MAZILU MIHNEA MIHAI 333AA

POWER-LAW GRAY LEVEL TRANSFORMATION

Image Enhancement: este procesul care imbunatateste calitatea imaginii pentru o aplicatie specifica.

Imbunătățește practic interpretarea sau percepția informațiilor din imagini pentru spectatorii umani și oferă un acces mai bun pentru alte tehnici de procesare automată a imaginilor.

Obiectivul principal al îmbunătățirii imaginii este de a modifica atributele unei imagini pentru a o face mai potrivită pentru o sarcină dată și pentru un observator specific.

In timpul acestui proces, unul sau mai multe atribute ale imaginii sunt modificate. Alegerea atributelor și modul în care sunt modificate sunt specifice unei sarcini date.

Mai mult, factorii specifici observatorului, precum sistemul vizual uman și experiența observatorului, vor introduce o mare subiectivitate în alegerea metodelor de îmbunătățire a imaginii.

Functiile de transformare Gray-level, denumite si functii de intensitate,sunt considerate cele mai simple tehnici de imbunatatire ale imaginii

Valorile pixelilor inainte de post-procesare vor fi notate cu r si s. Aceste valori au o relatie de forma:

$$s = T(r)$$

unde T este o transformare ce mapeaza valoarea r a unui pixel intr-o valoare s.

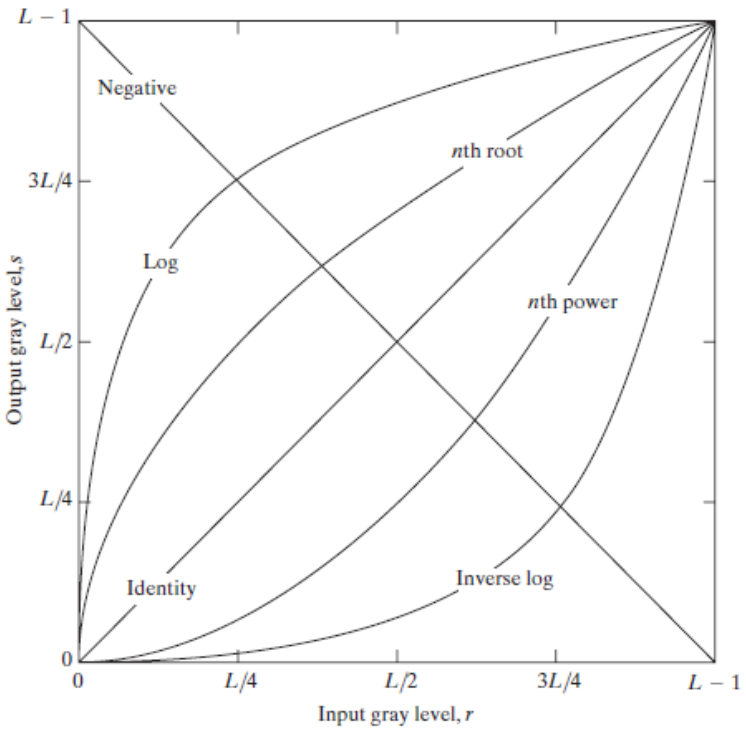

FIGURE 3.3 Some basic gray-level transformation functions used for image enhancement.

MAZILU MIHNEA MIHAI 333AA

Consideram figura prezentata care ne arata trei tipuri de functiuni de baza folosite frecvent pentru imbunatatirea imaginii.

Cele trei tipuri de functii de baza folosite frecvent pentru imbunatatirea imaginii sunt:

- **Linear Functions**:
  - Negative Transformation
  - Identity Transformation
- **Logarithmic Functions**:
  - Log Transformation
  - Inverse-log Transformation
- **Power-Law Functions**:
  - n^{th} power transformation
  - n^{th} root transformation

Transformarile Power-law au forma de baza de tipul:

$$s = cr^{\gamma}$$

unde c si γ sunt constante positive

Se obtin diferite curbe de transformare prin varierea parametrului γ (gamma):

