

Miniprojekt: Automatisk thresholding for gråtonebilleder

af Krestina Petersen Warming, Kristian Bonde Jensen, Sigrød Boesen,

Steffen Damkjær Hansen og Tina Øvad Pedersen, rum A3-103

Produkt- og Designpsykologi

Princip

Thresholding er et princip der bygger på, at det ofte er smart at opdele et billede i to niveauer (segmentering), således at et gråtone billede laves om til et binært billede.

Et billede består af en matrix af pixels med hver deres værdi fra gråskalaen (0 - 255). Hvis man ønsker at lave et binært billede, eksempelvis for at detektere kanter på bestemte objekter, bestemmer man en repræsentativ grænseværdi for det enkelte tilfælde, eksempelvis 145. I praksis vil det således sige, at alle pixels i et billede med en værdi under 145 inkl. skal være sorte, og alle pixels med en værdi over 145 bliver hvide. På den måde får man klart optrukne linjer, hvis grænseværdien er valgt realistisk.

Automatisk thresholding er en overbygning til thresholding konceptet, hvor det handler om, at lave en algoritme, der selv kan forudsige hvilken threshold der vil være passende for det aktuelle billede. Det kan evt. være på et billede, at alle pixels er forholdsvis lyse, hvorfor grænseværdien bør være rimeligt høj (200 eksempelvis). På et andet billede kan det modsatte være gældende. I stedet for at man selv manuelt skal sidde og estimere en god grænseværdi, kan man opstille en algoritme, der ud fra enten fikserede punkter, via gennemsnitsintensitet eller medianalgoritmer etc. kan finde en god grænseværdi for forskellige billeder.

Anvendelse

Thresholding bruges i store træk til at lokalisere og tælle antal objekter i et billede, da det adskiller forgrunden i et billede fra baggrunden, som i disse tilfælde anses for at være støj. Hvilken threshold værdi, der giver det bedste binære billedresultat afhænger af det individuelle billede, hvilket gør, at det hurtigt kan blive tidskrævende at omdanne mange gråtonebilleder til binære. For at spare tid og undgå menneskelige fejl, er det derfor smart at automatisere denne beregning af den optimale threshold værdi.

Automatisk thresholding bruges indenfor adskillige områder bl.a. røntgenfotografering, MRI-scanninger, satellit fotoer, post-scanning, pengeoptælling, sikkerhedskontrol osv. hvorefter billedet analyseres videre med f.eks. BLOB, dilation eller erosion for at fjerne eventuelt støj fra billedet.

For at kunne segmentere billedet bedst muligt, er det vigtigt at histogrammet over gråtonebilledet er af bimodal karakter. Det vil sige at billedet i høj grad består af to tonerområder med stor kontrast hvilket medfører at billedets histogram har to peaks helst i hver sin ende.

En anden vigtig faktor for præcisionen af resultatbilledet efter thresholding, er lysforholdene startbilledet er taget under. Kommer der lys fra vinkler, der skaber mange store skygger kan dette bevirke at skygger eller meget oplyste områder vil blive analyseret som objekter efter thresholding. Derfor er det vigtigt, at billederne af objekterne bliver taget i lys-kontrollerede omgivelser og dermed uden dagslys.

Historie

Der eksisterer overordnet to typer thresholding. Automatisk eller adaptiv/dynamisk thresholding er den ene type, der baserer sin tærskelværdi på en del af billedet (fx fra nabopixels eller et referencebillede). Den anden type kaldes global thresholding, hvor der søges at finde én overordnet tærskelværdi, som skal gælde for hele billedet. Generelt for begge typer anvendes one stage thresholding, der finder tærskelværdien, uanset type, hurtigst muligt. Dette er gældende for billeder, der er relativt uniformt eksponeret for lys. Hvis et billede skifter meget karakter, fx en solopgang, hvor en stor del af billedet har mørke pixels, men hvor solen og det omkringliggende indeholder ret kraftigt oplyste pixels, vil en one stage thresholding ikke være optimal. I dette tilfælde vil en lokal thresholding i stedet være fordelagtig. Samlet set kan det altså betale sig at anvende lokal thresholding, hvis man kender billedets lysforhold på forhånd, men ellers vil global thresholding være at foretrække. Ulempen ved lokal thresholding kan være, at resten af billedet påvirkes negativt af denne tærskelværdi, som kun med fordel kan anvendes på en lille del af billedet, fx omkring solen. Thresholding arbejder kun med lysintensiteten og således ikke med forholdet imellem pixels, der kan være sammenhængende, fx fadende solopgang.

Løsningen på dette kan være multistage thresholding, der kan anvendes til mere komplekse billeder med blandede lysforhold. Multistage thresholding går ud på at opdele processen i to dele, hvor den første del måler en global threshold for hele billedet. Anden del foretager en lokal thresholding med udgangspunkt i de "regions of interests/features", som den globale threshold har fundet. Dette kan fx fungere således, at et billede af en solopgang med to flyvende måger på vil blive opdelt sådan, at baggrunden (himlen) bliver identificeret og segmenteret igennem global threshold og mågerne regions of interest, der kan segmenteres i anden del.



