Informática Audiovisual
Grado en Ingeniería Informática del Software
Escuela de Ingeniería Informática – Universidad de Oviedo

Tratamiento de imagen con Processing 3

Juan Ramón Pérez Pérez

jrpp@univoi.es

Imágenes y pixels

Antes de tratar, cargar una imagen en memoria

```
PImage img;
size(512,640);
img = loadImage("lighthouse.jpg");
image(img,0,0);
```

s2ohola_imagen

Transformaciones de la imagen: zoom y desplazamiento

```
scale(escala);
image(img,x,y);
```

s21imagen_transforma

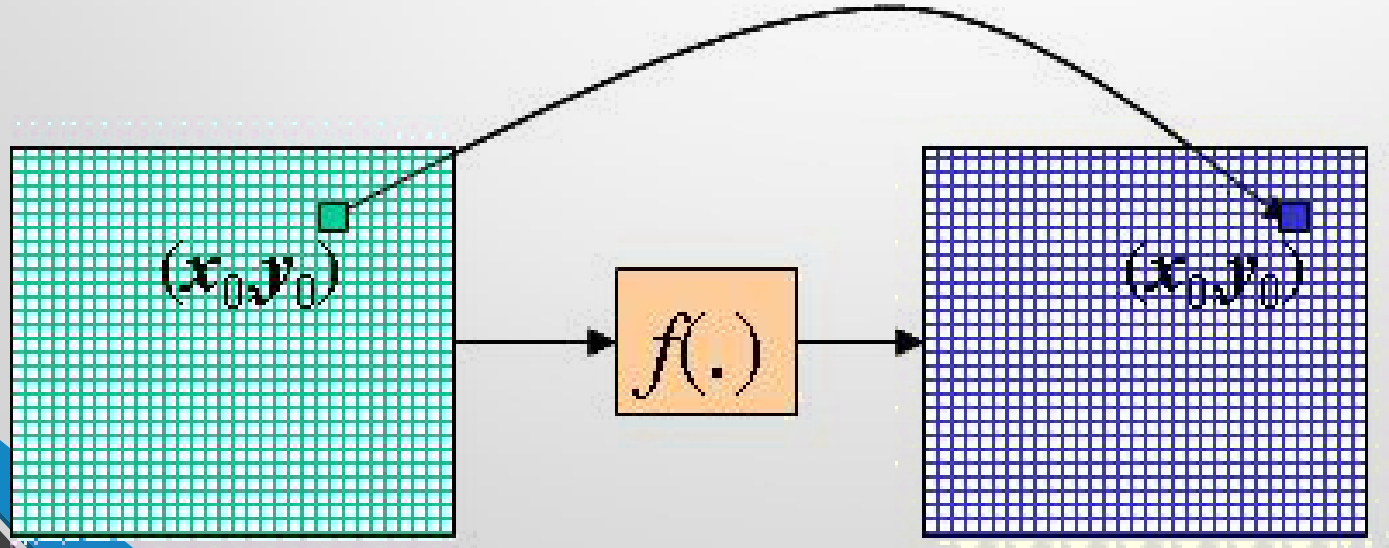
Ejercicio: Galería de imágenes

- Cargar varias imágenes
- Visualizar una imagen en pantalla
- Permitir intercambiar las imágenes
- Realizar zoom y desplazamientos

Mejora de imágenes

Operaciones puntuales

Se procesa cada pixel de la imagen por separado, independientemente del valor de los vecinos.



$$v = f(u)$$

Binarización o umbral

$$f(u) = \begin{cases} 0 \; si \; 0 \leq u < umbral \\ 1 \; si \; umbral \leq u < 2^B \end{cases}$$

Se utiliza para obtener imágenes monocromas a partir de imágenes a color o gris

Negativo de una imagen

$$f(u) = 2^B - u$$

Se obtiene el negativo digital de la imagen.

