HDR-avbildning: rekonstruksjon og rendring

Prosjektoppgave i IMT3881 Vitenskapelig programmering

Våren 2019

1 Innledning

Ved fotografering av utendørs scener kan det dynamiske omfanget, altså forholdet mellom luminansen til lyseste og mørkeste punkt i scenen, være meget høyt. Det er ikke uvanlig at det dynamiske omfanget nærmer seg en million, 10^6. Dette gjelder f.eks. for scener med direkte sollys og områder med mørke skygger, eller fotografier som inneholder elementer både innendørs og utendørs. For billedsensorer oppgis som regel det dynamiske omfanget i EVs – «Exposure Values» – på en log2-skala, og de beste sensorene ligger nå på 14.8 EVs som gir et dynamisk omfang på $2^{14.8} \approx 28526$,[1] som jo er vesentlig lavere enn 10^6.

For å bøte på dette problemet – som riktignok er blitt mindre med årene ettersom sensorene er blitt bedre – er ulike teknikker for HDR-avbilding («High Dynamic Range Imaging») utviklet. Den vanligste teknikken, som ble utviklet av Debevec og Malik i 1997 [1], går ut på at man tar en sekvens med bilder med ulik eksponeringstid, slik at alle punkter i bildet får minst én, helst flere, gode eksponeringer der de hverken er over- eller undereksponert. Disse ulike bildene settes så sammen ved hjelp av beregningsteknikker til et HDR-bilde med mye høyere dynamisk omfang enn det sensoren kan fange i én eksponering.

Den neste utfordringen får man når slike bilder skal vises frem. En god LCD-skjerm har i dag et dynamisk omfang (for skjermer kalles den ofte statisk kontrast) på ca. 30 000, mens projektorer, for ikke snakke om skrivere, har en vesentlig lavere dynamikk. Med andre ord må HDR-bildene tilpasses enheten de skal gjengis på. Denne prosessen kalles rendring av HDR-bilder, eller «tone mapping». Det finnes to hovedkategorier rendring av HDR-bilder: global og lokal. I en global rendring finner man en funksjon («kurve») og benytter denne for alle pixler i bildet. I en lokal rendring forsøker man heller å gjenskape de lokale forskjellene i bildet uten å ta hensyn til at alle pixler nødvendigvis bør behandles likt.

I dette prosjektet skal dere få implementere og prøve ut noen vanlige metoder for HDR-avbildning; både rekonstruksjon og rendring.

[1] `https://www.dxomark.com/`

2 Rekonstruksjon av HDR-bilder

En teknikk utviklet av Debevec og Malik [1] for å rekonstruere HDR-bilder for *statiske scener*, er å ta flere bilder (med kameraet på stativ) med ulik eksponeringstid, og så sette dem sammen til et HDR-bilde («High Dynamic Range») i ettertid. Dette gjøres ved at man estimerer kameraresponskurven samtidig som man estimerer den faktiske lysheten (radiansen) i scenen som et stort, sammensatt minste-kvadraters problem. For å unngå ustabiliteter i løsningen som følge av støy i bildene, må man gjerne legge noen føringer om glatthet på kameraresponskurven i tillegg. Debevec og Maliks original-artikkel gjengir en MATLAB-kode for algortimen, så den skulle være grei å implemenentere i Python. Bare husk på at array-er i MATLAB indekseres fra 1, mens de i Python indekseres fra 0. Bruk `np.linalg.leastsq` til å løse ligningssystemet.

I praksis får man raskt problemer med størrelsen på systemmatrisen A. En måte å håndtere dette på, er å bruke færre eksponeringer og vesentlig lavere oppløsning for bildene som brukes for å genrere responskurven, for så å benytte den rekonstruerte responskurven til å rendre fullskala-bildene. Et annet praktisk problem man raskt kommer borti er at de ulike eksponeringene ikke er helt nøyaktig opplinjert og at man derfor må korrigere for dette først.

3 Global HDR-rendring

3.1 Én felles kurve per kanal

Den enkleste, men langt fra beste, måten å rendre et HDR-bilde på, er ved å anvende en global funksjon («kurve») på alle pixel-verdiene. Altså å mappe $u \mapsto f(u)$. Vanlige mappinger er logaritme, $f(u) = \ln(u)$, eller en såkalt gammafunksjon, $f(u) = u^\gamma$, der $\gamma \in (0, 1)$. Man kan også benytte andre mer avanserte funksjoner.

