Detectarea liniilor din imagini folosind Transformarea Hough

1 Introducere

În această lucrare, vom aborda problema recunoașterii liniilor în imagini. Aceasta este o problemă clasică în domeniul vederii artificiale și există mai multe tehnici populare de rezolvare. Vom prezenta una dintre aceste tehnici, care funcționează în două etape, anume mai întâi determină punctele-muchii dintr-o imagine, apoi determină liniile pe care se așază punctele-muchii.

Detectarea liniilor reprezintă un pas important în înțelegerea conținutului dintr-o imagine. Există diverse domenii în care aceste tehnici sunt deosebit de utile, precum object/pattern matching (în domeniul vederii artificiale) sau extragerea caracteristicilor și detecția anomaliilor (în domeniul învățării automate). Câteva exemple de aplicații includ recunoașterea benzilor de circulație, identificarea codurilor de bare și analiza urbanistică a structurii clădirilor.

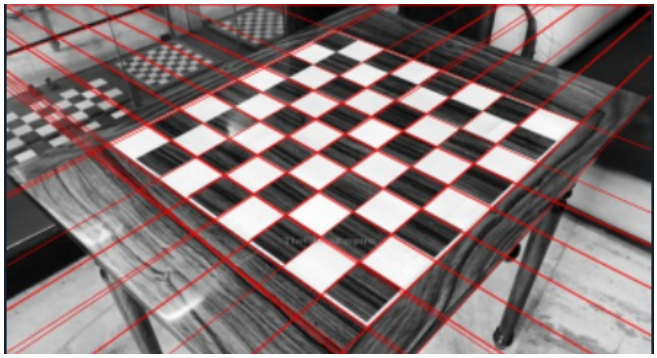


Figure 1: Set de șah

2 Strategie

Detectarea liniilor în imagini implică două subprobleme distincte, pe care le vom aborda separat. În primul rând, este necesar să identificăm muchiile (sau punctele-muchii) din imagine, acestea reprezentând pozițiile în care apare o schimbare bruscă a luminozității pixelilor. Ulterior, vom căuta liniile care sunt definite de aceste muchii.

În continuare, vom presupune că lucrăm cu imagini în tonuri de gri (grayscale). În cazul imaginilor color, se impune un pas adițional pentru conversia la grayscale. De-

sigur, rezultatele sunt identice, având în vedere că o imagine grayscale va avea aceleași contururi ca imaginea corespunzătoare color.

Deoarece ne vom concentra exclusiv pe imagini în tonuri de gri, vom utiliza termenii de luminozitate și intensitate a pixelilor în mod interschimbabil. Această mărime este reprezentată de o valoare întreagă cuprinsă între 0 și 255.

3 Identificarea muchiilor

Muchiile reprezintă schimbări bruște în luminozitatea pixelilor într-o regiune mică a imaginii. Aceste schimbări sunt determinate, în general, de:

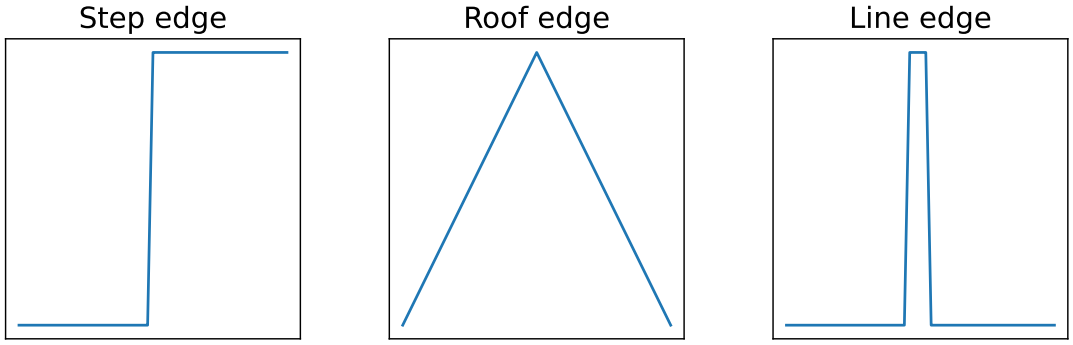
- discontinuități în adâncimea cadrului (o persoană care stă în fața unei clădiri)
- schimbări ale texturii materialelor (scrisul negru pe fundal alb, un perete vopsit în două culori)
- variație a iluminării scenei (umbrele copacilor în lumina soarelui)

În această secțiune, vom prezenta algoritmi pentru a identifica programatic regiunile din imagini care prezintă aceste caracteristici.

