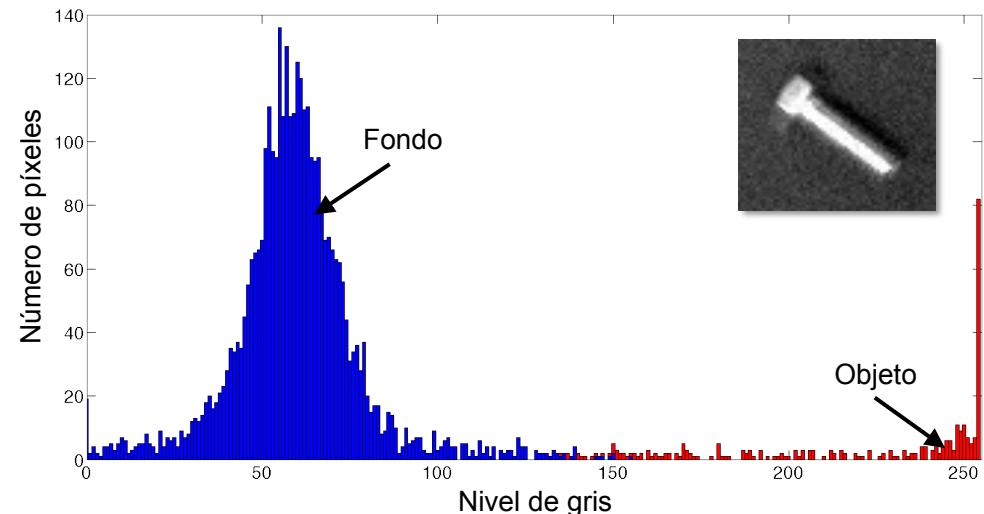


- **Implementación numérica**

```
w = strel('square',75); %Crea ventana cuadrada de tamaño 75x75
E = imerode(J,w); %Realiza la erosión de la imagen original
D = imdilate(E,w); %Realiza la dilataciónn de la erosión
TH = imsubtract(J,D); %Resta la dilatación a la imagen original
```

Segmentación

- El **método de Otsu** analiza el **histograma** de niveles de gris de la imagen para determinar un umbral, t , que segmente adecuadamente los objetos de el fondo.



- La **imagen segmentada** es una **imagen binaria**, donde el valor '0' corresponde a un píxel del fondo y el valor '1' corresponde a un píxel del objeto y se computa como:

$$B_{i,j} = \begin{cases} 1, & \text{si } I_{i,j} \geq t \\ 0, & \text{otro caso} \end{cases}$$

Segmentación

Teoría

- ★ El umbral óptimo, t^* , que separa los objetos de el fondo se encuentra en el nivel de gris que **maximiza la varianza interclase**:

$$t^* = \arg \max_{t=0,1,\dots,255} \left\{ A_t \cdot (A_{255} - A_t) \cdot (\mu_t - \nu_t)^2 \right\}$$

donde la media de los niveles de gris de los objetos y el fondo para el umbral t :

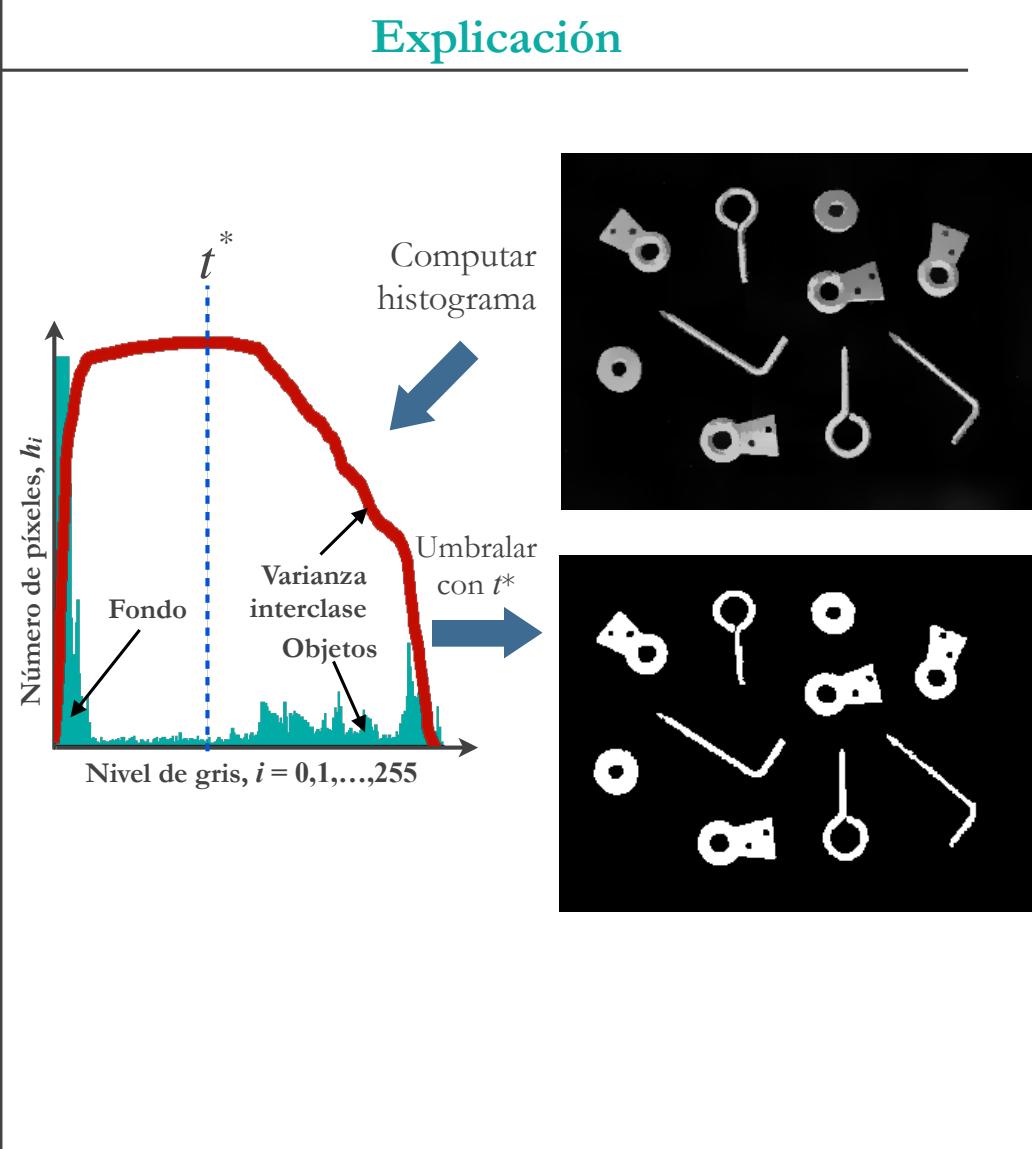
$$\mu_t = \frac{B_t}{A_t} \quad \text{y} \quad \nu_t = \frac{B_{255} - B_t}{A_{255} - A_t}$$

y las sumas acumuladas:

$$A_t = \sum_{i=0}^t h_i \quad \text{y} \quad B_t = \sum_{i=0}^t i \cdot h_i$$

donde h_i es el histograma de la imagen.

