Nota: Algunas de las imágenes que aparecen en esta presentación provienen del libro:
Visión por Computador: fundamentos y métodos.
Arturo de la Escalera Hueso. Prentice Hall.

Intelligent Systems Lab

Sistemas de Percepción

Visión por Computador

Arturo de la Escalera
José María Armingol
Fernando García
David Martín
Abdulla Al-Kaff

uc3m | Universidad Carlos III de Madrid

Intelligent Systems Lab

Transformaciones morfológicas

Transformaciones morfológicas

Intelligent Systems Lab

- **Dilatación**

dilate

Dilates an image by using a specific structuring element.

C++: void **dilate**(InputArray **src**, OutputArray **dst**, InputArray **kernel**, Point **anchor**=Point(-1,-1), int **iterations**=1, int **borderType**=BORDER_CONSTANT, const Scalar& **borderValue**=morphologyDefaultBorderValue())

Parameters:

- **src** – input image; the number of channels can be arbitrary, but the depth should be one of `CV_8U`, `CV_16U`, `CV_16S`, `CV_32F` or `CV_64F`.
- **dst** – output image of the same size and type as `src`.
- **kernel** – structuring element used for dilation; if `elemenat=Mat()`, a `3 x 3` rectangular structuring element is used. Kernel can be created using `getStructuringElement()`
- **anchor** – position of the anchor within the element; default value `(-1, -1)` means that the anchor is at the element center.
- **iterations** – number of times dilation is applied.
- **borderType** – pixel extrapolation method (see `borderInterpolate` for details).
- **borderValue** – border value in case of a constant border

Transformaciones morfológicas

Intelligent Systems Lab

- **Erosión**

erode

Erodes an image by using a specific structuring element.

C++: void `erode`(InputArray **src**, OutputArray **dst**, InputArray **kernel**, Point **anchor**=Point(-1,-1), int **iterations**=1, int **borderType**=BORDER_CONSTANT, const Scalar& **borderValue**=morphologyDefaultBorderValue())

Parameters:

- **src** – input image; the number of channels can be arbitrary, but the depth should be one of `CV_8U`, `CV_16U`, `CV_16S`, `CV_32F` or ``CV_64F`.
- **dst** – output image of the same size and type as `src`.
- **kernel** – structuring element used for erosion; if `element=Mat()`, a `3 x 3` rectangular structuring element is used. Kernel can be created using `getStructuringElement()`.
- **anchor** – position of the anchor within the element; default value `(-1, -1)` means that the anchor is at the element center.
- **iterations** – number of times erosion is applied.
- **borderType** – pixel extrapolation method (see `borderInterpolate` for details).
- **borderValue** – border value in case of a constant border

Transformaciones morfológicas

Intelligent Systems Lab

- Creación de un nuevo elemento estructurante definido por el usuario (círculo, rectángulo, elipse, …)

getStructuringElement

Returns a structuring element of the specified size and shape for morphological operations.

C++: Mat `getStructuringElement`(int **shape**, Size **ksize**, Point **anchor**=Point(-1,-1))

Parameters:

- **shape** –

  Element shape that could be one of the following:

  - **MORPH_RECT** - a rectangular structuring element:

    $$E_{ij} = 1$$

  - **MORPH_ELLIPSE** - an elliptic structuring element, that is, a filled ellipse inscribed into the rectangle `Rect(0, 0, esize.width, 0.esize.height)`
  - **MORPH_CROSS** - a cross-shaped structuring element:

    $$E_{ij} = \begin{cases} 1 & \text{if i=anchor.y or j=anchor.x} \\ 0 & \text{otherwise} \end{cases}$$

  - **CV_SHAPE_CUSTOM** - custom structuring element (OpenCV 1.x API)

Note: When using OpenCV 1.x C API, the created structuring element `IplConvKernel* element` must be released in the end using `cvReleaseStructuringElement(&element)`.

Transformaciones morfológicas

Intelligent Systems Lab

- Creación de un nuevo elemento estructurante definido por el usuario (círculo, rectángulo, elipse, …)

getStructuringElement

Returns a structuring element of the specified size and shape for morphological operations.