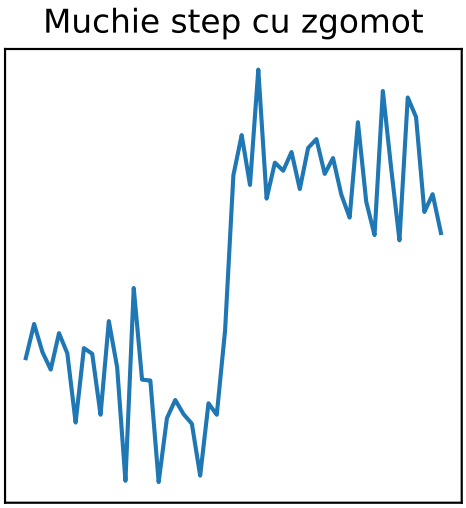
3.1 Tipuri de muchii

Nu toate muchiile care apar natural în imagini prezintă aceeași structură. Se pot clasifica în trei categorii distincte:

- Muchii step (step edges): acestea separă regiuni în care pixelii au diferite niveluri de luminozitate
- Muchii roof (roof edges): acestea separă regiuni în care o parte din pixeli prezintă o intensitate în creștere monotonă, în timp ce a doua parte prezintă o intensitate în descreștere monotonă
- Muchii linie: pixelii dintr-o regiune au o schimbare temporară în intensitate (aceasta poate fi asociată cu imagini care conțin fascicule fine, fire de iarbă, păr etc.)



În continuare, ne vom concentra în special asupra muchiilor step, deoarece acestea sunt cele mai frecvente în imagini și cele mai ușor de identificat. Imaginile sunt reprezentări discrete și cuantizate, putând fi afectate de zgomot, iar în practică, muchiile de tip treaptă nu prezintă întotdeauna forma ideală descrisă mai sus.

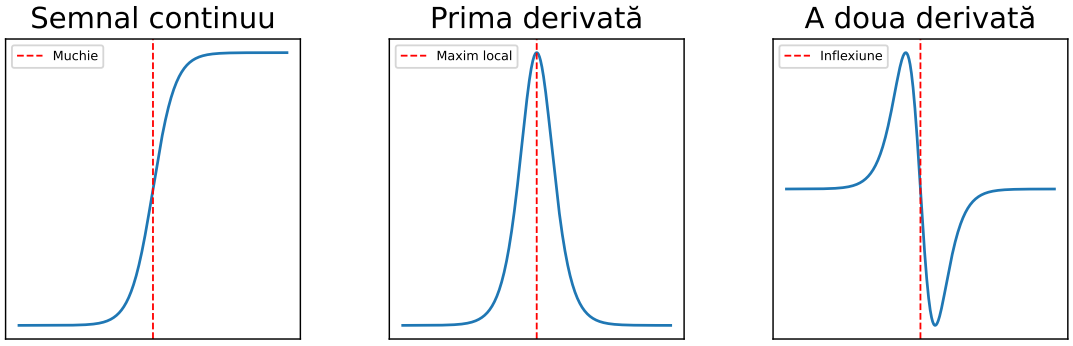


3.2 Problemă simplificată: semnal 1D continuu

Pentru a ajunge la un algoritm de identificare a muchiilor din imagini, vom rezolva mai întâi o variantă simplificată a problemei. Considerăm că avem un semnal unidimensional continuu și dorim să determinăm schimbările bruște în magnitudinea acestuia.

