**Detecția segmentelor de dreaptă în imagini grayscale**

Alexandru PEREANU

1.Descrierea problemei abordate

Obiectivul acestui proiect este de a implementa un algoritm de detecție a celor mai lungi *k* segmente de dreaptă dintr-o imagine grayscale, unde *K* este un parametru introdus de la tastatură. La rulare trebuie să afișăm pe ecran imaginea inițială, imaginea cu punctele de muchie imaginea acumulatorului Hough și imaginea finală care reprezintă cele k cele mai lungi segmente suprapuse peste imaginea de test.

Cel mai important obiectiv al proiectului este acela de a reuși să se facă detecția segmentelor de dreaptă din imaginea grayscale. Ținând cont că o imagine este compusă din multe forme geometrice, noi trebuie sa deosebim segmentele de dreaptă de celelalte forme geometrice. Pentru acest lucru se vor folosi mai mulți algoritmi de specialitate.

Prin tema pe care am ales-o încerc să scot în evidență câteva modalități prin care matematica ne ajută foarte mult în procesarea de imagini.

2.Considerații teoretice

**Transformata Hough**

Principalul algoritm folosit va fi transformata Hough. Transformata Hough a fost dezvoltată de Paul Hough în 1962 și este o metodă importantă în prelucrarea imaginilor și anume o metodă de separare a liniilor și curbelor dintr-o imagine.

Transformata Hough a fost propusă de Paul Hough şi în varianta iniţială a fost o metodă de timp real pentru a număra câte puncte sunt plasate pe fiecare posibilă dreaptă într-o imagine. Această metodă se baza pe reprezentarea dreptei în forma pantă-termen liber, (y=ax+b), şi pe construirea unui spaţiu de parametric, numit şi acumulator Hough. Pentru fiecare punct de interes din imagine, se calculează toate posibilele drepte ce trec prin el, şi se incrementează elementele din spaţiul parametric. Dreptele relevante sunt localizate în maximele locale ale spaţiului parametric.

Varianta iniţială a fost orientată pe detecţia dreptelor în imagini video, pe baza reprezentării dreptelor ca pantă şi termen liber. Această reprezentare este sub-optimă, deoarece nu este mărginită: pentru a reprezenta toate posibilele drepte din imagine, panta şi termenul liber trebuie să varieze în domeniul -∞ şi +∞. Rezultatele lui Duda şi Hart au făcut transformata Hough populară în domeniul viziunii artificiale. Principala problemă, parametrii nemărginiţi, a fost rezolvată prin parametrizarea normală. Parametrizarea normală a unei drepte consistă în reprezentarea dreptei prin vectorul ce trece prin origine şi este perpendicular pe dreaptă.

Pe lângă faptul că parametrii sunt limitaţi, cuantizarea parametrilor joacă un rol important în descreşterea complexităţii computaţionale. Cuantizarea are legătură cu dimensiunea acumulatorului Hough. Pentru fiecare din cei doi parametric ai dreptei se stabileşte un nivel de cuantizare, ce depinde de acurateţea cerută (ex. acurateţea lui ρ poate fi 10, 1, 0.5 pixeli etc, iar acurateţea lui θ poate fi 10 grade, 1 grad, 0.5 grade etc). Parametrii ρ şi θ au un interval de variaţie limitat deoarece imaginea are o dimensiune finită. Valoarea maximă pentru ρ este diagonala imaginii, ρmax. În funcţie de intervalul ales pentru θ, există mai multe configuraţii echivalente pentru domeniul parametrilor.

Alegerea unui nivel de cuantizare adecvat este foarte importantă. Dacă cuantizarea este prea fină, rezoluţia creşte odată cu timpul de procesare, şi cresc şi şansele ca puncte aparent colineare să fie dea parametri în celule diferite din acumulator (acest lucru va cauza detecţii multiple ale aceleiaşi drepte, sau fragmentarea unei drepte).

**Metoda Canny**

Pentru detecția punctelor de muchie din imagine se va folosi detectorul Canny. Detectorul de margini Canny a fost dezvoltat de John F. Canny în 1986 și este un operator de detectare a marginilor care utilizează un algoritm în mai multe etape pentru a detecta o gamă largă de margini în imagini.

Detectarea marginilor Canny este o tehnică de extragere a informațiilor structurale utile din diferite obiecte de viziune și reduce dramatic cantitatea de date care urmează să fie procesate. A fost aplicat pe scară largă în diferite sisteme de viziune computerizată. Canny a constatat că cerințele pentru aplicarea detecției muchiilor pe sisteme de viziune diverse sunt relativ similare. Astfel, o soluție de detectare a muchiilor pentru a face față acestor cerințe poate fi implementată într-o gamă largă de situații.

