INSTITUTT FOR INFORMATIKK, UNIVERSITETET I OSLO

Obligatorisk oppgave 1

INF2310, vår 2017

Dette oppgavesettet er på 9 sider, og består av 2 bildebehandlingsoppgaver.

Besvarelsen av denne og neste obligatoriske oppgave må være godkjent for at du skal få anledning til å gå opp til endelig skriftlig eksamen i kurset.

Besvarelsene kan utarbeides i smågrupper på opptil to studenter, men det er ikke noe i veien for å arbeide alene. Studenter i samme smågruppe kan levere identisk besvarelse, men samarbeidet må framgå av navnene på forsiden av besvarelsen. Av side 1 skal det fremgå hvem som har utarbeidet besvarelsen.

Det forventes at arbeidet er et resultat av egen innsats. Å utgi andres arbeid for sitt eget er uetisk og kan medføre sterke reaksjoner fra IFIs side. Se `http://www.uio.no/studier/admin/obligatoriske-aktiviteter/mn-ifi-oblig.html`.

Den skriftlige rapporten leveres primært som en PDF-fil som inneholder hele besvarelsen, med figurer og bilder. Kode skal leveres i tillegg til PDF-en.

Besvarelsen skal leveres via `https://devilry.ifi.uio.no`.

Legg merke til følgende:

- Alle filene må lastes opp hver gang man skal levere.
- PDF-en skal ha følgende navn: `inf2310-oblig1-brukernavn.pdf`, der `brukernavn` byttes ut med ditt eget sådan.
- Oppgaven skal kunne kjøres fra Matlab- eller Python- skript med navn: `oppgave1.m` eller `oppgave1.py` (evt. `oppgave1a.m` osv.).
- Spørsmål angående innlevering: `krisbhei@student.matnat.uio.no`.

Bildene det refereres til vil være å finne under: `http://www.uio.no/studier/emner/matnat/ifi/INF2310/v17/undervisningsmateriale/bilder/`.

Oppgaven utleveres *onsdag 01. mars 2017.*
Innleveringsfrist er *fredag 17. mars 2017.*

1 PREPROSESSERING AV PORTRETT FOR ANSIKTSGJENNKJENNING

Viktig: Dere skal her programmere transformene "fra bunnen av", altså ikke bruke ferdige programpakker.

Denne oppgaven går ut på å klargjøre (preprosessere) portrettbilder slik at det senere kan sendes til en ansiktsgjenkjenningsalgoritme.

Bildet vi skal jobbe med er `portrett.png`. Som vi ser er bildet både skjevt og det har lav kontrast. For å lettere kunne sammenligne dette bildet med andre bilder i en database, vil vi her standardisere både geometrien og kontrasten. I denne oppgaven skal vi gjøre dette på følgende måte:

1. Standardisere kontrasten ved benytte en lineær gråtonetransform spesifisert slik at resultatbildet får en middelverdi på 127 og standardavvik på 64. Bildet skal kunne lagres med 8 bit (`uint8`, altså ha verdier mellom 0 og 255.)

2. Standardisere geometrien ved å utføre en affin transform slik at øyne og munn passer over en forhåndsdefinert maske. Masken er gitt ved filen `geometrimaske.png`.

Lag en implementasjon av både forlengs- og baklengstransformasjon. Ved baklengstranformasjonen, prøv ut både nærmeste nabo og bilineær interpolasjon.

Dere skal altså ende opp med et resultatbilde som er like stort som maske-bildet (512×600 piksler), har god kontrast, og hvor øyne og munn finnes på de samme pikselkoordinatene som i maske-bildet.
NB!
Dere kan benytte ferdige Matlab/Python-funksjoner til å lese/skrive fra/til fil. Gråtonetransformen og den geometriske transformen MÅ dere implementere selv.

2 KANTDETEKSJON

I denne oppgaven skal du implementere Cannys algoritme for kantdeteksjon for a finne kanter i bildet `houses.png` (se fig. 2.1). En detaljert beskrivelse av algoritmen er å finne i Gonzalez & Woods side 719 – 725, og den består i hovedsak av:

1. Glatting av inputbildet med et gaussisk lav-pass filter.
2. Beregne gradientmagnitude og gradientvinkler fra det filtrerte bildet i punkt 1.
3. Tynning av kantene i gradientmagnitudebildet.
4. Finn sterke, "ekte" kanter ved hysteresetersкling.