3.2 Luminans og kromatisitet hver for seg

Et problem med å benytte én felles kurve per kanal, er at fargene har en tendens til å bli vasket ut. Dette er fordi mettede farger kjennetegnes ved at det er store forskjeller på pixelverdiene i de ulike fargekanalene, og med en komprimering vil disse forskjellene bli mindre, og dermed også fargene bli mindre mettet. En måte å unngå dette på, er ved å dele bildet opp i luminans (lyshet) og kromatisitet (fargeinformasjon). Dersom bildet har kanalene $[R, G, B]$, kan vi for enkelhets skyld definere luminansen som $L = R + G + B$ og kromatisitet som $[R/L, G/L, B/L]$. Da kan HDR-rendringen gjøres på luminanskanalen alene, $L \mapsto f(L)$, og det rendrede fargebildet rekonstrueres som $f(L)[R/L, G/L, B/L]$.

Denne teknikken kan igjen gi en kunstig høy fargemetning. En måte å løse dette på, er å lage en (vektet) blanding av disse to løsningene.

4 Lokal HDR-rendring

Det kan være vanskelig å finne en global funksjon som yter rettferdighet til både lyse og mørke partier i bildet. For å komme videre da, må vi over på metoder som behandler pixelverdiene i bildet ulik avhengig av hvor de er og hvordan konteksten i bildet er. Her finnes det et vell av metoder; vi skal se på noen.

4.1 Lineær spatiell filtrering

En metode går ut på å skille mellom små lokale detaljer og mer storskala endringer i bildet. Storskala-endringene i bildet kan finnes ved å lavpassfiltrere det originale bildet, $u_l = F(u_0)$ (gjøre det uskarpt, altså). Detaljene vil da være differansen mellom originalbildet og det lavpassfiltrerte, $u_h = u_0 - u_l$. Man kan så gjøre en rendring bare av storskalaendringene, $f(u_l)$ og så legge tilbake detaljene, u_h, altså $u_0 \mapsto u_h + f(u_l)$. Dette kan gjøres direkte i de lineære verdiene, eller i det logaritmiske domenet, og enten i alle kanaler, eller kun i luminanskanalen.

4.2 Ikke-lineær spatiell filtrering

Problemet med lineær spatiell filtrering er at det kan oppstå artefakter (særlig «haloer») nær skarpe kanter/overganger i bildet med stor kontrast. Dette problemet kan reduseres hvis det lineære filteret byttes ut med et kantbevarende lavpassfilter, f.eks. bilateral filtrering [2]. Ellers samme metode som over. Også dette kan gjøres direkte i de lineære verdiene, eller i det logaritmiske domenet, og enten i alle kanaler, eller kun i luminanskanalen.

4.3 Komprimering i gradientdomenet

Istedenfor å gjøre endringer direkte på pixelverdiene, foreslo Fattal & al. [3] å heller komprimere *gradienten* (den deriverte) til bildet, og så rekonstruere det rendrede bildet fra den komprimerte gradienten ved hjelp av en teknikk som kalles «Poisson Image Editing» utviklet av Pérez & al. [4]. Metoden går i korthet ut på at man representerer bildet man ønsker å komme frem til som en funksjon $u : \Omega \to C$, der $\Omega \subset \mathbb{R}^2$ er det rektangulære området hvor bildet er definert, og C er fargerommet, vanligvis $C = [0, 1]$ for gråtonebilder og $C = [0, 1]^3$ for fargebilder. Bildet fremkommer som en løsning av Poisson-ligningen

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \equiv \nabla^2 u = h,$$

der randverdier på $\partial\Omega$ og funksjonen $h : \Omega \to \mathbb{R}^{\dim(C)}$ spesifiseres avhengig av hvilket problem som skal løses, og u begrenses til C.

I vårt tilfelle er da

$$h = \nabla \cdot f(\nabla u_0),$$

der $f : \mathbb{R}^2 \to \mathbb{R}^2$ er en funksjon som komprimerer *gradienten* til bildet. Som regel konstrueres f slik at det bare er lengden til gradientvektoren som endres, ikke retningen. Her kan man, som tidligere, benytte ulike komprimeringsfunksjoner for lengden, f.eks. logaritme, gamma etc.