Funciones proporcionadas por processing

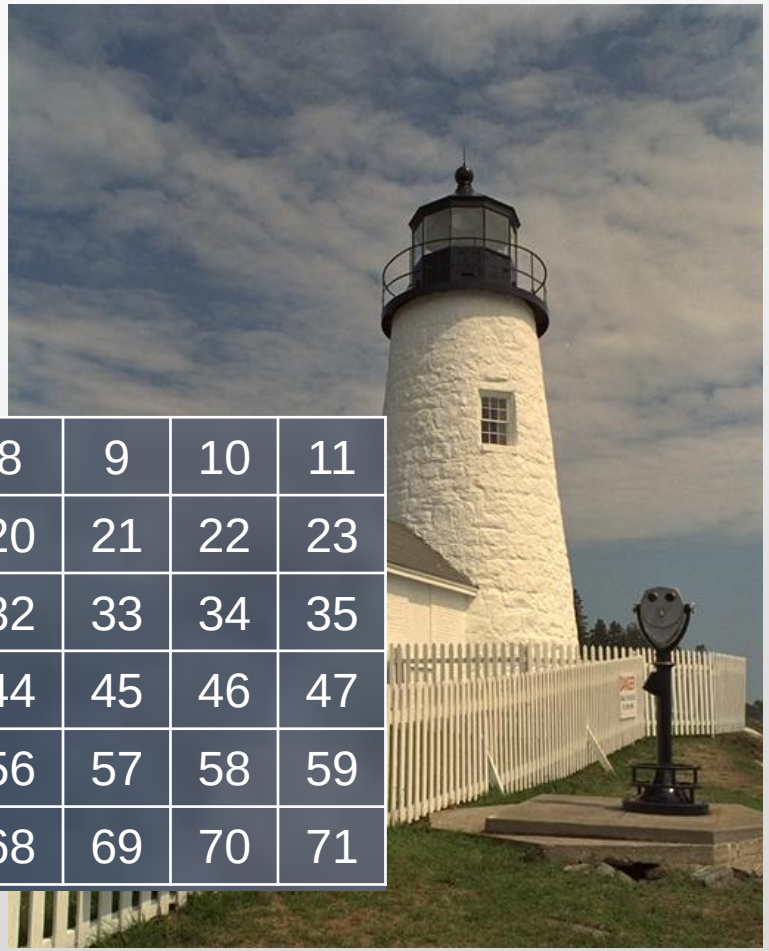
- tint(color) – modifica el color de cada pixel de la imagen
- filter(TIPO_FILTRO, parámetros)
  - THRESHOLD, filtro umbral
  - INVERT, cambia el color a su valor inverso
  - BLUR, distorsión gaussiana
  - DILATE, ERODE: incrementa o disminuye zonas iluminadas.

s24imagen_filtro_puntual

s25filtro_umbral

Modelo de pixels de una imagen

0	1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	19	19	20	21	22	23
24	25	26	27	28	29	30	31	32	33	34	35
36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59
60	61	62	63	46	65	66	67	68	69	70	71

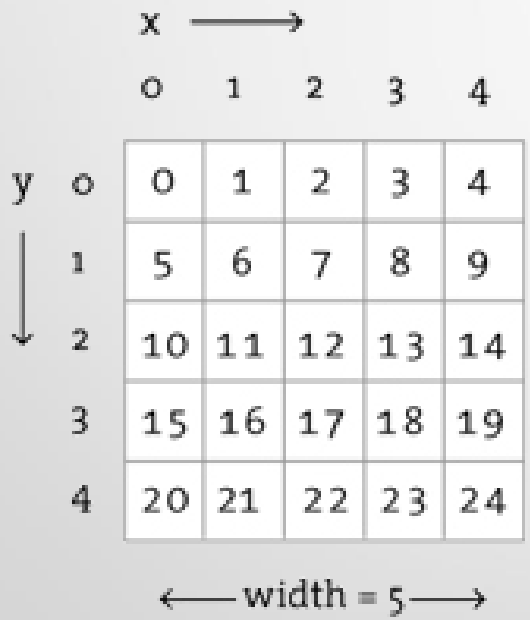


Accedemos al array de pixels

```
size(200,200);
loadPixels();        // Antes de acceder a los pixels
// Bucle para recorrer los pixels
for (int i = 0; i < pixels.length; i++ )
{
  float rand = random(255);
  color c = color(rand);
  // Acceder pixels individuales indexando el array pixels.
  pixels[i] = c;
}
// Indicamos que hemos finalizado el proceso de pixels
updatePixels();
```

s23aacceso_pixels

Cómo acceder a un pixel concreto



x= 3, y= 2 se corresponde con el pixel 13

x + (y * width)