Criteriile generale pentru detectarea muchiilor includ:

* Detectarea muchiei cu o rată de eroare scăzută, ceea ce înseamnă că detectarea ar trebui să prindă cu acuratețe cât mai multe muchii afișate în imagine.
* Punctul de muchie detectat de la operator ar trebui să localizeze cu exactitate în centrul muchiei.
* O muchie dată din imagine ar trebui marcată o singură dată și, atunci când este posibil, zgomotul imaginii nu ar trebui să creeze margini false.

Pentru a satisface aceste cerințe, Canny a folosit calculul variațiilor - o tehnică care găsește funcția care optimizează o funcțională dată. Funcția optimă în detectorul lui Canny este descrisă prin suma a patru termeni exponențiali, dar poate fi aproximată prin primul derivat al unui Gaussian.

Printre metodele de detectare a muchiilor dezvoltate până acum, algoritmul de detectare a muchiilor Canny este una dintre cele mai strict definite metode care oferă o detectare bună și fiabilă. Datorită optimității sale pentru a îndeplini cele trei criterii pentru detectarea muchiilor și simplitatea procesului de implementare, a devenit unul dintre cei mai populari algoritmi pentru detectarea muchiilor.

Algoritmul de detectare a muchiilor creat de Canny poate fi împărțit in 5 etape:

1. Aplicarea **filtrului Gaussian** pentru a netezi imaginea cu scopul de a elimina zgomotul
2. Găsirea gradienților de intensitate ai imaginii
3. Aplicarea supresiei pentru a scăpa de răspunsurile false la detectarea muchiilor
4. Aplicarea pragului dublu pentru a determina muchiile potențiale
5. Finalizarea detecției muchiilor prin suprimarea tuturor celorlalte muchii slabe si neconectate la muchiile puternice

**Filtrul Gaussian**

În procesarea imaginilor, o estompare gaussiană (cunoscută și sub denumirea de netezire gaussiană) este rezultatul estompării unei imagini de către o funcție gaussiană (numită după matematicianul și omul de știință Carl Friedrich Gauss). Este un efect utilizat pe scară largă în software-ul grafic, de obicei pentru a reduce zgomotul imaginii și a reduce detaliile.

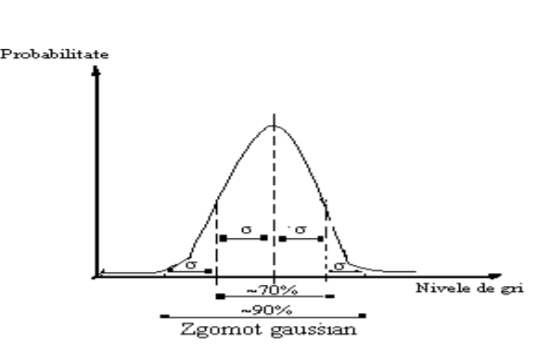
Efectul vizual al acestei tehnici de estompare este o încețoșare lină asemănătoare cu cea a vizualizării imaginii printr-un ecran translucid, diferit în mod distinct de efectul Bokeh produs de o lentilă în afara focalizării sau de umbra unui obiect sub iluminare obișnuită. Netezirea gaussiană este, de asemenea, utilizată ca etapă de pre-procesare în algoritmi de viziune computerizată pentru a îmbunătăți structurile de imagine la diferite scări.

3.Specificații de implementare

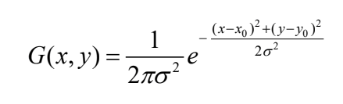
**Filtrul Gaussian**

Filtrul Gaussian este frecvent folosit în procesarea imaginilior cu scopul de a reduce zgomotul dintr-o imagine. Zgomotul este o informație care deteriorează calitatea unei imagini. El se definește ca orice proces care afectează imaginea achiziționată în memorie și nu face parte din memorie. Zgomotul din imaginile digitale poate proveni dintr-o mulțime de surse, iar acesta poate fi modelat fie printr-o histogramă sau o funcție a densității de probabilitate care se suprapune peste cea a imaginii originale.