Magnitudinea semnalului este reprezentată de o funcție continuă f. Prima sa derivată, f', reprezintă rata de schimbare a magnitudinii semnalului în fiecare punct (real). Pentru a determina schimbările bruște ale magnitudinii, vom lua în considerare maximele și minimele locale ale primei derivate f'. Acestea sunt punctele de inflexiune ale funcției f, adică rădăcinile celei de-a doua derivate, f''.

Figura de mai jos arată cum putem folosi această metodă: creșterea bruscă în jurul punctului x=0 a unei funcții sigmoid parametrizate poate fi identificată prin maximul local al primei derivate.



3.3 Imaginea: semnal 2D discret

Putem utiliza strategia de identificare a maximelor locale ale primei derivate și în cazul imaginilor. Pentru aceasta, va trebui să trecem de la cazul 1D la 2D și să renunțăm la asumpția de continuitate.

Vom lua în considerare un semnal bidimensional f(x, y). Derivatele parțiale în raport cu x și y reprezintă rata de schimbare a semnalului în raport cu fiecare dintre cele două intrări. În cazul imaginilor, aceste derivate parțiale reprezintă rata de schimbare pe orizontală și pe verticală, respectiv.

Gradientul este vectorul determinat de cele două derivate parțiale. Acesta indică direcția celei mai rapide creșteri, iar norma sa reflectă rata de schimbare (o normă mai mare înseamnă o schimbare mai bruscă).

Cu toate acestea, imaginile sunt semnale bidimensionale discrete. Semnalul nostru f(x, y) este definit doar în punctele x, y întregi, unde x ia valori între 0 și lățimea imaginii, iar y ia valori între 0 și înălțimea imaginii.

Pentru a calcula rata de schimbare pe orizontală, avem nevoie de cel puțin 2 pixeli adiacenți pe orizontală:

$$\frac{\delta f}{\delta x}(x+0.5, y) = \frac{f(x+1, y) - f(x, y)}{\gamma} \tag{1}$$

iar pentru a calcula rata de schimbare pe verticală, avem nevoie de cel puțin 2 pixeli adiacenți pe verticală:

$$\frac{\delta f}{\delta y}(x, y+0.5) = \frac{f(x, y+1) - f(x, y)}{\gamma} \tag{2}$$

Observați că am calculat aceste derivate parțiale la poziții care se află între doi pixeli adiacenți, anume x+0.5 și y+0.5. Cu γ am notat distanța dintre doi pixeli adiacenți, și este inversa DPI (dots per inch) a display-ului pe care afișăm imaginea.

Pentru a calcula gradientul (și a obține direcția celei mai rapide creșteri) ne vom poziționa la intersecția a 4 pixeli:

$$\frac{\delta f}{\delta x}(x+0.5, y+0.5) = \frac{(f(x+1, y+1) - f(x, y+1)) + (f(x+1, y) - f(x, y))}{2 * \gamma} \tag{3}$$

$$\frac{\delta f}{\delta y}(x+0.5, y+0.5) = \frac{(f(x+1, y+1) - f(x+1, y)) + (f(x, y+1) - f(x, y))}{2 * \gamma} \tag{4}$$

În ecuațiile de mai sus, am luat în considerare media aritmetică (de unde și numitorul $2 * \gamma$) a celor două rate de schimbare pe orizontală, respectiv pe verticală, calculate cu formulele (1) și (2).

3.4 Convoluția imaginii cu un filtru

Factorul $1/(2*\gamma)$ poate fi ignorat; vom presupune că înmulțim (scalăm) toate derivatele parțiale cu aceeași constantă. Astfel, gradientul scalat devine o combinație liniară a celor patru pixeli adiacenți. Deoarece vom calcula gradientul pentru fiecare poziție din imagine, putem privi acest calcul ca pe o convoluție a imaginii cu un filtru/kernel.

