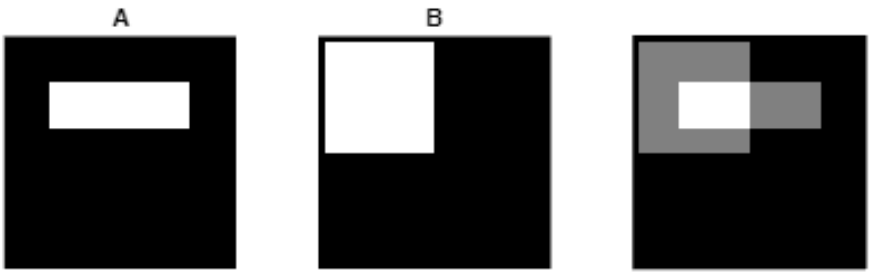
Cvičení 11 - Matematická morfologie

Množinové operace

množiny A a B (1 představuje fakt, že prvek (pixel na souřadnici) leží v množině, 0, že neleží)

celý obrázek je univerzum

```
A = zeros(500);
B = zeros(500);

A(100:200, 100:400) = 1;
B(15:250, 15:250) = 1;

figure
subplot(1,3,1)
imshow(A)
title('A')
subplot(1,3,2)
imshow(B)
title('B')
subplot(1,3,3)
imshowpair(A, B,'blend');
```



Množinové operace

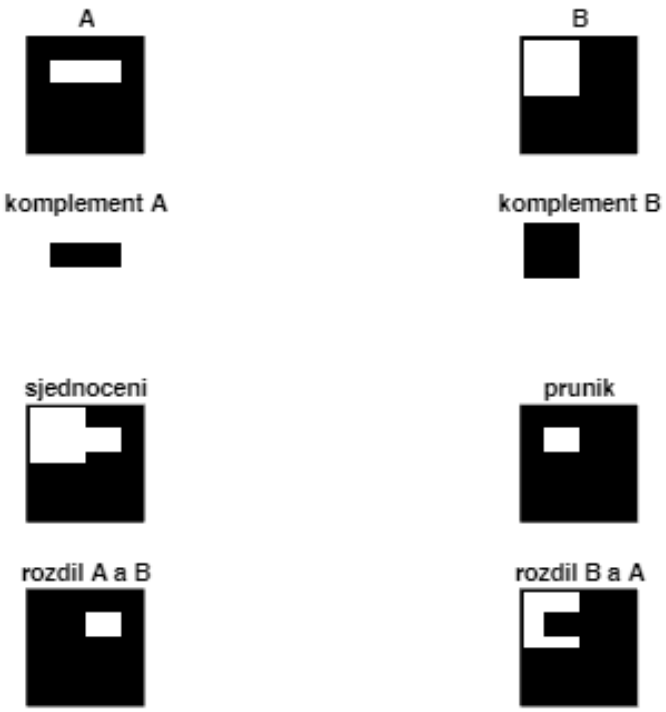
- komplement

- sjednoceni

- průnik

- rozdíl

```
komplement_A = 1 - A;
komplement_B = 1 - B;
sjednoceni = max(A,B);
prunik = min(A,B);
rozdil_A_B = max(A - B,zeros(size(A)));
rozdil_B_A = max(B - A,zeros(size(A)));

figure
subplot(4,2,1)
imshow(A)
title('A')
subplot(4,2,2)
imshow(B)
title('B')
subplot(4,2,3)
imshow(komplement_A)
title('komplement A')
subplot(4,2,4)
imshow(komplement_B)
title('komplement B')
subplot(4,2,5)
imshow(sjednoceni)
title('sjednoceni')
subplot(4,2,6)
imshow(prunik)
title('prunik')
subplot(4,2,7)
imshow(rozdil_A_B)
title('rozdil A a B')
subplot(4,2,8)
imshow(rozdil_B_A)
title('rozdil B a A')
```



ÚKOL 1

V praxi se pro modelování množinových operací používají logické operátory. Popište (naprogramujte) operace komplement, sjednocení, průnik a rozdíl pomocí logických operátorů.

`and()`, `or()`, `not()`, `xor()`

```
```

Další operace

Posunutí - translace

Množiny o bod `z = (z1,z2)`

$A_z = \{c | c = a + z, \forall a \in A\}$

```
z = [50,150];
Az = imtranslate(A,z);
figure
subplot(1,2,1)
imshow(A)
title('A')
subplot(1,2,2)
imshow(Az)
title('Az')
```

Reflexe

$\widehat{A} = \{w | w = -a, \forall a \in A\}$

Morfologické operátory

Dilatace

Rozšiřuje objekty v obrázku na základě strukturálního elementu.

$$A \oplus B = \bigcup_{b \in B} A_b$$

Strukturální element vytvoříme pomocí fce `strel()`

```
SE = strel([0 0 1; 0 1 0; 1 0 0]);
obr = zeros(5);
obr(3,3) = 1;
obr
```

```
obr = 5×5
     0     0     0     0     0
     0     0     0     0     0
     0     0     1     0     0
     0     0     0     0     0
     0     0     0     0     0
```