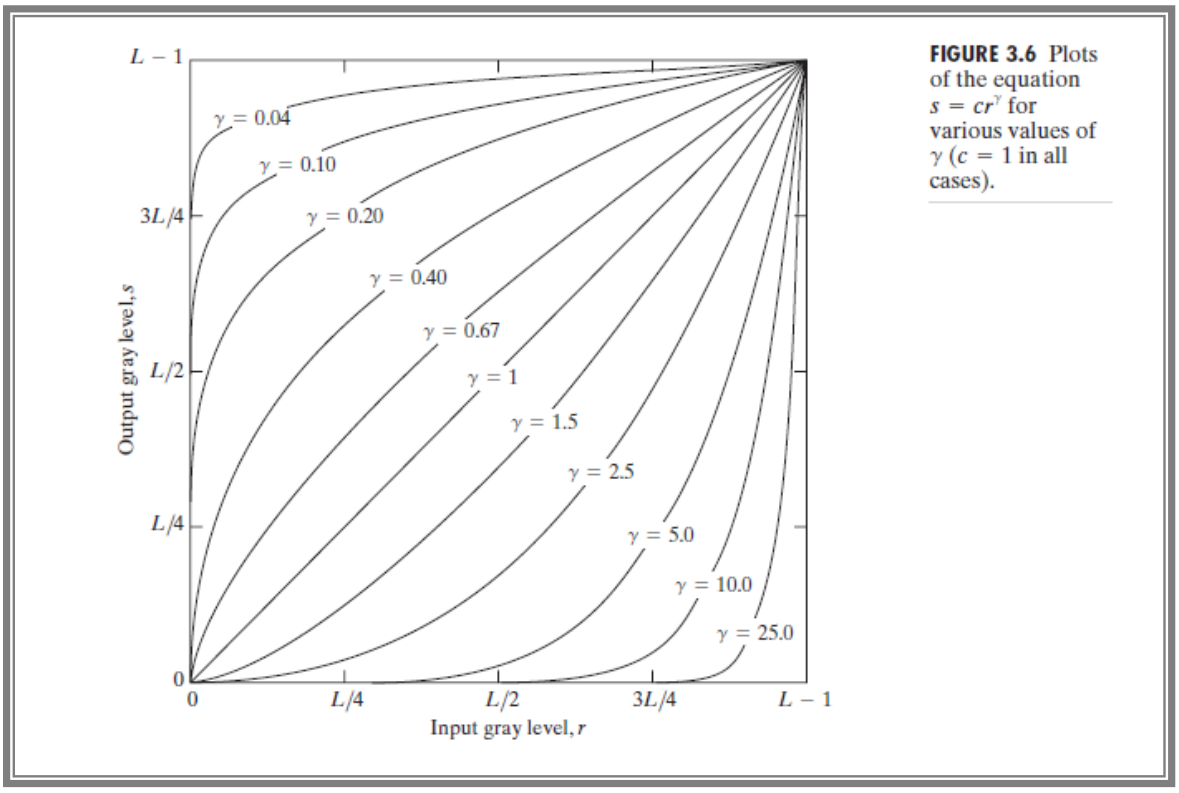

FIGURE 3.6 Plots of the equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases).

MAZILU MIHNEA MIHAI 333AA

Majoritatea dispozitivelor utilizate pentru captarea, tipărirea și afișarea imaginilor răspund în conformitate cu o lege a puterii. Procesul folosit pentru corectarea acestui fenomen de răspuns la legea puterii se numește ***corecție gamma***.

De exemplu, dispozitivele cu tuburi cu raze catodice (CRT) au un răspuns de intensitate la tensiune care este o funcție de putere, cu exponenți variind de la aproximativ 1,8 la 2,5. Cu referire la curba pentru g = 2,5 din Fig. 3.6, vedem că astfel de sisteme de afișare ar tinde să producă imagini mai întunecate decât s-a intenționat.

In completarea corectiilor gamma, transformarile power-law sunt utile pentru ajustarea contrastului imaginilor.

Descrierea aplicatiei

Aplicatia are 3 parti

- Partea I: Citirea din fisier si transferarea imaginii intre Threaduri
- Partea II: Modificarea Imaginii
- Partea III: Simularea unui cache

CITIRE SI TRANSFER INTRE THREADURI

Ca si structura prima parte este formata din 4 clase dintre care una abstracta:

- Buffer – clasa ce reprezinta modul de comunicare intre cele 2 threaduri si de asemenea locul in care se va stoca cate un sfert de imagine
- Worker – clasa abstracta ce extinde clasa Thread
- Producer – clasa ce extinde clasa Worker, este Threadul ce va face citirea si punerea informatiei in Buffer
- Consumer – clasa ce extinde tot clasa Worker, este Threadul ce va prelua informatia din Buffer si o va scrie intr-o variabila “imagineFinala”

Clasa Buffer:

- Are ca si atribut un obiect “img” de tip BufferedImage unde va stoca informatia primita
- Are un constructor fara parametrii ce initializeaza “img” cu null deoarece la inceput nu se afla nimic in Buffer
- Are o metoda set() ce seteaza “img” si o metoda get() ce transmite BufferedImage-ul

MAZILU MIHNEA MIHAI 333AA

Clasa Worker:

