

Statistisk bildeanalyse og læring

Øving 1

Gruppe 1

Jørgen Bergquist

Espen Brill

Helge Titlestad

Lars Andreas Eidsheim

Erling Hagen

Innhold

1	Bildesampling: støy og apriori-fordelinger	1
1.1	Simulering av bildedegradering	1
1.2	Simulering fra diskret apriori-fordelinger	3
1.3	Simulering fra kontinuerlig apriori-fordeling	6
1.4	Simulering fra hierarkisk apriori-fordeling	9

Øving 1

Bildesampling: støy og apriori-fordelinger

1.1 Simulering av bildedegradering

Vi tar utgangspunkt i et bilde (1.1) som vi simulerer forskjellig degraderinger på. Først jevnet vi ut bildet med en utjevningssmatrise (1.1).



Figur 1.1: Originalbilde og utjevnet bilde

a)

Ved hjelp av Box-Muller algoritmen skal vi simulere ukorrelert additiv Gaussisk støy, med standardavvik 5.0 og 15.0 (1.2) med forventning 0.



Figur 1.2: Box-Muller med standardavvik 5.0 og 15.0

Vi ser at bildene blir støyete. Et høyere standardavvik gir mer synlig støy. Dette er fordi verdiene varierer mer.

Box-Muller:

$$\begin{aligned}
 B1 &= 2 * \pi * U1 \\
 B2 &= \sqrt{-2 * \ln(U2)} \\
 s1 &= \mu + \sigma * B2 * \cos(B1) \\
 s2 &= \mu + \sigma * B2 * \sin(B1) \\
 nyX(i, j) &= X(i, j) + s1
 \end{aligned}$$

$U1$ og $U2$ er tilfeldige tall mellom 0 og 1, $s1$ og $s2$ er to uavhengige normalfordelte variable med forventning μ (0) og standardavvik σ (5 og 15). I oppgaven trengte vi ikke $s2$, så droppet utregningen for denne.

b)

Deretter skal vi bruke ukorrelert multiplikativ eksponensiell støy med standardavvik 1.0 (1.3)

Med multiplikativ støy blir det betydelig mer støy enn i a), og bildet blir nesten borte i all støyen.

Vi bruker algoritme 3.2 fra kompendiet for å finne de nye verdiene:

$$U1 = \frac{-\ln(U)}{\lambda}$$



Figur 1.3: Multiplaktiv støy

$$X(i, j) = X(i, j) * U1$$

Hvor U er et tilfeldig tall mellom 0 og 1, X er bildet og λ er standardavviket.

1.2 Simulering fra diskret apriori-fordelinger

I denne oppgaven skal generere bilder ved hjelp av en apriori fordeling. Pixelverdiene i bildet kan få verdier fra settet $1, 2, 3, 4$. Fordelingen er en Gibbsfordeling på formen $p(x) \propto \exp(\frac{-E(x)}{T})$ Hvor energien $E(x) = \sum(V(x))$. $V(x)$, også kalt klikkfunksjonen, er i dette tilfellet funksjonen

$$V_I(x, z) = \begin{cases} \frac{1}{3} & \text{if } y = z, \\ -\frac{1}{3} & \text{if } y \neq z \end{cases}$$

T kalles temperaturen i Gibbsfordelingen og er gitt ved konstanten $1/\beta$

For å kunne implementere sampling fra denne fordelingen i MatLab og C++ må vi isolere alle ledd som inneholder $X_{i,j}$. Resten av leddene i summen kan sees på som konstanter da vi regner resten av bildet som kjent. Vi ender opp med følgende fordeling som skal implementeres:

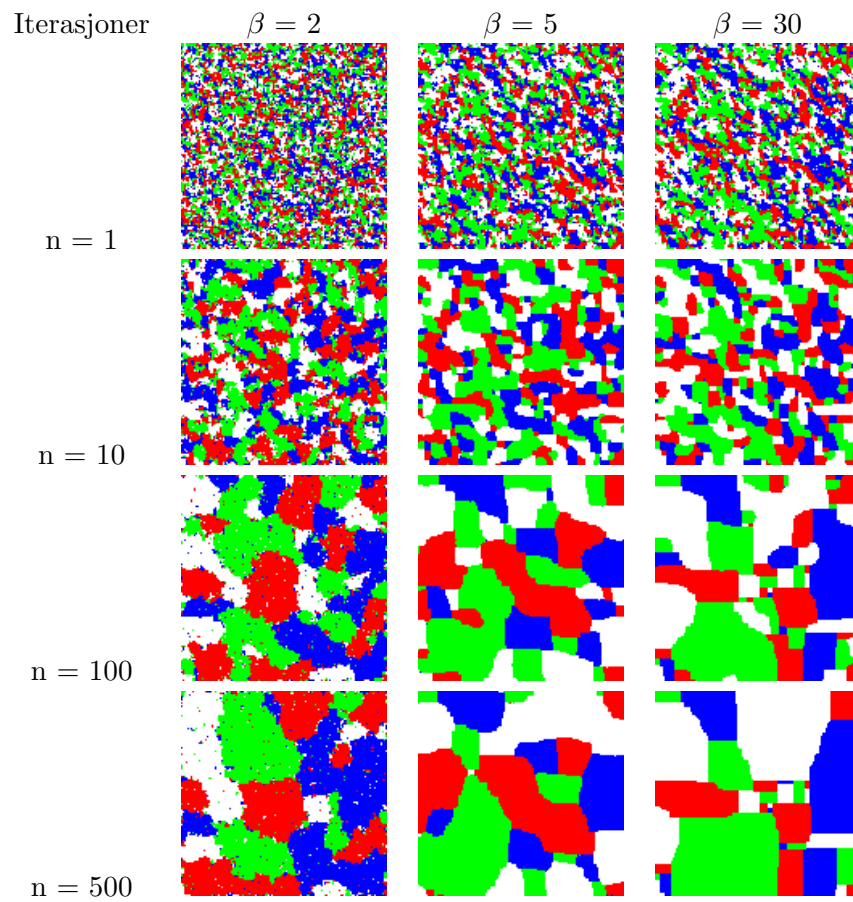
$$p(x_{i,j} | x_{k,l \neq i,j}) \propto \exp(V_I(x_{i,j}, x_{i+1,j}) + V_I(x_{i,j}, x_{i,j+1}) + V_I(x_{i-1,j}, x_{i,j}) + V_I(x_{i,j-1}, x_{i,j}))$$

a)

I denne deloppgaven skal vi sample med forskjellig antall iterasjoner og med forskjellige β . Figur 1.4 viser resultater med forskjellige verdier av n og β . Størrelsen på feltene varierer både med β og med antall iterasjoner. Feltene blir større uansett hvilken beta man bruker, men de øker raskere i størrelse og bli mer uniforme ved høye β . Ved $\beta = 1$ ser vi at bildet har en del "støy" i feltene selv etter 500 iterasjoner.

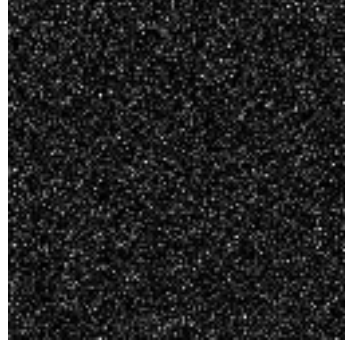
b)

Resultatene fra testingen viser at feltene blir større ved høye beta. Dette kan forklares ved å se på klikkfunksjonen $V_I(x, z) = \frac{1}{3}$ if $y = z$, $-\frac{1}{3}$ if $y \neq z$. Når et gitt pixel $x_{i,j}$ har pixelverdi som er lik naboene, vil det gi en negativ verdi ut fra klikkfunksjonen. Siden dette skal ganges med $-\beta$ for å få sansynligheten, vil stor likhet med naboene gi den høy sannsynlighet. Størrelsen på β bestemmer da hvilken vektlegging likhet med nabopixlene skal få. Høye β -verdier vil være aktuell ved restaurering av bilder med store flater. Tilsvarende vil små beta være aktuelle for restaurering av bilder med mange mindre flater.

**Figur 1.4:** Bilder fra kjøring av oppgave 2

1.3 Simulering fra kontinuerlig apriori-fordeling

Vi bruker samme initialbilde for begge deloppgavene, generert med eksponentialfordelig med standardavvik lik 1.



Figur 1.5: Initialbilde

a)

Vi begynner med fordelingen

$$P(x) \propto \exp \left(-\beta \sum_{i=1, j=1}^{127, 127} \{ (x_{i,j} - x_{i+1,j})^2 + (x_{i,j} - x_{i,j+1})^2 \} \right)$$

og trekker ut $x_{i,j}$. Da ender vi opp med

$$P(x) \propto \exp \left(-\beta \{ 4x_{i,j}^2 - 2x_{i,j}(x_{i-1,j} + x_{i,j+1} + x_{i+1,j} + x_{i,j+1}) + C \} \right)$$

