Universidad de Alcalá

Sistemas de Visión Artificial

Grado en Ingeniería Computadores

Tema 2:Técnicas básicas de procesamiento de imágenes.

Autores: Sira Palazuelos, Luis M. Bergasa , Manuel Mazo, M. Ángel García, Marisol Escudero, J. Manuel Miguel
Departamento de Electrónica. Universidad de Alcalá.

Universidad de Alcalá

Índice

1. Mejora de imágenes
2. Transformaciones geométricas en imágenes.
3. Ejemplos en Matlab
4. Histograma de una imagen y procesamiento basado en el histograma
5. Transformadas espaciales de la imagen
6. Ruido
7. Transformaciones en el dominio de la frecuencia

Universidad de Alcalá

Mejora de imágenes

Departamento de Electrónica

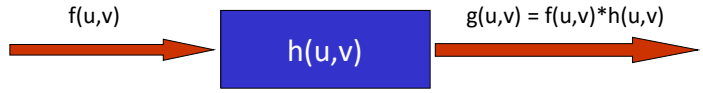
Universidad de Alcalá

Mejora de imágenes

- Objetivo: **mejorar la "calidad"** de las imágenes.
  - Para la perspectiva humana.
  - Para posteriores operaciones de procesamiento.
- Dos alternativas:
  - Técnicas en el **dominio del espacio** → Operando directamente sobre los píxeles de la imagen: sobre el histograma, con máscaras...
  - Técnicas en el dominio **de la frecuencia** → Operando sobre la transformada de Fourier (por ejemplo) de la imagen.
- No existe una teoría general para definir la "calidad visual".
  - Se asume: si parece mejor, es mejor.

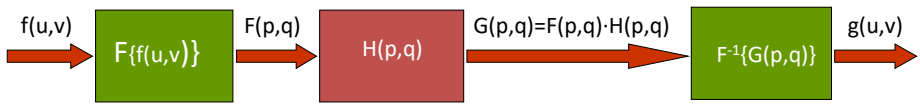
Alternativas de procesamiento de imágenes

❑ **Dominio del espacio**



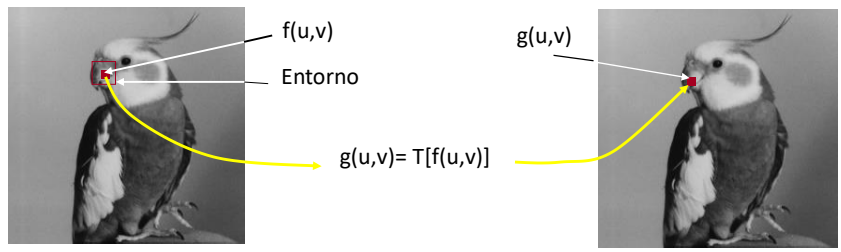
- u, v: coordenadas **espaciales** (plano imagen).
- h(u,v): **respuesta al impulso** (respuesta a un punto de luz intenso caracterizado por la función delta de Dirac).
- ***: operador **convolución**.

❑ **Dominio de la frecuencia** (transformada de Fourier, del Coseno, Wavelet, etc.)



- p, q: **frecuencias espaciales** (frecuencia en la dirección de u y v respectivamente).
- H(p,q): **función de transferencia** del filtro.

Métodos en el dominio del espacio



☐ La **transformación en el espacio** T puede ser:

- ☐ **Puntual**: píxel a píxel: el valor final del pixel solo depende de su valor en la imagen original, no del entorno (p. ej.: aumento de brillo).
- ☐ **Área**: área local a píxel. Para calcular el valor final del pixel se tiene en cuenta el valor del entorno (utilizando máscaras).
- ☐ **Global**: imagen entera a píxel. Para calcular el valor final del pixel se tiene en cuenta toda la imagen (p.ej.: ecualización del histograma).