For å oppnå et bra resultat til slutt, er det viktig at hver deloppgave er implementert korrekt, du blir derfor geleidet ganske tett i oppgaveteksten. I tillegg så er det en del hyperparametere som trenger fornuftige verdier, det blir foreslått omtrentlige verdier, men du står selvsagt fritt til å eksperimentere ut over det.

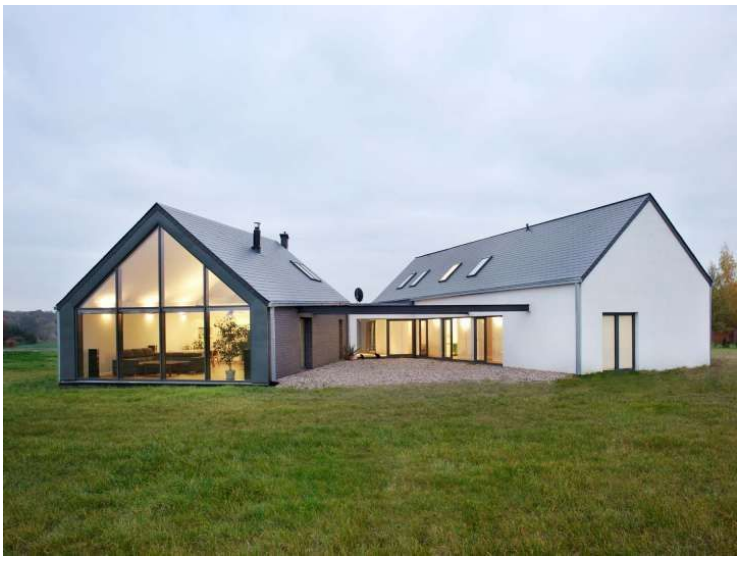

Figure 2.1: Originalt bilde `houses.png`

Du skal i denne oppgaven implementere hele Cannys algoritme "fra bunnen av", altså ikke bruke andres implementasjon av verken Cannys algoritme eller noen av dens bestanddeler, inkludert utvidelsen av innbildet, alle filtreringene, tynningen av gradientmagnitudebildet og hysteresetersklingen. Innebygde funksjoner til å plotte og lese/skrive bilder er selvsagt lov å bruke, samt innebygde funksjoner for trigonometri, kvadratrot, logaritme etc..

Generelle tips:

- Pass på hvilken datatype bildene har underveis. For eksempel, så er `uint8` begrenset til heltallsverdier i intervallet $[0,255]$, så om du gir verdier utenfor det intervallet til et bilde av typen `uint8` må du forvente uforventet oppførsel.
- Test delfunksjonene dine underveis. Du kan sjekke at implementasjonen din gir mening ved å sammenligne dine resultater mot resultater fra innebygde funksjoner. For eksempel, så skal både Matlab og Python ha lett tilgjengelige funksjoner for konvolusjon og gaussfordelingen. Om programmet ditt bruker lang tid, så kan det være lurt å bruke et mindre bilde når du utvikler programmet (for eksempel bildet `mnist_8_00011.png` fra MNIST[1]).
- Vær konsistent, og hold tunga rett i munnen når det gjelder retninger. Vi kommer bort i horisontale og vertikale gradientkomponenter, gradientvinkler, og tynning av gradientmagnitude normalt på kanten. Sørg for at du skjønner hva de forskjellige retningene er, og at de er konsistente gjennom hele oppgaven.

2.1 Generell konvolusjon

Implementer en funksjon som konvolverer et bilde med et generelt konvolusjonsfilter. Du kan anta at filteret er rektangulært med odde sidelengder. Resultatbildet skal ha samme form som inputbildet, og du kan bruke nullutvidelse (men symmetrisk utvidelse er anbefalt). Det er også sterkt anbefalt å implementere den slik at den kan håndtere separable filtre.

2.2 Cannys algoritme for kantdeteksjon

Implementer Cannys algoritme ved å følge de fire stegene nevnt ovenfor. Eksempler på delresultat gitt et sett parameterverdier vil bli gitt, men du står fritt til å velge andre verdier om du greier å finne noen bedre.

2.2.1 Gaussisk filtrering

I dette steget skal du konstruere et gaussfilter, og konvolvere inputbildet med dette filteret ved bruk av konvolusjonsfunksjonen du implementerte i kap. 2.1.