= 3 + (2 * 5)

= 13

Accedemos y modificamos el array de pixels

```
for (int y = 0; y < img.height; y++) {
   for (int x = 0; x < img.width; x++) {
     int loc = x + y * img.width;
     // Recuperamos componentes de color para dada pixel
     float r = red(img.pixels[loc]);
     float g = green(img.pixels[loc]);
     float b = blue(img.pixels[loc]);
     // El procesamiento de la imagen iría aquí
     r= 255;
     img.pixels[loc] =  color(r,g,b);
   }
 }
```

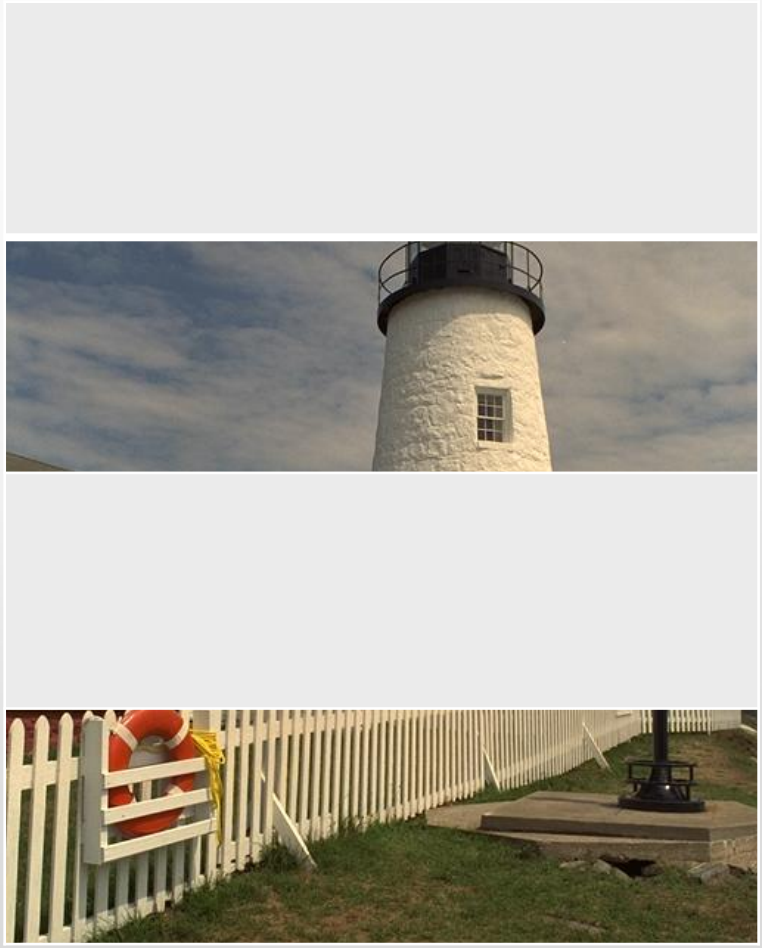
Ejercicio: Modificar pixels en la imagen

Carga una imagen cualquiera

Divídela en 4 partes verticalmente

Las partes 1 y 3 deben aparecer en gris.

Las partes 2 y 4 deben aparecer como en el original.