Universidad de Alcalá

Transformaciones geométricas en imágenes

Departamento de Electrónica

Universidad de Alcalá

Transformaciones geométricas
Generalidades

- Las **transformaciones geométricas** se utilizan para investigar ciertas **zonas** o regiones dentro de una imagen (regiones de interés).
- Para ello se realizan operaciones que modifican las **coordenadas espaciales** de la imagen: operaciones geométricas.
- Algunas de estas transformaciones son: **traslaciones, rotaciones y homotecias (zoom)**.

Imagen 1. Original

Imagen 2

Imagen 3

Imagen 4

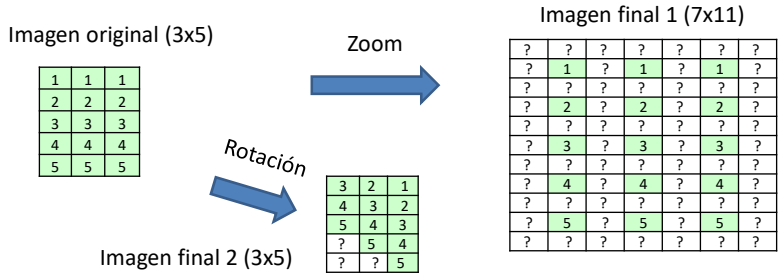
Imagen 5. Final

http://www2.diariomotor.com/imagenes/2012/10/posts/impuesto-de-matriculacion-01.jpg

Departamento de Electrónica

Transformaciones geométricas
Generalidades

- El objetivo de una operación geométrica es **transformar** los valores de una imagen para que se vean como podrían observarse desde otro punto de vista.
- Las imágenes son discretas (entre dos píxeles no existen valores de intensidad) formando una rejilla, donde las coordenadas de cada celda son números enteros.
- Al someter la imagen original a un desplazamiento, giro o zoom, en general para un píxel, no se va a obtener de la imagen original un valor entero en la imagen destino.
- Es necesario un **algoritmo de interpolación** que determine el nivel de intensidad de la imagen final a partir de uno o varios píxeles de la imagen original.



Coordenadas homogéneas

- Para poder representar las tres transformaciones (perspectiva, rotaciones y traslaciones) en forma matricial (como producto de matrices), es necesario representar los puntos en **coordenadas homogéneas.**
- Estas coordenadas permiten componer una sucesión de transformaciones en **una única matriz.**
- Estas **coordenadas** agregan una componente a las coordenadas cartesianas:

$$(x, y) \rightarrow (x', y', W).$$

- El punto representado en **coordenadas cartesianas** es el:

$$(x, y) = (x'/W, y'/W)$$

- El valor de W es generalmente 1.

Transformaciones geométricas
Traslación

- La **traslación** consiste en, dado un píxel f(u,v) si se desplaza u_d,v_d el píxel correspondiente en la imagen de salida será g($u+u_d$, $v+v_d$), siendo f = g (se mantiene la intensidad).

Ejemplo de: http://slideplayer.es/slide/4333895/

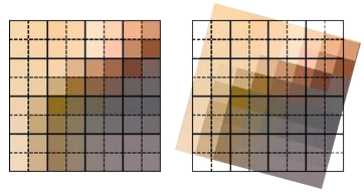
- En coordenadas homogéneas:

$$\begin{bmatrix} u_f \\ v_f \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & u_d \\ 0 & 1 & v_d \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix}$$

- Donde u_f y v_f son las **coordenadas finales**, u_i y v_i las **iniciales**, y u_d y v_d el **vector desplazamiento**.

Transformaciones geométricas
Giros

- Los **giros** se utilizan para producir efectos estéticos, y para simular la **rotación** de la cámara o del propio objeto.



