



# 13. előadás

## Matlab 8. (Képek kezelése)

Dr. Szörényi Miklós,  
Dr. Kallós Gábor

2015–2016





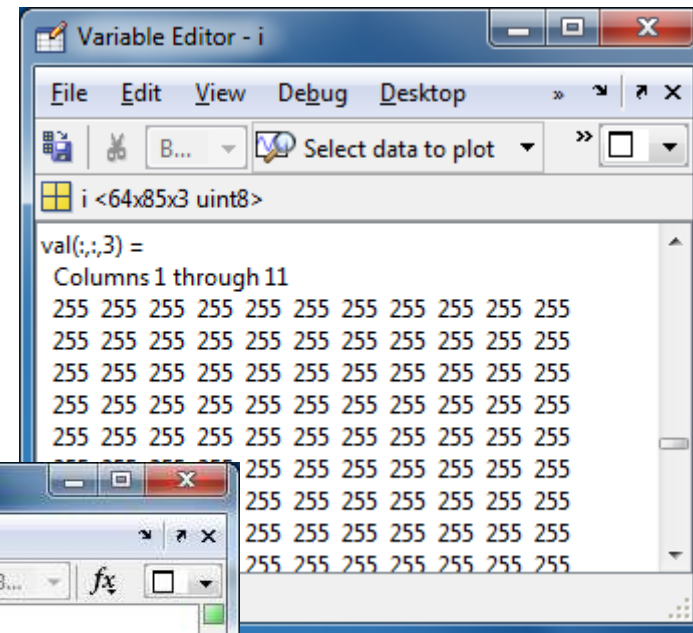
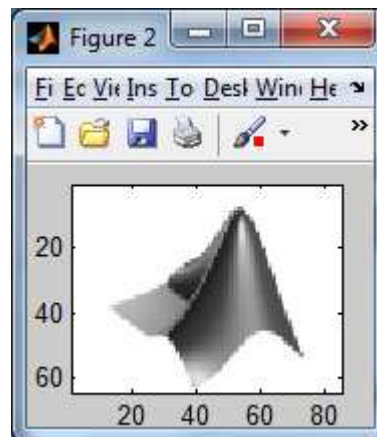
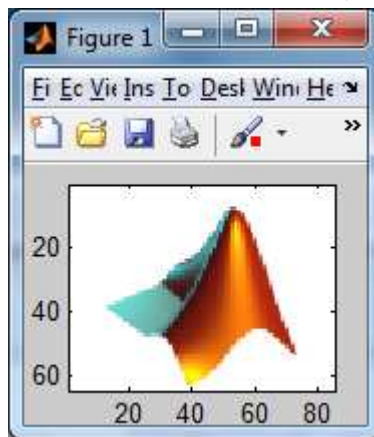
## Tartalom

- Áttekintés
- Képek betöltése
- Képtípusok a Matlabban
  - Indexelt, intenzitás, RGB
- Képtípus jellemzők lekérdezése
- Egyszerű módosítások
  - Részkép kivágása
  - Színcsatornák szétválasztása, invertálás
  - Vágási feladat
- Szűrési feladat
  - Zaj készítése
  - Lokális szűrés
- Éldetektálás
  - Eltérésmátrix, vágás, élvastagítás
- Kép háttérének cseréje

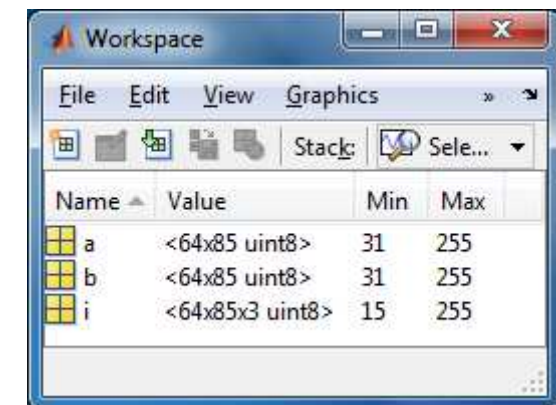


## Példa

- Színes logóból szürkeárnyaltos készítése és megjelenítése
  - Script, eredmény, változók



```
Editor - D:\Oktatás\Matlab-8-9\kepfeld-segedfajlok\szurke_uj.m*
File Edit Text Go Cell Tools Debug Desktop Window Help
1 i = imread('logo.png');
2 % egyszerű Matlab logo uint8 mátrixba olvasása
3 figure(1); imshow(i);
4 % a színes kép megjelenítése az 1. ablakban
5 a = .299*i(:, :, 1) + .587*i(:, :, 2) + .114*i(:, :, 3);
6 % az RGB összetevőkből szürkeintenzitás mátrix
7 b(:, :) = a;
8 % egycsatornás képmátrix szürkeintenzitással feltöltve
9 figure(2); imshow(b);
10 % a szürkeárnyaltos kép megjelenítése, és húzzuk arrébb
script Ln 10 Col 57 OVR
```





## Képek kezelése a Matlabban – áttekintés

- A Matlab a képek tárolására, kezelésére is mátrixokat (esetleg vektorokat) használ
  - Pl.: egy 300×400-as mátrixban tárolható egy ugyanilyen méretű kép
  - Bitmap filozófia (Bit-mapped images), az egyes képpontok a mátrix megfelelő elemei (+ esetleg szín információ)
    - Ha pl. RGB reprezentációt használunk, akkor 3D-típusú mátrixot kell bevetni
  - (A képek mátrixos tárolása miatt sok „normál” Matlab függvény használható lesz)
- Fontosabb műveletek képekkel
  - Olvasás, írás, megjelenítés
  - Információk lekérdezése (méretek, típus, colormap stb.)
  - Képtulajdonságok megváltoztatása (pl. vágás, élkiemelés, elmosás)
  - Grafikus formátumok közötti konverzió
- A képek megjelenítéséhez három számtípus használható: double, uint16, uint8
  - A double itt is az alapértelmezés, de célszerű takarékoskodni a memóriával (pl. gondoljunk egy 1000×1000-es képre...)
  - Idézzük fel a tárolási igényeket a típusoknál!
- Támogatott képformátumok
  - BMP, GIF, JPEG, PNG, TIFF, PCX stb.





## Képek betöltése

- Univerzális eszköz: `imread` függvény
  - Visszatérési értéke alapértelmezésben egy képmátrix, amelynek mérete a kép típusától függhet
  - Szürkeárnyaltos képeknél  $n \times n$ -es mátrixot kapunk
  - Ha három színcsatorna van (RGB), akkor az eredmény egy  $m \times n \times 3$ -as képmátrix
  - CMYK színtérben  $m \times n \times 4$ -es képmátrixot kapunk
- Szintaktika
  - `img = imread('filenév')`
    - Alapeset, a függvény a formátumot a kiterjesztésből határozza meg
  - `img = imread('filenév', 'formátumnév')`
    - A formátumot sztringként megadjuk második paraméterként (Ez általában egyértelmű – és így elhagyható –, de bizonyos képeknél nem derül mindig ki a kiterjesztésből)
  - `[X, map] = imread(...)`
    - Egyes spec. paraméterezések mellett a visszatérési érték a képmátrix mellett tartalmazhatja a képhez tartozó színskálát/színtérképet is (ha az létezik)
    - (Lásd példa a köv. oldalon)



## Képtípusok a Matlabban

### Indexelt

- Az indexelt kép két összetevőből áll
  - X – adatmátrix (képpontok)
  - map – colormap mátrix,  $n \times 3$ -as méretű,  $[0, 1]$ -beli értékekkel; tipikusan 256 sort tartalmaz
- Az X-beli értékek az adott sorszám szerint azonosítják a képpontok színeit (RGB) a map mátrixban (direkt leképezés)
  - Több ponthoz is tartozhat ugyanaz a színadatsor
  - Double típus esetén 1-től, uint8 vagy 16 esetén 0-tól sorszámozunk
- Indexelt kép megjelenítése (példa):
 

```
>> [X,map] = imread('canoe.tif');
% fontos a ;!
>> imshow(X,map)
% vagy
>> image(X);
>> colormap(map)
```