Explicación

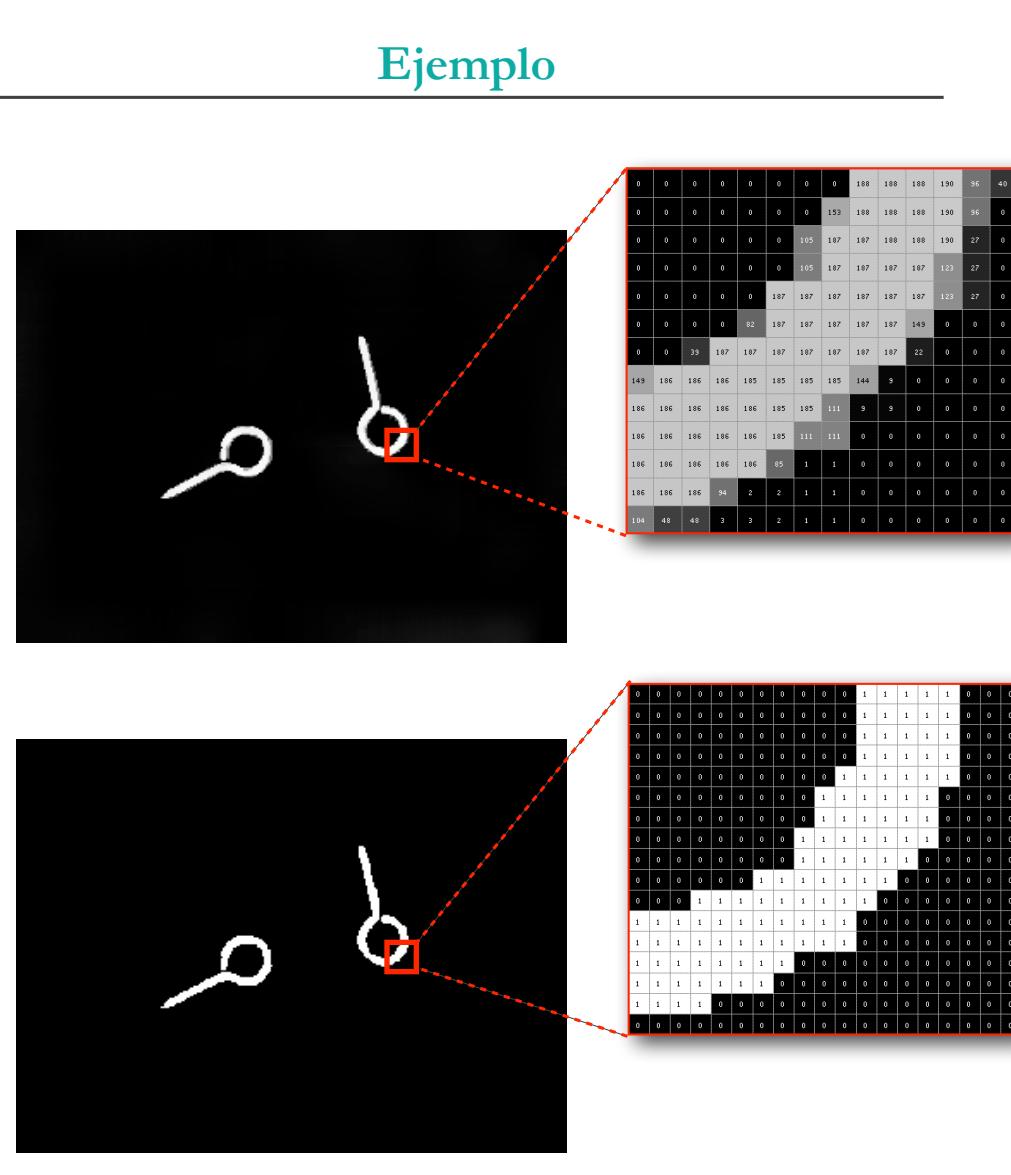


Segmentación

Implementación numérica

```
function B = otsu(I)
% Imagen de 8-bits (256 niveles de gris)
L = 2^8;
% Número de píxeles en la imagen
N = numel(I(:));
% Computa el histograma
hi = accumarray(I(:)+1,ones(N,1),[L 1],@sum,0);
% Computa sumas parciales
At = cumsum(hi)+eps;
Bt = cumsum((0:L-1)'.*hi)+eps;
A255 = At(L); B255 = Bt(L);
% Computa las medias de cada clase
mu_t = Bt./At;
nu_t = (B255-Bt)./((A255-At)+eps);
% Maximiza la varianza interclase
[~,t] = max(At.*(A255-At).* (mu_t-nu_t).^2);
% Umbraliza la imagen de entrada
B = I >= (t-1);
```