U strukturálního elementu se předpokládá, že střed je na souřadnicích `floor((size,B+1)/2)`.

dilatace - `imdilate()`

```
obr_dilate = imdilate(obr,SE)
```

```
obr_dilate = 5×5
     0     0     0     0     0
     0     0     0     1     0
     0     0     1     0     0
     0     1     0     0     0
     0     0     0     0     0
```

`strel()` implementuje vytvoření strukturálního elementu nejen ze zadané matice, ale i speciálních tvarů.

```
help strel
```

```
 strel Create morphological structuring element.
    SE = strel('arbitrary',NHOOD) creates a flat structuring element with
    the specified neighborhood.  NHOOD is a matrix containing 1's and 0's;
    the location of the 1's defines the neighborhood for the morphological
    operation.  The center (or origin) of NHOOD is its center element,
    given by FLOOR((SIZE(NHOOD) + 1)/2).  You can also omit the 'arbitrary'
    string and just use strel(NHOOD).

    SE = strel('diamond',R) creates a flat diamond-shaped structuring
    element with the specified size, R.  R is the distance from the
    structuring element origin to the points of the diamond.  R must be a
    nonnegative integer scalar.

    SE = strel('disk',R,N) creates a flat disk-shaped structuring element
    with the specified radius, R.  R must be a nonnegative integer.  N must
    be 0, 4, 6, or 8.  When N is greater than 0, the disk-shaped structuring
    element is approximated by a sequence of N (or sometimes N+2)
    periodic-line structuring elements.  When N is 0, no approximation is
    used, and the structuring element members comprise all pixels whose
    centers are no greater than R away from the origin.  N can be omitted,
    in which case its default value is 4.  Note: Morphological operations
    using disk approximations (N>0) run much faster than when N=0.  Also,
    the structuring elements resulting from choosing N>0 are suitable for
    computing granulometries, which is not the case for N=0.  Sometimes it
    is necessary for strel to use two extra line structuring elements in the
    approximation, in which case the number of decomposed structuring
    elements used is N+2.

    SE = strel('line',LEN,DEG) creates a flat linear structuring element
    that is symmetric with respect to the neighborhood center. DEG
    specifies the angle (in degrees) of the line as measured in a
    counterclockwise direction from the horizontal axis. LEN is
    approximately the distance between the centers of the structuring
    element members at opposite ends of the line.

    SE = strel('octagon',R) creates a flat octagonal structuring element
    with the specified size, R.  R is 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('rectangle',MN) creates a flat rectangle-shaped structuring
    element with the specified size.  MN must be a two-element vector of
    nonnegative integers.  The first element of MN is the number rows in the
    structuring element neighborhood; the second element is the number of
```

columns.

SE = **strel**('square',W) creates a square structuring element whose width is W pixels. W must be a nonnegative integer scalar.

SE = **strel**('sphere',R) creates a sphere-shaped structuring element whose radius is R pixels. R must be a nonnegative integer scalar.

SE = **strel**('cube',W) creates a cube structuring element whose width is W pixels. W must be a nonnegative integer scalar.

SE = **strel**('cuboid', XYZ) creates a cuboidal structuring element of size XYZ. XYZ must be a three-element vector of nonnegative integers specifying the number of rows, number of columns, and the number of planes in the third dimension.

Notes

For all shapes except 'arbitrary', structuring elements are constructed using a family of techniques known collectively as "structuring element decomposition." The principle is that dilation by some large structuring elements can be computed faster by dilation with a sequence of smaller structuring elements. For example, dilation by an 11-by-11 square structuring element can be accomplished by dilating first with a 1-by-11 structuring element and then with an 11-by-1 structuring element. This results in a theoretical performance improvement of a factor of 5.5, although in practice the actual performance improvement is somewhat less. Structuring element decompositions used for the 'disk' and 'ball' shapes are approximations; all other decompositions are exact.

The following syntaxes will still work, but are not recommended for use. The non-flat structuring element shapes listed below can be created using OFFSETSTREL.

SE = **strel**('pair',OFFSET) creates a flat structuring element containing two members. One member is located at the origin; the second member's location is specified by the vector OFFSET. OFFSET must be a two-element vector of integers.

SE = **strel**('periodicline',P,V) creates a flat structuring element containing 2*P+1 members. V is a two-element vector containing integer-valued row and column offsets. One structuring element member is located at the origin. The other members are located at 1*V, -1*V, 2*V, -2*V, ..., P*V, -P*V.

SE = **strel**('arbitrary',NHOOD,HEIGHT) creates a nonflat structuring element with the specified neighborhood. HEIGHT is a matrix the same size as NHOOD containing the height values associated with each nonzero element of NHOOD. HEIGHT must be real and finite-valued. You can also omit the 'arbitrary' string and just use **strel**(NHOOD,HEIGHT).