Variable Editor - X

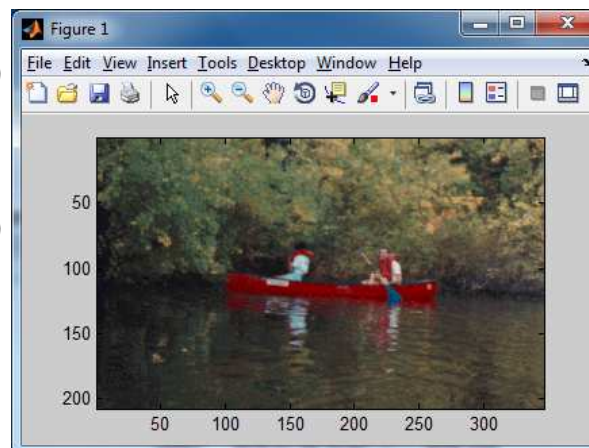
X <207x346 uint8>

	1	2	3	4	5
1	156	156	156	116	142
2	156	146	146	166	166
3	138	138	146	146	146
4	146	146	170	146	146
5	170	176	171	170	138
6	170	176	170	170	170
7	166	166	166	146	146

Variable Editor - map

map <256x3 double>

	1	2	3	4
1	1	1	1	
2	1	1	0	
3	1	0	1	
4	1	0	0	
5	0	1	1	
6	0	1	0	
7	0	0	1	
8	0	0	0	
9	0.5176	0	0	
10	0.4510	0	0	
11	0.3882	0	0	
12	0.5490	0.0627	0.0627	
13	0.4824	0.0627	0.0627	
14	0.4196	0.0627	0.0627	
15	0.5490	0.0941	0.0941	





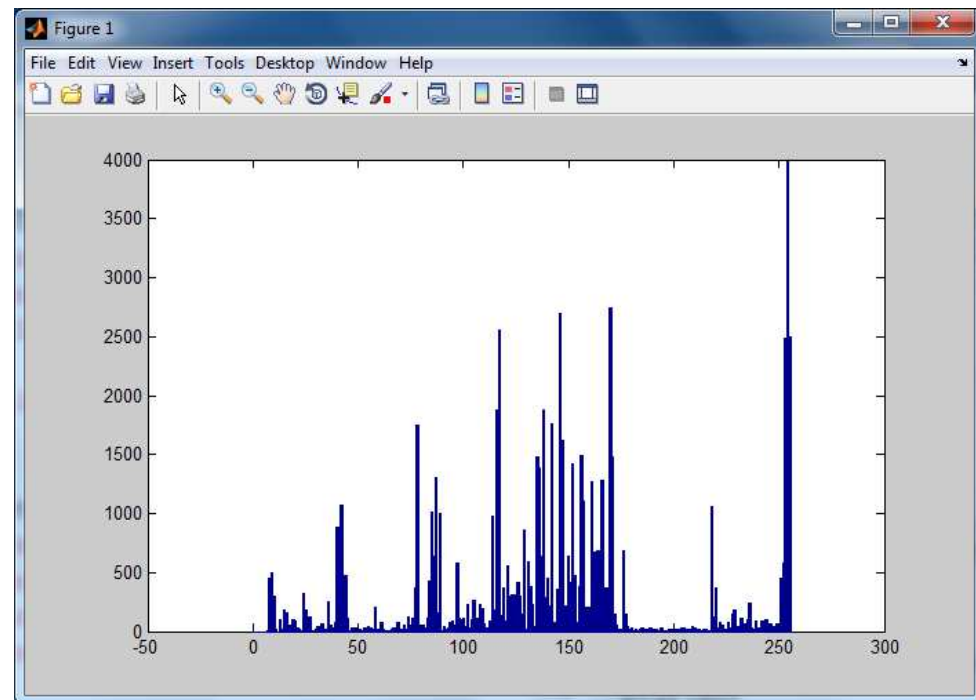
## Képtípusok a Matlabban

Indexelt (folyt.)

- Vizsgáljuk meg, hogy milyen az előző kép pontjainak színmegoszlása!
- Az X adatmárixra kérünk egy hisztogramot
  - 256 rekesz, a leggyakoribb színadatsorok ki fognak ugrani

■ Megoldás:

```
>> hist(X) % hiba,vektor paraméter kell
>> XU=reshape(X,1,207*346);
% átméretezünk
>> hist(XU) % még mindig
nem jó: integer típusra
nem működik
>> XU1=double(XU)
>> hist(XU1)
% 256 rekesz
>> hist(XU1, 256)
% Valóban nincs kicsi
érték?
>> min(XU1)
```



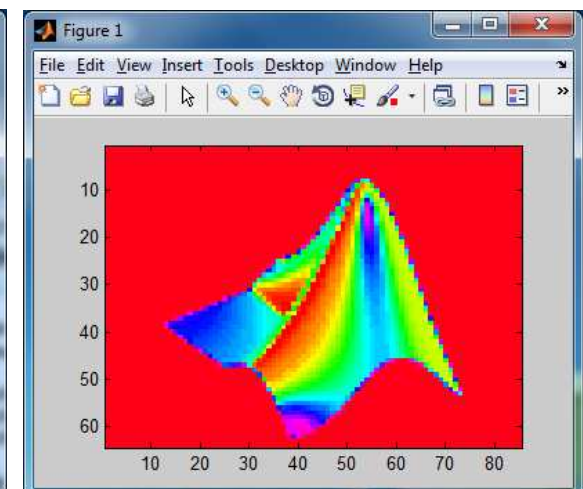
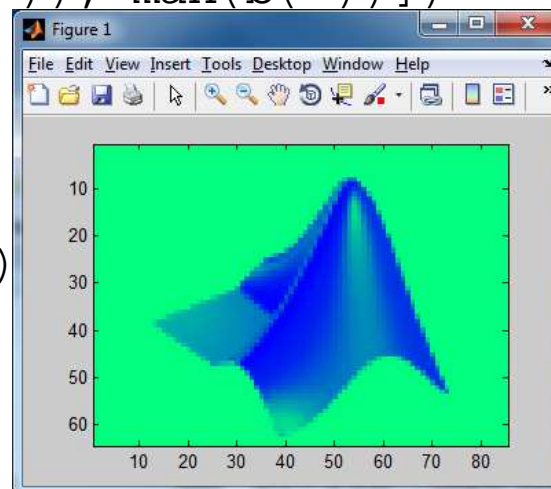
## Képtípusok a Matlabban

### Intenzitás

- A képmátrix adatai intenzitás értékeket jelentenek valamilyen tartományban
  - Ez lehet pl. [0 1], [0 255] stb.
- A megjelenítéshez az `imagesc` (image scale) függvény használható
  - Érdekes az alapértelmezett a colormap beállítást használni, de ez módosítható is

- Példa: a Matlab logó feldolgozása intenzitás képként

```
>> imagesc(b)
% nem kötelező megadni a tartományt, de lehet
>> imagesc(b, [0,255]) % vagy
>> imagesc(b, [min(b(:)), max(b(:))])
>> colormap(gray)
% colormap kísérletek
>> colormap(winter)
>> colormap(summer)
...
>> colormap('default')
```



