NTNU

Kunnskap for en bedre verden

Kunnskap for en bedre verden

TDT4105 Informasjonsteknologi, grunnkurs

Eksempler

Rune Sætre (satre@idi.ntnu.no)

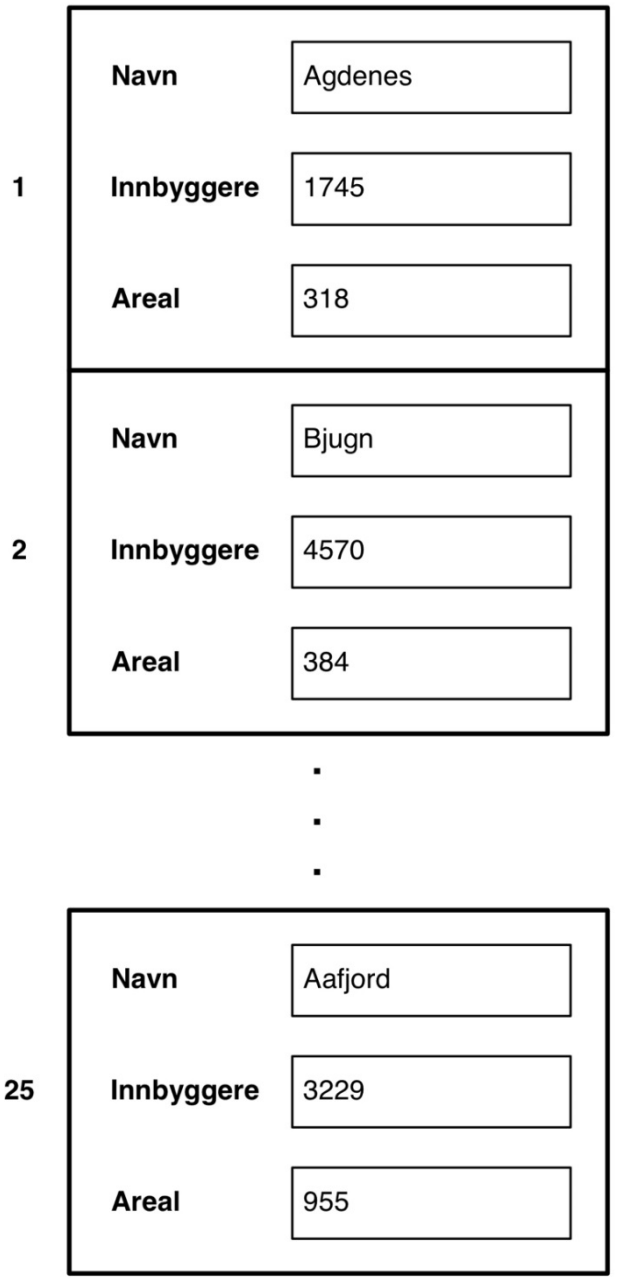
Kommunedata

- Sør-Trøndelag
  - 25 kommuner
    - Navn
    - Innbyggere
    - Areal (km^2)
  - Tekstfil med 25 linjer
- Må representere dette i programmet
  - Datastruktur
  - Mulige løsninger?

Agdenes	1745	318
Bjugn	4570	384
Frøya	4326	241
Hemne	4232	670
Hitra	4340	686
Holtålen	2048	1210
Klæbu	5894	186
Malvik	12677	169
Meldal	3903	613
Melhus	15114	695
Midtre-Gauldal	6050	1861
Oppdal	66912	274
Orkdal	11365	594
Osen	1025	387
Rennebu	2629	948
Rissa	6543	622
Roan	994	375
Røros	5581	1956
Selbu	3996	1235
Skaun	6756	224
Snillfjord	992	508
Trondheim	173486	342
Tydal	886	1329
Ørland	5133	74
Åfjord	3229	955

Datastruktur

- En struktur for hver kommune
  - Felter
    - Navn
    - Innbyggere
    - Areal
- En vektor med strukturer for fylket
- Lager funksjon som leser fil-data inn i slik datastruktur
  - Inndata: Filnavn
  - Utdata: Vektor med strukturer



lesKommuneFil.m

```
function kommuner = lesKommuneFil( filNavn )

    fid = fopen( filNavn, 'r', 'native', 'utf8');
    % 'UTF-8' / 'latin1' / ASCII %% utf8 i Matlab-versjon R2013a

    if fid == -1
        kommuner = -1;
    else
        indeks = 1;

        while ~feof(fid)
          % leser linje fra filen
          filLinje = fgetl(fid);
          % putter data inn i struktur
          kommuner(indeks) = lagPost(filLinje);
          % indeks for evt. neste post
          indeks = indeks + 1;
        end % while

        status = fclose(fid);
        if status ~= 0
           kommuner = -1;
        end
    end % if fid ok

end % function
```

lesKommuneFil.m

```
function post = lagPost(tekst)
% putter data for en kommune inn i struktur

    % plukker ut kommunenavn
    [post.navn, resten] = strtok(tekst);

    % plukker ut innbyggertall og areal
    tallData = str2num(resten);
    post.innbyggere = tallData(1);
    post.areal = tallData(2);

end % function

function post = lagPost2(tekst)
% alternativ måte å gjøre det

    % plukker ut kommunenavn
    [post.navn, resten] = strtok(tekst);

    % plukker ut innbyggertall
    [tall, resten] = strtok(resten);
    post.innbyggere = str2num(tall);

    % plukker ut areal
    [tall, resten] = strtok(resten);
    post.areal = str2num(tall);

end % function
```

```
>> kommuner = lesKommuneFil('datafil.txt')

kommuner =

1x25 struct array with fields:
    navn
    innbyggere
    areal

>> kommuner(1)

ans =

          navn: 'Agdenes'
    innbyggere: 1745
         areal: 318

>> kommuner(25)

ans =

          navn: 'Aafjord'
    innbyggere: 3229
         areal: 955

>>
```