Les først inn bildet som et gråtonebilde. Gaussfilteret konstruerer du ved å sample fra den kontinuerlige gaussiske sannsynlighet-tetthets- funksjonen (*probability density function (pdf)*) med forventning 0 og standardavvik σ. En bivariat gaussisk fordeling med forventning $(0,0)$ og kovarians $\sigma^2 I$ der I er en 2×2 identitetsmatrise, er gitt av

$$p(x,y) = \frac{1}{2\pi\sigma^2} \exp\left\{-\frac{x^2+y^2}{2\sigma^2}\right\}. \tag{2.1}$$

For et 2D filter med origo i senterpikselen, og bivariat gaussisk pdf $p(x,y)$ er verdiene i filteret altså $p(x,y)$ for heltallsindekser (x,y) rundt senterpikselen $(0,0)$. Dette

[1] http://yann.lecun.com/exdb/mnist/

filteret er separabelt, der hver del kan beregnes ved å sample fra en 1D gaussisk pdf rundt 0 og med standardavvik σ

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{x^2}{2\sigma^2}\right\}. \tag{2.2}$$

Filterstørrelsen og standardavviket bestemmer resultatet av filtreringen, og et eksempel er vist i fig. 2.2 der et 11×11 filter med standardavvik $\sigma = 2$ er benyttet.

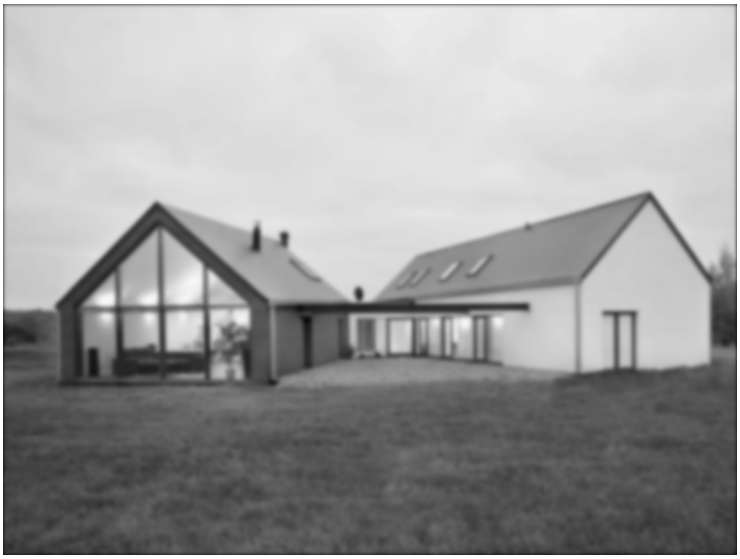

Figure 2.2: Resultat fra gaussisk filtrering.

2.2.2 Gradientmagnitude og gradientvinkel

Her skal du beregne gradientmagnituden og gradientvinkelen fra det filtrerte bildet fra kap. 2.2.1. Du står fritt til å velge hvilket gradientfilter du vil bruke. Fig. 2.3 viser et eksempel der et 3×3 Sobel filter er benyttet til å beregne gradientene.

Figure 2.3: Gradientmagnitude.

2.2.3 TYNNING AV GRADIENTMAGNITUDEBILDET

I dette steget skal man tynne kantene fra gradientmagnitudebildet ved å fjerne verdier som ikke er maksimale i et tverrsnitt normalt på kanten. For å få til dette benytter man en avrundet versjon av gradientvinkelbildet. Den følgende ligningen viser hvordan en enkelt kan transformere fra radianer til grader, runde alle vinkler til nærmeste 45 graders multiplum, og skifte alle vinkler til de to første kvadrantene

$$\hat{\theta} = \text{round}(\theta \frac{180}{\pi} \frac{1}{45}) \cdot 45 \mod 180. \tag{2.3}$$

Her er $\hat{\theta}$ den transformerte vinkelen i grader og θ er den opprinnelige vinkelen i radianer. mod er *modulo* operatoren, og round runder flyttall til nærmeste heltall, begge funksjonene er lett tilgjengelig i både Matlab og Python.

Et eksempel på et avrundet gradientmagnitudebilde er vist i fig. 2.4. Merk at koordinatsystemet er rotert slik at 0 grader er mot sør og 90 grader er mot øst, og verdier er i $\{0, 45, 90, 135\}$. Altså har horisontale kanter (vertikal gradientvinkel) verdi 0, og vertikale kanter (horisontal gradientvinkel) verdi 90.

Figure 2.4: Avrundet gradientvinkel.

