ლაბორატორია 9

სურათების დამუშავება

1 მოკლე შესავალი სურათების დამუშავებაში

სურათი ორ-განზომილებიანი ფუნქციაა $f(x, y)$, სადაც x და y სიბრტყის კოორდინატებია, და ნებისმიერ კოორდინატთა წყვილზე (x, y) არსებული ამპლიტუდა (ფუნქციის მნიშვნელობა), სურათის ინტენსიურობაა.

თუ x, y კოორდინატების და ამპლიტუდის მნიშვნელობები სასრული და დისკრეტული სიდიდეებია, ასეთ სურათს ციფრული სურათი ეწოდება. ციფრული სურათის შემადგენელი უჯრედი პიქსელია. თითოეულ პიქსელს გააჩნია მდებარეობა და ამპლიტუდა. ციფრული სურათების დამუშავება ძალიან ფართო სფეროა, რომელიც ნებისმიერი ტიპის ციფრული სურათის ციფრული სისტემით (მაგალითად ციფრული კომპიუტერით) დამუშავებას შეისწავლის.

1.1 სურათების დამუშავების ალგორითმები

სურათების დამუშავების ალგორითმების რამდენიმე ფართო კლასად დაყოფაა შესაძლებელი. სხვადასხვა ალგორითმი სხვდასხვა მიზანს ემსახურება და სხვდასხვა სამუშაოს ასრულებს.

1. სურათების გაუმჯობესება.
    a. გამკვეთრება, სურათის ფოკუსში მოყვანა, ზღვრების გამკვეთრება.
    b. სურათის კონტრასტის ან განათების გაუმჯობესება.
    c. ხმაურის გაფილტრვა.
2. სურათის აღდგენა.
    a. წრფივი მოძრაობისას გამოწვეული სურათი ბლარის მოშორება.
    b. ოპტიკური დისტორციის მოშორება.
    c. პერიოდული ინტერფერენციის მოშორება.
3. სურათის სეგმეტაცია.
    a. გეომეტრიული ფიგურების ამოცნობა.
    b. ობიექტების (მაქნანების, შენობების, გზის, ხეების) დეტექცია.

1.2 ციფრული სურათების ტიპები

არსებუბს ციფრული სურათის მრავალი ტიპი. ამჯერად სამ მათგანს განვიხილავთ:

1. ბინარული სურათი. სურათის თითოეული პიქსელი ან შავი ან თეთრია. შესაბამისად, თითოეული პიქსელის მხოლოდ ორი მნიშვნელობა არსებობს (0 ან 1) და ერთი პიქსელი ერთ ბიტ ინფორმაციას შეიცავს. ასეთი სურათის შენახვა მეხსიერების მხრივ მომგებიანია. მაგალითად, თუ გვჭირდება ტექსტური ფაილის ან თითის ანაბეჭდის შენახვა.

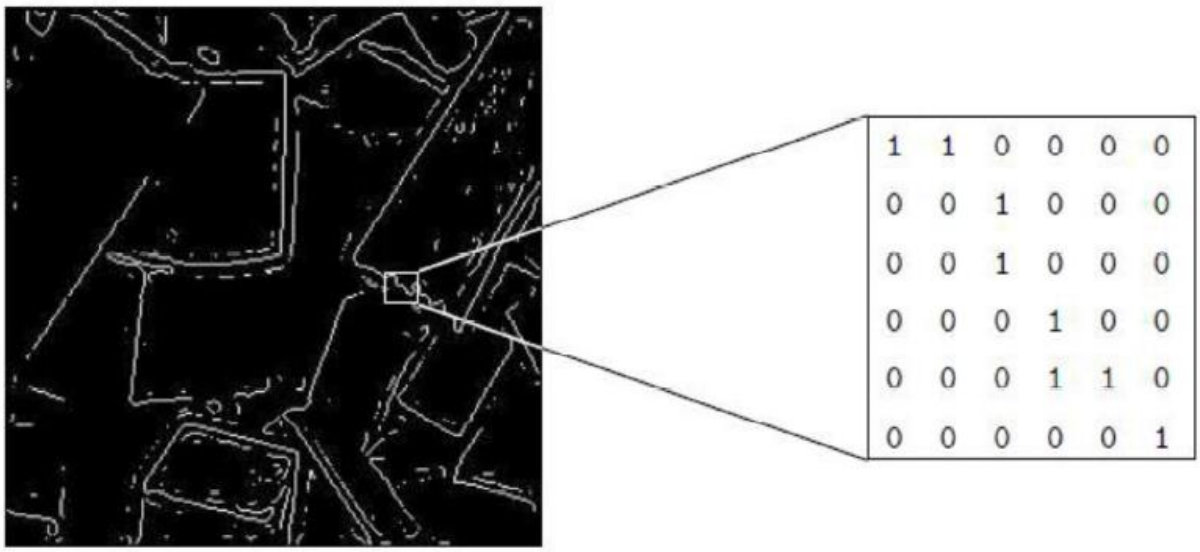
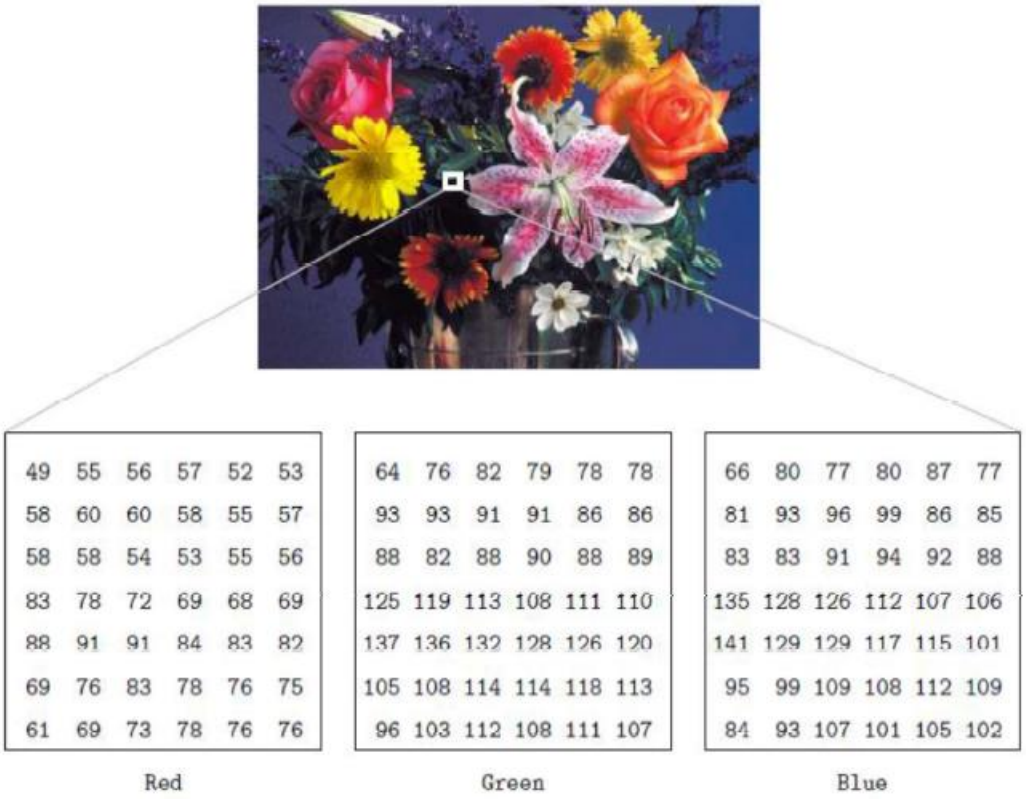