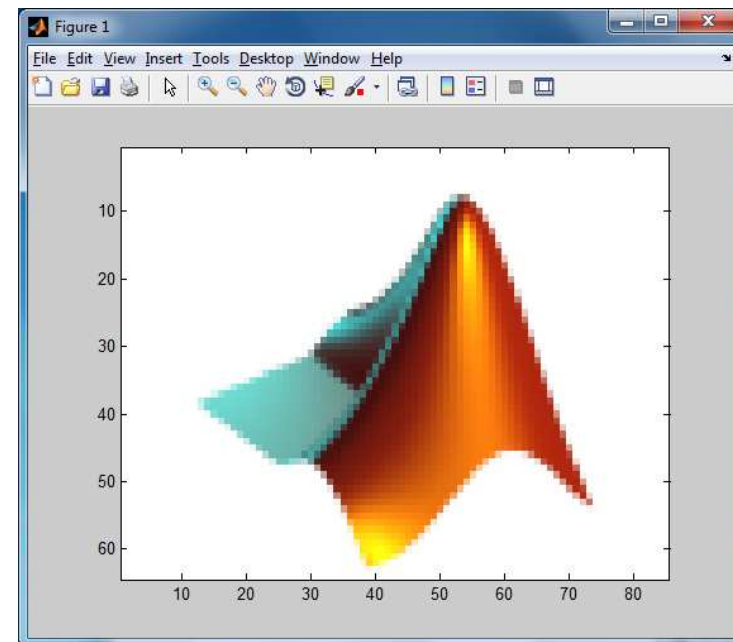


## Képtípusok a Matlabban

### RGB

- Truecolor kép,  $m \times n \times 3$ -as (esetleg  $m \times n \times 4$ -es; CMYK)
- Nincs külön colormap paletta, minden pixelre külön tárolódik az R, G és B összetevő
  - Eml.: a hagyományos megoldás szerint minden színre 8 bit jut
  - Pl. a kék összetevő előhívása a (10, 6) pontra: `tomb(10, 6, 3)`
- Ha double az alaptípus, akkor 0 és 1 közötti értékek tárolódnak
  - (0, 0, 0) – fekete, (1, 1, 1) – fehér
- Megjelenítés az `image` vagy az `imshow` paranccsal
  - (image-nél a map-ra figyelni kell, de RGB képeknél elvileg nem)
  - (Általában az `imagesc` is használható)
- Példa (Matlab logó)

```
>> image(i) % vagy
>> imshow(i)
% RGB összetevők előhívása
egy konkrét pontra
>> i(30, 40, 1:3)
```





## Képtípus jellemzők lekérdezése

- Általános lekérdezés: `imfinfo` parancs
  - Az eredmény egy lista, benne többek között: név, formátum, x és y méretek, colormap információk, színmélység
- Példák:

```
>> info = imfinfo('canoe.tif') % indexelt kép
>> imfinfo('ngc6543a.jpg')    % truecolor kép
```
- `img_size` függvény
  - Magasság (sorok), szélesség (oszlopok) és képcsatornák száma kérdezhető le vele
  - Szürkeárnyaltos képnél az utóbbi érték kiesik
- Példák

```
>> img_size = size(i)
% az előző Matlab logóval dolgozunk, i és a változók
>> h = img_size(1), v = img_size(2)
% sorok és oszlopok száma
>> n = img_size(3)
>> img_size = size(a)
% itt nincs színcsatorna
>> img = imread('ngc6543a.jpg', 'jpg');
>> size(img)
```

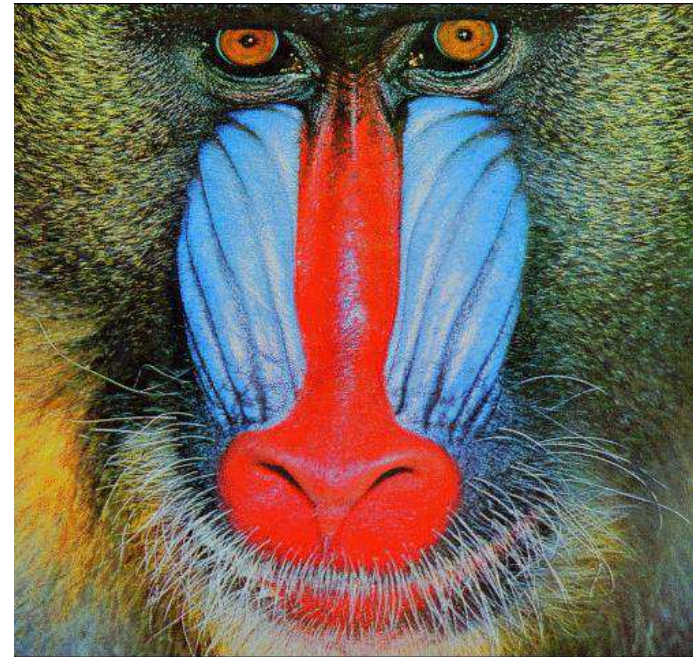


## Képmátrix fájlba írása

- `imwrite` függvény
  - Támogatott képtípusok: szürkeárnyaltos, true color, és az indexelt is
  - Támogatott fájlformátumok: GIF, TIFF, JPG stb.
- Szintaktika
  - `imwrite('fájlnev')`
  - `imwrite(A, 'fájlnev', 'formátum')`
  - `imwrite(X, map, 'fájlnev', 'formátum')`
  - (Használati eset példák az eddigiek szerint)
- Példa 1. (Súgó: `image`)

```
>> load mandrill
% X és map létrejön
>> figure('color','k') % új (fekete) kép létrehozása
>> image(X) % ábra megjelenítése
>> colormap(map) % színek helyesek
>> axis off % tengelyek levéve
>> axis image % skálázás négyzetesre
>> imwrite(X,map,'mandrill.jpg') % fájlba írás
```
- Példa 2.

```
>> load clown
>> imwrite(X,map,'clown.bmp')
```

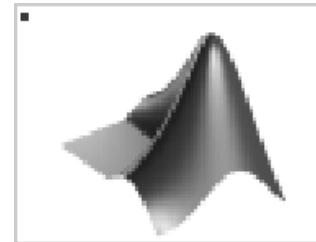


## Egyszerű módosítások

- Pontonként belenyúlunk a képbe
  - Az ismert mátrixos módon, de itt először az y, majd az x értéket kell megadni

- Példa

```
>> a(3:4,2:3) = 56;
% egyszerű ciklussal is lehetne
% (itt felesleges)
>> imshow(a)
```

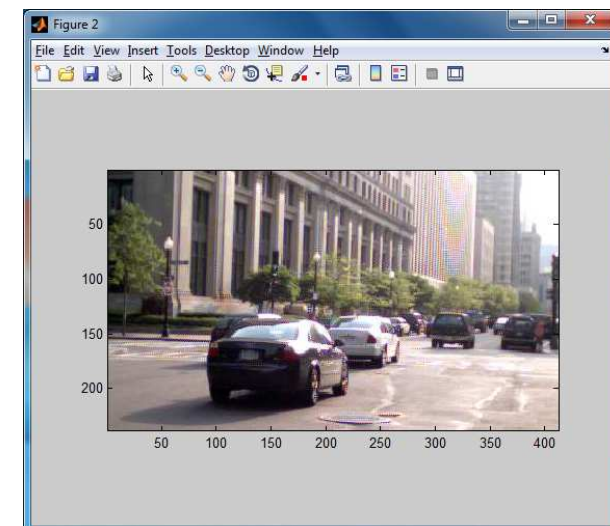


Workspace

Name	Value	Min	Max
MM	<238x412x3 uint8>	0	255
im	<480x640x3 uint8>	<T...	<T...
p	[107.4124, 77.5507; ...]	77...	517...
sp	[107, 77, 518, 314]	77	518

- Részkép kivágása

```
>> im = imread('street2.jpg');
>> image(im); axis image % megjelenítés,
% négyz. skálázás
>> p = ginput(2); % téglalap kijelölés (egér katt.)
>> sp(1) = min(floor(p(1)), floor(p(2)));
% megfelelő koordináták eltéve, xmin, ymin, xmax, ymax
>> sp(2) = min(floor(p(3)), floor(p(4)));
>> sp(3) = max(ceil(p(1)), ceil(p(2)));
>> sp(4) = max(ceil(p(3)), ceil(p(4)));
>> MM = im(sp(2):sp(4), sp(1):sp(3), :);
% új kép létrehozása
>> figure; image(MM); axis image
% új kép megjelenítése
>> imwrite(MM, 'street2_cropped.tif')
% és mentése
```