En måte å løse Poisson-ligningen på er å iterere seg frem til løsningen vha. såkalt gradientnedstigning («gradient descent»). I praksis gjøres dette ved å innføre en kunstig tidsparameter og la løsningen utvikle seg mot konvergens:

$$\frac{\partial u}{\partial t} = \nabla^2 u - h. \tag{1}$$

Når man velger denne fremgangsmåten, må man også velge en initialverdi for bildet, f.eks. lik originalbildet, $u(x, y, 0) = u_0(x, y)$, eller resultatet av en global/lokal komprimering, som over.

To diskrete numeriske skjemaer for (1) kan finnes ved henholdsvis eksplisitt og implisitt tidsintegrasjon og sentrerte differanser for de spatielle deriverte:

$$\frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t} = \frac{1}{\Delta x^2}(u_{i+1,j}^n + u_{i-1,j}^n + u_{i,j+1}^n + u_{i,j-1}^n - 4u_{i,j}^n) - h_{i,j}, \tag{2}$$

$$\frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t} = \frac{1}{\Delta x^2}(u_{i+1,j}^{n+1} + u_{i-1,j}^{n+1} + u_{i,j+1}^{n+1} + u_{i,j-1}^{n+1} - 4u_{i,j}^{n+1}) - h_{i,j}. \tag{3}$$

Det eksplisitte skjemaet (2) er noenlunde rett frem å implementere, mens det implisitte (3) er noe mer krevende, især for fargebilder. Sistnevnte løses enklest ved å skrive det på matriseform og bruke rutiner for glisne matriser i implementasjonen.

Som randverdi er det naturlig å benytte Neumannbetingelsen $\partial u/\partial n = 0$, som kan implementeres på både forenklet og litt mer nøyaktig måte, jf. arbeidskrav 6.

4.4 Reduksjon av haloer

Da også denne metoden noen ganger kan føre til artefakter nær sterke kanter i bildet, kan også denne forbedres ved bruk av kantbevaring, jf. det bilaterale filteret over. Dette kan i praksis gjøres ved å innføre en posisjonsavhengig diffusjonsparameter $D : \Omega \to [0, 1]$, der $D = 0$ gir null diffusjon og $D = 1$ gir full diffusjon. Diffusjonsparameteren kan f.eks. beregnes fra gradienten til bildet som

$$D(x, y) = \frac{1}{1 + \kappa||\nabla u_0(x, y)||^2}\,, \tag{4}$$

med κ som en passende valgt konstant. Andre løsninger er også mulig, se [5]. Diffusjonsligningen (1) endres da til

$$\frac{\partial u}{\partial t} = \nabla \cdot (D(\nabla u - f(\nabla u_0))), \tag{5}$$

og løses for Ω med samme betingelser som over på $\partial\Omega$. Merk at det da må lages egne numeriske skjemaer for denne modifiserte ligningen. Merk også at hvis $D = 1$ overalt, vil (5) bli identisk med (1).

5 Oppgave

5.1 Praktisk gjennomføring

Prosjektet gjøres i grupper à 1–4 studenter. Én på gruppen lager en *privat* fork av Bitbucket-repositoriet, men med *arvede egenskaper* slik at kun brukeren selv og emneansvarlig har innsyn. Legg så til de øvrige gruppemedlemmene i prosjektet. La både kode og rapport bo i repositoriet (det er fremdeles mulig å lenke repositoriet til Overleaf om man skulle ønske det. Gjør hyppige og små nok commits til repositoriet til at det blir mulig å følge utviklingen av prosjektet i ettertid. Bruk gjerne grener («branches») for enklere å kunne sjekke inn («commit») kode uten å ha en fungerende løsning. Det vil altså i praksis si flere og hyppigere innsjekkinger enn man strengt tatt ville hatt i en realistisk utviklingssituasjon.

5.2 Implementasjon

Implementer et selvvalgt utvalg – jo større, desto bedre, såklart – av metodene indikert over:

- Rekonstruksjon av HDR-bilder:
  - Implementer Debevec og Maliks metode for rekonstruksjon av HDR-bilder fra et sett med gråtonebilder med ulik eksponering [1], se avsnitt 2.
  - Utvid Debevec og Maliks metode til også å fungere for fargebilder (enkel liten utvidelse).
  - Utvid Debevec og Maliks metode til også å fungere med høyoppløselige bilder.
  - Utvid Debevec og Maliks metode til å ta høyde for at bildene ikke nødvendigvis er perfekt opplinjert. Korriger for dette vha. OpenCV[2] (eksempelbildene er allerede opplinjert).