Figure 1 ბინარული სურათი

2. Grey scale სურათი: თითოეული პიქსელი სხვადასხვა ნაცრისფერის სხვადასხვა ინტენსიობა გააჩნია (შავიდან თეთრამდე). თუ გვინდა, რომ სურათის ინტენსიობა 256 მნიშვნელობისგან შედგებოდეს, ასეთი სურათის თითოეული პიქსელი 8 ბიტისგან (ერთი ბაიტისგან) შედგება. ასეთი სურათები ხშირად გვხდება სამედიცინო სფეროში (რენტგენი). ასევე, გამოიყენება ბუნებრივი ობიექტების ამოცნობისთვის.

3. ფერადი სურათი, ანუ RGB. ასეთ სურათში ყველა პიქსელს ფერი გააჩნია, რომელიც აღიწერება პიქსელში წითელი, მწვანე და ლურჯის მეშვეობით. თუ

თითოეულ ფერის კომპონენტს (RGB) 0-255 ინტენსიობის მნიშვნელობა გააჩნია, ეს იძლევა $255^3 = 16{,}777{,}216$ სხვადასხვა ფერს. ამ შემთხვევაში პიქსელი 24 ბიტია.
ასეთი სურათი შეგვიძლია სამი მატრიცის წყობად წარმოვადგინოთ; თითოეული მათგანი წითლის, მწვანის და ლუქრჯის ინტენსიობას ინახავენ.



1.3 ციფრული სურათები MATLAB-ში

MATLAB-ში სურათების წაკითხვა imread ბრძანების გამოყენებითაა შესაძლებელი. მაგალითად გამოიძახეთ ბრძანება w=imread('cameraman.tif'); ეს ბრძანება იღებს gray-scale სურათის პიქსელებს და მარტრიცა w-ში წარმოადგენს.

ამ მატრიცის სურათად გამოსახვა შეგიძლიათ ბრძანებით:

figure

imshow(w)

impixelinfo

ბოლო ბრძანებას (impixelinfo) გრაფიკზე გამოაქვს ინფორმაცია პიქსელის ლოკაციის და ინტენსიობის შესახებ.

1.4 ციფრული სურათების წარმოდგენა

როგორც ვნახეთ x მატრიცის სურათად წარმოდგენა imshow(x) ბრძანებით არის შესაძლებელი. ხშირ შემთხვევაში მატლაბის ბრძანებების შედეგად მიღებული აუთფუთი არის მატრიცა, რომლის ელემენტები double ტიპის არიან. ასეთ შემთხვევაში შეგვიძლია ისინი გარდავქმნათ uint8 ტიპის სურათად და შემდეგ წარმოვადგინოთ, ან პირდაპირ წარმოვადგინოთ.

თუ double ტიპის სურათის imshow ბრძანებით წარმოდგენას ვეცდებით სურათი gray-scale სახით გამოჩნდება იმ შემთხვევაში, თუ მატრიცის წევრები 0-სა და 1-ს შორისაა.