Befolkningstetthet i Sør-Trøndelag

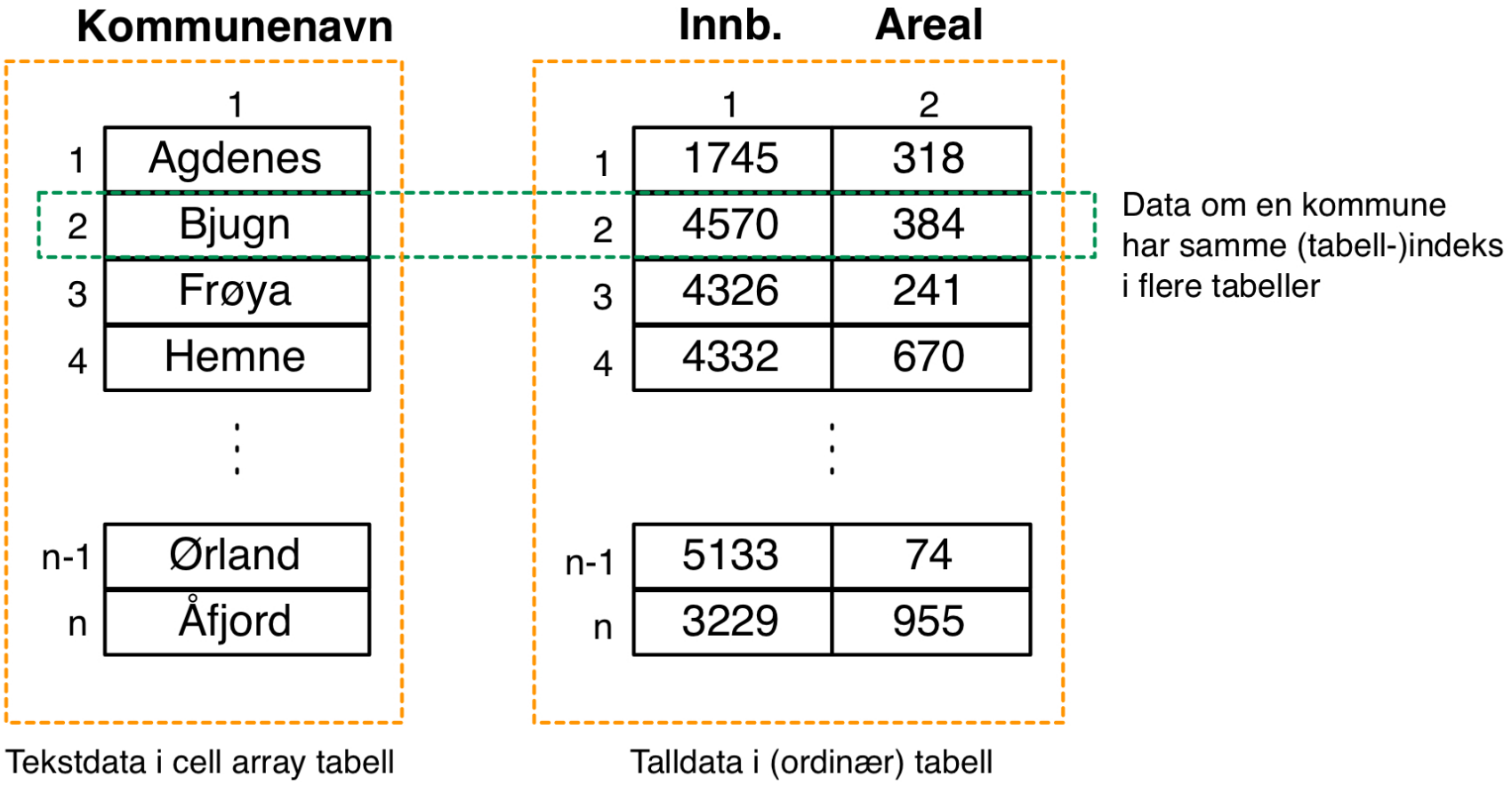
- Summere innbyggertall i alle kommuner
- Summere areal i alle kommuner
- Tetthet = sum(innbyggere) / sum(areal)
- Lager funksjon som gjør dette:
  - Inndata: Vektor med strukturer (kommunedataene)
  - Utdata: Befolkningstetthet

befolkningsTetthet.m

```
function tetthet = befolkningsTetthet(kommuner)

    % antall kommuner
    n = length(kommuner);

    % summerer befolkning og areal
    befolkning = 0;
    areal = 0;

    for i = 1:n
        befolkning = befolkning + kommuner(i).innbyggere;
        areal = areal + kommuner(i).areal;
    end % for

    % beregner befolkningstetthet
    tetthet = befolkning / areal;

end % function
```

```
>> t = befolkningsTetthet(k)

t =

   15.6027

>>
```