- Contine un atribut de tip Buffer “img” care este mostenit la fiecare clasa ce o va extinde, este de tip protected
- Contine o metoda abstracta run()

Clasa Producer:

- Contine 3 atribute: un String “path” ce ne va arata path-ul catre fisierul de unde se va citi imaginea, un Semafor “sem” ce va ajuta la sincronizare si un BufferedImage “initialImage” ce continue imaginea initiala citita din fisier
- Contine si atributul mostenit din Worker “img” de tip Buffer
- Are un contructor cu 3 parametri(cele 3 atribute)
- Si o metoda run() mostenita si suprascrisa

Clasa Consumer:

- Contine doar un atribut de tip Semafor “sem” ce ajuta la sincronizare
- Contine si atributul mostenit din Worker de tip Buffer “img”
- Are un constructor cu 2 parametri (cele 2 atribute)
- Si o metoda run() mostenita si suprascrisa

Modul de functionare:

In clasa MainClass sunt construiti 2 Workeri C si P, un Producer si un Consumer, amandoi initializati cu acelasi obiect de tip Buffer si acelasi Semafor (“buf” si “sem”).

Sunt pornite cele 2 threaduri.

Avand in vedere ca metoda run() a threadului Consumer incepe cu un wait() iar metoda run () a threadului Producer are un notify() inainte de wait(), inseamna ca primul thread care se va rula, trebuie sa fie neaparat Consumer, intrucat daca se executa Producer primul, acesta va apela notify() inainte ca Consumer sa intre in asteptare ceea ce va duce la blocare.

Pentru a ne asigura ca Consumer incepe inainte de Producer avem nevoie de un semafor initializat cu valoarea 0 care opreste threadul Producer iar cand threadul Consumer a intrat in zona critica acesta o sa incrementeze valoarea semaforului cu 1 astfel incat threadul Producer sa fie eliberat.

Odata intrat in zona critica, threadul Producer incepe sa citeasca imaginea din fisier si o salveaza in variabila “initialImage”. Apoi este creata o noua variabila auxiliara, in care se va salva doar un sfert din imaginea initiala, cu ajutorul careia vom umple Bufferul.

Dupa ce Producer seteaza Bufferul, da notify() threadului Consumer si apoi intra in asteptare.

Cand threadul Consumer este notificat ca poate continua, acesta preia din buffer sfertul de imagine si salveaza bucata cu bucata in “finalImage”.

Se repeta procesul de 4 ori dupa care cele 2 threaduri isi termina executia si se opresc.

MAZILU MIHNEA MIHAI 333AA

MODIFICAREA IMAGINII

Ca si structura, a doua parte este formata dintr-o singura clasa:

- PowerLawGrayT – clasa al carui obiect face transformarea imaginii conform algoritmului

Clasa PowerLawGrayT:

- Are 2 atribute de tip integer statice “rgb” variabila ce va reprezenta codul RGB unui pixel si variabila “gray” ce va reprezenta codul Gray al unui pixel (mai multe detalii la modul de functionare)
- metoda cu parametrii care preiau imaginea ce urmeaza sa fie modificata, path-ul catre fisierul de output si variabila gamma.
- Metoda va modifica imaginea si o va scrie in fisier

Modul de functionare:

Obiectul de tip PowerLawGrayT “funct” creat in main preia “finalImage”, variabila gamma cu ajutorul careia vom implementa algoritmul si path-ul fisierului de output.

Pe rand salvam fiecare pixel din imagine in “rgb”. Stim ca un obiect RGB este scris pe 24 de biti, fiecare 8 biti corespunzand fiecarei culori R, G, B. In fiecare culoare intra 8 biti deoarece valoarea maxima pe care poate sa o ia o culoare este 255, deci 2^8.

Ca imaginea sa fie gri rezulta ca fiecare obiect RGB va avea valorile celor 3 culori identice. Deci pe noi ne va interesa doar una dintre ele , ceea ce rezulta ca ne intereseaza doar primii 8 biti din “rgb”. Aplicand un & logic intre bitii “rgb” si 1111 1111 vom afla doar primii 8 biti.

Pe valoarea Gray aplicam formula transformarii si coeficientul gamma. Valoarea rezultata se va afla tot intre 0 si 255.

Dupa obtinerea noii valori o vom seta pe toate cele 3 culori , mai exact:

-siftam 0000 0000 cu 24 ca sa nu avem biti in plus la adunare deoarece rgb e un int pe 32 de biti

-siftam gray cu 16 biti dupa gray cu 8 biti si dupa gray cu 0 biti si adunam valorile rezultate

Dupa aceste calcule setam variabila “rgb” cu noul rezultat si setam pixelul curent al imaginii finale cu “rgb”.