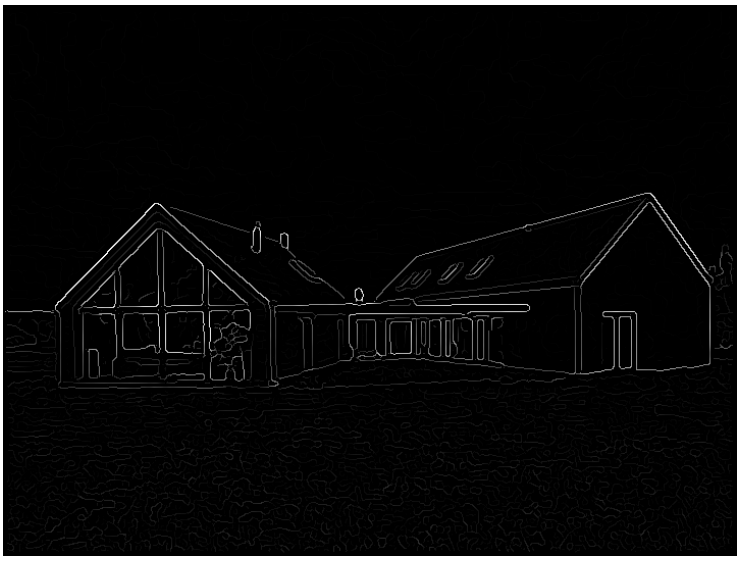

Figure 2.5: Tynnede kanter.

2.2.4 HYSTERESETERSKLING

I dette steget skal man finne de "egentlige" kantene fra det tynnede kantbildet fra kap. 2.2.3. Dette gjøres ved å dele kantbildet opp i tre. Ett som inneholder kanter større enn en terskel t_s, ett som inneholder kanter som er mindre enn t_s men større enn en terskel t_w, og ett bilde uten kanter overhode (alle verdier mindre enn t_w). Videre skal en flytte alle svake kantpiksler som er i kontakt med sterke kantpiksler over til bildet med de sterke kantene.

Fig. 2.6 viser et eksempel med $t_s = 60$ og $t_w = 30$ (og bildet med de tynnede kantene er reskalert til intervallet $[0,255]$).

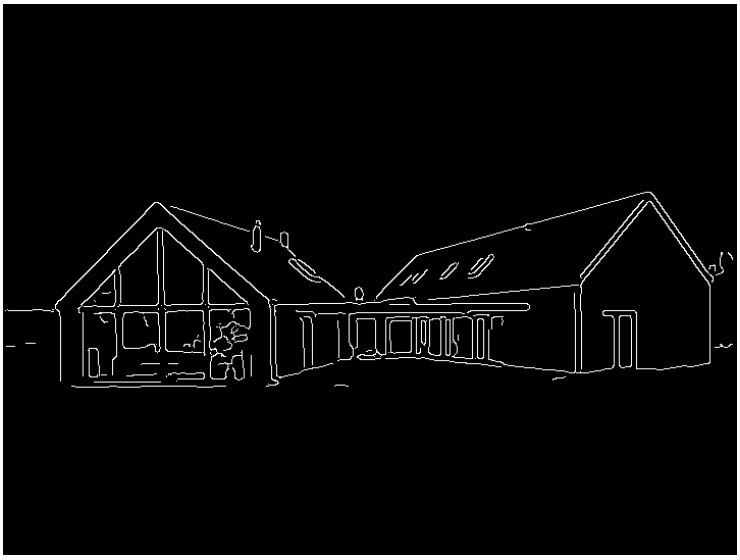

Figure 2.6: Endelig resultat

Merk at eksemplene ikke er ment som fasit, men heller som veiledning for å forsikre deg at du er på riktig vei. Det kan godt hende at du greier å få et bedre resultat.

3 HVA SKAL LEVERES

Oppgave 1

1. Mellom-resultat-bildet etter gråtonetransformen.
2. Forklaring på hvordan dere fant koeffisientene til den affine transformen.
3. Resultat-bilder for både forlengs- og baklengstranformasjonen, samt nærmeste nabo og den bilineære interpolasjonen.
4. Kommentarer/forklaringer på eventuelle forskjeller i resultatene ved forlengs- og baklengsmapping, og ved nærmeste nabo og bilineær interpolasjon.
5. Programkode.

Oppgave 2

1. Tekstlig beskrivelse av implementasjonen i kap. 2.1 og kap. 2.2.
2. Bilder fra delresultat i kap. 2.2 samt parameterverdier og implementasjonsdetaljer. Diskuter valg av parameterverdier, og hvordan de påvirker resultatene.
3. Drøfting av resultatbildet, inkludert det som kjennetegner det metoden er bra på og det den er mindre bra på, samt hvordan dette blir påvirket av å endre parameterverdiene.
4. Programkode.

Lykke til.