Filtro umbral accediendo al array de pixels

```
float umbral = 127;
// Manejamos los arrays de pixels de imagen origen y destino
origen.loadPixels();  destino.loadPixels();

for (int x = 0; x < origen.width; x++) {
  for (int y = 0; y < origen.height; y++ ) {
    int loc = x + y*origen.width;
    // Compara el brillo con el umbral
    if (brightness(origen.pixels[loc]) > umbral) {
      destino.pixels[loc]  = color(255);  // Blanco
    }  else {
      destino.pixels[loc]  = color(0);    // Negro
    }
  }
```

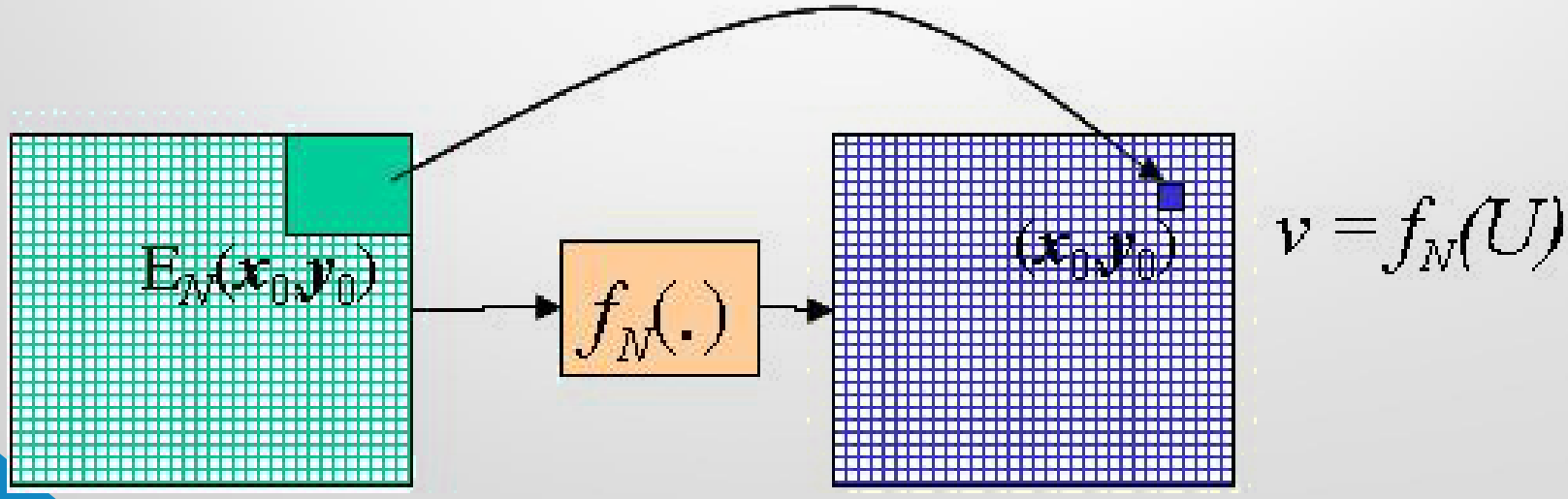
s25filtro_umbral_pixels

Realización de un análisis de histograma

s27histograma

Operaciones espaciales

Se modifica el valor de cada pixel en función del valor de un conjunto de pixels vecinos.



Máscara o plantilla

Máscara: matriz de coeficientes de la combinación lineal

El entorno del punto (x,y) que se considera en la imagen I para obtener $O(x,y)$ está determinado por el tamaño y forma de la máscara

El tipo de filtrado está determinado por el contenido de la máscara.