სცადეთ შემდეგი ბრძანებები:

```
c=imread(‘moon.tif');
cd=double(c);
imshow(c);

figure()

imshow(cd)
```

Figure 2 double-ად გარდაქმნის შემდეგ →

←

Figure 3 ორიგინალი სურათი

როგორც ხედავთ, მიღებული სურათი არაფრით წააგავს ორიგინალ სურათს. ეს იმით არის გამოწვეული, რომ imshow ბრძანება იღებს მნიშვნელობებს ნულსა და ერთს შორის. თუ მას მივაწვდით პიქსელის ინტენსიობის ერთზე დიდ მნიშვნელობებს ისინი გამოსახული იქნება, როგორც 1 (თეთრი), ნულზე ნაკლები მნიშვნელობები კი გამოსახული იქნება, როგორც 0 (შავი). ამ პრობლემის მოსაგვარებლად პიქსელის ინტენსიობის მნიშვნელობების ნორმალიზება გვჭირდება. ამას მატრიცის 255-ზე გაყოფით მივაღწევთ:

Imshow(cd/255)

ასევე სცადეთ imshow(cd/1024) და imshow(cd/128).

თუ სურათის double ტიპად გადაყვანა გვსურს, ამისთვის ბრძანება im2double შეგვიძლია გამოვიყენოთ, რომელიც შესაბამის ნორმალიზებას თვითონ ანხორციელებს.

1.5 სივრცული გაფართოება

სურათის სივრცული გაფართოება სურათის პიქსელების სიმკვრივეს აღწერს: რაც უფრო მაღალია გაფართოება, მით უფრო მეტი პიქსელი გააჩნია სურათს. მატლაბის imresize ბრძანების გამოყენები შეგვიძლია სურათის გაფართოება შევცვალოთ. დავუშვათ გვაქვს 256x256 8-ბიტიანი grayscale სურათი, რომელიც მატრიცა x-ად გვაქვს შენახული.

შემდეგი ბრძანების გამოყენებით სურათის ზომას ორჯერ შევამცირებთ:

imresize(x, 1/2) ;

ეს ბრძანება ზომის ცვლას სურათის მატრიცის სემპლინგით აღწევს. ორიგინალი სურათიდან, მხოლოდ ლუწი ინდექსის მქონე სტრიქონის და სვეტის შესაბამისი ელემენტები ამოირჩევა.

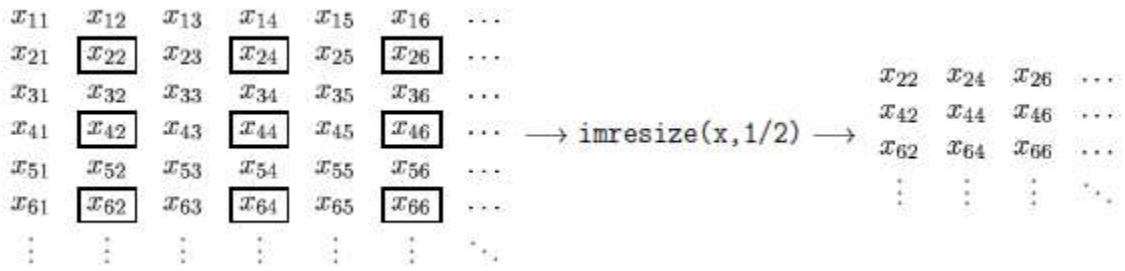


Figure 4 სურათის ზომის ცვლილება

ამ ბრძანების ერთხელ გამოყენების შემდეგ ვიღებთ სურათს, რომელიც 128x128 პიქსელისგან შედგება.

სცადეთ c=imread(‘tire.tif’) სურათის მატრიცის ზომის ცვლილება 1/2, 1/4, 1/8, 1/16 და 1/32-ჯერ. მიღებული სურათები გამოსახეთ გვერგიდგვერდ და დააკვირთით მიღებულ შედეგს.

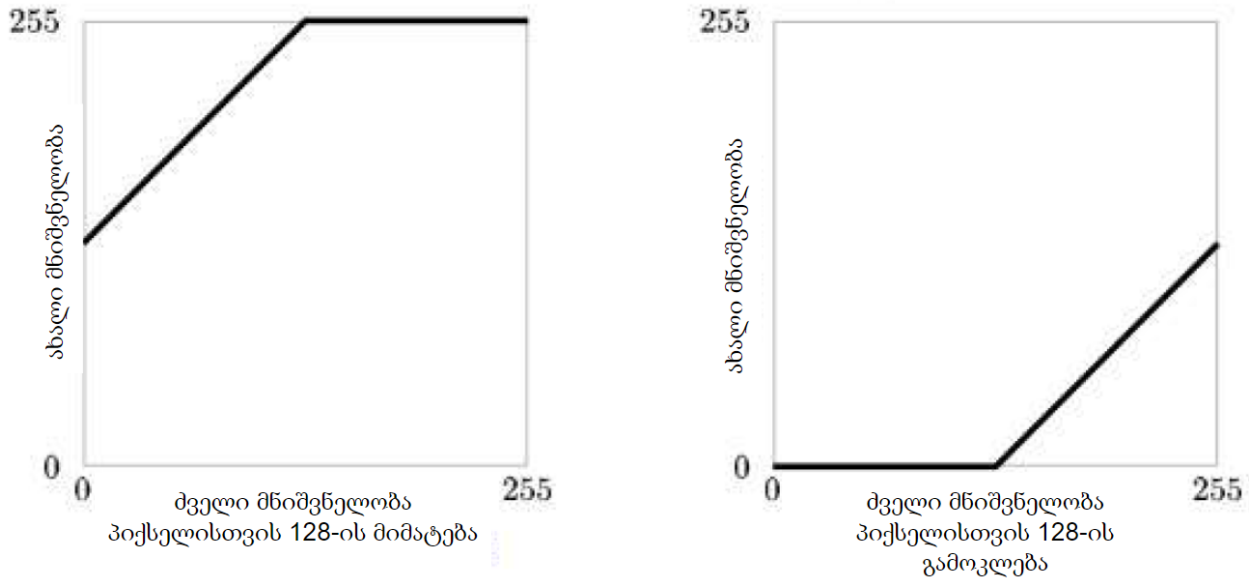
2 სურათების დამუშავების მეთოდები: წერტილოვანი დამუშავება