Algoritme

Overordnet set kan thresholding på et gråtone billede beskrives vha. nedenstående segmenteringsalgoritme:

$$g(x,y) = \begin{cases} 0 & \text{hvis } f(x,y) \leq T \\ 1 & \text{hvis } f(x,y) > T \end{cases}$$

Spørgsmålet er så på hvilken baggrund man vælger T , som her er threshold værdien. Som nævnt er der overordnet set to måder man kan bestemme threshold værdien ud fra, den globale og den lokale metode.

Den globale metode beror på at lave en overordnet threshold værdi, som er baseret på eksempelvis det samlede gennemsnit af hele billedet eller på median værdien for hele billedet. Umiddelbart er den globale metode ikke særligt stærk når det kommer til billeder med meget støj og forskellige lysforhold, hvorfor denne metode kun fungerer hvis man har billeder med lidt støj og en skarp opdeling af baggrund og figur.

Derfor er det mere interessant at se på de lokale metoder, hvor man opdeler billedet med henblik på at udregne T på baggrund af de enkelte dele.

En måde at gøre det på, er at analysere billedets histogram, som på x-aksen viser gråtone værdierne og på y-aksen viser frekvensen. Ved at se på histogrammet kan man således finde alle lokale maksima og derefter finde de lokale minima imellem ”toppene”. Dette kan dog også være uhensigtsmæssigt, da der kan være en del variation omkring gennemsnitsværdien, hvis der er flere, ikke klart definerede minima, hvorfor en præcis threshold er svær at identificere.

I 1979 opfandt Nobuyuki Otsu en metode til at konvertere gråtone-billeder til binære billeder ved at dele pixels op i to klasser, forgrund og baggrund, og finde den optimale threshold, der adskiller disse to. Mere teknisk vil det sige, at tærsklen beregnes således, at den samlede spredning for den interne varians i de to klasser er mindst muligt. Denne metode er af typen adaptiv thresholding, da der tages udgangspunkt i alle mulige threshold værdier for billedet samt pixelspredningen på hver side af threshold værdien. Det vil sige, at der ses på om de enkelte pixels enten falder i for- eller baggrunden. Den threshold værdi hvor summen af spredningerne er mindst, vælges. Algoritmen beskrives nedenfor først matematisk og dernæst illustreret i et eksempel ved et 4x4 billede med kun 3 gråtone niveauer:

Først regner man med at threshold værdien er 0, dernæst 1, så 2 osv. Dvs. man hele tiden arbejder med at billedet er delt op i to dele, en forgrund og en baggrund. Hver af disse ”grunde” skal køres igennem nedenstående matematiske formler:

$$Vægt = \frac{N_{g1} + N_{g2} + \dots + N_{gn}}{\text{antal pixels}}$$

$$Middelværdi \mu = \frac{(g_1 \times N_{g1}) + (g_2 \times N_{g2}) + \dots + (g_n \times N_{gn})}{N_{g1} + N_{g2} + \dots + N_{gn}}$$

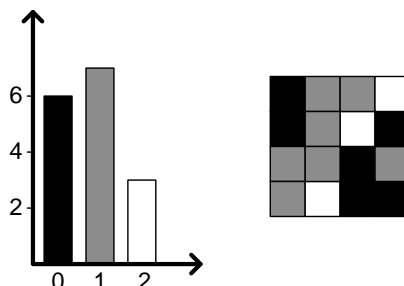
$$Varians \sigma^2 = \frac{((g_1 - \mu)^2 \times N_{g1}) + ((g_2 - \mu)^2 \times N_{g2}) + \dots + ((g_n - \mu)^2 \times N_{gn})}{N_{g1} + N_{g2} + \dots + N_{gn}}$$

Hvor g_n er en gråtone af værdien n , og N_{gn} er antallet af pixels med gråtone værdien g_n .

Når både forgrunden og baggrunden er kørt gennem disse formler, skal de til sidst bruges i samme formel, som er vist nedenfor, hvilken giver en værdi for "within-class variance". WCV skal i sidste ende bruges til at finde den rette thresholdværdi, den værdi med den mindste WCV, da det er den threshold værdi hvor der er mindst varians i de to dele af billedet.

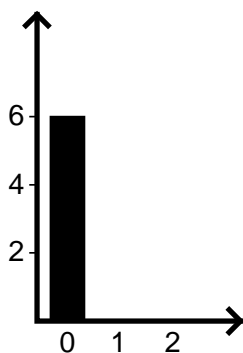
$$WCV \sigma_w^2 = V_b \times \sigma_b^2 + V_f \times \sigma_f^2$$

Et eksempel følger:



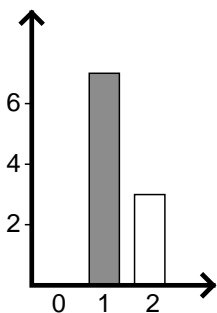
Som det ses på billedet ovenfor er der 3 gråtone niveauer, hvilket vil sige, at der er 3 mulige threshold værdier at anvende: 0, 1 eller 2. For hver af disse mulige T, køres algoritmen igennem. Som eksempel køres er T=1 igennem algoritmen.

