imdilate

Dilate image

Syntax

```
IM2 = imdilate(IM,SE)
IM2 = imdilate(IM,NHOOD)
IM2 = imdilate( __ ,PACKOPT)
IM2 = imdilate( __ ,SHAPE)
gpuarrayIM2 = imdilate(gpuarrayIM, __ )
```

Description

`IM2 = imdilate(IM,SE)` dilates the grayscale, binary, or packed binary image `IM`, returning the dilated image, `IM2`. The argument `SE` is a structuring element object, or array of structuring element objects, returned by the `strel` function.

If `IM` is logical and the structuring element is flat, `imdilate` performs binary dilation; otherwise, it performs grayscale dilation. If `SE` is an array of structuring element objects, `imdilate` performs multiple dilations of the input image, using each structuring element in succession.

`IM2 = imdilate(IM,NHOOD)` dilates the image `IM`, where `NHOOD` is a matrix of 0's and 1's that specifies the structuring element neighborhood. This is equivalent to the syntax `imdilate(IM,strel(NHOOD))`. The `imdilate` function determines the center element of the neighborhood by `floor((size(NHOOD)+1)/2)`.

`IM2 = imdilate( __ ,PACKOPT)` specifies whether `IM` is a packed binary image. `PACKOPT` can have either of the following values. Default value is enclosed in braces ({}).

Value	Description
`'ispacked'`	`IM` is treated as a packed binary image as produced by `bwpack`. `IM` must be a 2-D `uint32` array and `SE` must be a flat 2-D structuring element. If the value of `PACKOPT` is `'ispacked'`, `PADOPT` must be `'same'`.
`{'notpacked'}`	`IM` is treated as a normal array.

`IM2 = imdilate( __ ,SHAPE)` specifies the size of the output image. `SHAPE` can have either of the following values. Default value is enclosed in braces ({}).

Value	Description
`{'same'}`	Make the output image the same size as the input image. If the value of `PACKOPT` is `'ispacked'`, `SHAPE` must be `'same'`.
`'full'`	Compute the full dilation.

`gpuarrayIM2 = imdilate(gpuarrayIM, __ )` performs the operation on a graphics processing unit (GPU), where `gpuarrayIM` is a `gpuArray` that contains a grayscale or binary image. `gpuarrayIM2` is a `gpuArray` of the same class as the input image. Note that the `PACKOPT` syntax is not supported on a GPU. This syntax requires the Parallel Computing Toolbox™.

Code Generation support: Yes.

MATLAB Function Block support: Yes.

Class Support

`IM` can be logical or numeric and must be real and nonsparse. It can have any dimension. If `IM` is logical, `SE` must be flat. The output has the same class as the input. If the input is packed binary, then the output is also packed binary.

`gpuarrayIM` must be a `gpuArray` of type `uint8` or `logical`. When used with a `gpuarray`, the structuring element must be flat and two-dimensional. The output has the same class as the input.

Examples

collapse all

Dilate a Binary Image with a Vertical Line Structuring Element

Read a binary image.

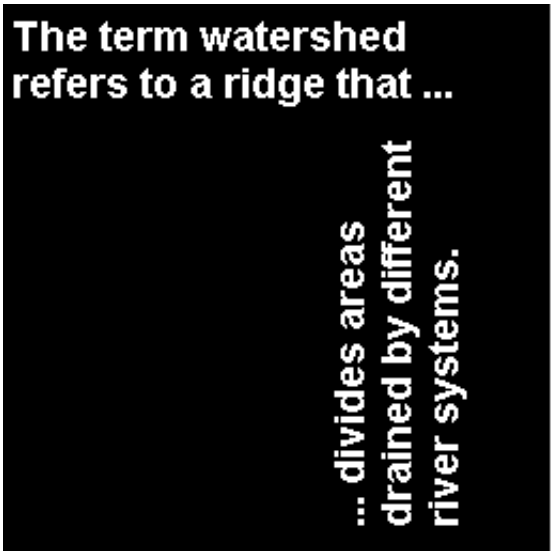
```
bw = imread('text.png');
```

Create a structuring element.

```
se = strel('line',11,90);
```

Dilate the image with a vertical line structuring element and compare the results.

```
bw2 = imdilate(bw,se);
imshow(bw), title('Original')
figure, imshow(bw2), title('Dilated')
```



Dilate a Grayscale Image with a Rolling Ball Structuring Element

Read a grayscale image.

```
I = imread('cameraman.tif');
```

Create a structuring element.

```
se = strel('ball',5,5);
```

Dilate a grayscale image with a rolling ball structuring element.

```
I2 = imdilate(I,se);
imshow(I), title('Original')
figure, imshow(I2), title('Dilated')
```

Determine the Domain of the Composition of Two Flat Structuring Elements

To determine the domain of the composition of two flat structuring elements, dilate the scalar value `1` with both structuring elements in sequence, using the `'full'` option.