Alternativ løsning: Parallelle vektorer



```
>> [kNavn, kData] = lesKommuneFil_alt2('datafil.txt')
kNavn =
  Columns 1 through 7
    'Agdenes'    'Bjugn'    'Froya'    'Hemne'    'Hitra'    'Holtaalen'    'Klaebu'
  Columns 8 through 14
    'Malvik'    'Meldal'    'Melhus'    'Midtre-Gauldal'    'Oppdal'    'Orkdal'    'Osen'
  Columns 15 through 21
    'Rennebu'    'Rissa'    'Roan'    'Roros'    'Selbu'    'Skaun'    'Snillfjord'
  Columns 22 through 25
    'Trondheim'    'Tydal'    'Oerland'    'Aafjord'
kData =
        1745         318
        4570         384
        4326         241
        4232         670
        4340         686
        2048        1210
        5894         186
       12677         169
        3903         613
       15114         695
        6050        1861
        6691        2274
       11365         594
        1025         387
        2629         948
        6543         622
         994         375
        5581        1956
        3996        1235
        6756         224
         992         508
      173486         342
         886        1329
        5133          74
        3229         955
>> t = befolkningsTetthet_alt2(kData)
t =
   15.6027
>>
```

```
function [kNavn, kData] = lesKommuneFil_alt2( filNavn )

    fid = fopen( filNavn, 'r', 'native', 'utf-8');
    if fid == -1
        disp('Får ikke åpnet filen')
    else
        indeks = 1;

        while ~feof(fid)
          % leser linje fra filen
          filLinje = fgetl(fid);
          % plukker ut dataelementene
          [navn, resten] = strtok(filLinje);
          tallData = str2num(resten);
          % putter data inn i datastrukturene
          kNavn{indeks} = navn;
          kData(indeks, 1) = tallData(1);
          kData(indeks, 2) = tallData(2);
          % indeks for evt. neste post
            indeks = indeks + 1;
        end % while

        status = fclose(fid);
        if status ~= 0
           disp('Kan ikke lukke filen')
        end
    end % if

end % function
```

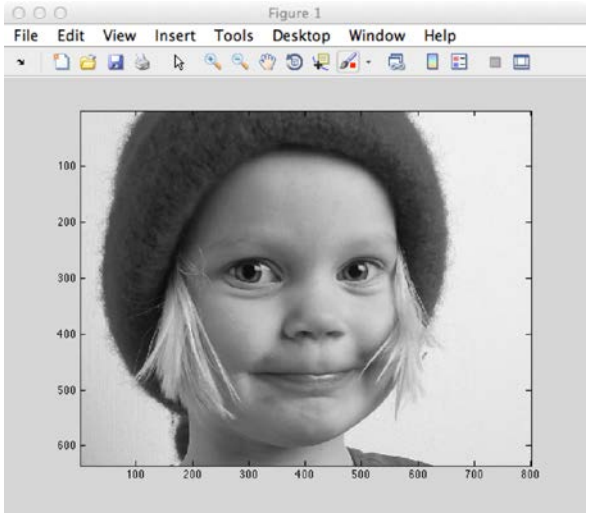
```
function tetthet = befolkningsTetthet_alt2(kommuneData)

    % beregner befolkningstetthet
    tetthet = sum(kommuneData(:,1)) / sum(kommuneData(:,2));

end % function
```

```
>> befolkningsTetthet_alt2(kData)

ans =

   15.6027

>>
```

Noen bildeeksempler

- Representasjon: To-dimensjonal tabell
  - Rader x kolonner med verdi for hver bildeelement (piksel)
- Gråtonebilder
  - 0 = sort, 255 = hvitt, gråtoner mellom
- Bilder har ”fargekart” som definerer hvilken farge hver pikselverdi skal ha.

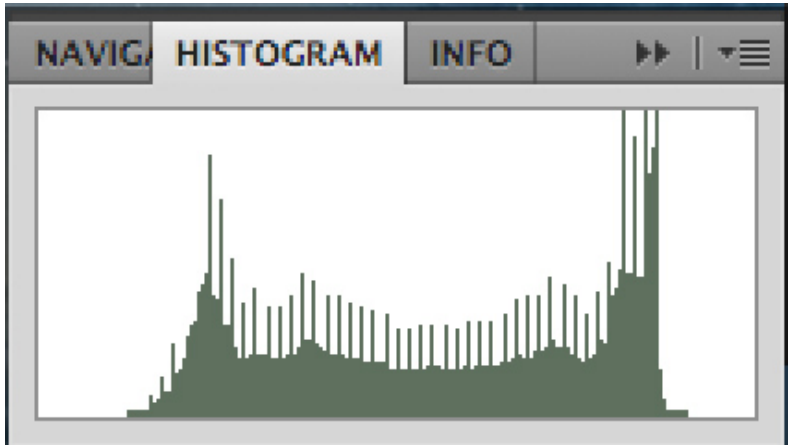
```
% leser inn bilde fra fil
[bilde, map] = imread('jente.gif');

% setter riktig "fargekart"
colormap(map)

% viser bildet
image(bilde)
```