SE = **strel**('ball',R,H,N) creates a nonflat "ball-shaped" (actually an ellipsoid) structuring element whose radius in the X-Y plane is R and whose height is H. R must be a nonnegative integer, and H must be a real scalar. N must be an even nonnegative integer. When N is greater than 0, the ball-shaped structuring element is approximated by a sequence of N nonflat line-shaped structuring elements. When N is 0, no approximation is used, and the structuring element members comprise all pixels whose centers are no greater than R away from the origin, and the corresponding height values are determined from the formula of the ellipsoid specified by R and H. If N is not specified, the default value is 8. Note: Morphological operations using ball approximations (N>0) run much faster than when N=0.

```
Examples
--------
    se1 = strel('square', 11)      % 11-by-11 square
    se2 = strel('line', 10, 45)     % length 10, angle: 45 degrees

    % Create and display a 2-D disk-shaped structuring element.
    se3 = strel('disk', 15)
    figure, imshow(se3.Neighborhood)

    % Create and display a 3-D sphere-shaped structuring element.
    se4 = strel('sphere', 15)
    figure, isosurface(se4.Neighborhood)

See also imdilate, imerode, offsetstrel.

Documentation for strel
```

ÚKOL 2

Vyzkoušejte různé SE a podívejte se na výsledky.

```
SE = strel('disk',4,0);
I = imread("text.png");
I_dilate = imdilate(I,SE);

figure
subplot(1,3,1)
imshow(I)
title('I')
subplot(1,3,2)
imshow(I_dilate)
title('dilate I')
subplot(1,3,3)
imshow(imfuse(I, I_dilate,'falsecolor'));
```

Eroze

Ztenčuje objekty v závislosti na zvoleném SE.

$$A \ominus B = \bigcap_{b \in B} A_{-b}$$

`imerode()`

```
SE = strel([0 0 0; 0 1 0; 1 0 0]);
obr = zeros(5);
obr(2:4,2:4) = 1;
obr
```

```
obr = 5×5
     0     0     0     0     0
     0     1     1     1     0
     0     1     1     1     0
     0     1     1     1     0
     0     0     0     0     0
```

```
obr_erode = imerode(obr,SE)
```

```
obr_erode = 5×5
     0     0     0     0     0
     0     0     1     1     0
     0     0     1     1     0
     0     0     0     0     0
     0     0     0     0     0
```

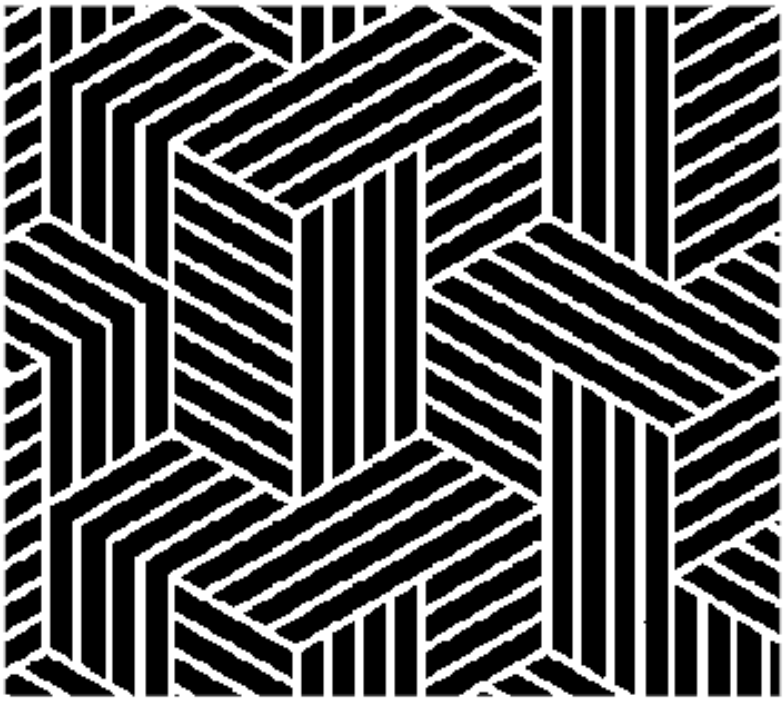
```
SE = strel('disk',4,0);
```

```
I = imread("text.png");
I_erode = imerode(I,SE);

figure
subplot(1,3,1)
imshow(I)
title('I')
subplot(1,3,2)
imshow(I_erode)
title('erode I')
subplot(1,3,3)
imshow(imfuse(I, I_erode,'falsecolor'));
```

ÚKOL 3

Vytvořte strukturální element tak, aby z následujícího obrázku odstranil pouze svislé čáry (pokud to jde).

```
I = imread("lines.png");

figure
imshow(I)
```



Otevření

$A \circ B = (A \ominus B) \oplus B$

Z obrázku odstraňuje objekty, do kterých se nevejde celý strukturální element.