C++: Mat **getStructuringElement**(int **shape**, Size **ksize**, Point **anchor**=Point(-1,-1))

Parameters:

- **ksize** – Size of the structuring element.
- **cols** – Width of the structuring element
- **rows** – Height of the structuring element
- **anchor** – Anchor position within the element. The default value $(-1, -1)$ means that the anchor is at the center. Note that only the shape of a cross-shaped element depends on the anchor position. In other cases the anchor just regulates how much the result of the morphological operation is shifted.
- **anchor_x** – x-coordinate of the anchor
- **anchor_y** – y-coordinate of the anchor
- **values** – integer array of `cols``*``rows` elements that specifies the custom shape of the structuring element, when `shape=CV_SHAPE_CUSTOM`.

Transformaciones morfológicas

Intelligent Systems Lab

- Transformaciones morfológicas avanzadas: apertura, cierre, gradiente, Top Hat, Black Hat

morphologyEx

Performs advanced morphological transformations.

C++: void `morphologyEx`(InputArray **src**, OutputArray **dst**, int **op**, InputArray **kernel**, Point **anchor**=Point(-1,-1), int **iterations**=1, int **borderType**=BORDER_CONSTANT, const Scalar& **borderValue**=morphologyDefaultBorderValue())

Parameters:

- **src** – Source image. The number of channels can be arbitrary. The depth should be one of `CV_8U`, `CV_16U`, `CV_16S`, `CV_32F` or ``CV_64F`.
- **dst** – Destination image of the same size and type as `src` .
- **kernel** – Structuring element. It can be created using `getStructuringElement()`.
- **anchor** – Anchor position with the kernel. Negative values mean that the anchor is at the kernel center.
- **op** –

  Type of a morphological operation that can be one of the following:

  - **MORPH_OPEN** - an opening operation
  - **MORPH_CLOSE** - a closing operation
  - **MORPH_GRADIENT** - a morphological gradient
  - **MORPH_TOPHAT** - "top hat"
  - **MORPH_BLACKHAT** - "black hat"
- **iterations** – Number of times erosion and dilation are applied.
- **borderType** – Pixel extrapolation method. See `borderInterpolate` for details.
- **borderValue** – Border value in case of a constant border. The default value has a special meaning.

Transformaciones morfológicas

Intelligent Systems Lab

```
//Dilatacion
dilate(img_src,img_dilatacion,Mat());

element = getStructuringElement(MORPH_ELLIPSE,Size(10,10));
dilate(img_src,img_dilatacion,element);

//Erosion
erode(img_src,img_erosion,Mat());

element = getStructuringElement(MORPH_ELLIPSE,Size(10,10));
erode(img_src,img_erosion,element);


//Gradiente morphologico
morphologyEx(img_src,img_morphology,MORPH_GRADIENT,Mat());
```

Transformaciones morfológicas

- e01_trans_morphology.cpp
  - Cargar la imagen.
  - Comprobar que se ha cargado correctamente.
  - Aplicar transformación morfológica (probar diferentes)
  - Mostrar las imágenes.
  - Esperar la pulsación de una tecla.
  - Liberar memoria.
  - Finalizar el programa.

Transformaciones morfológicas

Intelligent Systems Lab

```cpp
#include "opencv\cv.hpp"
#include <iostream>

using namespace cv;
using namespace std;

int main(int argc, char* argv[])
{
    // Objects
    Mat src_img, erode_img, dilate_img, morph_img;
    Mat kernel;
    // Load image from disk
    src_img = imread("open_close.png");
    if (!src_img.data){
        cout << "error loading image" << endl;
        system("pause");
        return 1;
    }

    // Create kernel of size 3x3. Try different sizes
    kernel = getStructuringElement(MORPH_RECT , Size(10, 10));
    //kernel = getStructuringElement(MORPH_CROSS, Size(10, 10));
    //kernel = getStructuringElement(MORPH_ELLIPSE, Size(10, 10));
    cout << kernel << endl;

    // Dilate
    dilate(src_img, dilate_img, kernel);

    // Erode
    erode(src_img, erode_img, kernel);

    // Morphologic transformation: could be:
    // - MORPH_OPEN
    // - MORPH_CLOSE
    // - MORPH_GRADIENT
    // - MORPH_TOPHAT
    // - MORPH_BLACKHAT
    morphologyEx(src_img, morph_img, MORPH_CLOSE, kernel);

    // Show image in the name of the window
    imshow("original", src_img);
    imshow("dilate", dilate_img);
    imshow("erode", erode_img);
    imshow("morphology",morph_img);

    waitKey(0);

    // Free memory
    src_img.release();
    dilate_img.release();
    erode_img.release();
    morph_img.release();
    destroyAllWindows();
    // End of the program
    return 0;
}
```