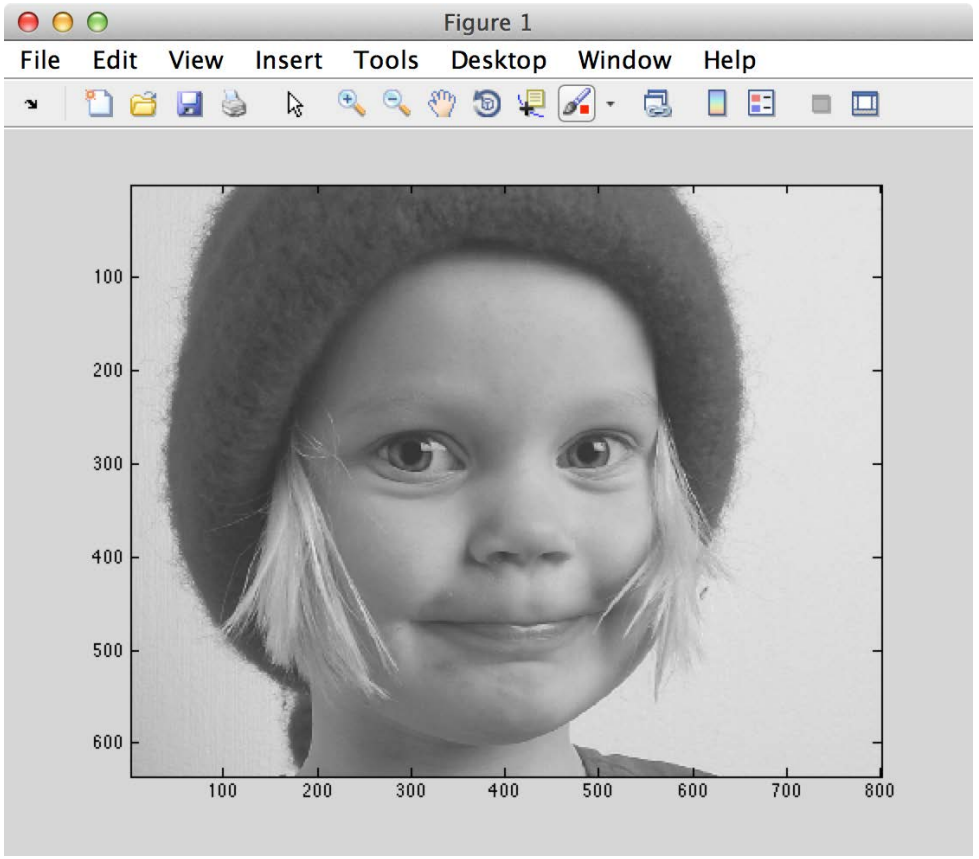


Problem: Lite kontrast

```
>> [bilde, map] = imread('jente1.gif');
>> colormap(map)
>> image(bilde)
>>
```

- Utnytter ikke toneomfanget
- Mørkeste piksel: 31
- Lyseste piksel: 233





Løsning

```
>> a = uint8(130);
>>
>> b = a + 100
b =
  230
>> c = a + b
c =
  255
>> d = 2*a
d =
  255
>>
```

- Skalerer 31-233 til 0-255
  - (pikselverdi-31)*255/(233-31)
    - 31 -> 0
    - 233 -> 255
- NB! Bildetabellene våre har datatype **uint8**
  - ”Arithmetic operations that involve both integers and floating-point always result in an integer data type.”
  - Det er lett å regne feil (se øverst til høyre)
- Typekonvertering (type cast)
  - B = cast(A,NEWCLASS)
  - minPiksel = cast(minPiksel, 'double');
    - Gjør om (”31”) til double-representasjon

maksKontrast.m

```
function bilde = maksKontrast(bilde)

    minPiksel = min(min(bilde));
    maksPiksel = max(max(bilde));

    minPiksel = cast(minPiksel,'double');
    maksPiksel = cast(maksPiksel,'double');

    strekkeFaktor = 255/(maksPiksel-minPiksel);

    bilde = round( (bilde-minPiksel)*strekkeFaktor );

end % function
```

```
>> nyttbilde = maksKontrast(bilde);
>> max(max(nyttbilde))
ans =
   255
>> min(min(nyttbilde))
ans =
     0
>> image(nyttbilde)
>>
```