`imopen()`

```
SE = strel('disk',4,0);
I = imread("text.png");
I_open = imopen(I,SE);

figure
subplot(1,3,1)
imshow(I)
title('I')
subplot(1,3,2)
imshow(I_open)
title('open I')
subplot(1,3,3)
imshow(imfuse(I, I_open,'falsecolor'));
```

Uzavření

$A \bullet B = (A \oplus B) \ominus B$

Vyhlazuje kontury (stejně jako otevření), zaceluje malé díry

```
SE = strel('disk',4,0);
I = imread("text.png");
I_close = imclose(I,SE);

figure
subplot(1,3,1)
imshow(I)
title('I')
subplot(1,3,2)
imshow(I_close)
title('close I')
subplot(1,3,3)
imshow(imfuse(I, I_close,'falsecolor'));
```

ÚKOL 4

Co s obrázkem udělá otevření a následné uzavření? Co naopak (nejprve uzavření a pak otevření)? Budou výsledky stejné?

Hit-or-Miss transformace

Občas je potřeba detekovat určitá uskupení pixelů v obraze. K tomu můžeme použít transformaci hit-or-miss

$A \otimes B$

B je strukturální element skládající se z dvojice elementů B_1, B_2

$A \otimes B = (A \ominus B_1) \cap (A^c \ominus B_2)$

```
B_1 = [0 1 0; 1 1 1; 0 1 0]
```

```
B_1 = 3×3
     0     1     0
     1     1     1
     0     1     0
```

```
B_2 = [1 0 1; 0 0 0; 1 0 1]
```

```
B_2 = 3×3
```

```
     1     0     1
     0     0     0
     1     0     1
```

```
B_1 = strel(B_1);
B_2 = strel(B_2);
M = load('hitmiss.txt')
```

```
M = 10×10
     0     0     0     0     0     0     0     0     0     0
     0     1     0     0     0     1     1     1     1     0
     1     1     1     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     1     0
     0     0     0     0     1     0     0     1     1     1
     0     0     0     1     1     1     0     1     1     1
     0     0     0     0     1     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
```

$(A \ominus B_1)$

identifikuje místa, kde shluky pixelů odpovídají prvnímu SE

```
M2 = imerode(M,B_1)
```

```
M2 = 10×10
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     1     0
     0     0     0     0     1     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
```

A^c

```
MC = imcomplement(M)
```

```
MC = 10×10
     1     1     1     1     1     1     1     1     1     1
     1     0     1     1     1     0     0     0     0     1
     0     0     0     1     1     1     1     1     1     1
     1     0     1     1     1     1     1     1     1     1
     1     0     1     1     1     1     1     1     0     1
     1     1     1     1     0     1     1     0     0     0
     1     1     1     0     0     0     1     0     0     0
     1     1     1     1     0     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
```

$(A^c \ominus B_2)$

identifikuje, pixely, u kterých pixely dané druhým SE tvoří pozadí

```
M3 = imerode(MC,B_2)
```

```
M3 = 10×10
     0     1     0     1     0     0     0     0     0     0
     0     0     0     0     1     1     1     1     1     1
     0     1     0     1     0     0     0     0     0     0
     0     0     0     0     1     1     1     0     1     0
     0     1     0     0     1     0     0     0     0     0
     0     1     0     0     0     0     0     0     0     0
     1     1     1     0     1     0     0     0     0     0
     1     1     0     0     0     0     0     0     0     0
     1     1     1     0     1     0     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
```

$$A \otimes B = (A \ominus B_1) \cap (A^c \ominus B_2)$$

```
M_hitmis = min(M2,M3)
```

```
M_hitmis = 10×10
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     1     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
```

V matlabu je tato transformace implementována funkcí `bwhitmiss()`

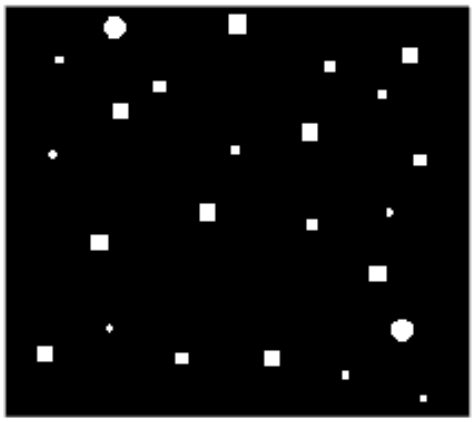
```
M_hitmis2 = bwhitmiss(M,B_1,B_2)
```

```
M_hitmis2 = 10×10 logical array
   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0
   0   1   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   1   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0
```

ÚKOL 5

V následujícím obrázku najděte všechny levé horní rohy čtverců.

```
I = imread("hitmiss.png");
figure
imshow(I)
```