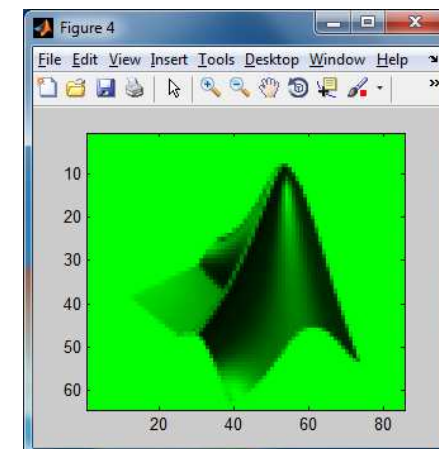
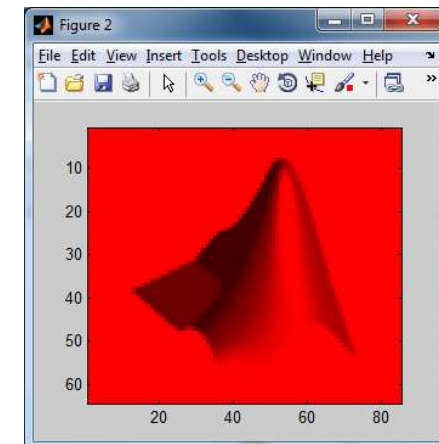
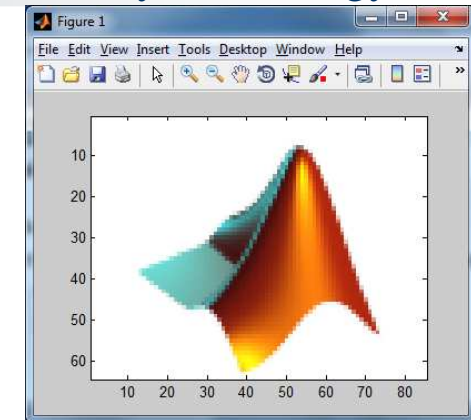


## Egyszerű módosítások

### Színcsatornák szétválasztása

- Vörös csatorna értékeinek kimentése új képre
  - Az új mátrixot kezdetben csupa nulla mátrixként hozzuk létre (szokásos ötlet, lásd később is)
  - Tdk. az `img_red` mátrixnak is három csatornája van, csak kettő üres
  - Egymásba ágyazott két ciklus helyett direkt mátrixműveletek is használhatók
  - Zöld és kék csatorna értékeinek kimentése: hasonlóan

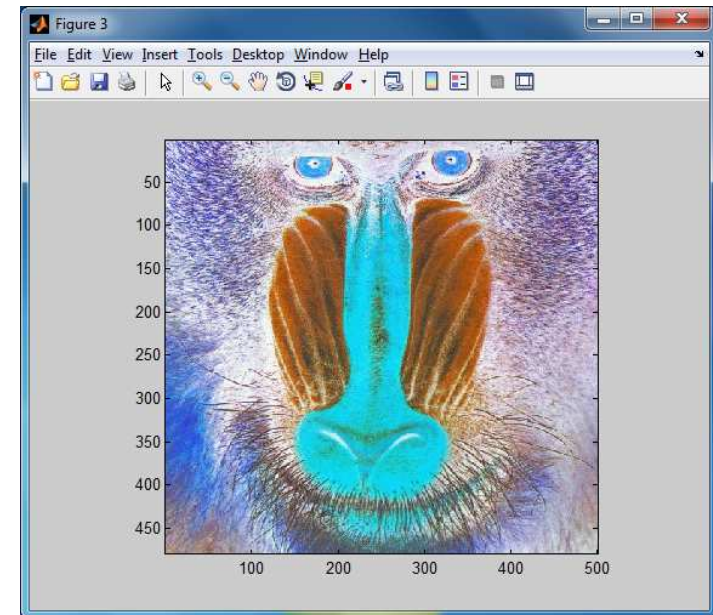
```
Editor - D:\Oktatás\Matlab-8-9\kepfeld-segedfajlok\voros_csar.m
File Edit Text Go Cell Tools Debug Desktop Window Help
img = imread('logo.png');
% egyszerű Matlab logo uint8 mátrixba olvasása
figure(1); image(img);
% a színes kép megjelenítése az 1. ablakban
img_size = size(img);
% méretjellemzők
width = img_size(1);
height = img_size(2);
img_red = zeros(img_size, 'uint8');
% megfelelő méretű új csupa 0 mátrix létrehozása
for y=1:height, for x=1:width, img_red(x,y,1) = img(x,y,1); end; end;
% img_red(:, :, 1) = img(:, :, 1); is használható lenne
% a piros csatorna értékei kimentve
figure(2); image(img_red);
% és megjelenítve
script Ln 15 Col 18 OVR...
```



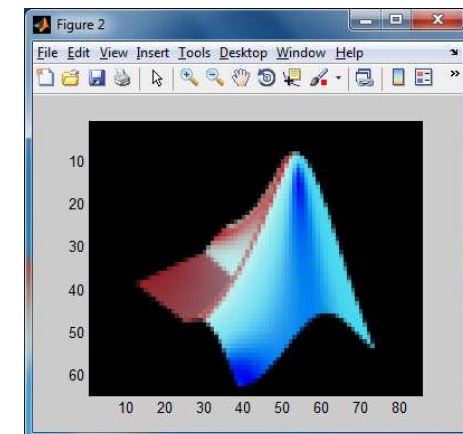
## Egyszerű módosítások

### Kép invertálása

- Három színcsatornás képmátrix inverzének előállítására függvényt készítünk
  - A függvény input paramétere a mátrix
- Az inverz értékeket csatornánként számoljuk ki
- Hasonlóan készíthető függvény szürkeárnyaltos kép invertálására is
- Hívás:  
`>> inv = myinverse(img);`



```
Editor - D:\Oktatás\Matlab-8-9\kepfeld-segedfajlok\myinverse.m
File Edit Text Go Cell Tools Debug Desktop Window Help
1 function [ inverse_image ] = myinverse( image )
2 % függvény képmátrix invertálása
3     dims = size(image);
4     inverse_image = zeros(dims, 'uint8');
5 % üres inverz képmátrix létrehozása
6     inverse_image(:,:,:) = 255 - image(:,:,:);
7 % invertálás, az összes színcsatornára
8 end
myinverse Ln 8 Col 4 OVR
```





