strel

Morphological structuring element

Syntax

```
SE = strel(nhood)
SE = strel("diamond",r)
SE = strel("disk",r)
SE = strel("disk",r,n)
SE = strel("octagon",r)
SE = strel("line",len,deg)
SE = strel("rectangle",[m n])
SE = strel("square",w)
SE = strel("cube",w)
SE = strel("cuboid",[m n p])
SE = strel("sphere",r)
```

Description

Arbitrary Neighborhood Shape

`SE = strel(nhood)` creates a flat structuring element with specified neighborhood `nhood`.

2-D Geometric Neighborhood Shapes

`SE = strel("diamond",r)` creates a diamond-shaped structuring element, where `r` specifies the distance from the structuring element origin to the points of the diamond.

`SE = strel("disk",r)` creates a disk-shaped structuring element, where `r` specifies the radius.

`SE = strel("disk",r,n)` creates a disk-shaped structuring element, where `r` specifies the radius and `n` specifies the number of line structuring elements used to approximate the disk shape. Morphological operations run much faster when the structuring element uses approximations.

`SE = strel("octagon",r)` creates a octagonal structuring element, where `r` specifies the distance from the structuring element origin to the sides of the octagon, as measured along the horizontal and vertical axes. `r` must be a nonnegative multiple of 3.

`SE = strel("line",len,deg)` creates a linear structuring element that is symmetric with respect to the neighborhood center, with approximate length `len` and angle `deg`.

`SE = strel("rectangle",[m n])` creates a rectangular structuring element of size `[m n]`.

`SE = strel("square",w)` creates a square structuring element whose width is `w` pixels.

3-D Geometric Neighborhood Shapes

`SE = strel("cube",w)` creates a 3-D cubic structuring element whose width is `w` pixels.

`SE = strel("cuboid",[m n p])` creates a 3-D cuboidal structuring element of size *m*-by-*n*-by-*p* pixels.

`SE = strel("sphere",r)` creates a 3-D spherical structuring element whose radius is `r` pixels.

Compatibility

The following syntaxes still work, but `offsetstrel` is the preferred way to create these nonflat structuring element shapes:

- `SE = strel("arbitrary",nhood,h)`, where `h` is a matrix of the same size as `nhood` containing the height values associated with each nonzero element of `nhood`.
- `SE = strel("ball",r,h,n)`

imerode

Erode image

Syntax

```
J = imerode(I,SE)

J = imerode(I,nhood)

J = imerode(__,packopt,m)

J = imerode(__,shape)
```

Description

`J = imerode(I,SE)` erodes the grayscale, binary, or packed binary image `I` using the structuring element `SE`.

`J = imerode(I,nhood)` erodes the image `I`, where `nhood` is a matrix of `0`s and `1`s that specifies the structuring element neighborhood.

This syntax is equivalent to `imerode(I,strel(nhood))`.

`J = imerode(__,packopt,m)` specifies whether input image `I` is a packed binary image. `m` specifies the row dimension of the original unpacked image.

`J = imerode(__,shape)` specifies the size of the output image.

imdilate

Dilate image

Syntax

```
J = imdilate(I,SE)

J = imdilate(I,nhood)

J = imdilate(__,packopt)

J = imdilate(__,shape)
```

Description

`J = imdilate(I,SE)` dilates the grayscale, binary, or packed binary image `I` using the structuring element `SE`.

`J = imdilate(I,nhood)` dilates the image `I`, where `nhood` is a matrix of `0`s and `1`s that specifies the structuring element neighborhood.

This syntax is equivalent to `imdilate(I,strel(nhood))`.

`J = imdilate(__,packopt)` specifies whether `I` is a packed binary image.

`J = imdilate(__,shape)` specifies the size of the output image.

References

1. **https://www.mathworks.com/help/images/ref/strel.html**
2. **https://www.mathworks.com/help/images/ref/imerode.html**
3. **https://www.mathworks.com/help/images/ref/imdilate.html**