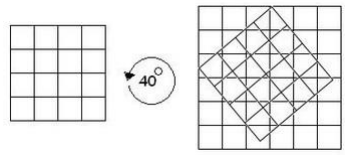
Ejemplo de: http://www.kolor.com/wiki-en/Interpolation_and_blenders

- Los parámetros necesarios para simular la rotación son el **ángulo de giro** y las coordenadas del **centro de rotación**.
- La rotación, respecto a un centro genérico (u_d, v_d) viene dada por:

$$\begin{bmatrix} u_f \\ v_f \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & u_d \\ 0 & 1 & v_d \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -sen\theta & 0 \\ sen\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -u_d \\ 0 & 1 & -v_d \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix}$$

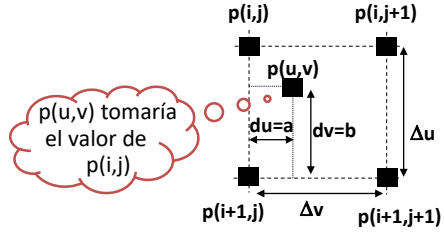
Transformaciones geométricas
Interpolación

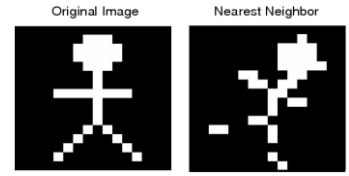
- La **interpolación** tiene por objetivo calcular el valor de **intensidad** de un píxel, en una posición cualquiera, como **función de los que le rodean: lo utilizamos en los giros, homotecias, etc.**.



Ejemplo de: http://www.codeproject.com/Articles/12230/Anti-Aliased-Image-Rotation

- Una solución: el píxel p(u,v) de la imagen final toma el **valor del píxel más cercano** (distancia euclídea) de los cuatro vecinos que le rodean.





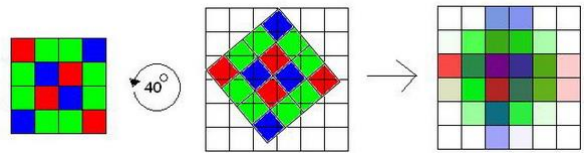
Ejemplo de: http://northstar-www.dartmouth.edu/doc/idl/html_6.2/Interpolation_Methods.html

Transformaciones geométricas
Interpolación bilineal

- La **interpolación bilineal** asigna un **valor medio ponderado** de las intensidades de los cuatro píxeles que le rodean.
- Los **factores de ponderación** vienen dados por la distancia entre el píxel y los del entorno:

$$p(u,v) = a_1 p(i,j) + a_2 p(i,j+1) + a_3 p(i+1,j) + a_4 p(i+1,j+1)$$
$$a_1 = (1-du)(1-dv); \quad a_2 = du(1-dv)$$
$$a_3 = (1-du)dv; \quad a_4 = dudv$$

- Da mejores resultados, pero tiene mayor coste computacional.



Ejemplo de: http://www.codeproject.com/Articles/12230/Anti-Aliased-Image-Rotation

Transformaciones geométricas
Interpolación bicúbica

- La **interpolación bicúbica**: El valor de un pixel interpolado es una combinación de los valores de los 16 píxeles más cercanos.
- Este método produce una superficie mucho más suave que la interpolación bilineal. Para usar la interpolación bicúbica, la primera y segunda derivada de la superficie deben ser continuas.
- Es más lenta y ocupa más memoria que la interpolación bilineal.

Transformaciones geométricas
Zoom

- **Zoom**: Se trata de seleccionar una parte de la imagen (sub-imagen), separarla del resto de la imagen original y realizar un proceso de **expansión**.

Transformaciones geométricas
Interpolación lineal

- Una forma de expansión frecuente, por ejemplo para agrandar una imagen (zoom): **interpolación lineal** (valor medio entre **dos** píxeles).
- Se pasa de imágenes de NxN a imágenes de (2N-1)x(2N-1) y se puede repetir las veces que se quiera.