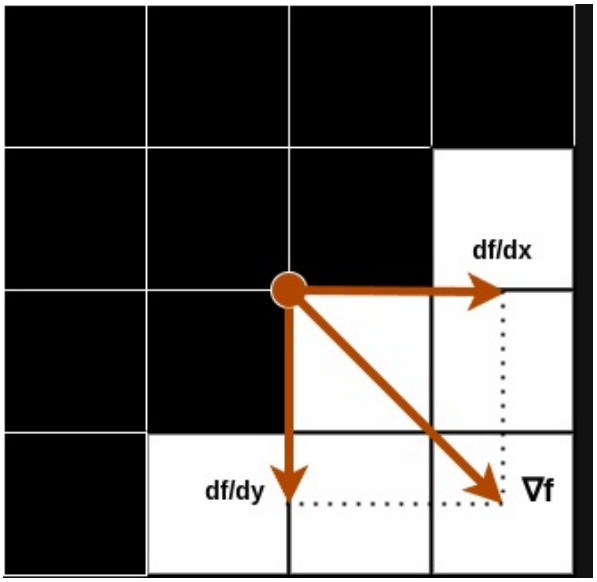
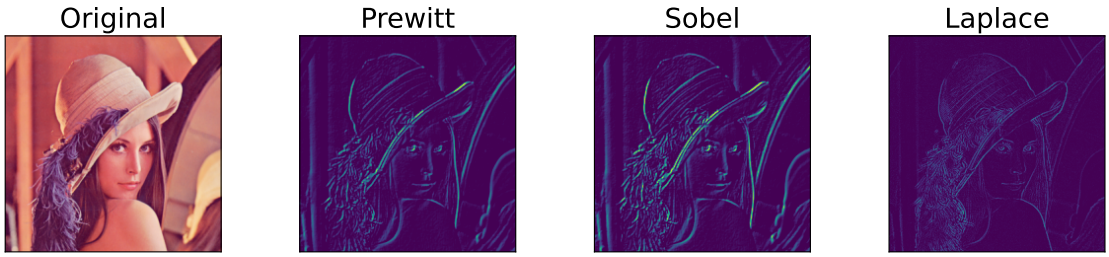


Figure 2: Exemplu de gradient la intersecția a 4 pixeli

În exemplul de mai sus, ne-am poziționat la intersecția a 4 pixeli și am calculat derivatele parțiale în raport cu pixelii adiacenți. Acestă vecinătate restrânsă de pixeli este susceptibilă la detecții false în prezența zgomotului, astfel că putem lua în considerare alte combinații liniare ale pixelilor dintr-o vecinătate mai mare, folosind filtre de dimensiuni mai mari (în general 3x3 sau 5x5). În practică, preferăm să realizăm convoluția cu un filtru pătratic de lungime impară, deoarece dacă aplicăm un padding adecvat, rezultatul convoluției va avea aceeași dimensiune cu imaginea inițială.

În figura de mai jos sunt prezentate câteva dintre filtrele cunoscute. Prewitt și Sobel sunt perechi de filtre 3x3, fiecare având un filtru pentru direcția orizontală și unul pentru cea verticală. Aceste filtre calculează independent rata de schimbare pe orizontală și pe verticală, furnizând gradientul în fiecare pixel al imaginii. Filtrul Laplacian nu conține o pereche de filtre, ci încearcă să identifice direct punctele de schimbare bruscă a intensității, fară să aducă informații privind direcția gradientului.

	Prewitt	Sobel	Laplace
Orizontal	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$
Vertical	$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$	