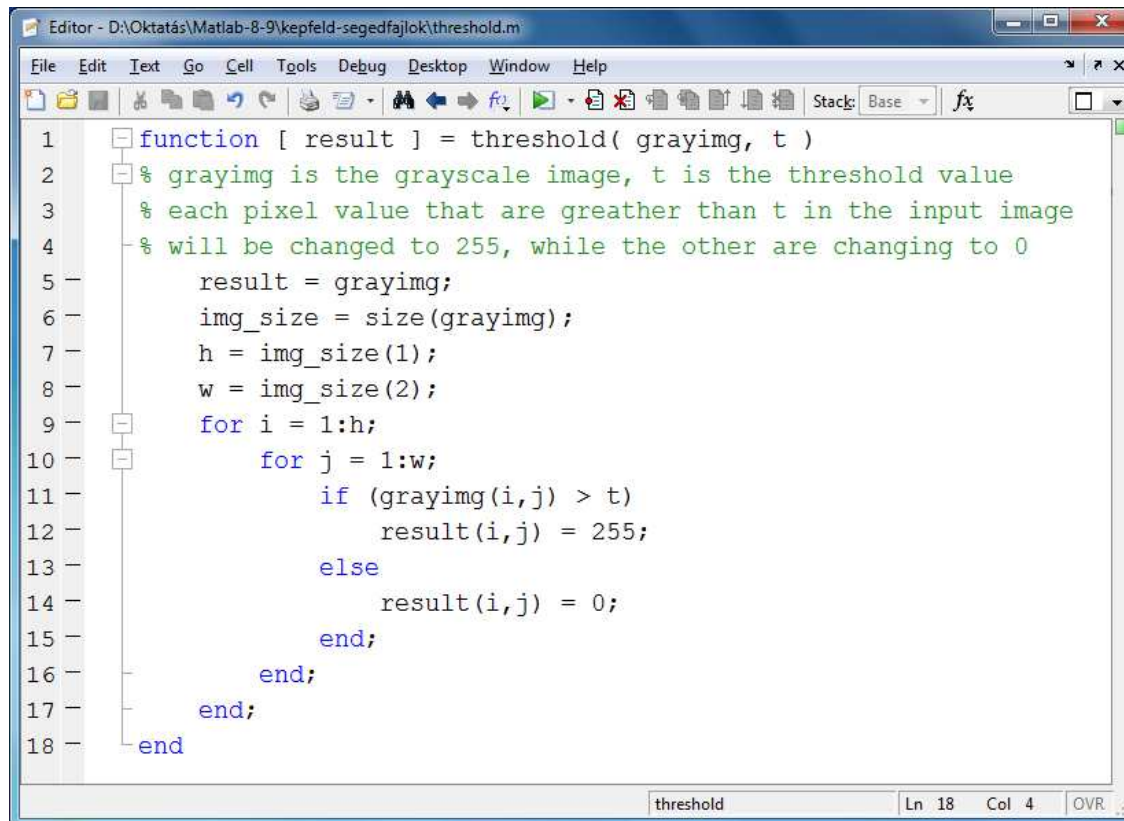
## Egyszerű módosítások

Klasszikus vágási feladat (szürkeárnyaltos képre)

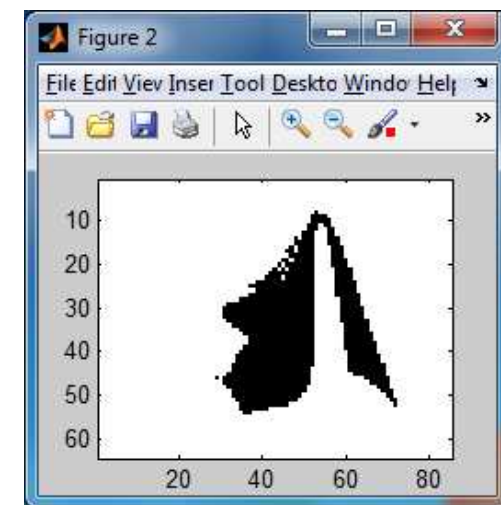
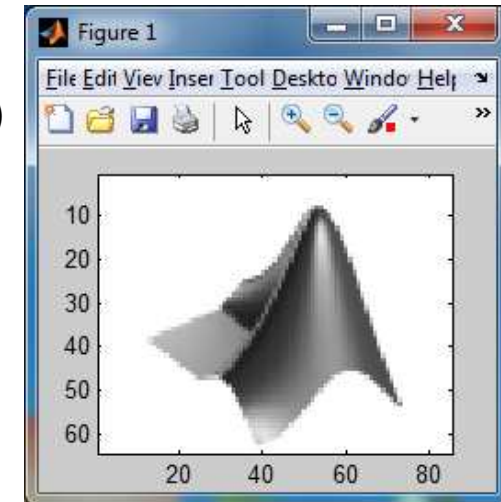
- Ha a színérték bizonyos korlát felett van, akkor fehér lesz, különben fekete (kontraszt szélsőséges kiemelése)
  - Szintén függvénnel oldjuk meg

- Hívás (pl.):

```
>> result = threshold(a,128);
```



```
1 function [ result ] = threshold( grayimg, t )
2 % grayimg is the grayscale image, t is the threshold value
3 % each pixel value that are greater than t in the input image
4 % will be changed to 255, while the other are changing to 0
5 result = grayimg;
6 img_size = size(grayimg);
7 h = img_size(1);
8 w = img_size(2);
9 for i = 1:h;
10     for j = 1:w;
11         if (grayimg(i,j) > t)
12             result(i,j) = 255;
13         else
14             result(i,j) = 0;
15         end;
16     end;
17 end;
18 end
```





## Szűrés

### Elméleti áttekintés

- A képfeldolgozás során általában képpontok csoportjaival dolgozunk (itt is)
- Egyszerű módszer: pixel értékek súlyozott összegét használjuk
  - Különböző súlyokkal kül. célokat lehet így elérni
  - Pl. simítani vagy élesíteni lehet a képet, zajt lehet eltávolítani, éleket lehet kiemelni
- Modell
  - Készítsünk egy ugyanakkora mátrixot, mint az eredeti kép
  - Írjuk minden pozícióra az adott képpontot körülvevő (eredeti) pixelek súlyozott összegét
  - A súlyozás – kezdetben, az első megoldásnál – minden képpont esetén ugyanolyan legyen (lineáris szűrő, dobozszűrő)
- Konvolúció: folyamat, amelynek során ezt az eljárást egy képre alkalmazzuk
  - Kernel: az a minta, amelyből a súlyozott összeg készül (pl. 9 db pont, az eredeti + 8 db szomszéd)
- Gond adódik a kép szélein (hiányoznak a szomszédok)
  - Néhány lehetséges megoldás:  
Levágjuk a kép széleit; 0-val (fekete értékek), szürke értékekkel, vagy a legközelebbi még létező pont(ok) értékeivel töltjük fel a hiányt

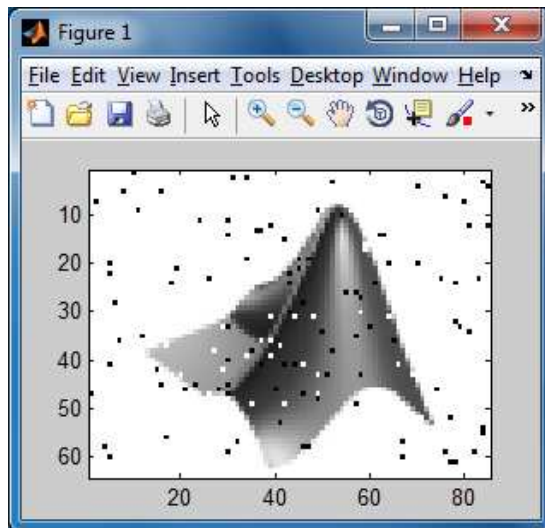


## Szűrés

### Előkészítés

- „Só és bors” zajt készítünk
  - Fekete vagy fehér (hibás) képpontok
- Ötlet: két ciklussal végigmegyünk a képfájlon, és véletlent használva „megszórjuk”
- Paraméterezhető függvény, megadhatjuk a zaj intenzitását
- Hívás példa

```
>> k=saltandpep(a, 0.96);  
>> image(k)
```

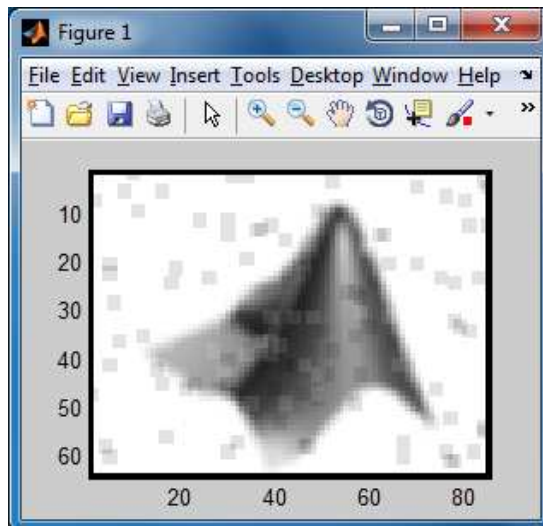


```
Editor - D:\Oktatás\Matlab-8-9\kepfeld-segedfajlok\saltandpep.m
File Edit Text Go Cell Tools Debug Desktop Window Help
1 function [k] = saltandpep(img, l)
2 % só és bors zaj készítése
3 % img a mátrix, l a valószínűség
4 k = img;
5 for i = 1:size(k, 1)
6     for j = 1:size(k, 2)
7         r = rand;
8         if r >= l
9             e = randi([0, 1]);
10             % 0 vagy 1
11             % fekete vagy fehér zaj
12             k(i, j) = e*255;
13         end
14     end
15 end
16 end
saltandpep Ln 7 Col 22 OVR
```