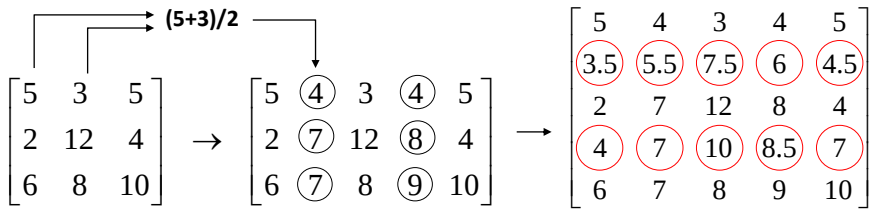


Imagen original **Imagen con filas expandidas** **Imagen con filas y columnas expandidas**

Transformaciones geométricas
Promedio del entorno de vecindad

- Otra forma de expansión: **promedio del entorno de vecindad.**

1ª. Expandir:

$$I=\begin{bmatrix}\mathbf{2} & \mathbf{6} & \mathbf{4}\\ \mathbf{8} & \mathbf{10} & \mathbf{2}\\ \mathbf{4} & \mathbf{6} & \mathbf{8}\end{bmatrix} \rightarrow I_{exp} = \begin{bmatrix}0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & \mathbf{2} & 0 & \mathbf{6} & 0 & \mathbf{4} & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & \mathbf{8} & 0 & \mathbf{10} & 0 & \mathbf{2} & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & \mathbf{4} & 0 & \mathbf{6} & 0 & \mathbf{8} & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}$$

2º. Promediar (convolución):

$$h=\begin{bmatrix}\frac{1}{4} & \frac{1}{2} & \frac{1}{4}\\ \frac{1}{2} & 1 & \frac{1}{2}\\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4}\end{bmatrix} \qquad I_{zoom} = h * I_{exp} \qquad \begin{bmatrix}0.5 & 1 & 2 & 3 & 2.5 & 2 & 1\\ 1 & 2 & 4 & 6 & 5 & 4 & 2\\ - & - & - & - & - & - & -\\ - & - & - & - & - & - & -\\ - & - & - & - & - & - & -\\ - & - & - & - & - & - & -\\ - & - & - & - & - & - & -\end{bmatrix}$$

Representación de imágenes en Matlab

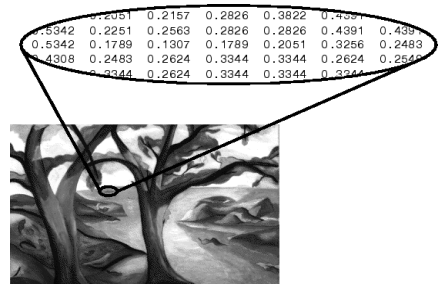
Representación de imágenes en Matlab

- ***Formato de las imágenes.*** Matlab puede trabajar con imágenes en los siguientes **formatos**:
  - Graphics Interchange Format (.GIF)
  - Tagged Image File Format (.TIFF)
  - Bit Map Format (.BMP)
  - ISO Standard Format (.JPEG , .JPG , .JPE)
  - Otros formatos (.PCX , .CGI , .PNG, etc.)
- ***Representación de las imágenes (I)***
  - Matlab puede **representar** las imágenes de diferentes maneras, entre otras:
  - **Imágenes binarias**. La imagen es una matriz B = f(n,m) donde cada pixel puede valer 0 o 1: f(i,j) = 0 or 1.

Representación de imágenes en Matlab

Representación de las imágenes (II)

- **Imágenes de intensidad (escala de grises)**: cada imagen se representa con una matriz, I = f(n,m) donde cada pixel es un número real (r) con 0 =< r <= 1 (0 : black; 1 : white).



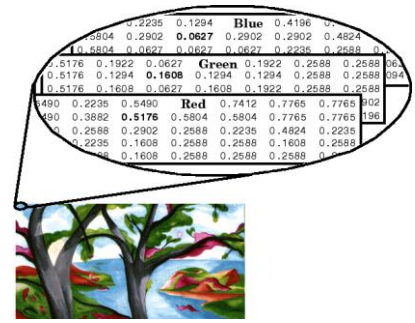
https://es.mathworks.com/

Representación de imágenes en Matlab

Representación de las imágenes (III)