Create a flat structuring element.

```
se1 = strel('line',3,0)
```

```
se1 =

Flat STREL object containing 3 neighbors.
Neighborhood:
     1     1     1
```

Create another flat structuring element.

```
se2 = strel('line',3,90)
```

```
se2 =

Flat STREL object containing 3 neighbors.
Neighborhood:
     1
     1
     1
```

Dilate the scalar value 1 using both structuring elements in sequence

```
composition = imdilate(1,[se1 se2],'full')
```

```
composition =
     1     1     1
     1     1     1
     1     1     1
```

Dilate a Binary Image with a Vertical Line Structuring Element on a GPU

Read a binary image.

```
originalBW = imread('text.png');
```

Create a structuring element.

```
se = strel('line',11,90);
```

Dilate the image with a vertical line structuring element and compare the results. Note how the example passes the image to the `gpuArray` function as it passes it to `imdilate`.

```
dilatedBW = imdilate(gpuArray(originalBW),se);
figure, imshow(originalBW), figure, imshow(dilatedBW)
```

Dilate a Grayscale Image with a Disk Structuring Element on a GPU

Read a grayscale image.

```
originalI = imread('cameraman.tif');
```

Create a structuring element.

```
se = strel('disk',5);
```

Dilate a grayscale image with a rolling ball structuring element.

```
dilatedI = imdilate(gpuArray(originalI),se);
figure, imshow(originalI), figure, imshow(dilatedI)
```

Definitions

The *binary dilation* of *A* by *B*, denoted $A \oplus B$, is defined as the set operation:

$$A \oplus B = \left\{ z \middle| \left(\hat{B}\right)_z \cap A \neq \emptyset \right\},$$

where $\hat{B}$ is the reflection of the structuring element *B*. In other words, it is the set of pixel locations *z*, where the reflected structuring element overlaps with foreground pixels in *A* when translated to *z*. Note that some people use a definition of dilation in which the structuring element is not reflected.

In the general form of *gray-scale dilation*, the structuring element has a height. The gray-scale dilation of *A(x,y)* by *B(x,y)* is defined as:

$$\left(A \oplus B\right)(x, y) = \max\left\{ A(x - x', y - y') + B(x', y') \middle| \left(x', y'\right) \in D_B \right\},$$

where D_B is the domain of the structuring element *B* and *A*(*x*,*y*) is assumed to be −∞ outside the domain of the image. To create a structuring element with nonzero height values, use the syntax `strel(nhood,height)`, where `height` gives the height values and `nhood` corresponds to the structuring element domain, D_B.

Most commonly, gray-scale dilation is performed with a flat structuring element (*B*(*x*,*y*) = 0). Gray-scale dilation using such a structuring element is equivalent to a local-maximum operator:

$$\left(A \oplus B\right)(x, y) = \max\left\{ A(x - x', y - y') \middle| (x', y') \in D_B \right\}.$$

All of the `strel` syntaxes except for `strel(nhood,height)`, `strel('arbitrary',nhood,height)`, and `strel('ball', ...)` produce flat structuring elements.

For more information about binary dilation, see [1].

More About

expand all

Code Generation

This function supports the generation of C code using MATLAB® Coder™. Note that if you choose the generic `MATLAB Host Computer` target platform, the function generates code that uses a precompiled, platform-specific shared library. Use of a shared library preserves performance optimizations but limits the target platforms for which code can be generated. For more information, see Understanding Code Generation with Image Processing Toolbox.

When generating code, the input image, `IM`, must be 2-D or 3-D. The structuring element argument `SE` must be a single element—arrays of structuring elements are not supported. To obtain the same result as that obtained using an array of structuring elements, call the function sequentially. When the target is `MATLAB Host Computer`, the `PACKOPT` and `SHAPE` arguments must be compile-time constants. When the target is any other platform, the `PACKOPT` syntax is not supported.

MATLAB Function Block

You can use this function in the MATLAB Function Block in Simulink.

Algorithms

`imdilate` automatically takes advantage of the decomposition of a structuring element object (if it exists). Also, when performing binary dilation with a structuring element object that has a decomposition, `imdilate` automatically uses binary image packing to speed up the dilation.

Dilation using bit packing is described in [3].

References

[1] Gonzalez, R. C., R. E. Woods, and S. L. Eddins, *Digital Image Processing Using MATLAB*, Gatesmark Publishing, 2009.

[2] Haralick, R. M., and L. G. Shapiro, *Computer and Robot Vision*, Vol. I, Addison-Wesley, 1992, pp. 158-205.

[3] van den Boomgard, R, and R. van Balen, "Methods for Fast Morphological Image Transforms Using Bitmapped Images," *Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing*, Vol. 54, Number 3, pp. 254-258, May 1992.

See Also

`bwpack` | `bwunpack` | `conv2` | `filter2` | `gpuArray` | `imclose` | `imerode` | `imopen` | `strel`

Introduced before R2006a