Resultat

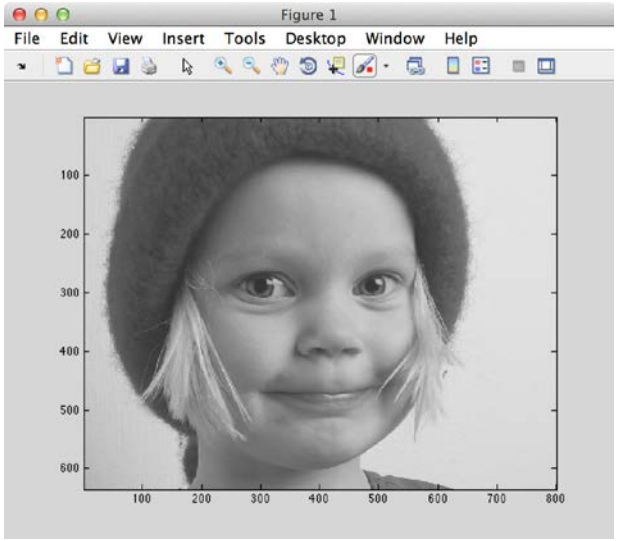
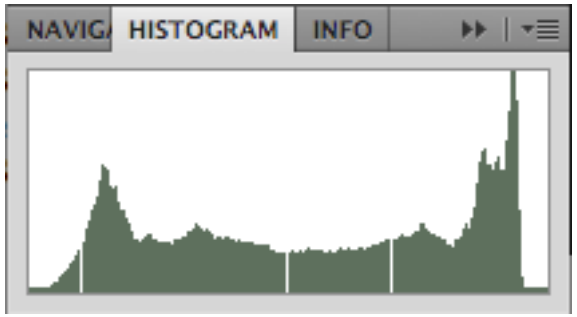


Figure 1
File Edit View Insert Tools Desktop Window Help
100
200
300
400
500
600
100 200 300 400 500 600 700 800