Først regnes variansen i baggrunden, for det tilfælde at T=1, som ses på tegningen nedenfor.



$$\begin{aligned} \text{Vægt } V_b &= \frac{6}{16} = 0,375 \\ \text{Middelværdi } \mu_b &= \frac{(0 \times 6)}{6} = 0 \\ \text{Varians } \sigma_b^2 &= \frac{((0 - 0)^2 \times 6)}{6} = 0 \end{aligned}$$

Dernæst regnes variansen i forgrunden, for det tilfælde at T=1.



$$\begin{aligned} \text{Vægt } V_b &= \frac{7 + 3}{16} = 0,625 \\ \text{Middelværdi } \mu_b &= \frac{(1 \times 7) + (2 \times 3)}{7 + 3} = 1,3 \\ \text{Varians } \sigma_b^2 &= \frac{((1 - 1,3)^2 \times 7) + ((2 - 1,3)^2 \times 3)}{7 + 3} = 0,21 \end{aligned}$$

Til sidst for tilfældet T=1, regnes "Within-class variance", som er det samme som summen af varianserne ganget med deres vægtning.

$$WCV \sigma_w^2 = 0,375 \times 0 + 0,625 \times 0,21 = 0,1313$$

Dette eksempel er ikke helt illustrativt, da den eneste threshold man egentlig kan regne på her er hvis T=1. Men havde det for eksempel været et 8x8 billede med flere gråtone niveauer, så skulle man have gjort det samme for hver threshold mulighed. Når WCV er udregnet for hver T, er det den med den laveste værdi (mindste varians), der vælges som threshold værdi for hele billedet.

Vores implementering

Vi har valgt at lave et program i MatLab, som egentlig udfører de beregninger som er gennemgået ovenfor, koden ses nedenfor, inddelt i "afsnit", for delene af programmet:

```
%% Hent billede

clear
A = imread('toad.jpg');
B = rgb2gray(A);
figure(1)
imshow(B)

T = 200;
t=1;

%% lav histogram, som beregninger laves på baggrund af

for I=0:255
    H(I+1)=length(find(B==I));
    x(I+1)=I; %en x-akse laves, så plottet blive pænt
end

%% Mellemregninger baggrund og forgrund

Utal=x.*H;

for t=1:256
    Vb(t)=sum(H(1:t))/(size(B,1)*size(B,2));
    Vf(t)=sum(H(t+1:256))/(size(B,1)*size(B,2));
    Ub(t)=sum(Utal(1:t))/(sum(H(1:t))+1e-8); %1e-8 indsættes, da matlab ellers ikke fatter
                                                at 0/0 = 0, derfor indsættes et meget lille
                                                tal for at snyde den.
    Uf(t)=sum(Utal(t+1:256))/(sum(H(t+1:256))+1e-8);
end

for t = 1:256
    Sb(t)=sum(((x(1:t)-Ub(1:t)).*(x(1:t)-Ub(1:t))).*H(1:t))/(sum(H(1:t))+1e-8);
    Sf(t)=sum(((x(t+1:256)-Uf(t+1:256)).*(x(t+1:256)-
                                                Uf(t+1:256))).*H(t+1:256))/(sum(H(t+1:256))+1e-8);
end

%% Algoritmen

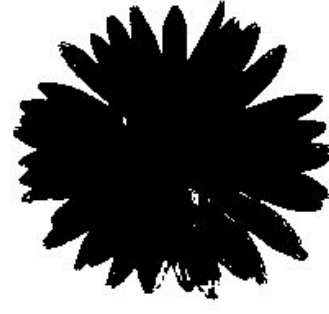
WCV = Vb.*Sb+Vf.*Sf;
T = find(WCV==min(WCV(1:255)));

%% plot med det fundne T

for i=1:size(B,1)
    for j=1:size(B,2)
        if B(i,j)>T
            C(i,j)=255;
        else
            C(i,j)=0;
        end
    end
end
figure(2)
imshow(C)
```

Resultat

Som resultat ses nedenfor 2 forskellige typer billede, der er kørt igennem programmet. Først har man et farvebillede, som MatLab kan lave om til et gråtone billede vha. funktionen `rgb2gray` . Derefter køres gråtonebilledet igennem algoritmen, således det sidste binære billede opnås.



Kilder:

Automatic Thresholding of Gray-level Using Multi stage Approach, Sue Wu and Adnan Amin, 2003 og http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MORSE/threshold.pdf fra The University of Edinburgh, Information Sciences, Bryan S. Morse, 2000

Otsus metode (figurer fra http://en.wikipedia.org/w/index.php?title=Otsu%27s_method&oldid=339348801)

