Képfeldolgozás MATLAB-ban

A Matlab rendszert hatékonysága miatt számos területen használják. Segíti a gyors fejlesztést, hatékonyan vannak benne implementálva a nagy számításigényű algoritmusok. A Matlabhoz készült jónéhány olyan programcsomag, amelyben az alkalmazott informatika egyes területén közismertté vált algoritmusokat, számításokat, fejlesztési eszközöket gyűjtötték össze. Az egyik ilyen programcsomag az ***Image Processing Toolbox***.

A következőkben azt tekintjük át, hogyan lehet egyszerű képműveleteket végezni Matlabban.

Képek megnyitása és mentése:

MATLAB-ban a képek megnyitására az `imread(...)` függvény szolgál. A függvénynek több paramétere is megadható, ezt most nem részletezzük. Hagyományos képformátumoknál nem szükséges megadni a formátumot paraméterként, azonban speciális képeknél, mint például a DICOM formátum, ezt meg kell adni. (A DICOM képek sokszor egy képsorozatból állnak, minden szelet egy külön fájlban van, a szeletek sorrendjét a fájlnév alapján lehet meghatározni.)
Az `imread(...)` függvény egy mátrixban tárolja el a képeket.

```
img = imread('fájlnév')
```

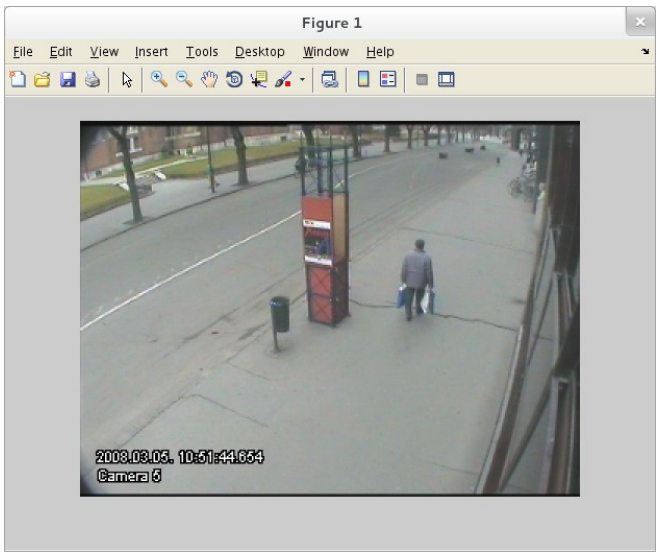
Egyes paraméterezésnél a színtérkép (color map) is lekérhető.

```
[img, map] = imread(URL, ...);
```

A képeket az imshow(...) függvénnyel lehet megjeleníteni.

Példa:

```
img = imread('cam1.jpg');
imshow(img);
```



Figyeljük meg, hogy a betöltött kép egy 360×480×3 méretű, `uint8` típusú mátrix. Vagyis a sorok száma (a kép magassága) 360 pixel, az oszlopok száma (a kép szélessége) 480 pixel, és a kép három csatornát tartalmaz.

A képek mentésére az `imwrite(...)` függvény szolgál. A függvényt a következőképpen lehet meghívni (sokféle változat, paraméterezés létezik).

```
imwrite(img, 'fájlnév')
imwrite(img, 'fájlnév', 'formátum')
```

Pixelek elérése, kezelése

Csakúgy, mintha „hagyományos” mátrixokat kezelnénk MATLAB-ban, a képeknél is így tudjuk elérni a pixeleket. Figyelnünk kell azonban arra, hogy hagyományos képpont eléréshez képest (amely szerint a kép koordinátarendszerének középpontja a kép bal felső sarkában található, az x-tengely balról jobbra, az y-tengely felülről lefelé nő) meg kell cserélni a koordinátákat (vagyis először az y-koordinátát kell megadni, mint sor indexet, utána pedig az x-koordinátát kell megadni, mint oszlop indexet). A harmadik dimenzió a képcsatornára vonatkozik (RGB kép esetén az vörös, a zöld és a kék színcsatorna).

A képmátrix szélességét és magassát, valamint a csatornák számát a size(...) függvénnyel kérhetjük le. Az eredményül kapott vektor első komponense a sorok száma, a második az oszlopok száma, a harmadik pedig a csatornák száma. Tehát, ha az img változó tartalmazza a képmátrixot, akkor

```
>> img_size = size(img);
>> w = img_size(2);
>> h = img_size(1);
```

Készítsünk szürkeárnyalatos képet egy RGB képből! Emlékeztetőül, a szürkeárnyalatok meghatározásához az R,G és B színcsatornák lineáris kombinációját kell venni, ahol a legyakrabban javasolt súlyozás:

$$Gray = 0.33 * R + 0.56 * G + 0.11 * B$$

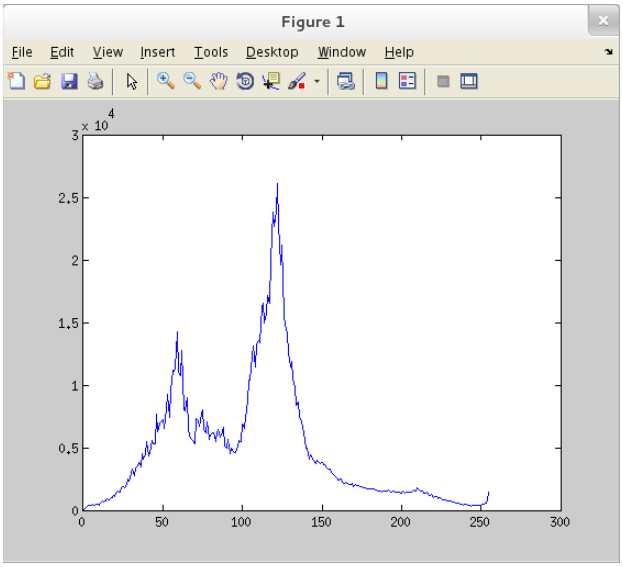
```
>> szeder_gray = zeros(h,w);
>> for x=1:w; for y= 1:h; szeder_gray(y,x)=
   uint8(0.33*szeder(y,x,1))+uint8(0.56*szeder(y,x,2))+
   uint8(0.11*szeder(y,x,3)); end; end;
>> imshow(uint8(szeder_gray));
```

Természetesen ez az alap képfeldolgozó művelet nem hiányzik az Image Processing Toolbox függvényei közül. Az `rgb2gray(...)` beépített függvény segítségével konvertálhatjuk

szürkeárnyalatossá RGB képeinket.

Készítsük el az imént létrehozott szürkeárnyalatos kép hisztogramját!

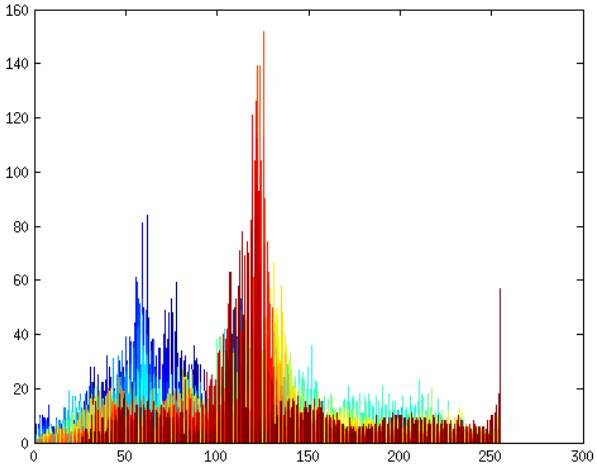
```
>> histogram2 = zeros(256,2);
>> for i =0:255; histogram2(i+1,1) = i; end;
>> for x=1:w; for y = 1:h; histogram2(szeder_gray(y,x)+1,2) =
   histogram2(szeder_gray(y,x)+1,2) +1; end; end
>> plot(histogram2(:,1), histogram2(:,2))
```



A hisztogram számítás is szerepel a függvények között. A hisztogram oszlopdiagrammal történő megjelenítését a

```
>> [n,xout] = hist(szeder_gray,256);
>> bar(xout,n);
```

függvényekkel végezhetjük el.



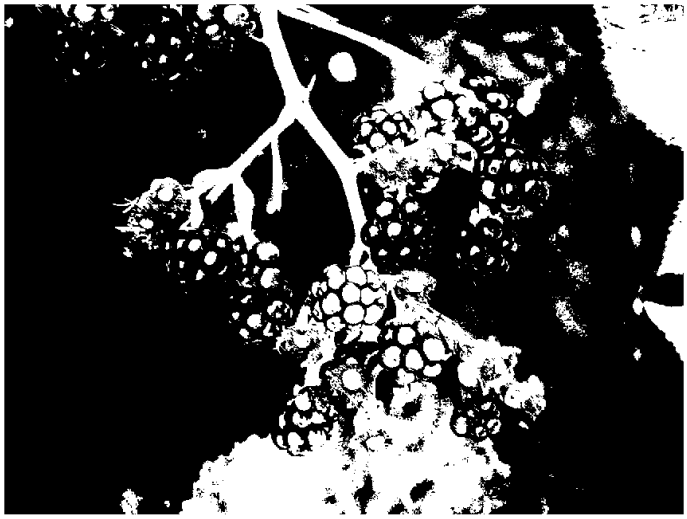
Hajtsunk végre egy egyszerű küszöbölést a *szeder.jpg*-ből készített képen. Készítsünk ehhez egy függvényt!

```
function result = mythreshold(grayimg, t)
    % grayimg is the grayscale image
    % t is the threshold value
    % each pixel value that are greather than t in the input image
    % will be changed to 1, while the other are changing to 0

    result = grayimg;
    img_size = size(grayimg);
    h = img_size(1);
    w = img_size(2);
    for i = 1:h;
        for j = 1:w;
            if ( grayimg(i,j) > t)
                result(i,j) = 1;
            else
                result(i,j) = 0;
            end;
        end;
    end;
end
```

A függvény meghívása:

```
>> thres_gray = mythreshold(szeder_gray, 128);
```



Feladat:

1. Készíts függvényt egy paraméterként kapott RGB kép szürkeárnyalatossá konvertálására!
2. Készíts függvényt a hisztogram számítására! Kiegészítés: készíts függvényt, amely egy RGB kép hisztogramját számítja ki mindhárom csatornára.