## Szűrés

Lineáris szűrő (dobozszűrő) – megvalósítás

- Függvényt írunk a feladatra, ennek paramétere a simítandó képmátrix
- Az új mátrix kezdetben csupa nulla elemből áll
- A pontozott szorzást használjuk egy megfelelő részmátrixra és a kernelre
- A szélső 1-1 sor és oszlop fekete
- Hívás példa  
 >> l = simit(k);  
 >> image(l)



```

Editor - D:\Oktatas\Matlab-8-9\kepfeld-segedfajlok\simit.m
File Edit Text Go Cell Tools Debug Desktop Window Help
1 function [ ujM ] = simit( A )
2     % 9 pontos simító fv.
3     x=size(A, 1);
4     y=size(A, 2);
5     ujM = zeros(x, y);
6     Amas = double(A);
7     % új csupanulla mátrix
8     kernel=[1 1 1; 1 1 1; 1 1 1];
9     % 9 pontos kernel
10    for i=2:x-1
11        for j=2:y-1
12            Aresz = Amas(i-1:i+1, j-1:j+1);
13            % megfelelő részmátrix kivétele
14            ujM(i, j) = sum(sum(Aresz.*kernel))/9;
15            % simított képpontok előállítása
16            % pontozott szorzás kell!
17        end
18    end
19    ujM = uint8(ujM);
20    % double mátrixszal számolunk, de az eredmény uint8
21 end
simit Ln 1 Col 30 OVR

```



## Szűrés

### Szűrőfajták (átlagszűrők)

- A szűrők mérete leggyakrabban páratlan, mert így lehet egyértelműen meghatározni a középpontpixelt
- Tipikus megoldás, hogy a súlyok csökkennek a középponttól mért távolsággal
  - A középponttól távolabb levő képelemek kisebb hatással vannak az eredményre
- 3×3-as átlagszűrők
  - A normálótényező mindig a maszkelemek összege
  - A bal oldali az egyszerű dobozszűrő

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \frac{1}{6} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \frac{1}{8} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

- 5×5-ös átlagszűrők

$$\frac{1}{81} \begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 4 & 6 & 4 & 2 \\ 3 & 6 & 9 & 6 & 3 \\ 2 & 4 & 6 & 4 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix} \quad \frac{1}{100} \begin{bmatrix} 0 & 3 & 4 & 3 & 0 \\ 3 & 6 & 7 & 6 & 3 \\ 4 & 7 & 8 & 7 & 4 \\ 3 & 6 & 7 & 6 & 3 \\ 0 & 3 & 4 & 3 & 0 \end{bmatrix}$$

- Az átlagszűrő mindig elmosza az éleket!



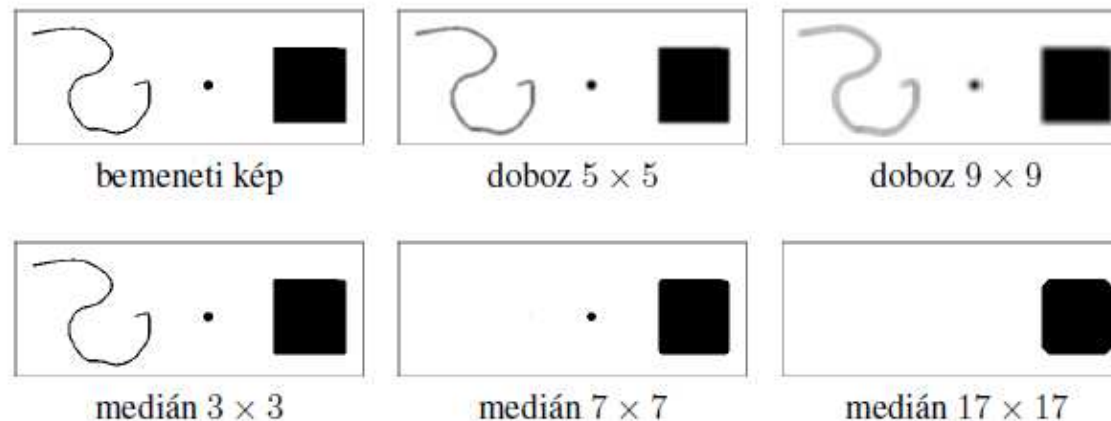




## Szűrés

Szűrőfajták folyt. – mediánszűrő

- A mediánszűrő eredménye a szűrési ablakban levő értékek mediánja
- Rendezzük az értékeket, és kiválasztjuk a megfelelőt
  - Ez tekinthető úgy is, mint egy „többségi szavazás”, ahol a szélső értékek kiesnek
- Főbb tulajdonságok
  - A „só és bors” zaj eltávolítására kiválóan alkalmas
  - Nem mossa el az éleket és nem csökkenti a kontrasztot
  - Az igen vékony vonalakat törli (ha a vonalvastagság kisebb, mint a szűrőméret fele)
  - Lekerekíti a sarkokat (fehér háttér előtti objektumoknál)
- Példa: dobozszűrő és mediánszűrő alkalmazása





## Éldetektálás

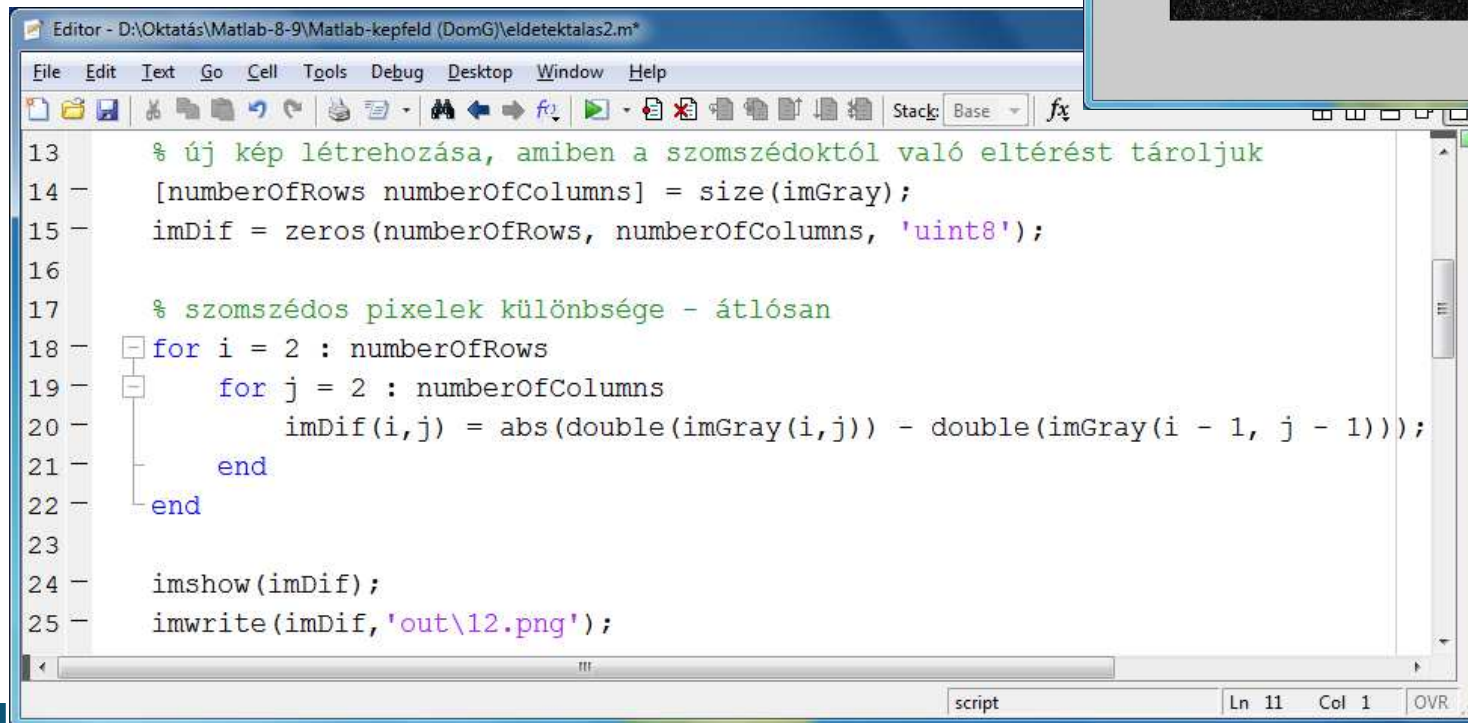
- Első összetettebb feladat: éldetektálás
  - Nagyobb részben már ismert apparátussal dolgozunk
  - 1. lépés: betöltés
  - 2. lépés: szürkeárnyaltos konverzió  
(itt Image Processing Toolbox fv.)
    - Enélkül is meg tudjuk oldani, lásd bev. slide,  
tipikus súlyozás:  $\text{Gray} = 0,33 \cdot R + 0,56 \cdot G + 0,11 \cdot B$

```
Editor - D:\Oktatás\Matlab-8-9\Matlab-kepfeld (DomG)\eldetektalas2.m*
File Edit Text Go Cell Tools Debug Desktop Window Help
1 % Élek detektálása képen
2
3 % kép betöltése
4 im = imread('horse.jpg');
5 image(im);
6 imwrite(im, 'out\10.png');
7
8 % kép átalakítása szürkeárnyaltossá
9 imGray = rgb2gray(im);
10 imshow(imGray);
11 imwrite(imGray, 'out\11.png');
12
script Ln 14 Col 1 OVR
```