NAVIG
HISTOGRAM
INFO

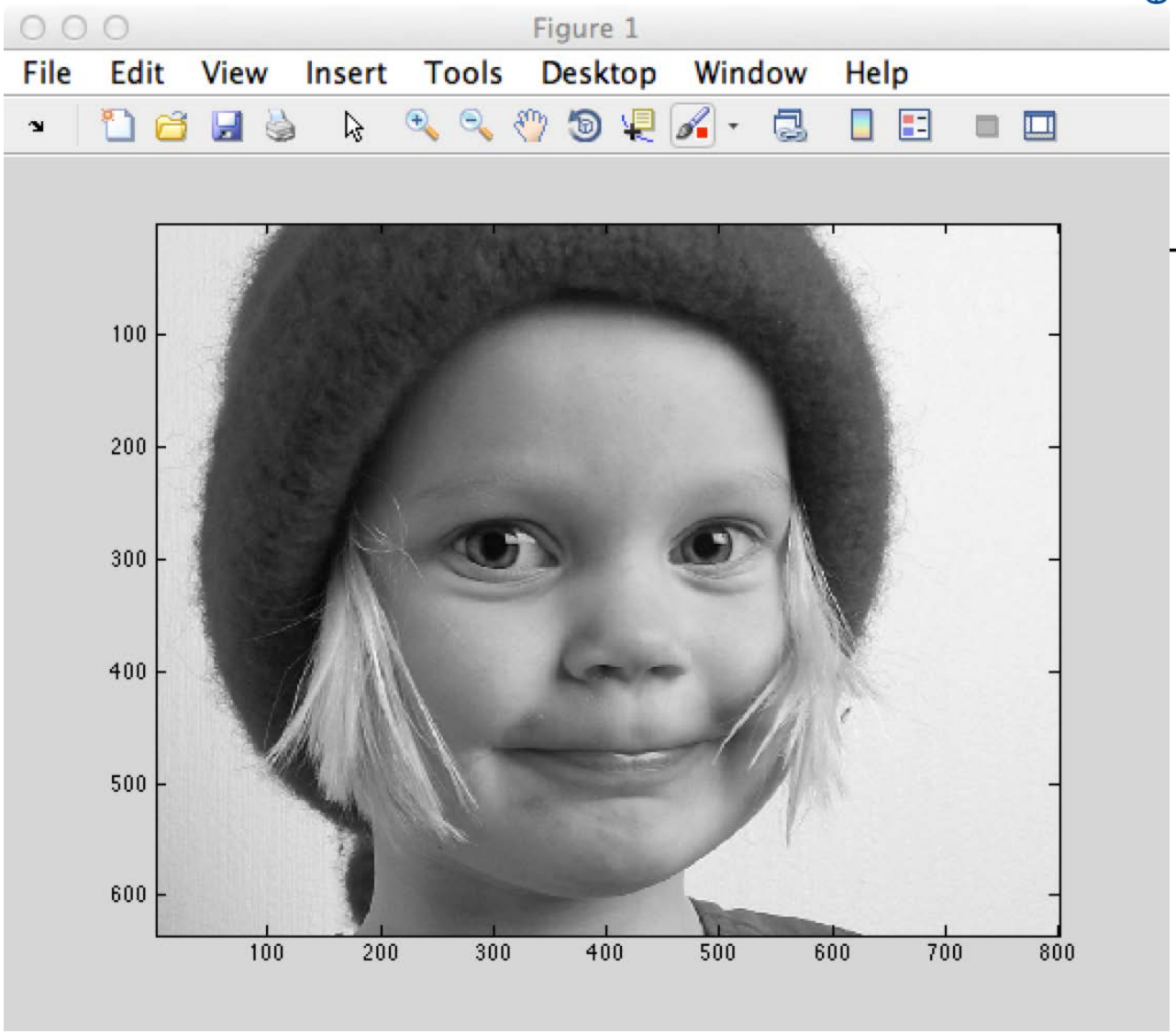
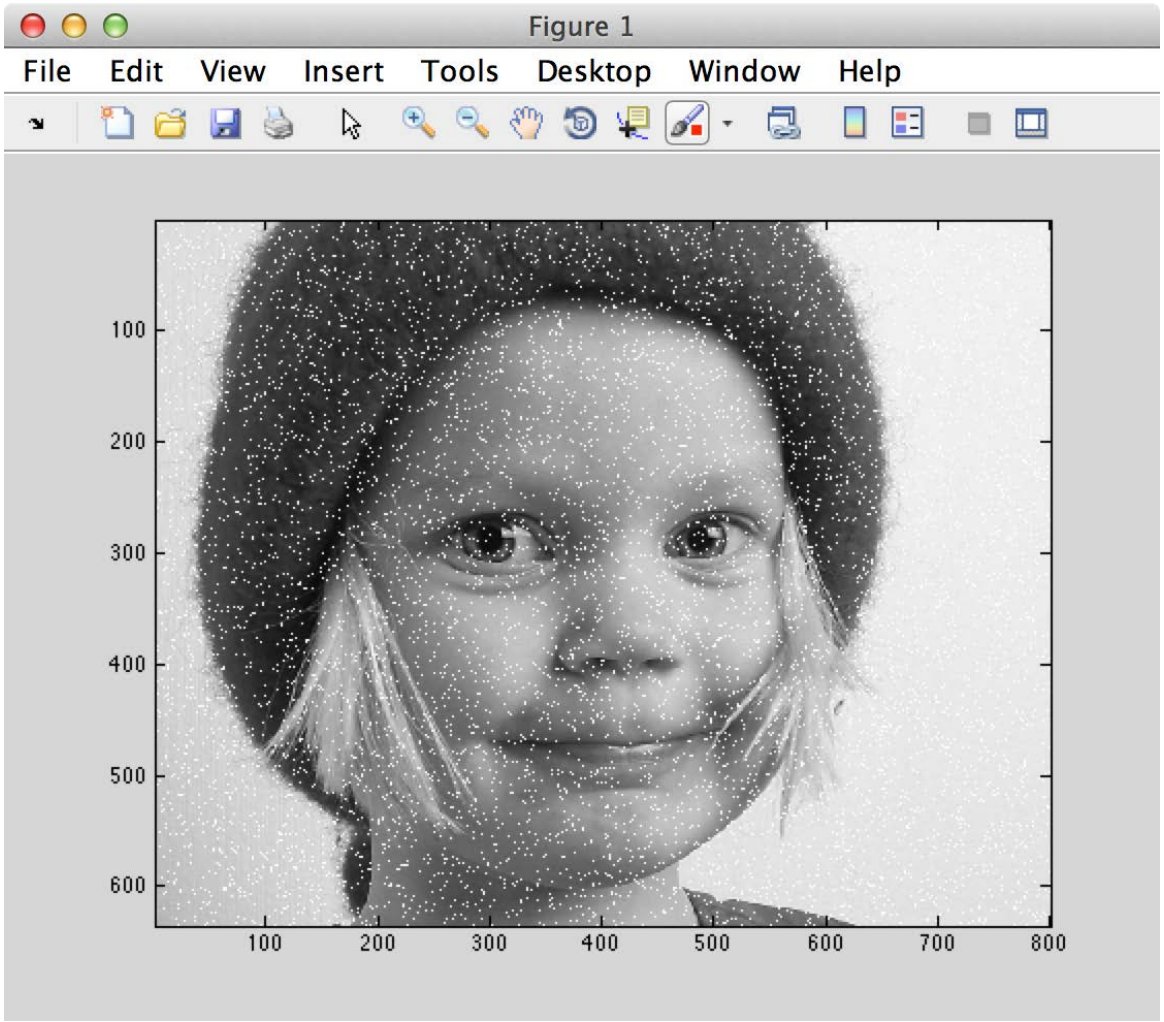
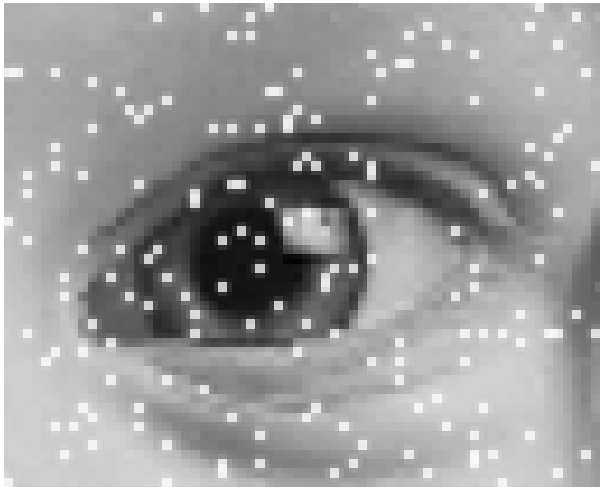


Figure 1
File Edit View Insert Tools Desktop Window Help
100
200
300
400
500
600
100 200 300 400 500 600 700 800

Bilde med støy (hvite piksler)