ნებისმიერი სურათის დამუშავების მეთოდი სურათის პიქსელების მოდიფიცირებას ახდენს, თუმცა სურათების დამუშავების მეთოდები სამ ფართო კლასად იყოფა.

1. გარდაქმნები. ამ შემთხვევაში გარდაქმნის ოპერაცია მთლიან სურათზე სრულდება და მასში ყველა პიქსელი მონაწილეობს.
2. სივრცითი ფილტრები. ვახდენთ სურათის დამუშავებას (პიქსელის ცვლილებას), რომელშიც პიქსელის ირგვლივ მდებარე მცირე პიქსელების რაოდენობა იღებს მონაწილეობას.
3. წერტილოვანი ოპერაციები. პიქსელის ცვლილება ხდება, თუმცა მასში სხვა პიქსელები არ მონაწილეობენ.

2.1 არითმეტიკული ოპერაციები

ასეთი ოპერაციების ქმედება ძველი და ახალი პიქსელის მნიშვნელობების შედარებით შეგვიძლია შევისწავლოთ. ქვემო სურათი გვიჩვენებს თითოეული პიქსელიდან 128-ის მიმატების და გამოკლების შედეგს. მაგალითად, როდესაც სურათის ყველა პიქსელს 128-ს ვუმატებთ, ყველა 127-ზე მაღალი მნიშვნელობა 255 ხდება.



შეგვიძლია ეს პროცედურა გამოვცადოთ ‘rice.png’ სურათზე:

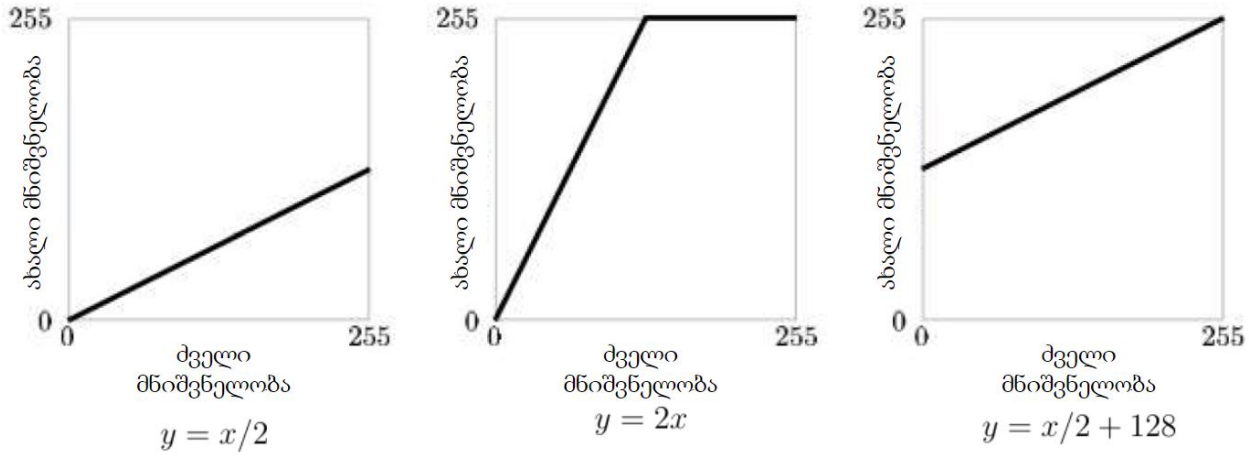
```
b=imread('rice.png');
b1=imadd(b,128); % ეს ბრძანება სურათის ქველა პიქსელს 128-ს უმატებს.
```

იგივენაირად გამოკლების შემთხვევაშიც:

b2=imsubtract(b,128);

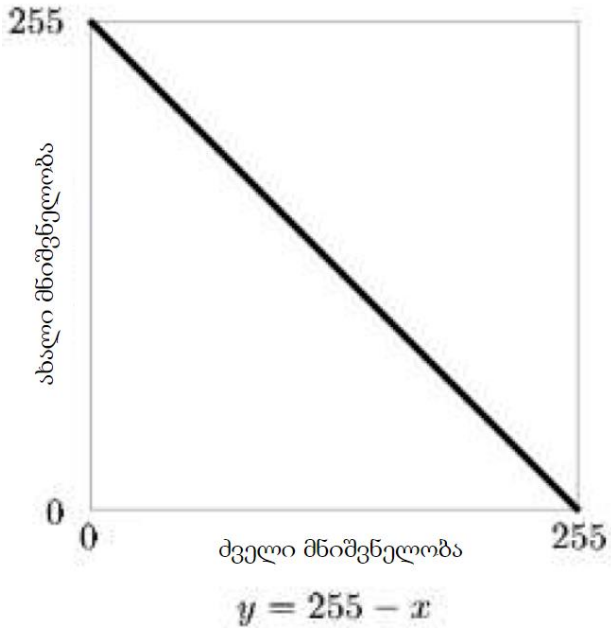
გამოსახეთ ამ დამუშავებით მიღებული შედეგები imshow ფუნქციის გამოყენებით.