MAZILU MIHNEA MIHAI 333AA

SIMULAREA UNUI CACHE

Ca si structura, a treia parte este formata din 2 clase si 2 interfete:

- AxaX – interfata ce descrie latimea imaginii salvate cat si limita acesteia
- AxaY – interfata ce descrie inaltimea imaginii salvate cat si limita acesteia
- Matrix – clasa abstracta ce implemeteaza cele 2 interfete
- Picture – clasa ce extinde clasa Matrix

Interfata AxaX:

- Are un atribut static final XMAX = 1920
- metoda ce urmeaza sa fie implementata getWidth()

Interfata AxaY:

- Are un atribut static final YMAX = 1080
- metoda ce urmeaza sa fie implementata getHeight()

Clasa Matrix:

- Are o matrice de tip String “descript” in care se va salva descrierea pozei salvate
- Are un constructor cu parametri si unul fara
- metoda de set()
- metoda abstracta display() ce va fi implemetata in clasa ce o va mosteni
- De observant ca foloseste atat in constructor cat si in metode Varargs

Clasa Picture:

- Continue matricea mostenita “descript”
- Are un constructor cu parametri
- Metodele getHeight() si getWidth() mostenite si suprascrise din interfete
- Metoda display() mostenita si suprascrisa

MAZILU MIHNEA MIHAI 333AA

SIMULAREA UNUI CACHE

Aici se observa ca am folosit atat o mostenire pe 3 nivele cat si interfete, clase abstracte si varargs.

Cache ul simulat functioneaza in urmatorul mod. In main am create un vector de Picture “cache[]” in care imaginile modificate sunt salvate daca si numai daca dimensiuniile lor corespund cu maximul permis in interfete.

De ex avem setat ca si maxim un format de 1920x1080 pixeli, daca incercam sa incarcam o imagine de dimensiune mai mare aceasta nu se va salva.

Descrierea fiecare imagini ramane chiar daca imaginea va fi salvata sau nu.

Modul de functionare este structurat dupa cum urmeaza:

Interfetele au maximul dimensiunii admise

- Clasa Matrix seteaza descrierea unei imagini

- Clasa Picture ce le mosteneste pe restul salveaza atat imaginea in sine daca nu depaseste limita cat si descrierea acesteia

In clasa Picture constructorul este apelat prin mostenire cu super(). Iar getHeight() si getWidth() retuneaza cele 2 dimensiuni ale imaginii salvate.

FUNCTIONAREA MAINCLASS

In MainClass ordinea este urmatoarea:

- Programul poate citi oricati parametri de la tastatura in felul urmator: primul parametru este gamma, al doilea este folderul de output iar urmatoarele n sunt locatia pozelor ce urmeaza sa fie modificate.
- In functie de cate imagini sunt transmise avem un for care va rula acelasi proces de un numar de ori egal cu numarul de imagini care a fost dat ca parametru in linia de comanda.
- Se pornesc cele 2 threaduri Producer Consumer si dupa ce se termina de executat se reunesc in main.
- Se aplica functia de transformare cu ajutorul obiectului “funct”.
- Daca imaginea are dimensiuniile potrivite se salveaza in vectorul cache, iar daca nu, obiectul din vectorul cache va avea valoarea NULL.
- Programul o ia de la inceput pentru urmatoarea imagine.

MAZILU MIHNEA MIHAI 333AA

REZULTATE

MAZILU MIHNEA MIHAI 333AA

BIBLIOGRAFIE

1. Image enhancement methods in the spatial domain, www.philadelphia.edu.jo › academics
2. 2.Raman Maini and Himanshu Aggarwal, A Comprehensive Review of Image Enhancement Techniques, JOURNAL OF COMPUTING, VOLUME 2, ISSUE 3, MARCH 2010, ISSN 2151-9617,HTTPS://SITES.GOOGLE.COM/SITE/JOURNALOFCOMPUTING
3. S P Vimal and P K Thiruvikraman , Automated image enhancement using power law transformations, Vol. 37, Part 6, December 2012, pp. 739–745.c Indian Academy of Sciences
4. 4.Evans Baidoo, Implementation of Gray Level Image Transformation Techniques, May 2018, International Journal of Modern Education and Computer Science 10 (5) DOI:10.5815/ijmecs.2018.05.06
5. 5.https://ocw.cs.pub.ro/courses/apd
6. https://online-converting.com/image/convert2bmp/
7. https://www.rapidtables.com/web/color/white-color.html
8. https://www.programmersought.com/article/24706288717/
9. https://onlinejpgtools.com/convert-jpg-to-grayscale
10. https://stackoverflow.com

MAZILU MIHNEA MIHAI 333AA