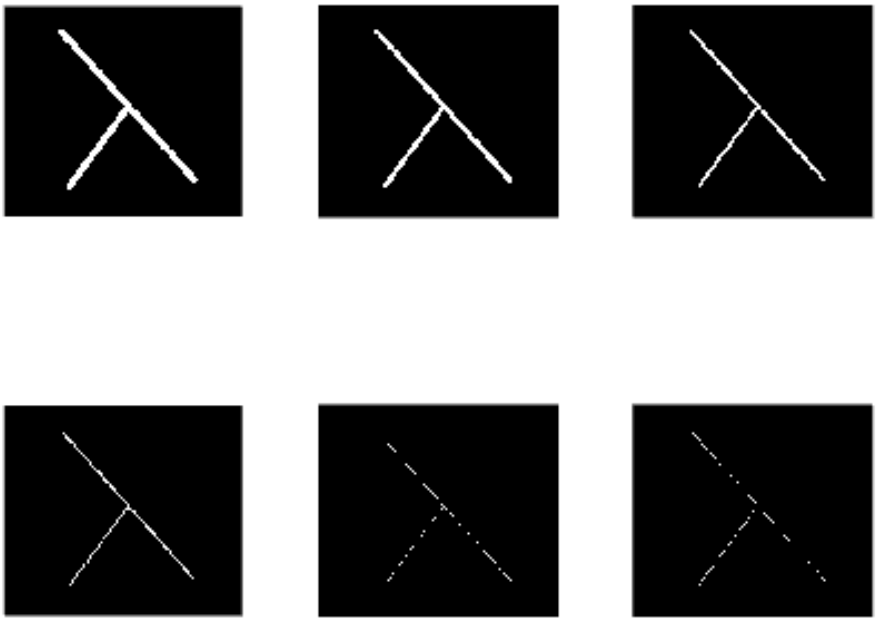


Ztenčení

V matlabu jsou ve funkci `bwmorph()` implementovaní další transformace založené na dilataci a erozi. Například ztenčení.

Výsledkem této operace je obrázek, který má o šířku zmenšenou o 1 pixel. Pro větší ztenčení se aplikuje víckrát.

```
I = imread("ztenceni.png");

figure,

for i = 1 : 6
    subplot(2,3,i), bwmorph(I, 'thin', i);
end
```



Od určitého opakování se obrázek nemění. Není nutné znát, kdy přestane. Počet opakování se nastaví na `Inf`. Jakmile se obrázek přestaně měnit, operace skončí.

Morfologická rekonstrukce

2 vstupní obrázky a jeden SE. Jeden z obrázků představuje značky, kde transformace začíná, 2. kde končí (maska). SE definuje sousednost (4 nebo 8)

F ... značky

G ... maska

Rekonstrukce $R_G(F)$:

1. h_1 inicializujeme F
2. Vytvoříme SE (pro 8 sousednost 3x3 jednotková matice)
3. Opakujeme $h_{k+1} = (h_k \oplus B) \cap G$ dokud $h_{k+1} = h_k$
4. $R_G(F) = h_k$

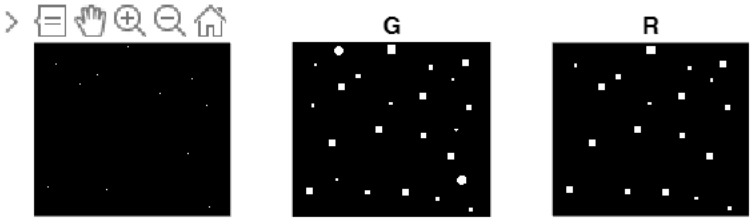
Příklad

Vezmeme obrázky z úkolu 5. F jsou nalezené levé rohy v obrázku G.

V matlabu `imreconstruct()`

```
G = imread("hitmiss.png");
```

```
F = imread("hitmiss2.png");
R = imreconstruct(F, G);

figure
subplot(1,3,1)
imshow(F)
title('F')
subplot(1,3,2)
imshow(G)
title('G')
subplot(1,3,3)
imshow(R);
title('R')
```



Výsledkem jsou v obrázku pouze čtvercové objekty.

Otevření rekonstrukcí

Při klasickém otevření jsou erozí odstraněny malé objekty v obraze a následnou dilatací se snažíme vrátit do původního stavu. Výsledek ale závisí na tvaru SE a původního tvaru. V této transformaci objekt vrátíme do původního stavu za pomoci rekonstrukce.

$R_G(G \ominus B)$

```
SE = strel('disk',4,0);
I = imread("text.png");
I_open = imopen(I,SE);

figure
subplot(2,3,1)
```

```
imshow(I)
title('I')
subplot(2,3,2)
imshow(I_open)
title('open I')
subplot(2,3,3)
imshow(imfuse(I, I_open,'falsecolor'));

Ierode = imerode(I,SE);
Ir = imreconstruct(Ierode,I);

subplot(2,3,4)
imshow(I)
title('I')
subplot(2,3,5)
imshow(Ir)
title('reconstruct')
subplot(2,3,6)
imshow(imfuse(I, Ir,'falsecolor'));
```

Další operace založené na iteračním procesu rekonstrukce

Vždy je potřeba vymyslet, jak vypadá obrázek značek.