- **Imágenes RGB**: Usa 3 matrices: R(n,m), G(n,m), B(n,m). En MATLAB, es posible extraer las 3 componentes. Si F es una imagen RGB, FR = F(: , : , 1); FG = F(: , : , 2); FB = F(: , : , 3).

https://es.mathworks.com/

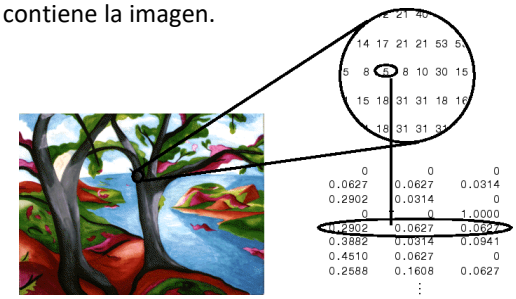


Representación de imágenes en Matlab

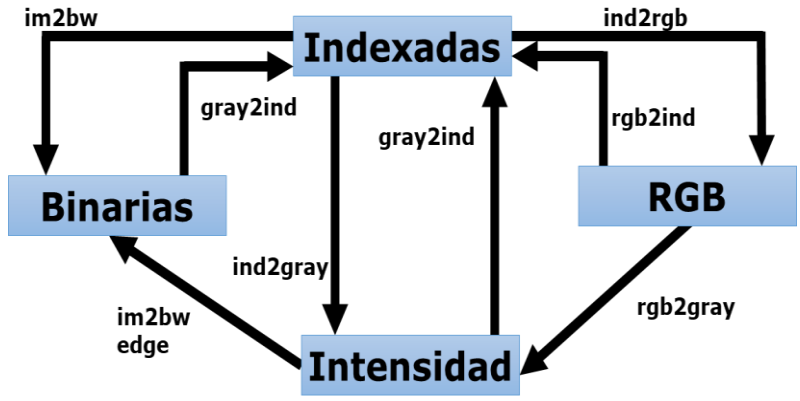
Representación de las imágenes (IV)

- **Imágenes indexadas**: utilizan una matriz aparte llamada **mapa de color**. image(X); colormap(map). Los elementos de la imagen indexada NO son colores, son índices que determinan el número de fila en el mapa de color. Cada fila contiene un número (en el caso de imágenes de intensidad) o un triplete (para imágenes RGB), que representan el color que debe tener ese pixel. El número total de filas del mapa de color, L, es el número de colores diferentes que contiene la imagen.

https://es.mathworks.com/



Conversión entre formatos en Matlab



Ejemplos en Matlab

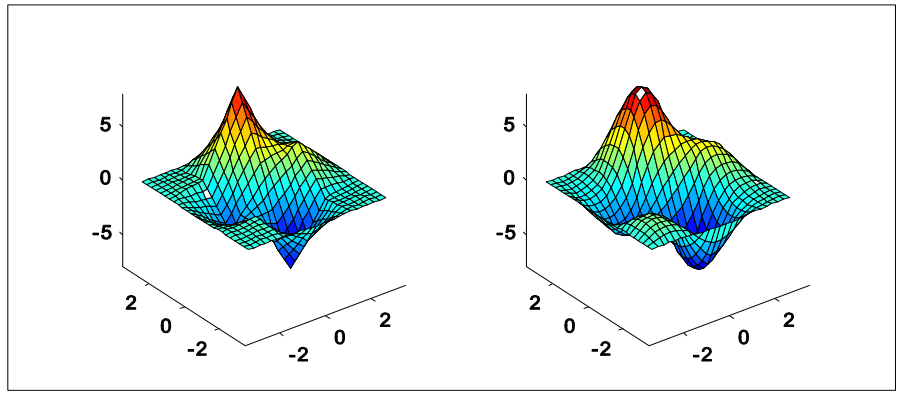
- **La Image Processing Toolbox** permite realizar rotaciones, homotecia (zoom) y selección, así como operaciones más especializadas.
- Estas funciones soportan cualquier tipo de imagen, aunque se debe procesar cada una de las componentes de una imagen RGB por separado.
- Proporciona tres métodos de interpolación:
  - Vecino más cercano
  - Interpolación bilineal
  - Interpolación bicúbica