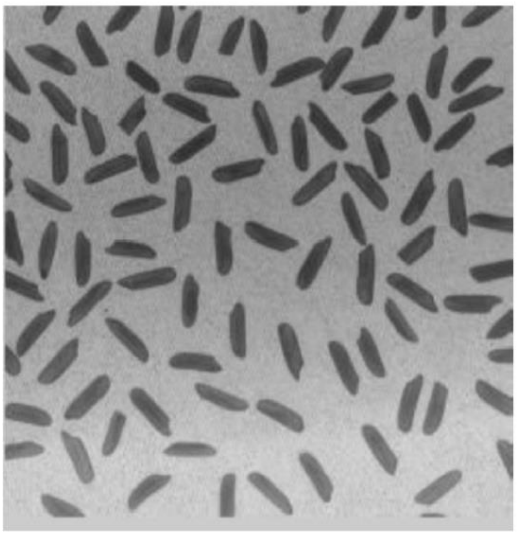
ასევე, გამრავლებით შესაძლებელია სურათის დანათება ან დაბნელება. ქვემო სურათზე მოცემულია რამდენიმე ასეთი ოპერაციის მაგალითი. სცადეთ თითოეული მათგანით იგივე სურათის დამუშავება.



კომპლემენტური სურათი არის სურათის „ნეგატივი“, რომელიც imcomplement ბრძანებით მიიღება. ტრანსფორმაცია, რომელსაც სურათი ამ შემთხვევაში გადის არის y=255-x.

B=imread(‘rice.png’); x=imcomplement(b);
imshow(xc)



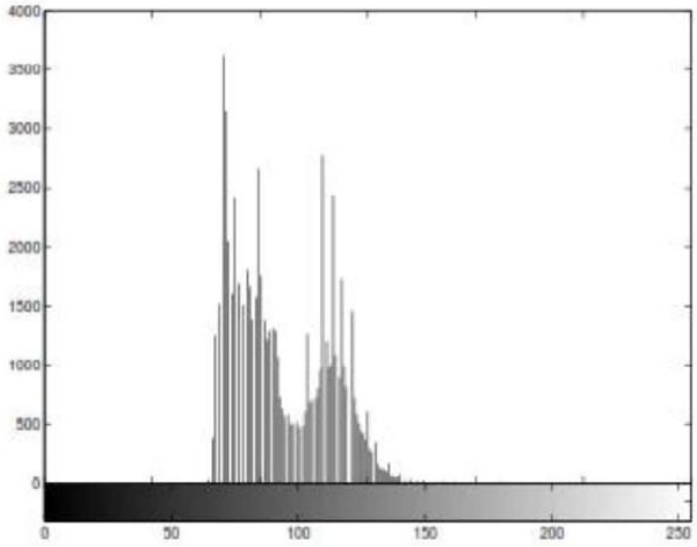


2.2 ჰისტოგრამები და ჰისტოგრამის გათანაბრება (ეკვალიზაცია)

Greyscale სურათის ჰისტოგრამა შედგება მასში ნაცრისფერის სხვდასხვა ტონის ჰისტოგრამისგან. ანუ გრაფიკი იძლევა კონკრეტული ნაცრისფერი ტონის შესაბამისი პიქსელის რაოდენობას მთლიან სურათში. სურათის ჰისტოგრამიდან ბევრი რამის თქმა შეიძლება მისი განათების შესახებ:

- ჩაბნელებულ სურათში, ნაცრისფრის ტონების უმეტესობა შედარებით დაბალი მნიშვნელობებისკენ არის კონცენტრირებული.
- გადაანათებულ სურათში ნაცრისფრის ტონების უმეტესობა მაღალი მნიშვნელობებისკენ იყრის თავს.
- კარგი კონტრასტის მქონე სურათში ტონები მეტნაკლებად თანაბრად არიან გადანაწილებული.

მატლაბში გახსენით სურათი ‘pout.tif’ და ააგეთ მისი ჰისტოგრამა ბრძანებით imhist().



როგორც ხედავთ სურათი არ არის კარგად განათებული, რადგან ნაცრისფრის ტონების უმეტესობა მკვრივად ცენტრშის სიახლოვეშია თავმოყრილი. ჰისტოგრამის გათანაბრებით შეგვიძლია ცუდი კონტრასტის მქონე სურათის გამოსწორება. გათანაბრებისას ნაცრისფერი ტონების უფრო თანაბრად განაწილება გვსურს.