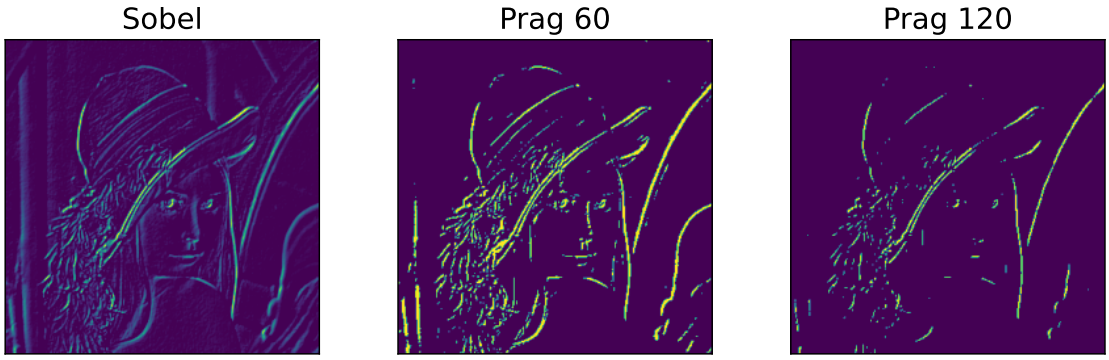
Observați structura acestor filtre. Cele care măsoară rata de schimbare pe orizontală sunt simetrice pe axa verticală, în timp ce filtrele care măsoară rata de schimbare pe verticală sunt simetrice pe axa orizontală. Practic, filtrele pe orizontală calculează o diferență ponderată între pixelii din partea dreaptă și cei din partea stângă, în timp ce filtrele pe verticală calculează o diferență ponderată între pixelii de sus și cei de jos.

Prin ajustarea ponderilor și dimensiunilor filtrelor, avem posibilitatea de a crea o varietate de filtre distincte.

3.5 Thresholding

După calcularea gradienților, pentru a obține muchiile semnificative, aplicăm un prag. Astfel, eliminăm gradienții cu normă scăzută, care nu reprezintă schimbări bruște ale luminozității.

Pragul reprezintă un hiperparametru al detectorului și trebuie setat înainte de aplicarea acestuia. Un prag mai mic poate recunoaște zgomotul ca fiind muchie, în timp ce un prag mai mare poate ignora anumite muchii.



Pentru a îmbunătăți performanța detectorului de muchii, putem utiliza două praguri, $W \leq S$. Astfel, gradienții cu normă mai mică decât W sunt neglijați, cei cu normă între W și S sunt denumiți muchii slabe, iar cei cu normă mai mare decât S sunt denumiți muchii puternice. Muchiile imaginii sunt identificate prin muchiile puternice și prin muchiile slabe care se învecinează cu cel puțin o muchie puternică. Acest proces de suprimare a muchiilor slabe este denumit Hystereses, iar un detector care implementează acest algoritm este detectorul Canny [2].

3.6 Algoritm pentru detectarea muchiilor dintr-o imagine

După ce trecem prin pașii menționați anterior, vom obține o hartă a muchiilor, adică o matrice binară de dimensiuni identice cu imaginea de intrare (lățime, înălțime), în care valorile de 1 reprezintă pixeli-muchie.

Algorithm 1 Algoritm de detectare a muchiilor O(WHlog(WH))

Data:
    Imagine I de dimensiune $W \times H$ sau $W \times H \times 3$
    Filtru Laplacian L
    Threshold muchii slabe W
    Threshold muchii puternice S
Result: Hartă a muchiilor E de dimensiune $W \times H$

if I not grayscale **then**
    $I \leftarrow$ to_grayscale(I)
$G \leftarrow$ convolve_with_filter(I, L)
$E \leftarrow$ thresholding_canny_edges(G, W, S)

4 Identificarea liniilor

După ce am identificat muchiile, următorul pas este să identificăm liniile pe care se așează aceste muchii. În continuare, atunci când menționăm un punct în planul imaginii, ne referim la un punct-muchie, obținut conform pașilor descrisi anterior (valorile de 1 din harta de muchii).

4.1 Reprezentarea în planul dual

Dualitatea punct-linie este un concept în geometrie care stabilește o corespondență între puncte și linii într-un plan. Potrivit acestui concept, fiecare punct dintr-un plan are o linie corespunzătoare în planul dual, și invers. Această dualitate reflectă relația fundamentală dintre puncte și linii în geometrie.

În cazul nostru, planul primal este planul imaginii. Fie un punct (x_i, y_i) din planul primal. Acesta are o linie corespunzătoare în planul dual: $c = -m * x_i + y_i$

Dacă reprezentăm fiecare punct din planul imaginii în planul dual, vom obține o mulțime de linii. Punctele situate pe aceeași linie în planul imaginii vor genera linii care se vor intersecta în același punct $C(m, c)$ în planul dual.