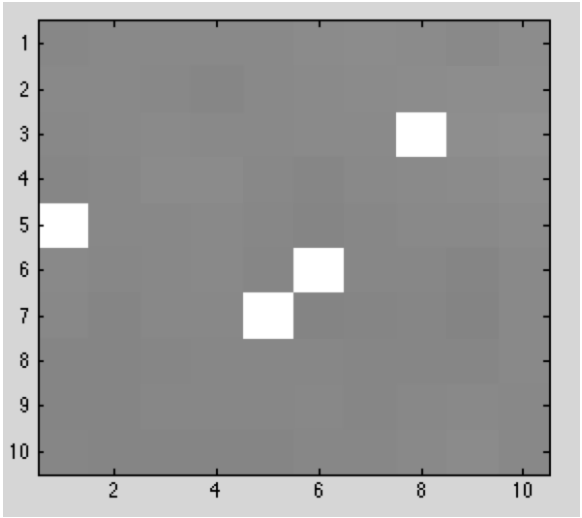


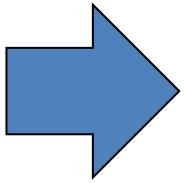


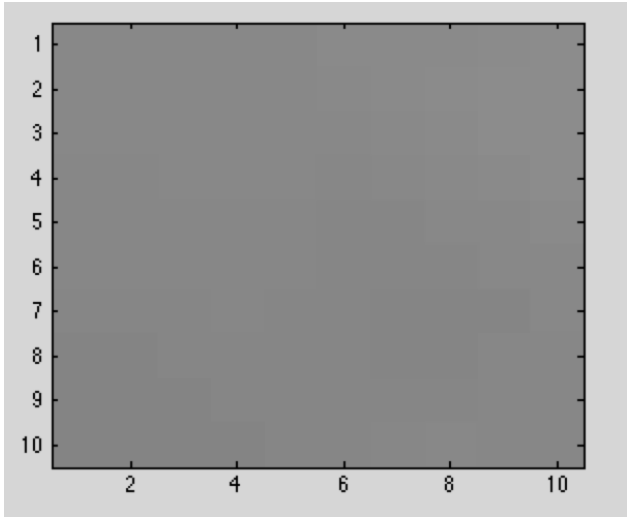
Filtrere støy med medianfilter

- Observasjon: Støypikslene skiller seg (stort sett) mye ut
- Idé: Erstatte støypikslene med en riktigere verdi fra omgivelsene
- Alle piksler har 8 naboer (unntatt kantpikslene som vi ignorerer)
- Får ”nabolag” på 3x3-piksler
  - Bruke gjennomsnittet av nabolaget (støypikslene ”trekker”)
  - Sortere pikselverdiene i nabolaget og bruke medianen (støypikslene blir marginalisert)
- Median-filter
  - Fjerner støy
  - Mister litt bildekvalitet (utenom støypikslene)

Utsnitt fra panna



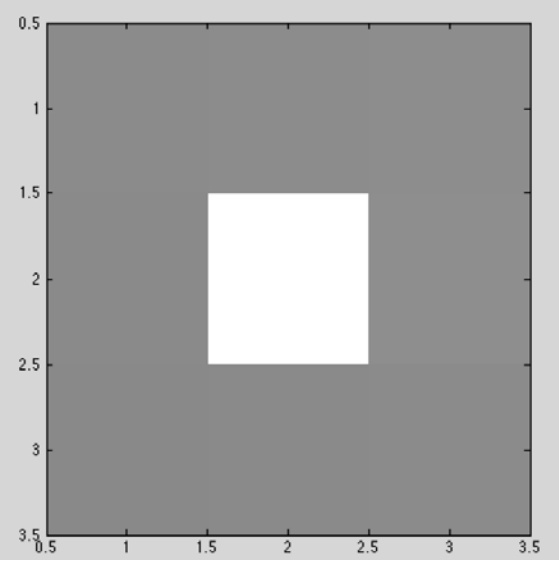




116	118	118	118	118	120	121	120	118	121
118	118	117	115	118	119	120	121	122	122
117	118	119	118	118	119	119	255	123	125
115	117	120	120	117	115	118	119	120	122
255	117	117	118	116	114	116	118	118	120
118	116	117	118	115	255	116	116	114	118
117	114	117	117	255	113	114	115	113	118
114	114	115	116	116	116	115	115	115	118
114	114	116	116	116	117	115	117	118	117
115	114	114	114	114	116	116	118	120	117

118	118	118	118	118	120	120	120	121	122
118	118	118	118	118	119	120	121	122	122
118	118	118	118	118	118	119	120	122	122
117	117	118	118	118	117	118	119	120	122
117	117	117	117	117	116	116	118	118	120
117	117	117	117	117	116	116	116	118	118
116	116	116	117	116	116	115	115	115	118
114	114	116	116	116	116	115	115	117	117
114	114	114	116	116	116	116	116	117	117
114	114	114	114	116	116	117	118	117	117

Detalj
(for en piksel)