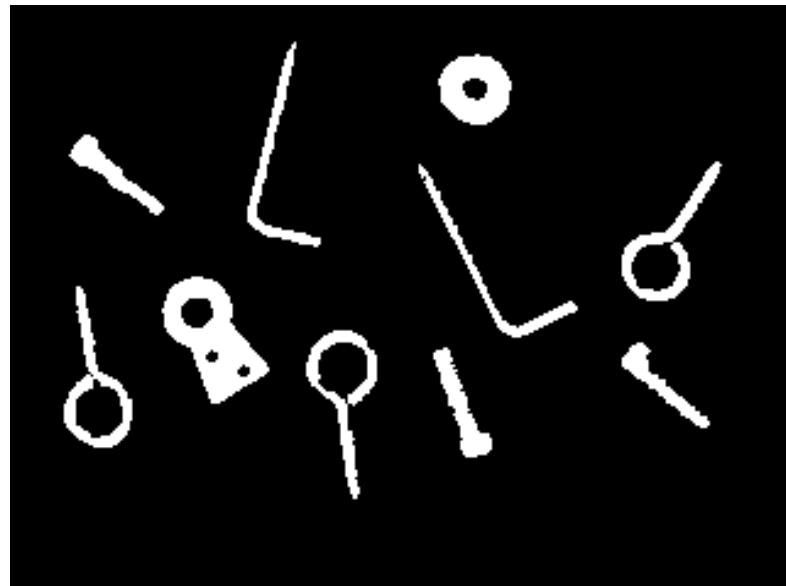
Ejemplo



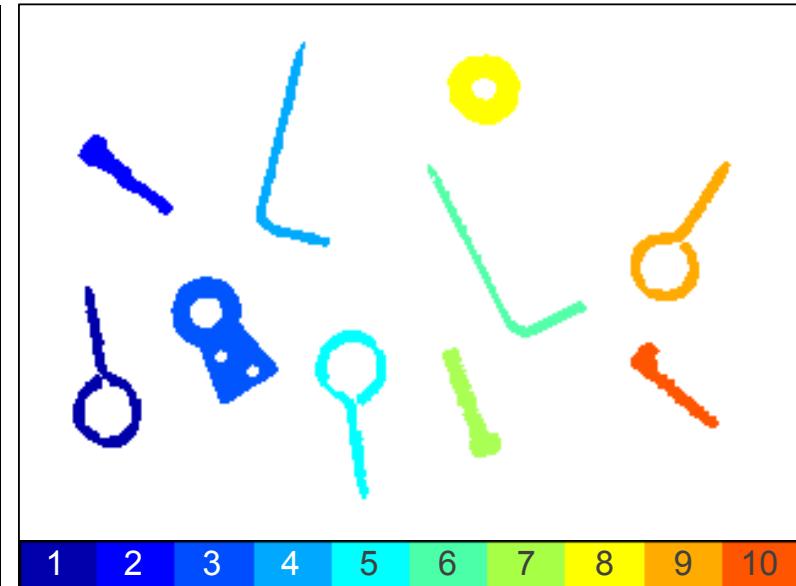
Segmentación

- La salida del método de segmentación es una imagen binaria, donde los píxeles con valor '1' involucran a todos los objetos.
- Con este formato binario, el sistema de reconocimiento no sabría dónde se localiza cada objeto de manera individual.
- El etiquetado transforma la imagen binaria en una **imagen etiquetada**, donde los píxeles que pertenecen a un sólo objeto posee una etiqueta numérica única $L = 1, 2, \dots, N$, donde N es el número total de objetos en la imagen.

B = otsu(I)



L = bwlabel(B)



Extracción de rasgos

- La forma característica de un objeto se puede cuantificar mediante sus **momentos**, los cuales describen la distribución de los píxeles sobre el plano.
- Los momentos deben ser **invariantes** a las transformaciones geométricas que puede sufrir el objetos y **discriminantes** para objetos con formas distintas.

Teoría

- ★ Los **momentos de Hu** son siete descriptores invariantes que cuantifican la forma de un objeto, donde los dos primeros momentos son:

$$\text{Rasgo 1: } \phi_1 = \eta_{20} + \eta_{02}$$

$$\text{Rasgo 2: } \phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

donde los **momentos centrales normalizados** de segundo orden se calculan como:

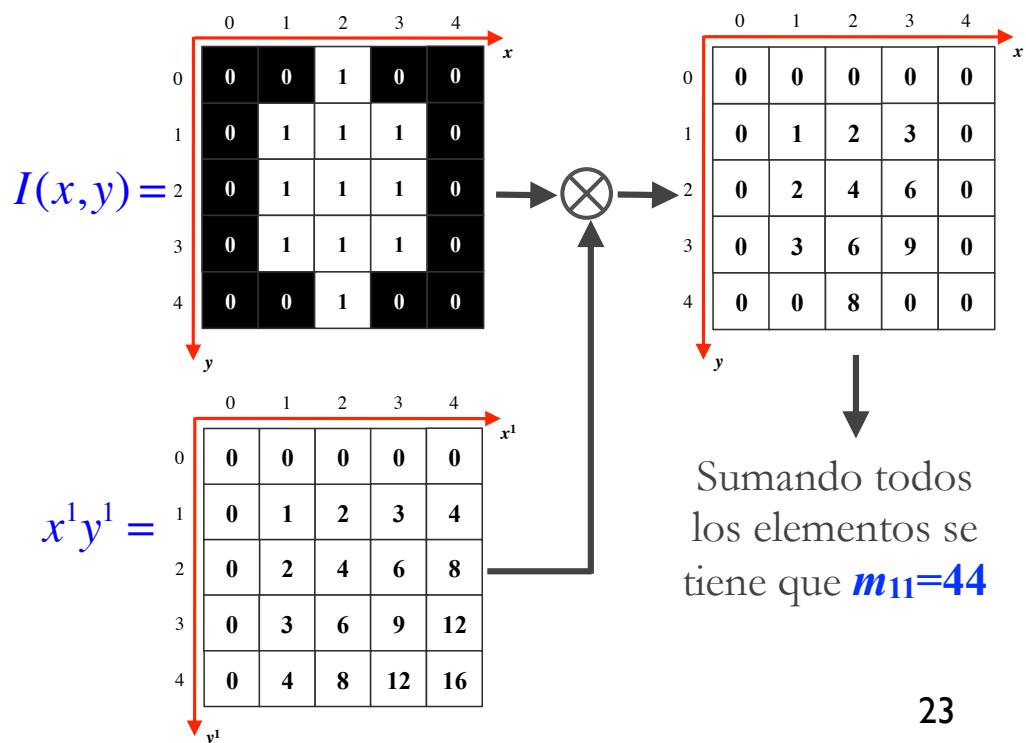
$$\eta_{20} = \frac{m_{20} - \frac{m_{10}^2}{m_{00}}}{m_{00}^2}, \quad \eta_{02} = \frac{m_{02} - \frac{m_{01}^2}{m_{00}}}{m_{00}^2}, \quad \eta_{11} = \frac{m_{11} - \frac{m_{10}m_{01}}{m_{00}}}{m_{00}^2}$$

y los **momentos geométricos** de orden $p+q$ se calculan como:

$$m_{pq} = \sum_x \sum_y x^p y^q I(x,y)$$

Explicación

- ★ La base de los momentos de Hu son los momentos geométricos, por ejemplo calcular m_{11} :



Sumando todos los elementos se tiene que $\mathbf{m_{11}=44}$

Extracción de rasgos

- La forma característica de un objeto se puede cuantificar mediante sus **momentos**, los cuales describen la distribución de los píxeles sobre el plano.
- Los momentos deben ser **invariantes** a las transformaciones geométricas que puede sufrir el objetos y **discriminantes** para objetos con formas distintas.