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Expresión matemática

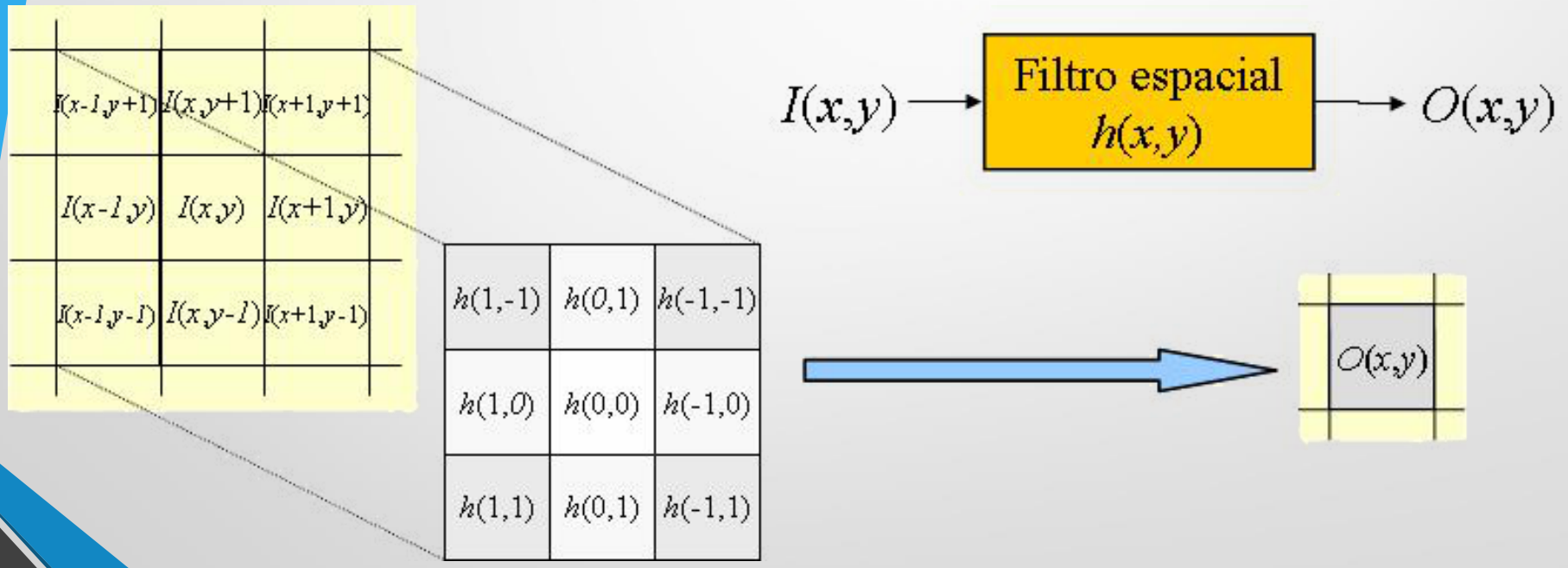
$$O(x,y) = \sum_{(k,l)\in E_N} \sum h(k,l) I(x-k, y-l)$$

x, y posición del pixel que se está filtrando

E_N conjunto de pixels utilizados en los cálculos (vecindario)

h(k,l) coeficientes de ponderación del filtro

Representación gráfica



Trabajando con pixels vecinos en Processing

```
// Como miramos los vecinos izquierdos saltamos la 1a columna
for (int x = 1; x < width; x++) {
  for (int y = 0; y < height; y++ ) {
    int loc = x + y*origen.width; // Pixel objetivo
    color pix = origen.pixels[loc];
    int leftLoc = (x-1) + y*origen.width;  // Pixel izquierda
    color leftPix = origen.pixels[leftLoc];

    // La diferencia entre los dos será el Nuevo color
    float diff = abs(brightness(pix) - brightness(leftPix));
    destino.pixels[loc] = color(diff);
  }
}
```

s28dif_vecinos

Filtrado paso bajo

Permite suavizar, eliminar ruido

Produce efecto desenfoque

Todos los coeficientes son positivos y suman 1

Suavizado direccional → todos los coeficientes ≠0 están en una dirección

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

0	1/8	0
1/8	½	1/8
0	1/8	0

Promediado de 5 puntos

Filtrado paso alto

Resalta transiciones

Aparecen valores negativos

Los coeficientes suman o

Pueden ser direccionales

-1/8	-1/8	-1/8
-1/8	1	-1/8
-1/8	-1/8	-1/8

-1	0	1
-2	0	2
-1	0	1

Sobel horizontal

-1	-2	1
0	0	0
1	2	1

Sobel vertical

Ampliación de imágenes

Repetición: repetir cada pixel *k* veces y cada línea *k* veces

Interpolación lineal:

- Añadimos 1 pixel y 1 línea a 0
- Aplicamos la plantilla de interpolación
- Repetimos el proceso para duplicar *k*

¼	½	¼
½	1	½
¼	½	¼

Ejercicio: Filtrado de imágenes

- Probar los distintos tipos de filtros: paso bajo y paso alto sobre una imagen.
- Aplicar la matriz de interpolación lineal para ampliar una imagen.