121	120	118
120	121	122
119	255	123
118	119	120
116	118	118

1	120
2	119
3	118
4	121
5	255
6	119
7	122
8	123
9	120

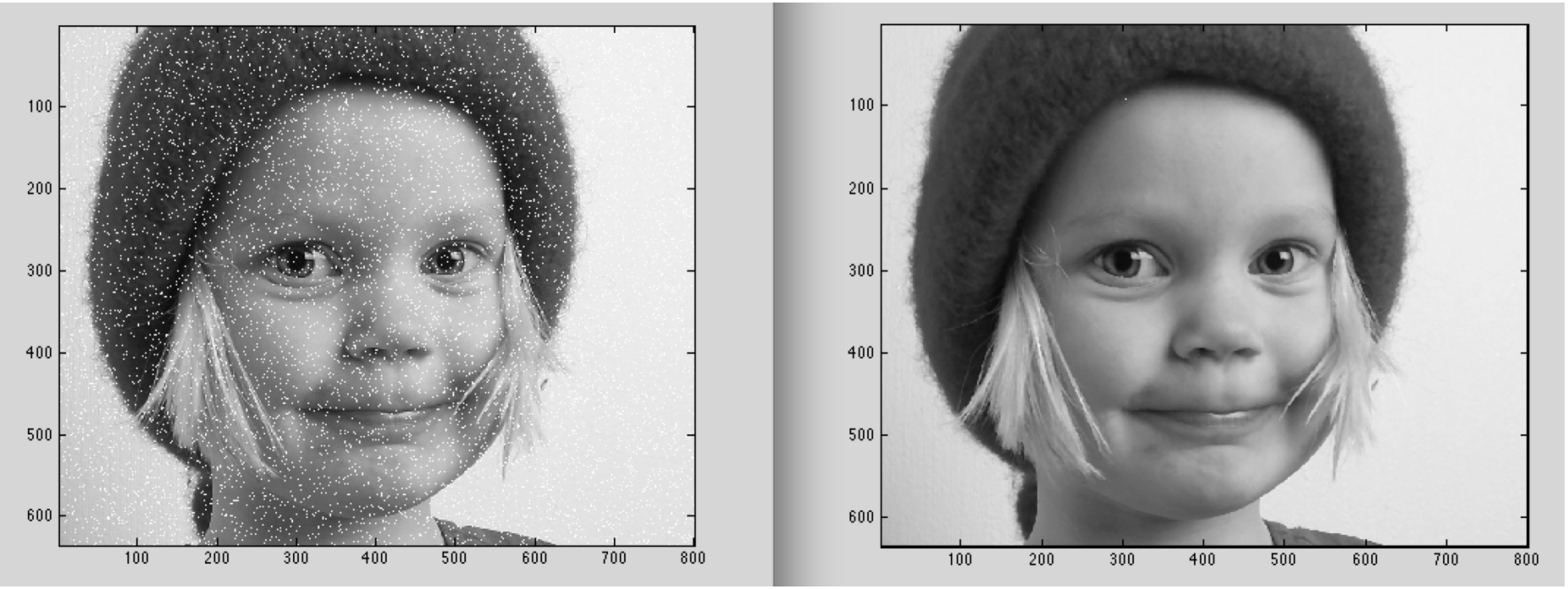
Sorterer

1	118
2	119
3	119
4	120
5	120
6	121
7	122
8	123
9	255

medianFilter.m

```
function bildeUt = medianFilter(bildeInn)

    [rader, kolonner] = size(bildeInn);

    bildeUt = zeros(rader, kolonner);

    for r = 2:rader-1
        for k = 2:kolonner-1

            bildeUt(r,k) = medianPiksel(bildeInn(r-1:r+1, k-1:k+1));

        end %for
    end %for

end % function

function piksel = medianPiksel(M)

     vektor = sort(M(:));
     piksel = vektor(5);

end % medianPiksel
```

Resultat



Kant-filter

- Ser på nabolaget rundt hver piksel (3x3)
- En maske for å oppdage ”horisontale” kanter
  - $Gx = (z_7+2z_8 + z_9) – (z_1 + 2z_2 + z_3)$
- En maske for å oppdage ”vertikale” kanter
  - $G_y = (z_3+2z_6 + z_9) – (z_1 + 2z_4 + z_7)$
- Kantstyrken i hvert punkt
  - $G = |G_x| + |G_y|$
- Dette kalles en Sobel-operator
  - Finne alternative måter å finne kanter

z1	z2	z3
z4	z5	z6
z7	z8	z9

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Eksempel

```
>> bildebit
bildebit =
     0     0     0     0     0     0
     0     9     9     9     9     0
     0     9     9     9     9     0
     0     9     9     9     9     0
     0     0     0     0     0     0
>> kanter = kantFilter(bildebit)
kanter =
     0     0     0     0     0     0
     0    54    36    36    54     0
     0    36     0     0    36     0
     0    54    36    36    54     0
     0     0     0     0     0     0
>>
```

kantFilter.m

```
function bildeUt = kantFilter(bildeInn)

    [rader, kolonner] = size(bildeInn);

    bildeInn = cast(bildeInn, 'double');
    bildeUt = zeros(rader, kolonner);

    for r = 2:rader-1
        for k = 2:kolonner-1

            bildeUt(r,k) = sobelOperator(bildeInn(r-1:r+1, k-1:k+1));

        end %for
    end %for

end % function

function gradient = sobelOperator(M)

    gX = -M(1,1) - 2*M(1,2) - M(1,3) + M(3,1) + 2*M(3,2) + M(3,3);
    gY = -M(1,1) - 2*M(2,1) - M(3,1) + M(1,3) + 2*M(2,3) + M(3,3);

    gradient = abs(gX) + abs(gY);

end % function
```

Resultat