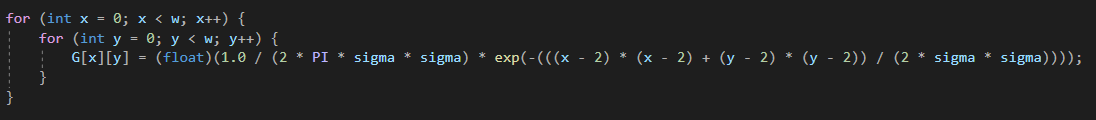
Zgomotul Gaussian este un zgomot al cărui funcție a densității de probabilitate are o forma gaussiană:



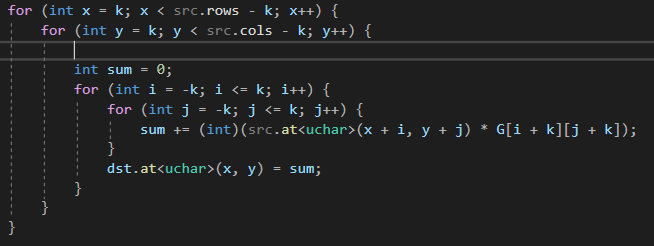
Pentru acest proiect am ales implementarea unui filtru Gaussian de dimensiune 5. Asta înseamnă că se creează o matrice de 5x5 (numită nucleu) iar valorile acestui nucleu vor fi create folosind formula specifica filtrului Gaussian și anume:



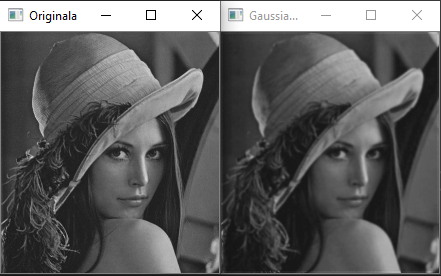
Unde (x0, y0) sunt coordonatele liniei și coloanei centrale a nucleului.



În acest fel am creat și populat toate elementele matricii nucleu, în acest proiect matricea având numele G.



Astfel, după creearea nucleului am parcurs matricea destinație și am aplicat fiecărui element al acesteia filtrul Gaussian.



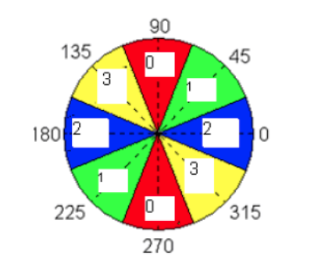
Rezultatele sunt semnificativ vizibile. Se observă cu ochiul liber că imaginea din dreapta este diferită față de imaginea din stânga, diferență fiind făcută de filtrul Gaussian. Cea de-a doua imagine reflectă aplicarea filtrului Gaussian pe prima imagine.

**Detectorul Canny**

Metoda Canny este folosită pentru detecția muchiilor dintr-o imagine. În acest proiect a fost implementată metoda Canny cu scopul de a detecta muchiile din imagine ca mai apoi să fie aplicată transformata Hough pentru detecția celor mai lungi K segmente de dreaptă.

Pentru implementarea metodei Canny s-au urmat pașii:

1. Se face filtrarea imaginii cu un filtru Gaussian de dimensiune 5
2. Se calculează modulul și direcția gradientului (se realizează convoluția dintre imaginea obținută după aplicarea filtrului Gaussian și nucleul Sobel)
3. Supresia non-maximelor sau subțierea muchiilor (fiecare valoare din imaginea rezultată la pasul anterior este comparată cu modulele celor 2 vecini corespunzători direcției)
4. Extinderea muchiilor prin histereză (se iau valorile p = 0.08 și k = 0.4)
   1. Calcularea pragului adaptiv de binarizare
   2. Binarizarea cu histereză a modulului normalizat
   3. Extinderea muchiilor (rezultatul final după aplicarea acestui pas nu este cel așteptat )
5. Rezultatul filtrării imaginii cu filtrul Gaussian este afișat mai sus.
6. Pentru realizarea convoluției am parcurs imaginea începând de la elementul cu indexul 1 (nu 0) și am parcurs-o până la elementul cu indexul (length -1). Am procedat în acest fel ca să putem realiza convoluția pe o vecinătate de N8 aplicând nucleul Sobel. Pentru ca rezultatul convoluției să fie în intervalul [0,255] s-a făcut normalizarea printr-o împărțire cu 5.65 = 4 \* sqrt(2).
7. Pentru supresia non-maximelor ne-am folosit de cercul trigonometric pentru a putea afla valoarea cuantizată a direcției gradientului.



Folosind acest cerc trigonometric am parcurs imaginea noastră (de la indexul 1 pana la indexul length-1) și am verificat în ce intervale ne aflăm. În cazul în care valoarea modulului este mai mică decât oricare dintre cele 2 valori ale vecinilor săi înseamnă că acel modul nu este un maxim local si se elimină din lista maximelor locale primind valoarea 0. Dacă valoarea modulului este mai mare decat valoarea vecinilor săi înseamnă că acesta este un maxim local.