Ejemplos en Matlab

- Los comandos que se muestran a continuación crean una figura de dos picos que ilustra estas diferencias:

```
[x,y,z]=peaks(5)                %Figura inicial de valores x,y,z
[xi,yi]=meshgrid(-3:.25:3);     %Valores a interpolar
zlin=interp2(x,y,z,xi,yi,'linear');     %Interpolacion lineal
zcub=interp2(x,y,z,xi,yi,'cubic');      %Interpolacion cubica
figure,colormap(jet);
surf(x,y,z);title('Figura inicial: con pocos puntos');
figure,
subplot(1,2,1),   surf(xi,yi,zlin)
axis([-3.5    3.5  -3.5 3.5  -8    8]);title('Figura tras interpolacion lineal');
subplot(1,2,2),   surf(xi,yi,zcub)
axis([-3.5    3.5  -3.5 3.5  -8    8]);title('Figura tras interpolacion cubica');
```

Ejemplos en Matlab



Ejemplos en Matlab

- **Rotación de imágenes**
- El comando *imrotate* rota una imagen usando un método de interpolación especificado y un ángulo de rotación. Si no se especifica un método de interpolación, la función determina el tipo de imagen y automáticamente elige el mejor método.
- Por ejemplo, para rotar la imagen *"trees"* 35°:
  - load trees
  - Y=imrotate(X,35);
  - figure, imshow(Y,map)

- Como este ejemplo no especifica el método de interpolación a *imrotate*, la función elige el mejor método para la imagen. *"trees"* es una imagen indexada, por lo que *imrotate* usa la interpolación por vecino más cercano.

Selección dentro de una imagen

- La función *imcrop* extrae una porción rectangular de una imagen. imcrop permite definir el rectángulo de corte con el ratón o a través de una lista de argumentos. Por ejemplo, para extraer un rectángulo de 92x95 de la imagen "trees" comenzando en la coordenada (71,107), se utiliza:

- load trees
- figure, subplot(1,2,1), imshow(X,map)
- subplot(1,2,2), imshow(imcrop(X,[71, 107, 92, 95]),map);

- Para definir el rectángulo de corte con el ratón, se usa imcrop sin argumentos de entrada. La actual figura debe contener una imagen en la cual imcrop pueda operar. Tras llamar a imcrop, presionar el botón izquierdo del ratón mientras se arrastra por la imagen visualizada. Soltar el botón del ratón cuando se haya definido el área deseada.

Cambio de tamaño de imágenes (I)

- La función *imresize* cambia el tamaño o la relación de muestreo de una imagen usando un método de interpolación especificado. Si no se especifica un método, la función determina el tipo de imagen y automáticamente elige el mejor.
- *imresize* puede recalibrar una imagen por un factor, o a un tamaño de fila-columna especificado. Por ejemplo, para recalibrar X a dos veces su actual tamaño usaremos,
- Y=imresize(X,2);
- Para obtener una imagen de tamaño 100x150 usaremos,
- Y=imresize(X,[100 150]); %Puede comprobar el nuevo tamano con el comando: size(Y)

- **Cambio de tamaño de imágenes (II)**
  - Si se reduce el tamaño de la imagen, *imresize* aplica un filtro pasobajo a la imagen antes de la interpolación. Esto reduce el efecto de las muestras de *Moiré*, rizados que resultan del aliasing durante el muestreo.
  - Se puede aplicar la función *truesize* para recalibrar imágenes. Esta función visualiza una imagen con un pixel de pantalla por cada pixel de la imagen.
- **Zoom de una imagen**
  - En Matlab, una vez visualizada una imagen en una figura de Matlab se puede dentro del menú *Figure* → *Tools* → *zoom in* ampliar la imagen para ver mejor los detalles.