Vi setter dette inn i Bayes, forkorter vekk konstantene C og får til slutt

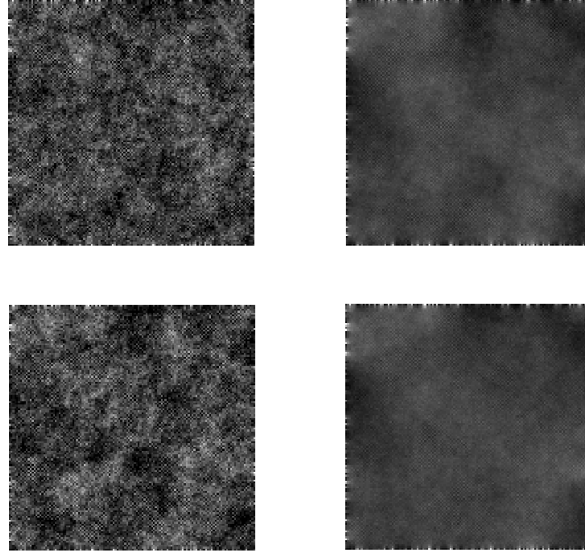
$$P(X_{i,j} | X_{k,l \neq i,j}) \propto \frac{e^{(-\beta \{ 4x_{i,j}^2 - 2x_{i,j}(x_{i-1,j} + x_{i,j+1} + x_{i+1,j} + x_{i,j+1}) \})}}{\int_{x=-\infty}^{+\infty} e^{(-\beta \{ 4(x_{i,j}=x)^2 - 2(x_{i,j}=x)(x_{i-1,j} + x_{i,j-1} + x_{i+1,j} + x_{i,j+1}) \})}} dx$$

β bestemmer hvor fort pixelverdiene jevnes ut. Ved mange iterasjoner ser vi at kantene får ganske mye å si for resultatene. Muligens kunne vi unngått denne effekten ved å legge inn en test for om vi er nær en kant, men dette har vi ikke valgt å gjøre av ytelsesgrunner.

b)

Vi går ut i fra fordelingen i oppgaven, bruker samme fremgangsmåte som i deloppgave b, og kommer fram til:

$$P(x_{i,j} | x_{k,l \neq i,j}) \propto$$

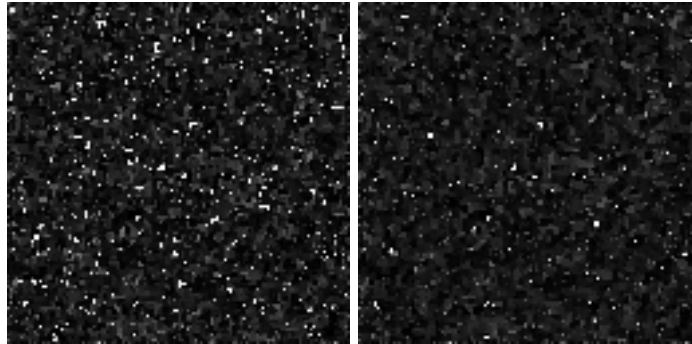


Figur 1.6: Resultatbilder. Øverst til venstre: 50 iterasjoner, 1 β . Øverst til høyre: 50 iterasjoner, 30 β . Nederst til venstre: 200 iterasjoner, 1 β . Nederst til høyre: 200 iterasjoner, 30 β .

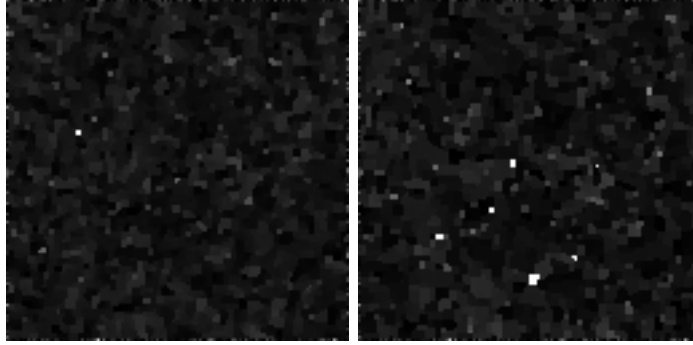
$$\frac{e^{-\beta \left\{ V_{GM}\left(\frac{x(i-1,j)-x(i,j)}{\delta}\right) + V_{GM}\left(\frac{x(i,j-1)-x(i,j)}{\delta}\right) + V_{GM}\left(\frac{x(i+1,j)-x(i,j)}{\delta}\right) + V_{GM}\left(\frac{x(i,j+1)-x(i,j)}{\delta}\right) \right\}}}{\int_{x=-\infty}^{\infty} e^{-\beta \left\{ V_{GM}\left(\frac{x(i-1,j)-x(i,j)=x}{\delta}\right) + V_{GM}\left(\frac{x(i,j-1)-x(i,j)=x}{\delta}\right) + V_{GM}\left(\frac{x(i+1,j)-x(i,j)=x}{\delta}\right) + V_{GM}\left(\frac{x(i,j+1)-x(i,j)=x}{\delta}\right) \right\}}}$$

Med denne fordelingen vil det framheves skarpe kanter. Dette skjer bare ved $0 < \gamma < 1$, hvor fordelingen virker sterkt ikke-interpolerende. Ved $\gamma \geq 1$ vil bildet fort bli utjevnet (og ubrukelig for bilderestaurering). Sterke overganger mellom regioner er viktig for å kunne skille objekter i bildet.