დავუშვათ, რომ სურათს L ნაცრისფრის ტონი გააჩნია: 0, 1, 2 ... L-1 და რომელიღაც i-ური ტონი n_i-ჯერ გვაქვს სურათში წარმოდგენილი. სურათის პიქსელების სრული რაოდენობა არის n, სადაც $n_0 + n_1 + n_2 + \cdots + n_{L-1} = n$. უკეთესი კონტრასტის მიღებისთვის i-ური ტონის შემდეგი გარდაქმნა ხდება:

$$\left(\frac{n_0 + n_1 + \cdots + n_i}{n}\right)(L-1)$$

მიღებული მნიშვნელობა შეგვიძლია უახლოეს მთელ რიცხვამდე დავამრგვალოთ.

გამოიყენეთ შემდეგი ბრძანება და დააკვირდით მიღებული სურათის კონტრასტს.

```
ch=histeq(p);
imshow(ch)
figure
imhist(ch);
```

უკეთ არის განათებული სურათი ორიგინალთან შედარებით?

2.3 ზღვრული ფილტრაცია

თუ გვსურს grayscale სურათის ბინარულში გარდაქმნა, ორიგინალი სურათიდან ვირჩევთ ზღვრულ ნაცრისფრის ტონს T, რომელსაც გარდაქმნისას ზღვრად გამოვიყენებთ. პიქსელები, რომელთან T-ზე დიდი მნიშვნელობა აქვთ თეთრები ხდებიან, ხოლო მასზე ნაკლები მნიშვნელობის მქონე პიქსელები შავები.

$$\textit{ახალი პიქსელის მნიშვნელობა} = \begin{cases} 1, & \textit{თუ ნაცრისფრის ტონი} > T, \\ 0, & \textit{თუ ნაცრისფრის ტონი} < T. \end{cases}$$

ზოგ შემთხვევაში ზღვრული ფილტრაციის გამოყენებით ობიექტების ფონიდან გამოყოფა არის შესაძლებელი.

ზღვრული ფილტრაციისთვის მატლაბში ვიყენებთ ოპერაციას x>T, სადაც x სურათის მატრიცაა, T კი სასურველი ზღვარია.

სცადეთ შემდეგი:

```
r=imread('rice.png');
imshow(r)
figure
imshow(r>120).
```

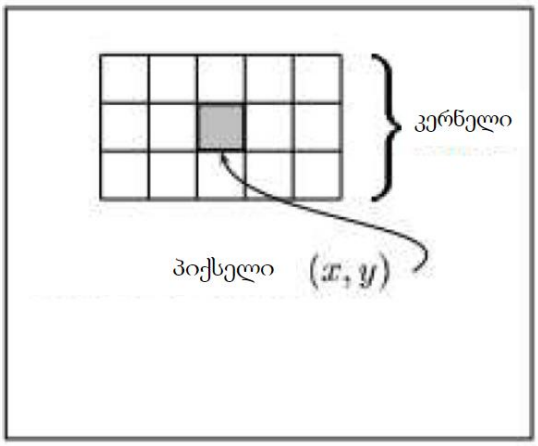
მიღებულ სურათზე, ორიგინალთან შედარებით, მკვეთრად უნდა იყოს გამოსახული ბრინჯის მარცვლები.

მატლაბის ფუნქცია im2bw(x,T) იგივე ოპერაციას არსულებს, სადაც x სურათია, T კი ზღვრის მნიშვნელობა, რომელიც 0-დან 1-მდეა განსაზღვრული.

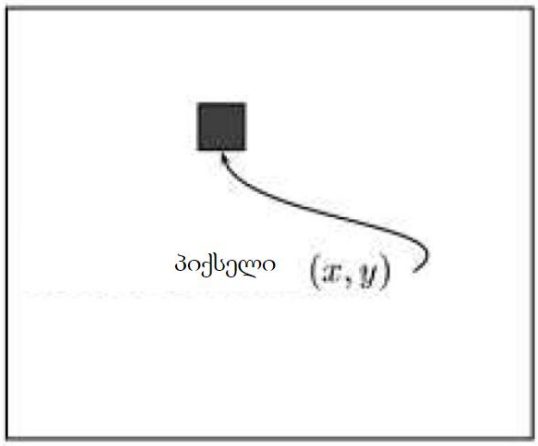
2.3 სივრცული ფილტრაცია

ვნახეთ, რომ სურათის მოდიფიცირება შესაძლებელია თითოეულ პიქსელზე მოქმედებით. სივრცული ფილტრაცია ამ მეთოდის გაფართოებას გულისხმობს, სადაც პიქსელის მეზობლად მყოფი პიქსელების გამოყენებით ვაგენერირებთ თითოეული პიქსელის ახალ მნიშვნელობას.

ასეთ მეთოდებში, მართკუთხა კერნელს (რომელსაც კენტი სიგანე და სიგრძე გააჩნია) გადავატარებთ მოცემულ სურათს. კერნელის ცენტრალური პიქსელის მნიშვნელობის მინიჭება ხდება კერნელის მიერ დაფარული პიქსელებზე ოპერაციის ჩატარებით. ასეთ დროს კერნელს და პიქსელებზე ოპერციებს ფილტრს ვუწოდებთ. თუ ახალი პიქსელის გამოთვლა წრფივი ფუნქციის გამოყენებით ხდება, ვამბობთ, რომ ფილტრი წრფივია.