[2] `https://opencv.org/`

- Rendring av HDR-bilder. Hvis metodene for rekonstruksjon av HDR-bilder over ikke er implementert (ennå), kan dere lese inn HDR-bilder direkte fra OpenExr-filene:
  - Implementer ulike globale metoder for HDR-rendring med ulike kurver per kanal og ved å skille mellom behandling av luminans og kromatisitet, se avsnitt 3.1. og 3.2
  - Implementer lokal HDR-rendring med lineær spatiell filtrering. Bruk forskjellige kurver for storskala luminans, se avsnitt 4.1.
  - Implementer lokal HDR-rendring med bilateral spatiell filtrering. Bruk forskjellige kurver for storskala luminansen, se avsnitt 4.2.
  - Implementer det eksplisitte skjemaet (2) for Fattals metode for HDR-rendring [3] av gråtonebilder. Bruk forenklede Neumann-betingelser med null derivert for randen (jf. arbeidskrav 6).
  - Forbedre Fattal-implementasjonen med bruk av mer nøyaktige Neumann-betingelser (jf. arbeidskrav 6).
  - Utvid Fattals metode for HDR-rendring [3] til også å fungere for fargebilder (enkel liten utvidelse).
  - Implementer den forbedrede varianten av Fattals metode med lokal diffusjonsparameter. Prøv med ulike funksjoner for å bremse diffusjonen, se avsnitt 4.4.
  - Implementer det implisitte numeriske skjemaet (3) vha. glisne matriser («sparse matrices»)[3] og benytt det for alle de implementerte anvendelsene.
  - Undersøk nøyaktigheten av de implementerte løsningene vha. av metoden med konstruerte løsninger (jf. arbeidskrav 6). Se spesielt på forskjellen mellom de to implementasjonene av randverdier.
- Det virkelige liv ... :
  - Lag en applikasjon med grafisk brukergrensesnitt som gir brukeren mulighet til å utføre alle de implementerte operasjonene interaktivt. Bruk f.eks. PyQt5.[4]
  - Hvis en på gruppen har et systemkamera, ta en serie med bilder av en scene med høyt dynamisk omfang og prøv ut de implementerte metodene. Det er viktig at alt av instillinger settes til manuell kontroll (hvitbalanse, ISO, blender, eksponeringstid), og at kun eksponeringstiden varieres. Resultatet blir best hvis man lagrer bildene i et RAW-format. Dere kan bli nødt til å gjøre noe med både opplinjering og hvitbalanse, og det er jo et studium i seg selv ...

[3] `https://docs.scipy.org/doc/scipy/reference/sparse.html`
[4] `https://www.riverbankcomputing.com/software/pyqt/intro`

Det er også anledning til å gå utover det foreslåtte; et raskt litteratursøk på «HDR» og «tone mapping» skulle gi rikelig med idéer til utvidelser.

5.3 Rapport

Beskriv problemstillingen, løsningen og resultater i en velformet rapport med god og korrekt bruk av ligninger, kode, grafer, eksempelbilder, kryssreferanser og referanser. Rapporten *skal inneholde en lenke* til det forkede repositoriet.

5.4 Vurderingskriterier

Ved vurderingen av prosjektoppgaven vil det bli lagt vekt på

- hvilke metoder for HDR-rekonstruksjon, HDR-rendring, GUI etc. som er implementert
- kvalitet på koden, herunder
  - struktur og gjenbruk av kode
  - gjenbrukbarhet i form av moduler
  - dokumentasjon i form av velformede doc-strings
  - variabelnavn
  - automatiserte (enhets)tester med tilhørende rapportering (f.eks. med `coverage`)
- kvalitet på rapporten, herunder
  - struktur
  - språk
  - formler
  - referanser
  - kryssreferanser
  - figurer
  - tabeller
  - kodelistinger
- prosessen (slik den fremkommer av git-historikken)

5.5 Innlevering

Rapporten innleveres som PDF i Inspera. Rapporen *skal inneholde en lenke* til det forkede Bitbucket-repositoriet. Repositoriet må minimum få leve til etter at sensuren er gitt, og noe lenger dersom man ønsker å ha muligheten til å klage på karakteren.

Referanser

[1] P. E. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, pages 369–378, 1997.

[2] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, pages 839–846, Washington, DC, USA, 1998. IEEE Computer Society.

[3] Raanan Fattal, Dani Lischinski, and Michael Werman. Gradient domain high dynamic range compression. *ACM Transactions on Graphics (TOG)*, 21(3):249–256, 2002.

[4] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Transactions on Graphics*, 22(3):313–318, 2003.

[5] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.