Teoría	Explicación
<p>★ Los momentos de Hu son siete descriptores invariantes que cuantifican la forma de un objeto, donde los dos primeros momentos son:</p> <p>Rasgo 1: $\phi_1 = \eta_{20} + \eta_{02}$</p> <p>Rasgo 2: $\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$</p> <p>donde los momentos centrales normalizados de segundo orden se calculan como:</p> $\eta_{20} = \frac{m_{20} - \frac{m_{10}^2}{m_{00}}}{m_{00}^2}, \quad \eta_{02} = \frac{m_{02} - \frac{m_{01}^2}{m_{00}}}{m_{00}^2}, \quad \eta_{11} = \frac{m_{11} - \frac{m_{10}m_{01}}{m_{00}}}{m_{00}^2}$ <p>y los momentos geométricos de orden $p+q$ se calculan como:</p> $m_{pq} = \sum_x \sum_y x^p y^q I(x,y)$	<p>★ Realizando este proceso para todos los órdenes se tienen los siguientes momentos geométricos:</p> <p>Orden cero: $m_{00}=11$</p> <p>Primer orden: $m_{10}=22, m_{01}=22$</p> <p>Segundo orden: $m_{20}=50, m_{02}=58, m_{11}=44$</p> <p>★ Sustituyendo los valores en los momentos centrales normalizados:</p> $\eta_{20} = 0.0496, \quad \eta_{02} = 0.1157, \quad \text{y} \quad \eta_{11} = 0$ <p>★ Finalmente, los dos primeros momentos de Hu son:</p> $\phi_1 = 0.1653 \quad \text{y} \quad \phi_2 = 0.0044$

Extracción de rasgos

Implementación numérica

```

function [phi1, phi2] = hu_moments(I)
% Convierte imagen a double
I = double(I);
% Tamaño MxN de la imagen
[N,M] = size(I);
% Crea plano coordenado (x,y)
[x,y] = meshgrid(0:M-1,0:N-1);
% Calcula momentos geométricos
m00 = momgeom(I,x,y,0,0);
m10 = momgeom(I,x,y,1,0);
m01 = momgeom(I,x,y,0,1);
m20 = momgeom(I,x,y,2,0);
m02 = momgeom(I,x,y,0,2);
m11 = momgeom(I,x,y,1,1);
% Calcula momentos centrales normalizados
n20 = (m20-((m10^2)/m00))/m00^2;
n02 = (m02-((m01^2)/m00))/m00^2;
n11 = (m11-((m10*m01)/m00))/m00^2;
% Calcula momentos de Hu 1 y 2
phi1 = n20 + n02;
phi2 = (n20-n02)^2 + 4*n11^2;

%***** Momentos geométricos de orden p+q
function mpq = momgeom(I,x,y,p,q)
mpq = sum(sum(I.*(x.^p).* (y.^q)));

```

Ejemplo

TORNILLO



$$\phi_1 = \begin{bmatrix} 0.54 & 0.50 & 0.45 & 0.47 & 0.52 \end{bmatrix}$$

$$\phi_2 = \begin{bmatrix} 0.25 & 0.21 & 0.17 & 0.19 & 0.24 \end{bmatrix}$$

RONDANA



$$\phi_1 = \begin{bmatrix} 0.19 & 0.19 & 0.19 & 0.19 & 0.19 \end{bmatrix}$$

$$\phi_2 = \begin{bmatrix} 0.10 & 0.40 & 0.25 & 0.33 & 0.25 \end{bmatrix} \times 10^{-4}$$