Vyplnění děr

F je definován tak, že na okraji obrázku jsou pixely rovny 1-G, jinak jsou rovny 0.

G je nastaven na negativ obrázku.

Výsledkem je negativ obrázku s vyplněnými dírami.

V matlabu je naprogramovaná funkce `imfill()`

```
I = imread("text.png");
I_fill = imfill(I,'holes');

figure
subplot(1,2,1)
imshow(I)
title('I')
subplot(1,2,2)
imshow(I_fill)
title('fill')
```

ÚKOL 6

Naprogramujte iterační proces rekonstrukce a vyzkoušejte například na vyplňování děr. (Je možné v jiném jazyce, nebo jen popsat postup)

Vyčištění okrajů

Odstanění objektů, které leží na okraji obrázku.

Značkovací obrázek je definovaný tak, že hodnoty pixelů na okraji obrázku jsou rovny hodnotám obrázku G. Ostatní jsou nastaveny na 0.

V matlabu `imclearborder(obrazek,sousednost)` (sousednost = 4 nebo 8)

```
I = imread("text.png");
I = I(:, 1:600);
I_fill = imclearborder(I,8);

figure
subplot(1,2,1)
imshow(I)
title('I')
subplot(1,2,2)
imshow(I_fill)
title('fill')
```

Morfologie šedotónových obrázků

Všechny dříve zmíněné operace (kromě hit-or-miss) mají přímé rozšíření na šedotónové obrázky. Dilatace a eroze jsou definovány minimem a maximem okolních pixelů (to kterých je dáno strukturálním elementem).

SE se skládá ze dvou složek, jednak definující okolí pixelu (doména) a pak výšku. Pokud je výška rovna 0 pro všechny prvky definující okolí, pak mluvíme o flat operátoru a definujeme ho stejně, jako v případě černobílých obrázků.

Dilatace

$$A \oplus b(x, y) = \max \{I(x - x', y - y') + b(x', y')|(x', y') \in D_b\}$$

D_b označuje doménu strukturálního elementu (složka definující okolí)

Je vidět, že operace pracuje obdobně, jako konvoluce.

V případě flat operátoru se operace zjednodušuje a představuje maximální hodnotu z okolí pixelu, které je dané SE.

Vytvoření SE pomocí `strel()`. V případě operátoru, který není flat se předávají dvě matice.

Pro dilataci opět použijeme `imdilate()`

```
SE = strel('disk',4,0);
I = imread("gray.png");
I_dilate = imdilate(I,SE);

figure
subplot(1,3,1)
imshow(I)
title('I')
subplot(1,3,2)
imshow(I_dilate)
title('dilate I')
subplot(1,3,3)
imshow(imfuse(I, I_dilate,'falsecolor'));
```

Výsledkem je rozmazaný obrázek.

Eroze

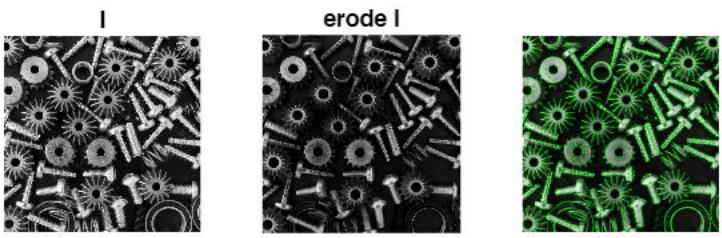
$$A \ominus b(x, y) = \min \{I(x - x', y - y') + b(x', y') | (x', y') \in D_b\}$$

D_b označuje doménu strukturálního elementu (složka definující okolí)

local minimum operator

Pro dilataci opět použijeme `imerode()`

```
SE = strel('disk',4,0);
I = imread("gray.png");
I_erode = imerode(I,SE);

figure
subplot(1,3,1)
imshow(I)
title('I')
subplot(1,3,2)
imshow(I_erode)
title('erode I')
subplot(1,3,3)
imshow(imfuse(I, I_erode,'falsecolor'));
```



Kombinace dilatace a eroze

kombinací výsledků eroze a dilatace můžeme dosáhnout zajímavých výsledků.