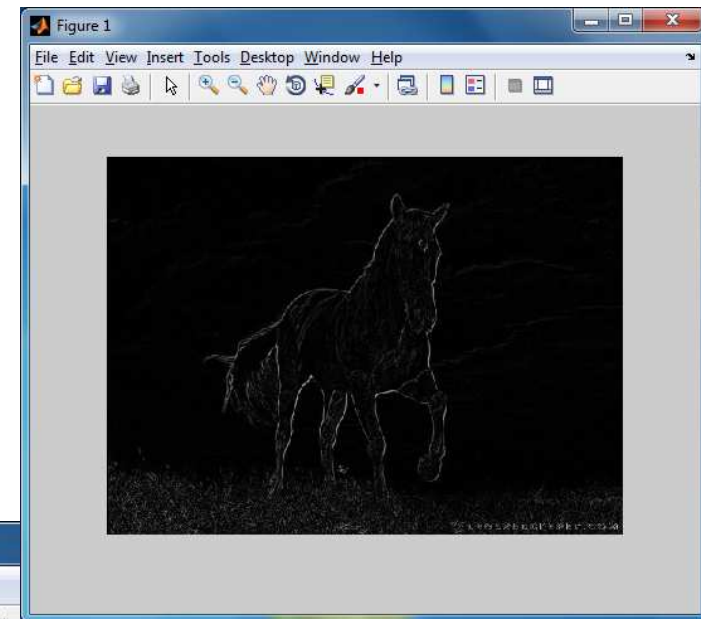


## Éldetektálás

- Mo. folyt.
  - 3. lépés: szomszédoktól való eltérésmátrix létrehozása  
(Az egyszerűség kedvéért csak a bal felső szomszédot nézzük)
    - A közbülső érték negatív is lehet! (Konverzió kell)



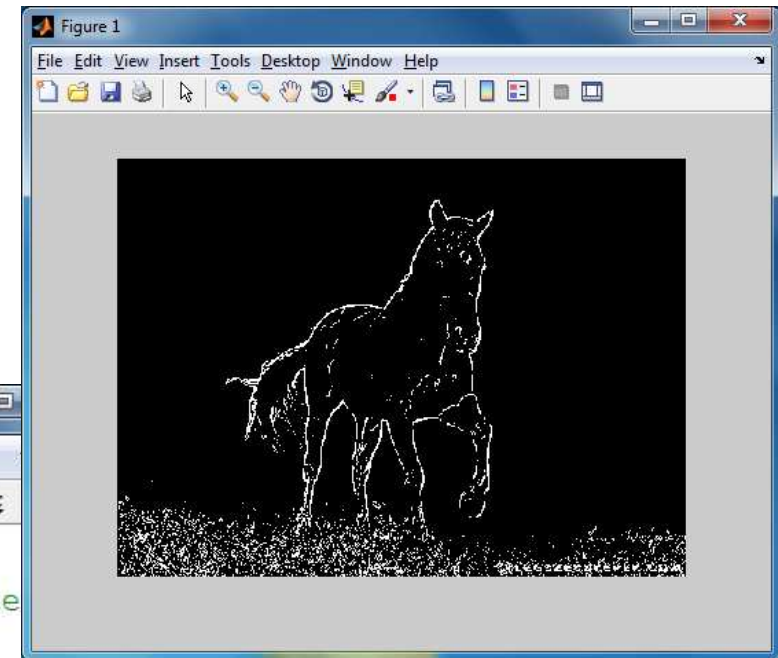
```
Editor - D:\Oktatás\Matlab-8-9\Matlab-kepfeld (DomG)\eldetektalas2.m*
File Edit Text Go Cell Tools Debug Desktop Window Help
13 % új kép létrehozása, amiben a szomszédoktól való eltérést tároljuk
14 [numberOfRows numberOfColumns] = size(imGray);
15 imDif = zeros(numberOfRows, numberOfColumns, 'uint8');
16
17 % szomszédos pixelek különbsége - átlósan
18 for i = 2 : numberOfRows
19     for j = 2 : numberOfColumns
20         imDif(i,j) = abs(double(imGray(i,j)) - double(imGray(i - 1, j - 1)));
21     end
22 end
23
24 imshow(imDif);
25 imwrite(imDif, 'out\12.png');
```



## Éldetektálás

- Mo. folyt.
- 4. lépés: vágás (a már ismert módon),  
a korlát most 30

```
Editor - D:\Oktatás\Matlab-8-9\Matlab-kepfeld (DomG)\eldetektalas2.m*
File Edit Text Go Cell Tools Debug Desktop Window Help
[Icons] fx
27 % vágás: ha az érték egy bizonyos korlát [cut]
28 % fölött van, akkor fehér, ha alatta, akkor fekete
29 cut = 30;
30 for i = 2 : numberOfRows
31     for j = 2 : numberOfColumns
32         if imDif(i,j) > cut
33             imDif(i,j) = 255;
34         else
35             imDif(i,j) = 0;
36         end
37     end
38 end
39
40 imshow(imDif);
41 imwrite(imDif, 'out\13.png');
42
script Ln 43 Col 1 OVR
```