1. Pentru extinderea muchiilor prin histereză s-au aplicat mai mulți pași. Primul dintre pași a fost declararea a două valori de tip float, și anume p și K. Pentru cele mai bune rezultate pentru setul meu de imagini de intrare s-au considerat a fi necesară valoarea lui p = 0.08, iar valoarea lui K = 0.4; După declararea acestor valori s-a calculat histograma modulului normalizat până la pasul curent. Histograma unei imagini este văzută ca un vector de apariții. Se parcurge întreaga imagine și se incrementează în vectorul histrograma valoarea pixelului din imagine la coordonatele (i,j). După aflarea și afișarea histogramei se calculează numărul de pixeli diferiți de zero care nu vor fi puncte de muchie. Numărul acestora se află după aplicarea formulei nrNonMuchie = (1-p)\*(height\*width-histograma[0]). După aflarea acestui rezultat se continuă cu calcularea pragului adaptiv însumând elementele din histograma (atenție: însumarea elementelor se face începând cu indexul 1) și oprindu-ne în momentul în care suma elementelor din histogramă este mai mare decât valoarea nrNonMuchie.

Se stabilesc cele 2 praguri de binarizare: pragInalt care ia valoarea pragului adaptiv calculat mai sus și pragJos care ia valoarea 0.4 \* pragInalt. Cu aceste valori se parcurge imaginea modulului normalizat și se realizează binarizarea cu histereză după pragurile pragInalt și pragJos. Daca valoarea modulului este mai mică decât pragJos atunci trebuia sa ia valoarea 0, daca este mai mare decât valoarea pragului Inalt atunci ia valoarea 256, altfel ia valoarea 128.

Pentru implementarea ultimei părți a metodei Canny ne vom folosi de clasa container Queue. Algoritmul este asemănător cu parcurgerea grafurilor în lățime. Ne declarăm o coadă în care să stocăm doar Puncte(i,j) pentru memorarea coordonatelor pixelilor. Se parcurge imaginea modulului rezultat dupa aplicarea tuturor pașilor de mai sus și se verifică dacă valoarea pixelului curent este 256 și dacă nu a fost vizitat atunci se adaugă coordonatele lui în coadă. Acesta se marcheaza cu vizitat și se intră într-o buclă care se va termina doar în momentul în care coada noastră va fi goală. În interiorul buclei se parcurge pe o vecinătate de N8 ai pixelului și se verifica dacă valoarea vreunui dintre vecini este 128. În caz afirmativ acest pixel va primi valoarea 256 și va fi inserat în coadă.

**Transformata Hough**

Transformata Hough este o metodă ce rezolvă o problemă clasică din viziunea artificială: găsirea dreptelor într-o imagine ce conține o mulțime de puncte de interes.

Transformata Hough definește pe P ca fiind distanța perpendiculară de la origine la linie, unde P este aflat după formula: P = x \* cos(theta) + y \* sin(theta), unde theta ia valori din intervalul [0,360] iar x și y reprezintă valorile coordonatelor pixelului current.

Pentru implementarea transformatei Hough ne creeăm un accumulator numit hough care este un vector 2D, adică o matrice. Dimensiunea acestui vector este 360 X (D + 1) unde D reprezintă diagonala imaginii primite ca parametru de intrare. Diagonala este calculate după formula D = sqrt (height \* height + width \* width). După aflarea diagonalei se începe algoritmul de popularea a acumulatorului și anume:

1. Se inițializează fiecare celulă din H cu 0
2. Pentru fiecare punct P(x,y) (punct de muchie) :
   1. Pentru fiecare theta de la 0 la 360 se calculeaza P = x \* cos(theta) + y \* sin(theta)
   2. Daca P apartine intervalului [0, D] atunci se incrementează valoarea din H[p,theta]