ARMELLA



$$\phi_1 = \begin{bmatrix} 0.79 & 0.78 & 0.77 & 0.83 & 0.71 \end{bmatrix}$$

$$\phi_2 = \begin{bmatrix} 0.37 & 0.36 & 0.35 & 0.41 & 0.27 \end{bmatrix}$$

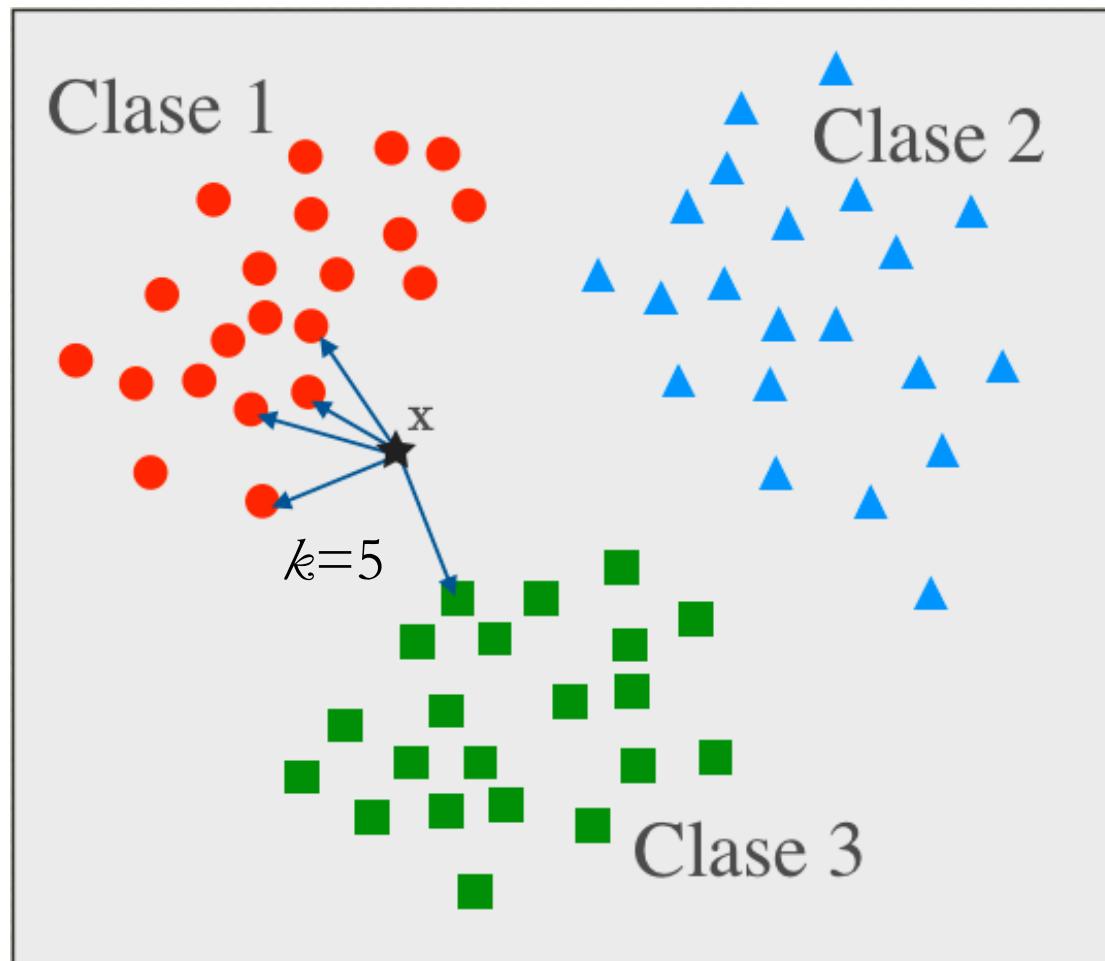
Clasificación de objetos

- Un objeto es caracterizado mediante un vector con d rasgos, $\mathbf{x}=[x_1, x_2, \dots, x_d]$, denominado patrón y está asociado a una etiqueta de clase, $\mathbf{y} \in \{1, 2, \dots, C\}$.
- Para el problema de reconocimiento de piezas de ferretería un objeto se representa por un vector de dos rasgos, $\mathbf{x}=[\phi_1, \phi_2]$, y está asociado a una de las cinco clases conocidas: tornillo ($\mathbf{y}=1$), rondana ($\mathbf{y}=2$), alcayata ($\mathbf{y}=3$), armella ($\mathbf{y}=4$) y grapa cola de pato ($\mathbf{y}=5$).
- El reconocimiento de objetos se puede realizar mediante la comparación entre el patrón que se desea reconocer y **patrones de entrenamiento** de cada tipo de objeto conocido y asignarlo al grupo que más se parezca.
- ‘Más parecido’ se puede relacionar con la idea de ‘cercanía’ utilizando una métrica de **distancia**, por ejemplo, la distancia Euclídea:

$$D(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\phi_1^i - \phi_1^j)^2 + (\phi_2^i - \phi_2^j)^2}$$

Clasificación de objetos

- El algoritmo de los k -vecinos más cercanos (KNN, k -nearest neighbor) mide la distancia Euclíadiana de un patrón y sus k vecinos de entrenamiento más cercanos y lo clasifica en la clase que tenga mayor presencia.



Clasificación de objetos

- Pasos del algoritmo KNN:

1. Medir la distancia Euclíadiana entre el patrón que se desea reconocer y todas las muestras del conjunto de entrenamiento.
2. Identificar los k patrones de entrenamiento más cercanos al patrón que se desea reconocer (i.e., aquellos que obtuvieron la menor distancia Euclíadiana) y obtener la etiqueta de clase de cada vecino.
3. El patrón que se desea reconocer se asigna a la clase que obtuvo el mayor número de ocurrencias o votos.

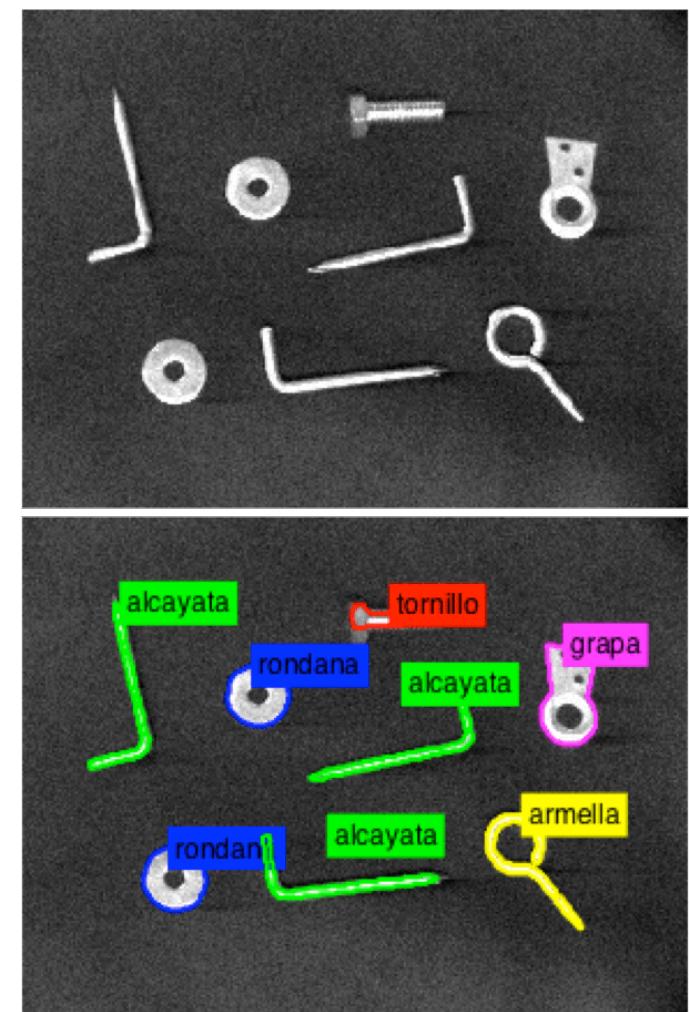
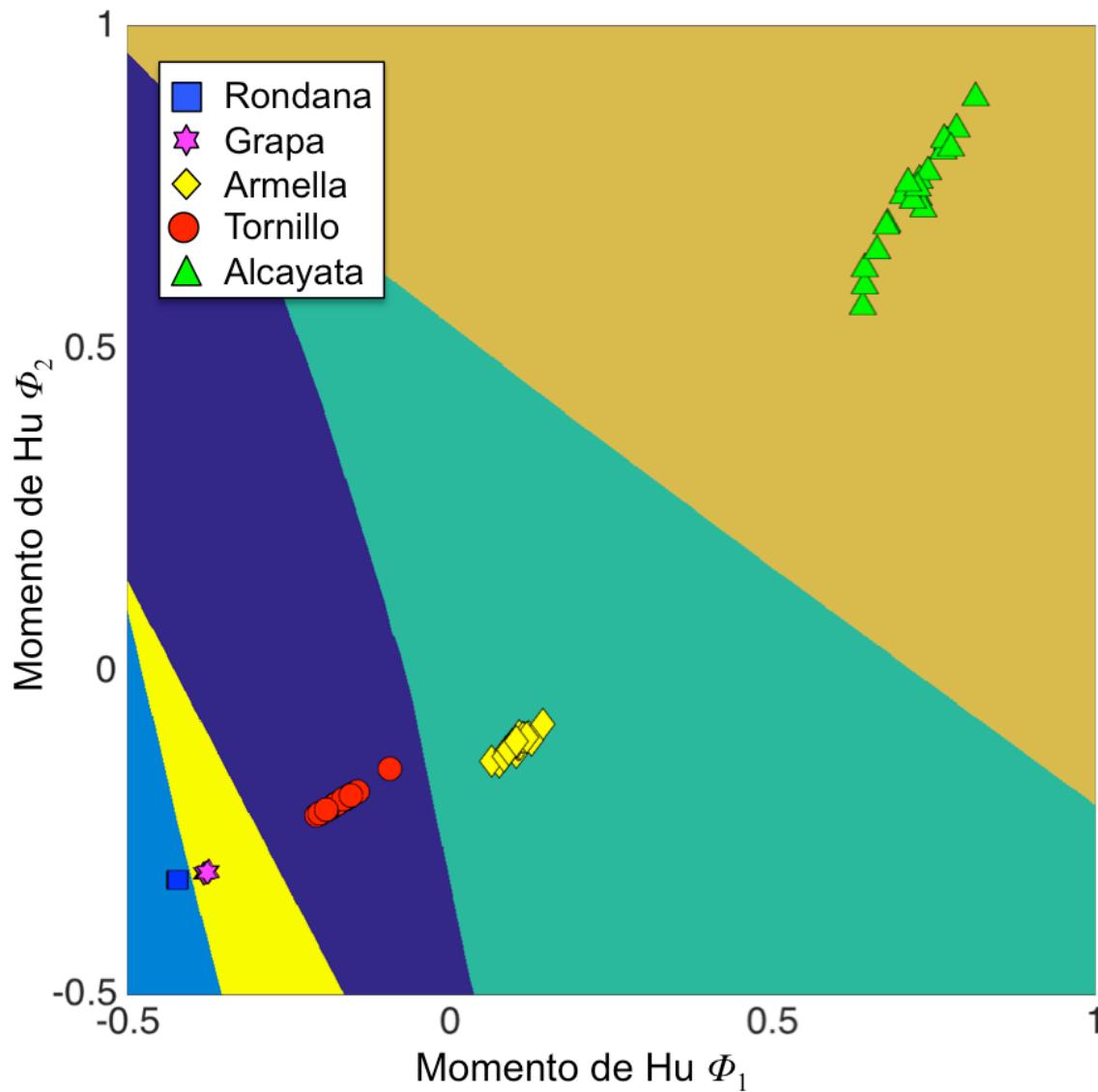
Clasificación de objetos