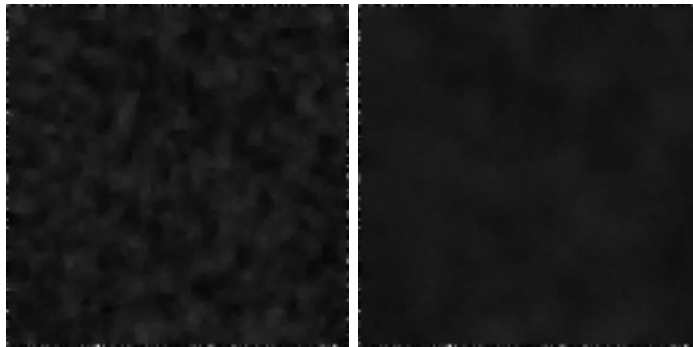
Vi er ikke helt sikker på hva spørsmålet om naturlighet i oppgaven var myn-
tet på, men det er mer naturlig å bruke fordelingen fra deloppgave b enn a i bilderestaurering.



Figur 1.7: Venstre: 10 iterasjoner, $\beta = 100$, $\gamma = 0,05$. Høyre: 10 iterasjoner, $\beta = 50$, $\gamma = 0,5$.



Figur 1.8: Venstre: 50 iterasjoner, $\beta = 50$, $\gamma = 0,5$. Høyre: 500 iterasjoner, $\beta = 100$, $\gamma = 0,05$.



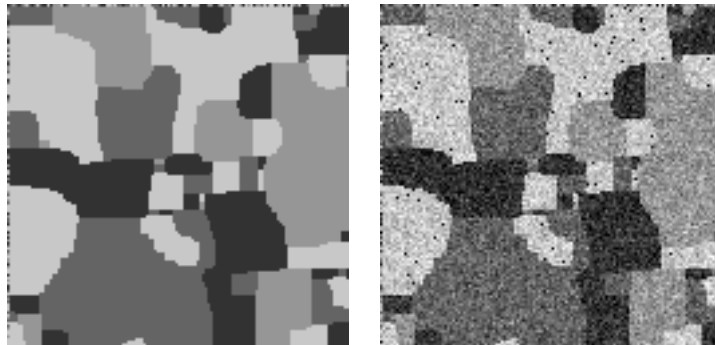
Figur 1.9: To bilder med $\gamma = 1$, for å vise utjevningseffekten. Venstre: 50 iterasjoner, høyre: 500 iterasjoner.

1.4 Simulering fra hierarkisk apriori-fordeling

Fra oppgave 2 og 3 har vi bilder vi ønsker å bruke til å simulere klassebilder.

a)

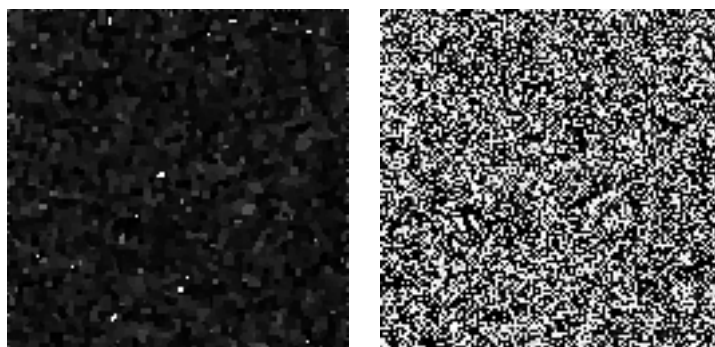
I klassebildet fra oppgave 2 ser man klart de forskjellige områdene.



Figur 1.10: Bilde fra oppgave 2 og samme bilde med Box-Muller med varierende forventningsverdi.

Vi ser at hvert av områdene har blitt støyet, og at den gjennomsnittlige gråtoneverdien har steget. For å gjøre forskjellen på områdene i bildet synlig måtte gange gråtoneverdiene med en konstant faktor. Vi fikk også noen negative gråtoneverdier som vi satte til 0.

b)



Figur 1.11: Bilde fra oppgave 3b og samme bilde med Box-Muller med varierende avvik og forventning lik 0.

Siden forventningsverdien er lik 0, så vil de nye verdiene være både positive og negative. Dette fører til at ca. halvparten av verdiene blir forkastet (negative verdier). Disse er satt til 0 for å vise bildet. Resultatet er et ugjenkjennelig bilde med bare støy.