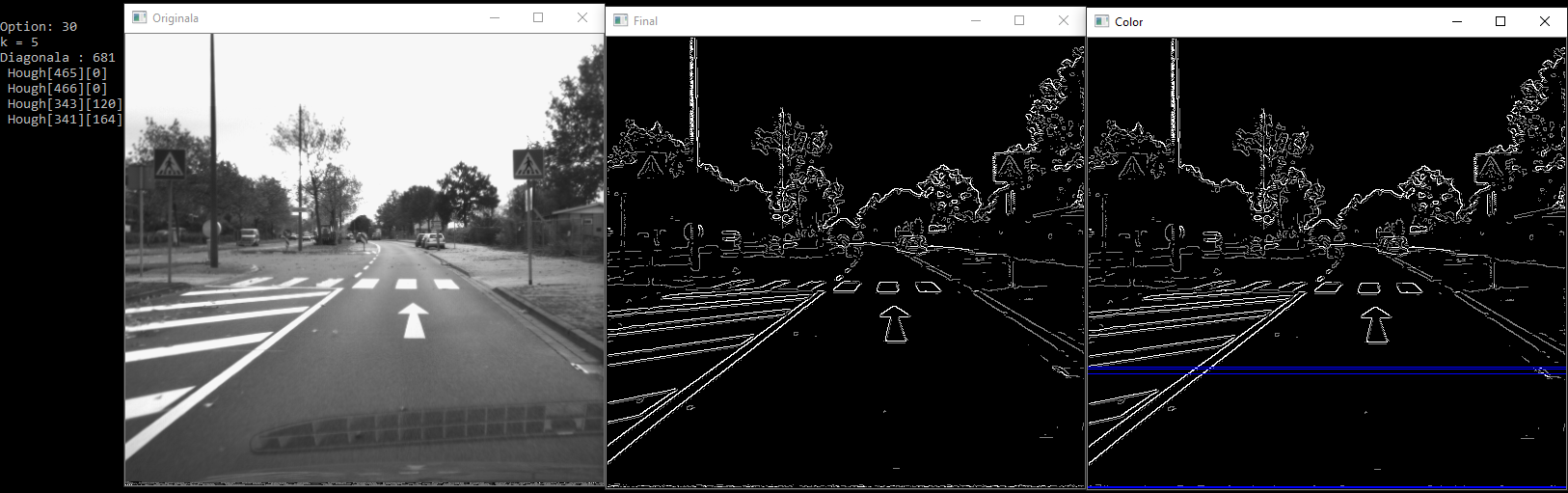
După popularea acumulatorului se verifică pentru fiecare element dacă este un maxim local într- fereastră pătrată ( 7 x7 ) centrată pe element. Acele elemente care sunt maxime locale le reținem într-o structura sub forma (P, theta, valoare), maxime locale care au valoarea mai mare decât pragul 10. Aceste elemente reținute în structura noastră vor fi sortate descrescător astfel încât să avem posibilitatea sa alegem doar primele K elemente care ne interesează, ele având cele mai mari valori dintre toate punctele care sunt maxime locale. Acestea vor fi de fapt dreptele relevante din imagine.

**Ghid de utilizare**

1. Se deschide proiectul
2. Se rulează aplicția cu CTRL+F5
3. Se va selecta opțiunea 30
4. Se selectează o imagine de test
5. Se introduce valoarea lui k. Această valoare reprezintă numărul de linii dorit a se afișa
6. În funcție de valoarea lui k se așteaptă o anumită perioadă de timp. Pentru k = 6 se va aștepta aproximativ 20 de secunde
7. După trecerea timpului se vor afișa în consolă valoarea diagonalei imaginii primite ca input, primele K elemente din structură sub forma: Hough[valoarea coordonatei x][valoarea coordonatei y]
8. Se va afișa pe ecran imaginea originală, imaginea după aplicarea metodei Canny și imaginea cu cele K linii colorate cu albastru

**4.Rezultate experimentale**

Pentru K = 5 și imaginea test1 se afiseaza urmatoarele imagini



**5.Concluzii**

Pentru acest proiect s-a dorit realizarea de detecție a segmentelor de dreaptă în imagini grayscale. Pentru imaginea primită ca și parametru la intrare s-a aplicat filtrul Gaussian pentru reducerea zgomotului, apoi s-a aplicat metoda Canny pentru detecția punctelor de muchie din imagine, s-a aplicat nucleul Sobel, s-a utilizat vecinătatea N8 la extinderea muchiilor și s-a folosit transformata Hough pentru detecția segmentelor de dreaptă.

Consider că gradul de împlinire al obiectivelor propuse este în jur de 60-70%. Am avut dificultăți în implementarea ultimului pas de la metoda Canny si consider că rezultatul nu este suficient de corect astfel încât transformata Hough sa ofere rezultate corecte.

**5.Bibliografie**

<https://docs.google.com/document/d/1-S2hPjs_qKhBLeleajzM3ffPZ1SRo3DOlMzajkdBq6s/edit> - pentru implementarea metodei Canny

<https://drive.google.com/file/d/1CHrDWVJhxKnMBXc3sMg2sS_fvf2r_ZtA/view> - pentru implementarea transformatei Hough

<https://drive.google.com/file/d/1BlXVeB6MQrgsOy6V9ftcu_B6IVlN1orY/view> - pentru implementarea filtrului Gaussian

<https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html> - pentru înțelegerea funcționării transformatei Hough

<https://www.mathworks.com/help/images/ref/houghlines.html>