- **Implementación numérica**

```
function Yrec = knn1(Xent,Yent,Xrec)
% Número de muestras u objetos que serán reconocidos
NumObj = size(Xrec,1);
% Busca los índices de los 3 vecinos de entrenamiento más cercanos
% midiendo la distancia Euclidiana
idx = knnsearch(Xent,Xrec,'dist','euclidean','k',3);
% Identifica la clase de los 3 vecinos más cercanos
C = Yent(idx);
% Inicializa en ceros la matriz de conteo de votos
V = zeros(NumObj,max(Yent));
for i = 1:max(Yent) % Para cada clase conocida
    V(:,i) = sum(C==i,2); % Número de veces que aparece la clase i
end
% Gana la clase con el mayor número de votos
[~,Yrec] = max(V,[],2);
```

Clasificación de objetos

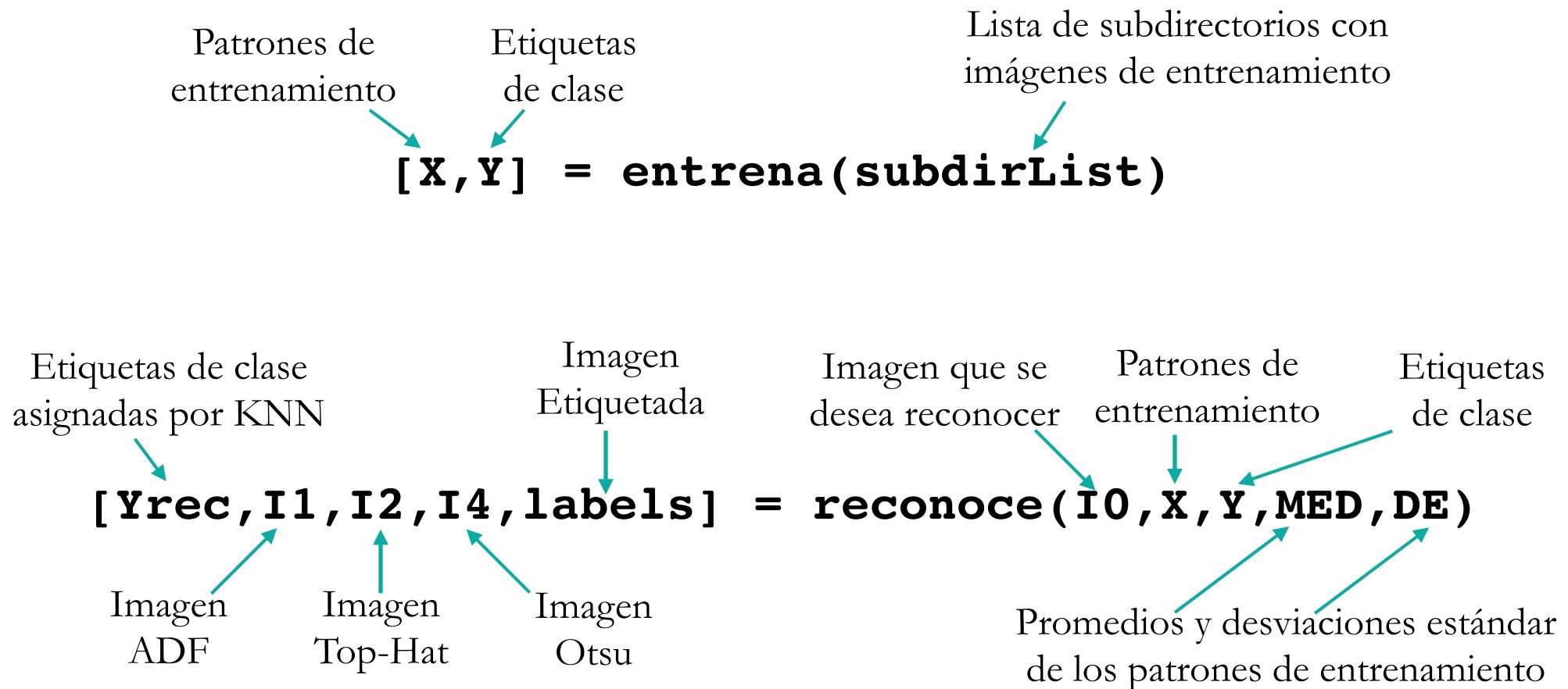
- Clasificación KNN del problema de piezas de ferretería.