ორიგინალი სურათი



სახეცვლილი პიქსელი
ფილტრაციის შემდეგ

წრფივი ფილტრის დანერგვა შესაძლებელია კერნელის ყველა წევრის შესაბამის პიქსელის მეზობელ პიქსელებზე გამრავლებით და მათი შეკრებით. შემდეგ სურათზე ნაჩვენებია ეს პროცედურა, რომელიც სამ ეტაპად შეგვიძლია წარმოვადგინოთ:

1. მოვათავსოთ კერნელი ახლანდელი პიქსელის ზევით,
2. გადავამრავლოთ კერნელის თითოეული პიქსელის მნიშვნელობა, მის ქვეშ არსებული პიქსელის მნიშვნელობაზე,
3. დავაჯამოთ მიღებული შედეგი (ეს ხდება ახალი პიქსელის მნიშვნელობა),
4. გავიმეოროთ 1-3 ნაბიჯები ყველა პიქსელისთვის.

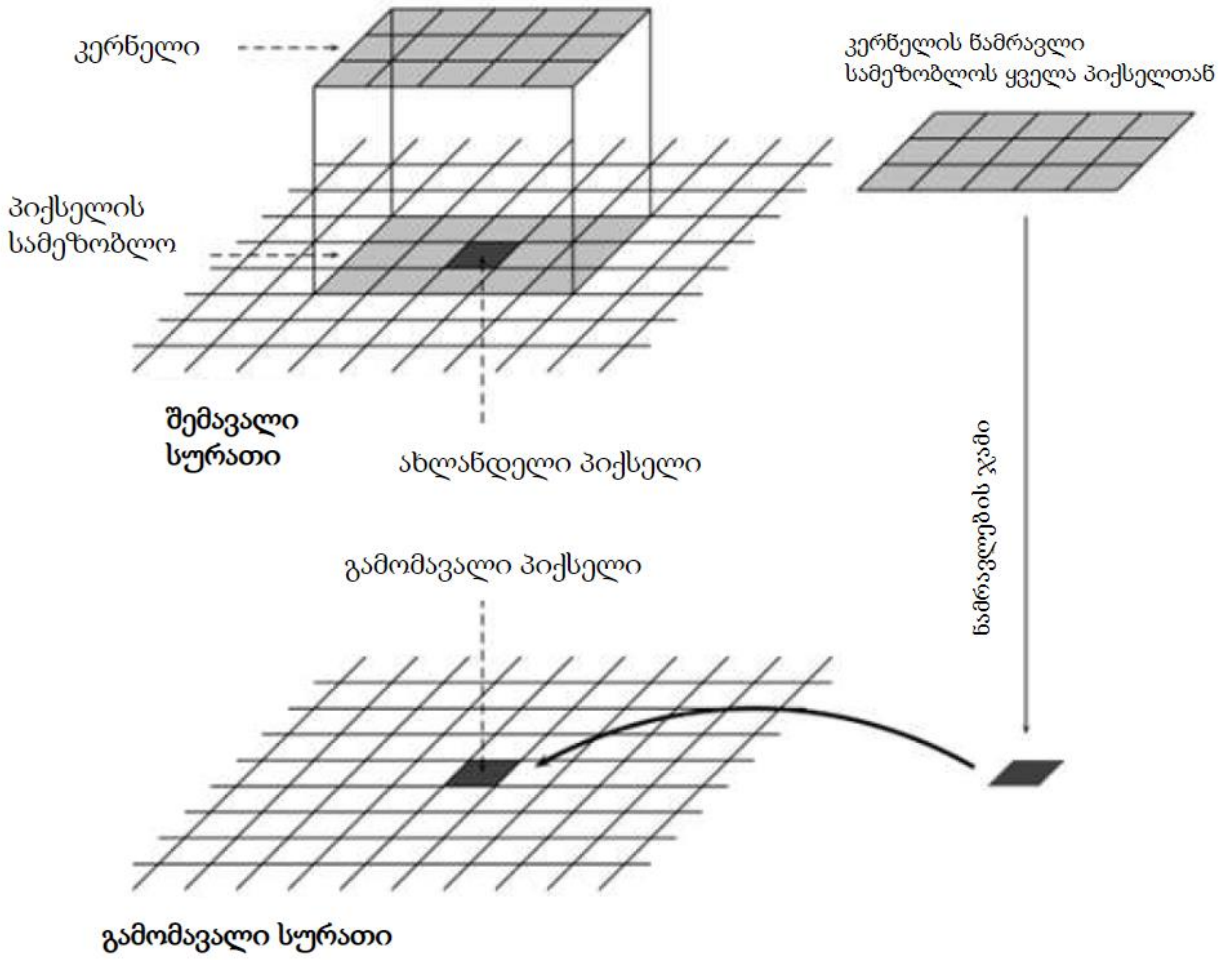


Figure 5 სივრცითი ფილტრის ქმედება

სურათის კიდეები

ამ მეთოდის გამოყენებისას სურათის კიდეებთან პრობლემას ვაწყდებით: კერნელის ნაწილი სურათის გარეთ არის მოთავსებული. ასეთ შემთხვევაში არ შეგვიძლია კერნელის ზოგი მნიშვნელობის გამოყენება, რადგან მის ქვემოთ დასამუშავებელი სურათის პიქსელები არ არის.

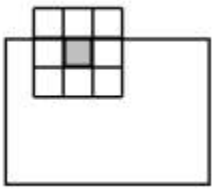


Figure 6 კერნელი სურათის კიდეზე

ამ პრობლემის რამდენიმე გადაჭრა არსებობს.