Morfologický gradient

odečtení erodovaného obrázku od dilatovaného.

```
I_morf = I_dilate - I_erode;

figure
subplot(1,3,1)
imshow(I_dilate)
title('dilate I')
subplot(1,3,2)
imshow(I_erode)
title('erode I')
subplot(1,3,3)
imshow(I_morf);
title('morfologicky gradient')
```

Otevření

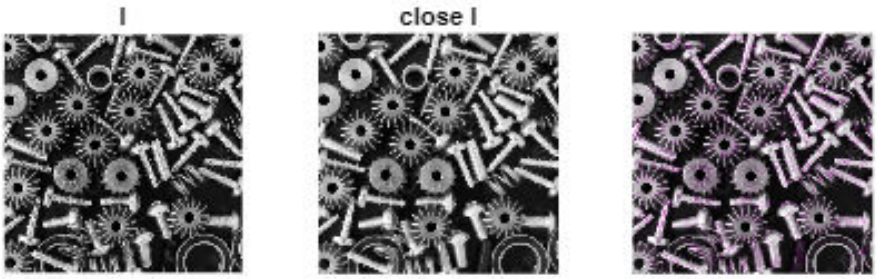
Potlačuje světlé části menší než strukturální element.

```
SE = strel('disk',4,0);
I = imread("gray.png");
I_open = imopen(I,SE);

figure
subplot(1,3,1)
imshow(I)
title('I')
subplot(1,3,2)
imshow(I_open)
title('open I')
subplot(1,3,3)
imshow(imfuse(I, I_open,'falsecolor'));
```

Uzavření

Potlačuje tmavé části menší než strukturální element.

```
SE = strel('disk',4,0);
I = imread("gray.png");
I_close = imclose(I,SE);

figure
subplot(1,3,1)
imshow(I)
title('I')
subplot(1,3,2)
imshow(I_close)
title('close I')
subplot(1,3,3)
imshow(imfuse(I, I_close,'falsecolor'));
```



Kombinace otevření a uzavření

open-close operace

uzavření obrázku získaného otevřením.

Vyhlazuje světlé části, tmavé části jsou téměř nedotčené.

```
SE = strel('disk',4,0);
I = imread("gray.png");
I_open = imopen(I,SE);
I_openclose = imclose(I_open,SE);

figure
subplot(1,3,1)
imshow(I)
title('I')
subplot(1,3,2)
imshow(I_openclose)
title('open-close I')
subplot(1,3,3)
imshow(imfuse(I, I_openclose,'falsecolor'));
```

close-open operace

uzavření obrázku získaného otevřením.

Ponechány pouze světlé detaily.

```
SE = strel('disk',4,0);
I = imread("gray.png");
I_close = imclose(I,SE);
I_closeopen = imopen(I_close,SE);

figure
subplot(1,3,1)
imshow(I)
title('I')
subplot(1,3,2)
imshow(I_closeopen)
title('close-open I')
subplot(1,3,3)
imshow(imfuse(I, I_closeopen,'falsecolor'));
```

Alternující sekvenční filtrování

Postupné aplikování open-close operace s postupně se zvětšujícím SE.

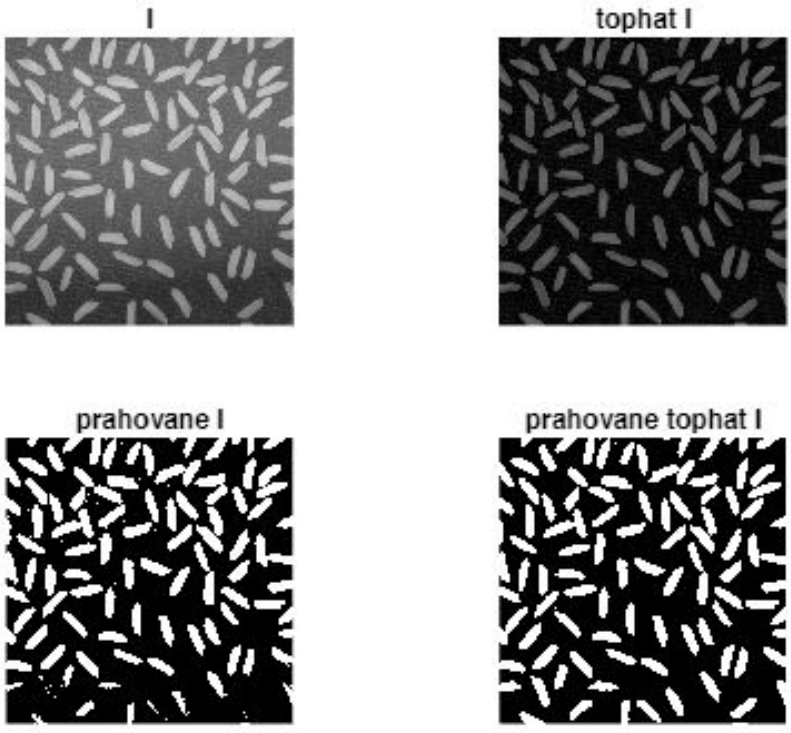
```
I = imread("gray.png");
I_alter = I;

for k = 2 : 5
    SE = strel('disk',k);
    I_alter = imclose(imopen(I_alter,SE), SE);
end

figure
subplot(1,3,1)
imshow(I)
title('I')
subplot(1,3,2)
imshow(I_alter)
title('alter I')
subplot(1,3,3)
imshow(imfuse(I, I_alter,'falsecolor'));
```

Výsledkem je více vyhlazený obrázek.