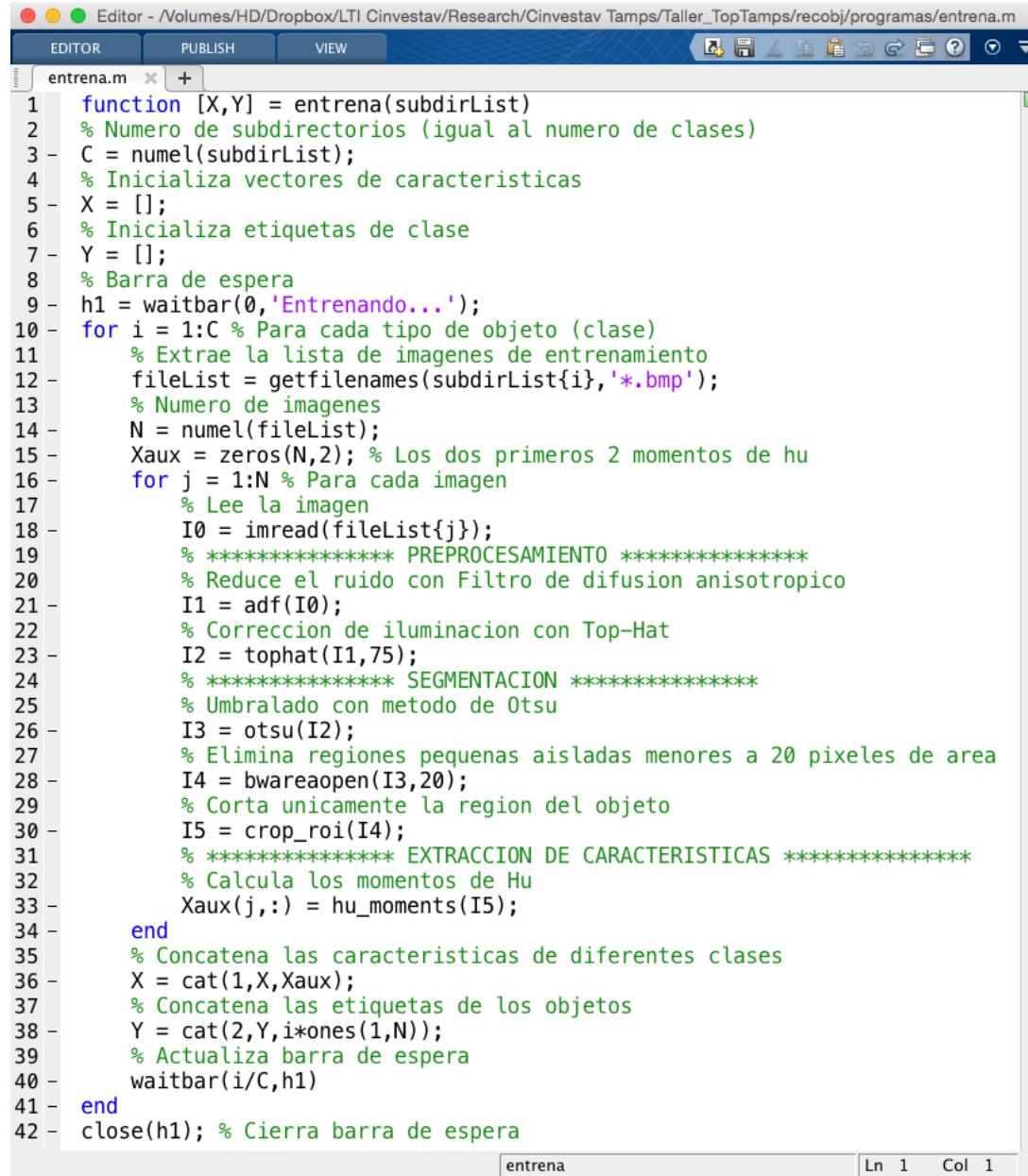
Ejercicio

Ejercicio

- Implementar en MATLAB dos funciones que realicen el entrenamiento y reconocimiento de piezas de ferretería, los cuales deben tener los siguientes argumentos de entrada y salida:



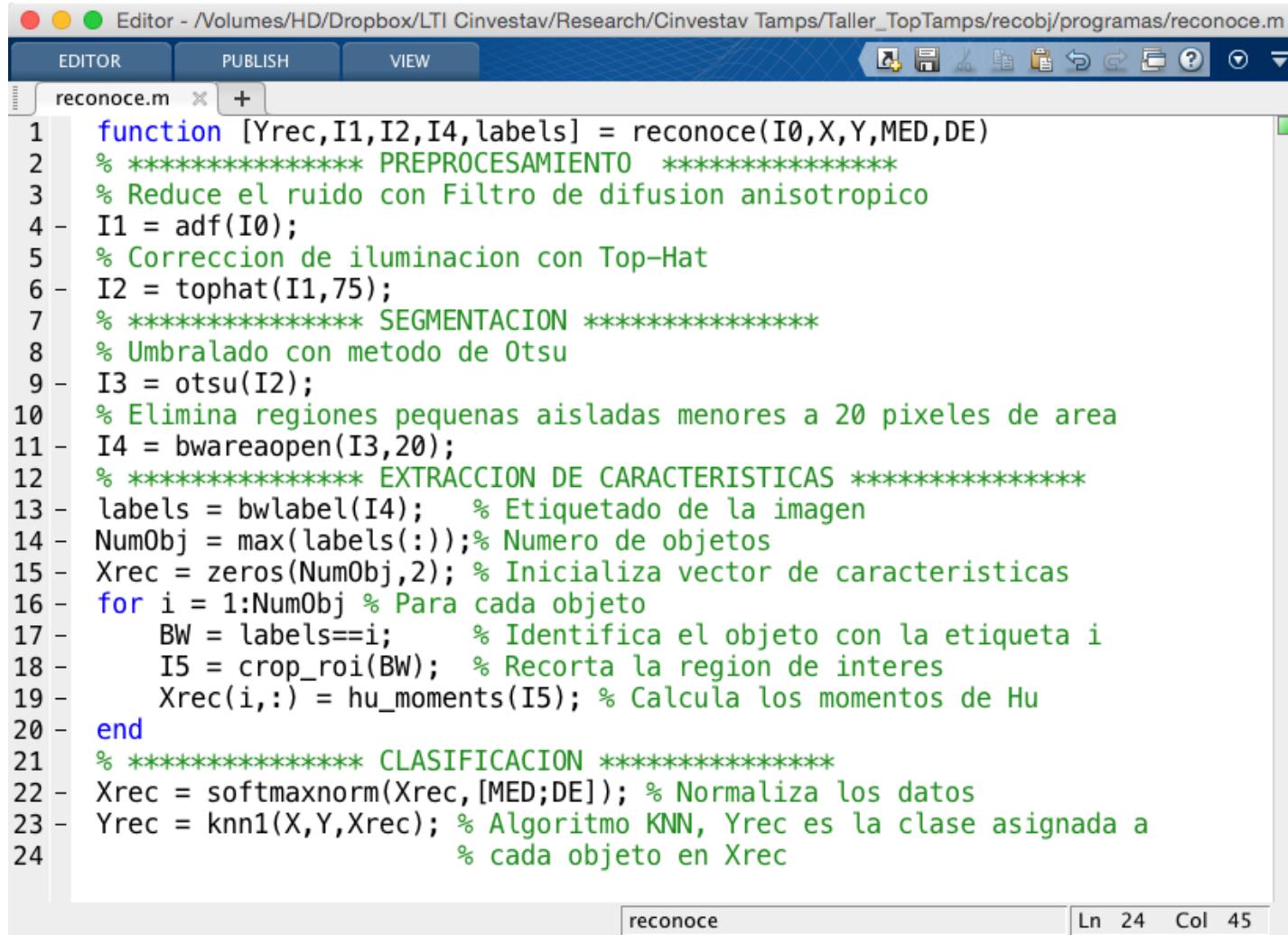
Ejercicio



```
function [X,Y] = entrena(subdirList)
% Numero de subdirectorios (igual al numero de clases)
C = numel(subdirList);
% Inicializa vectores de caracteristicas
X = [];
% Inicializa etiquetas de clase
Y = [];
% Barra de espera
h1 = waitbar(0,'Entrenando...');

for i = 1:C % Para cada tipo de objeto (clase)
    % Extrae la lista de imagenes de entrenamiento
    fileList = getfilenames(subdirList{i}, '*.bmp');
    % Numero de imagenes
    N = numel(fileList);
    Xaux = zeros(N,2); % Los dos primeros 2 momentos de hu
    for j = 1:N % Para cada imagen
        % Lee la imagen
        I0 = imread(fileList{j});
        % ***** PREPROCESAMIENTO *****
        % Reduce el ruido con Filtro de difusion anisotropico
        I1 = adf(I0);
        % Correccion de iluminacion con Top-Hat
        I2 = tophat(I1,75);
        % ***** SEGMENTACION *****
        % Umbralado con metodo de Otsu
        I3 = otsu(I2);
        % Elimina regiones pequenas aisladas menores a 20 pixeles de area
        I4 = bwareaopen(I3,20);
        % Corta unicamente la region del objeto
        I5 = crop_roi(I4);
        % ***** EXTRACCION DE CARACTERISTICAS *****
        % Calcula los momentos de Hu
        Xaux(j,:) = hu_moments(I5);
    end
    % Concatena las caracteristicas de diferentes clases
    X = cat(1,X,Xaux);
    % Concatena las etiquetas de los objetos
    Y = cat(2,Y,i*ones(1,N));
    % Actualiza barra de espera
    waitbar(i/C,h1)
end
close(h1); % Cierra barra de espera
```

Ejercicio



The image shows a MATLAB Editor window with the file 'reconoce.m' open. The code is a script for object recognition, divided into several sections: PREPROCESAMIENTO, SEGMENTACION, EXTRACCION DE CARACTERISTICAS, and CLASIFICACION. It uses various image processing functions like adf, tophat, otsu, bwareaopen, bwlabel, crop_roi, and hu_moments, and classification functions like knn1 and softmaxnorm.

```
function [Yrec,I1,I2,I4,labels] = reconoce(I0,X,Y,MED,DE)
% ***** PREPROCESAMIENTO *****
% Reduce el ruido con Filtro de difusion anisotropico
I1 = adf(I0);
% Correccion de iluminacion con Top-Hat
I2 = tophat(I1,75);
% ***** SEGMENTACION *****
% Umbralado con metodo de Otsu
I3 = otsu(I2);
% Elimina regiones pequenas aisladas menores a 20 pixeles de area
I4 = bwareaopen(I3,20);
% ***** EXTRACCION DE CARACTERISTICAS *****
labels = bwlabel(I4); % Etiquetado de la imagen
NumObj = max(labels(:)); % Numero de objetos
Xrec = zeros(NumObj,2); % Inicializa vector de caracteristicas
for i = 1:NumObj % Para cada objeto
    BW = labels==i; % Identifica el objeto con la etiqueta i
    I5 = crop_roi(BW); % Recorta la region de interes
    Xrec(i,:) = hu_moments(I5); % Calcula los momentos de Hu
end
% ***** CLASIFICACION *****
Xrec = softmaxnorm(Xrec,[MED;DE]); % Normaliza los datos
Yrec = knn1(X,Y,Xrec); % Algoritmo KNN, Yrec es la clase asignada a
% cada objeto en Xrec
```

Ejercicio

- Una vez que hayan escrito ambas funciones, se deben copiar en la carpeta **programas**, dentro de la carpeta principal **RecObj**, y ejecutar la interfaz gráfica del sistema de reconocimiento.



Referencias

- **Conjunto de imágenes:** R. R. Morales, J. H. Sossa Azuela, Procesamiento y Análisis Digital de Imágenes, Alfaomega Ra-Ma, 2012.
- **Filtro ADF:** P. Perona, J. Malik, Scale-space and edge detection using anisotropic diffusion, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7): 629–639, 1990.
- **Filtro Top-Hat:** P. Soille, Morphological Image Analysis, 2a Edición, Capítulo 4, Springer-Verlag, 2004.
- **Método de Otsu:** N. Otsu, A threshold selection method from gray-level histogram, *IEEE Transactions on Systems, Man, and Cybernetic*, 9(1): 62-66, 1979.
- **Momentos de Hu:** M. K. Hu, Visual pattern recognition by moment invariants, *IRE Transactions on Information Theory*, 8(2): 179–187, 1962.
- **Clasificador KNN:** S. Theodoridis, K. Koutroumbas, Pattern Recognition, 4a Edición, Capítulo 2, Academic Press, 2009.