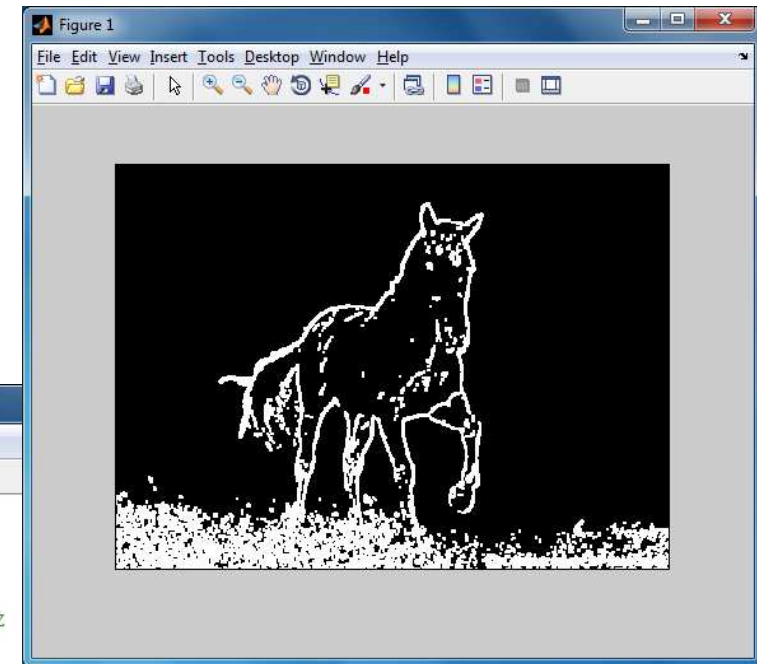


## Éldetektálás

- Mo. folyt.
- 5. lépés: élvastagítás, és végül a kimeneti mátrix elkészítése

```
Editor - D:\Oktatás\Matlab-8-9\Matlab-kepfeld (DomG)\eldetektalas2.m*
File Edit Text Go Cell Tools Debug Desktop Window Help
[Icons] Stack: Base fx

43 - imOut = zeros(numberOfRows, numberOfColumns, 'uint8');
44 - % kimeneti mátrix
45 -
46 - % élek vastagítása: ahol valamelyik szomszéd fehér, az fehér lesz
47 - for i = 2 : numberOfRows - 1
48 -     for j = 2 : numberOfColumns - 1
49 -         if (imDif(i - 1, j - 1) == 255 || imDif(i - 1, j) == 255 ...
50 -             || imDif(i - 1, j + 1) == 255 || imDif(i, j - 1) == 255 ...
51 -             || imDif(i, j) == 255 || imDif(i, j + 1) == 255 ...
52 -             || imDif(i + 1, j - 1) == 255 || imDif(i + 1, j) == 255 || ...
53 -             imDif(i + 1, j + 1) == 255)
54 -             imOut(i, j) = 255;
55 -         else
56 -             imOut(i, j) = 0;
57 -         end
58 -     end
59 - end
60 -
61 - imshow(imOut);
62 - imwrite(imOut, 'out\14.png');
63 -
```



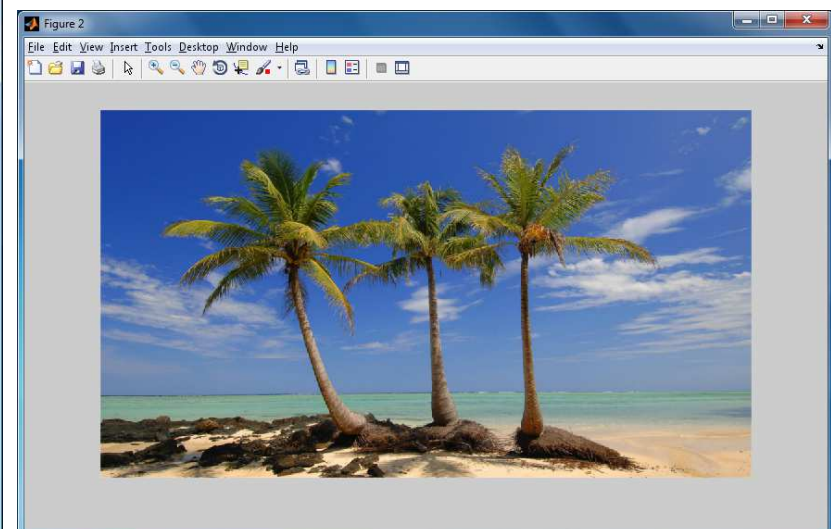
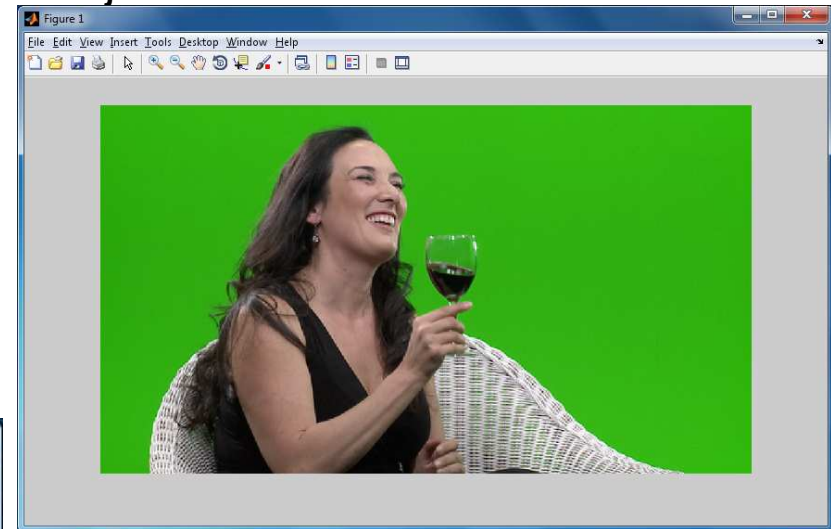


## Kép háttérének cseréje

Második összetettebb feladat: kép háttérének cseréje

- Egy stúdióban készült kép (zöld) háttérét le kell cserélnünk valamely adott másik háttérre
- Szintén az ismert apparátussal dolgozunk
- 1. lépés: képek betöltése

```
Editor - D:\Oktatás\Matlab-8-9\Matlab-kepfeld (DomG)\studio.m*
File Edit Text Go Cell Tools Debug Desktop Window Help
1 % stúdióban felvett jelenet zöld háttérének
2 % lecserélése tetszőleges háttérre
3
4 % stúdióban felvett kép betöltése
5 - im = imread('borozos.png');
6 - imshow(im);
7 - imwrite(im, 'out\20.png');
8
9 % háttér betöltése
10 - im2 = imread('island.png');
11 - imshow(im2);
12 - imwrite(im2, 'out\21.png');
13
script Ln 14 Col 15 OVR ...
```



## Kép háttérének cseréje

- Mo. folyt.
- 2. lépés: RGB komponensek kivétele
- 3. lépés: zöld háttér átörökítése  
(Mi a zöld? Vá.: Intenzív „g” érték)

```
Editor - D:\Oktatás\Matlab-8-9\Matlab-kepfeld (DomG)\studio.m*
File Edit Text Go Cell Tools Debug Desktop Window Help
26 - if numberOfRows == numberOfRows2 && numberOfColumns == numberOfColumns2
27 -
28 -     % szomszédos pixelek különbsége - átlósan
29 -     for i = 1 : numberOfRows
30 -         for j = 1 : numberOfColumns
31 -             % Az számít zöldnek, ahol a g nagyobb 10-el mint az r vagy a b
32 -             if (g(i,j) > (r(i,j) + 10) && g(i,j) > (b(i,j) + 10))
33 -                 r(i,j) = r2(i,j);
34 -                 g(i,j) = g2(i,j);
35 -                 b(i,j) = b2(i,j);
36 -             end
37 -         end
38 -     end
39 -
40 -     im(:,:,1) = r;
41 -     im(:,:,2) = g;
42 -     im(:,:,3) = b;
43 -
44 -     imshow(im);
45 -     imwrite(im, 'out\22.png');
46 - end
script Ln 24 Col 21 OVR
```

```
Editor - D:\Oktatás\Matlab-8-9\Matlab-kepfeld (DomG)\studio.m*
File Edit Text Go Cell Tools Debug Desktop Window Help
14 - r = im(:,:,1);
15 - g = im(:,:,2);
16 - b = im(:,:,3);
17 -
18 - r2 = im2(:,:,1);
19 - g2 = im2(:,:,2);
20 - b2 = im2(:,:,3);
21 -
22 - % A két kép mérete
23 - [numberOfRows numberOfColumns] = size(r);
24 - [numberOfRows2 numberOfColumns2] = size(r2);
script Ln 14 Col 15 OVR
```



## Kép háttérének cseréje

- Eredmény
  - Apró hibák, az eljárás még finomítható