Úprava pozadí - Tophat transformace

odečtení otevřeného obrázku od původního

```
SE = strel('disk',10,0);
I = imread("rice.png");
I_open = imopen(I,SE);

I_tophat = I - I_open;

figure
subplot(2,2,1)
imshow(I)
title('I')
subplot(2,2,2)
imshow(I_tophat)
title('tophat I')
subplot(2,2,3)
imshow(imbinarize(I));
title('prahovane I')
subplot(2,2,4)
imshow(imbinarize(I_tophat));
title('prahovane tophat I')
```



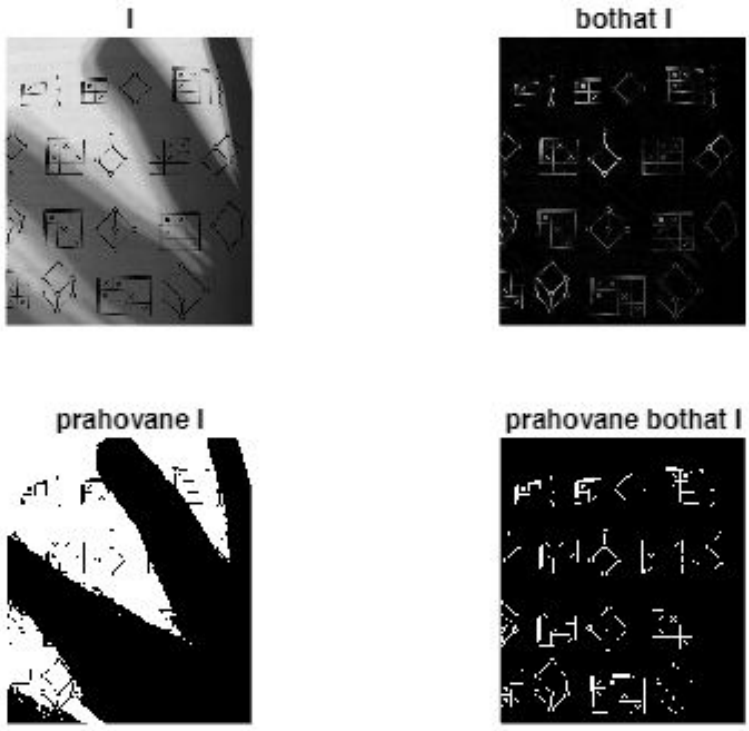
V matlabu `imtophat()`

Bottomhat transformace

odečtení uzavřeného obrázku od původního

V matlabu `imbothat()`

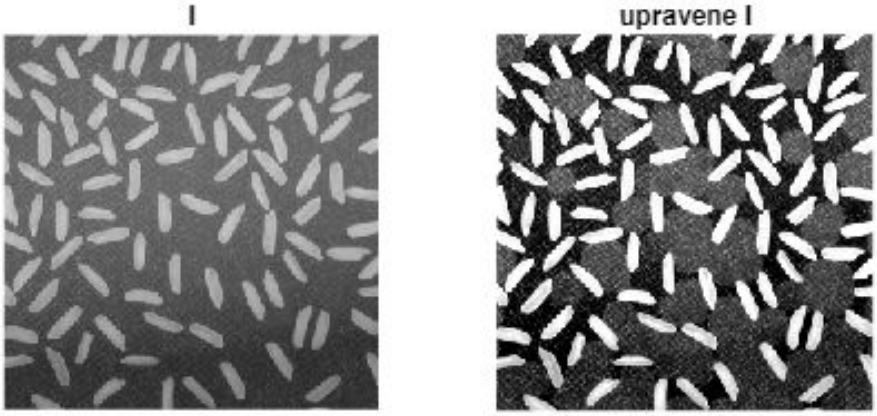
```
SE = strel('disk',10,0);
I = imread("lokalni.jpg");
I_bothat = imbothat(I,SE);


figure
subplot(2,2,1)
imshow(I)
title('I')
subplot(2,2,2)
imshow(I_bothat)
title('bothat I')
subplot(2,2,3)
imshow(imbinarize(I));
title('prahovane I')
subplot(2,2,4)
imshow(imbinarize(I_bothat));
title('prahovane bothat I')
```



Zvětšení kontrastu

obrázek + tophat - bothat

```
SE = strel('disk',10,0);
I = imread("rice.png");
I_kontrast = I + imtophat(I,SE) - imbothat(I,SE);


figure
subplot(1,2,1)
imshow(I)
title('I')
subplot(1,2,2)
imshow(I_kontrast)
title('upravene I')
```



Rekonstrukce pomocí morfologie

definovaná stejně jako pro binární obrázky.