- შეგვიძლია კიდეები არ გავითვალისწინოთ, ანუ კერნელი ის გადავატაროთ სურათზე, რომ ის სურათის კიდეს არასოდეს გაცდეს. ეს ნიშნავს, რომ მიღებული სურათი ორიგინალ სურათზე პატარა იქნება. ეს განსაკუთრებით დიდი დანაკარგია, როდესაც დიდი ზომის კერნელს ვიყენებთ.
- ნულებით შევსება. ამ შემთხვევაში სურათს გარე კანტს ვუკეთებთ, რომელიც შესაბამისი რაოდენობის ნულებისგან შედგება. ასეთ დროს გამომავალი სურათი შემავალი სურათის ზომისაა. შესაძლოა ამ მეთოდმა სურათში არასასურველი ეფექტები გააჩინოს სურათის კიდეების სიახლოვეში.

მატლაბში სურათის ფილტრაციისთვის filter2 ბრძანებას ვიყენებთ: filter2(filter, image, shape).

Filter - გამოყენებული ფილტრია (კერნელის ტიპი და ზომა);

Image - დასამუშავებელი სურათი;

Shape - ვუთითებთ რა მეთოდი გამოიყენოს კიდეებთან (თუ არ მივუთითებთ ნულებით შეივსება).

ფილტრების შექმნა ჩვენითაც შეგვიძლია, ან მატლაბის ჩაშენებულ ფილტრებს ვიყენებთ. ფილტრის შექმნა მატლაბის fspecial ფუნქციის გამოყენებით შეგვიძლია:

fspecial('average',[5,7]) ქმნის 5x7 ზომის „გამასაშუალოებელ“ ფილტრს. თუ კვადრატული ფორმის კერნელი გვსურს, შეგვიძლია მეორე არგუმენტად ვექტორის მაგივრად უბრალო რიცხვი ჩავაწოდოთ: fspecial('average',9). თუ მეორე არგუმენტი მითითებული არ არის მატლაბი 3x3 ზომის კერნელს ქმნის.

მაგალითად განვიხილოთ ასეთი ფილტრი:

```
c=imread('cameraman.tif');
f1=fspecial('average');
cf1=filter2(f1,c);
figure
imshow(c)
figure
imshow(cf1/255)
```

რა ეფექტი აქვს მასაშუალოებელ ფილტრს სურათზე? ცვალეთ მისი ზომა და აღწერეთ ზომის ცვლილებით გამოწვეული ეფექტი. დააკვირდით მიღებული სურათის კიდეს სხვადასხვა ზომის ფილტრების გამოყენებისას.

2.3.1 დაბალსიხშირული და მაღალსიხშირული ფილტრები

სურათის სიხშირე წარმოადგენს, რამდენად „სწრაფად“ იცვლება მისი პიქსელის მნიშვნელობები მანძილთან ერთად. მაღალსიხშირულ კომპონენტებს ახასიათებთ მოკლე მანძილზე დიდი ცვლილებები. მაღალსიხშირული კომპონენტებია სურათის კიდეები და ხმაური. დამალსიხშირული კომპონენტები, მანძილთან ერთად მცირედით იცვლებიან. ეს შეიძლება იყოს სურათის ფონი.

მაღალსიხშირული ფილტრი სურათიდან აქრობს ნელა ცვალებად კომპონენტებს, დაბალსიხშირული ფილტრი კი სურათის სწრაფად ცვალებად კონპონენტებს ფილტრავს.

მაგალითად, შემდეგი 3x3 მასაშუალოებელი ფილტრი დაბალსიხშირულია. შემდეგი კერნელის მქონე ფილტრი კი მაღალსიხშირული:

$$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

სცადეთ ამ ფილტრის ‘cameraman.tif’ სურათზე გამოყენება:

```
h = [1 -2 1; -2 4 -2; 1 -2 1];

imfilter(x,h)
```

2.3.2 გაუსის ფილტრები

გაუსის ფილტრები დაბალსიხშირული ფილტრების კლასია, რომლების გაუსის განაწილებაზეა დაფუძნებული:

$$f(x) = e^{-\frac{x^2}{2\sigma^2}}$$

ორ განხომილებიანი გაუსის ფუნქცია მოცემულია, როგორც:

$$f(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

გაუსის ფილტრების „ბლარის“ ეფექტი გააჩნიათ, რომელიც წააგავს მასაშუალოებელი ფილტრის ქმედებას. გამოიყენეთ სხვადასხვა ზომის გაუსის ფილტრი რომილემე სურათზე და დააკვირდით მის ქმედებას

```
g1=fspecial('gaussian',[5,5]);
g2=fspecial('gaussian',[5,5],2);
g3=fspecial('gaussian',[11,11],1);
g4=fspecial('gaussian',[11,11],5);
imshow(filter2(g1,c)/256), figure,imshow(filter2(g2,c)/256)
figure,imshow(filter2(g3,c)/256), figure,imshow(filter2(g4,c)/256)
```

2.3.3 კიდეების დეტექცია

სურათში კიდეების დეტექციისთვის სხვადასხვა კერნელის გამოყენებაა შესაძლებელი, გამოიყენეთ შემდეგი კერნელი რიმელიმე სურათზე და აღწერეთ მიღებული შედეგი:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

შეგიძლიათ შეიმუშავოთ კერნელი რომელიც სხვა მიმართულების კიდეების დეტექციასაც მოახდენს?