4.2 Sistemul de votare

Pentru a identifica liniile în imagine, trebuie să identificăm punctele de intersecție ale dreptelor din planul dual. Aceasta poate fi realizată prin discretizarea planului dual în celule, fiecare celulă contorizând câte drepte o intersectează.

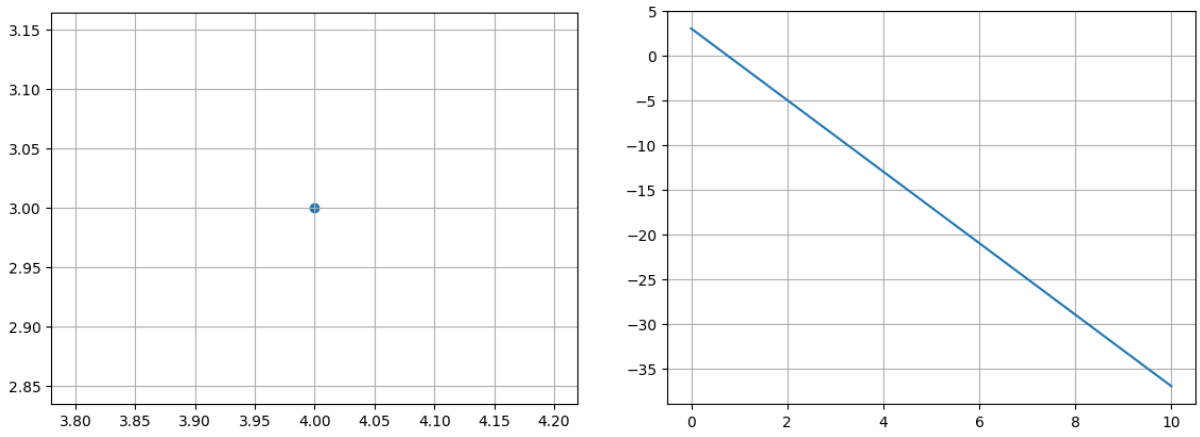


Figure 3: Planul primal și planul dual

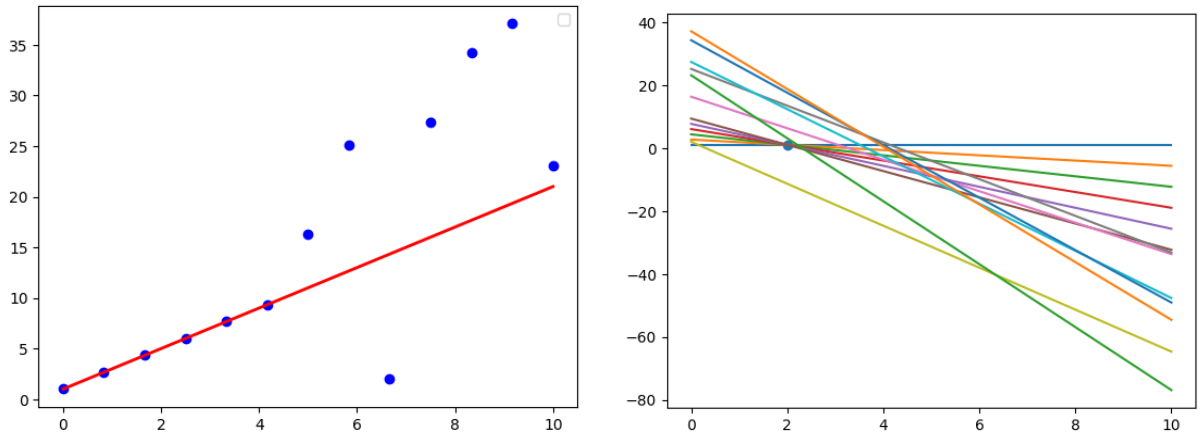


Figure 4: Planul primal și planul dual pentru puncte aparținând unei drepte

Vom utiliza o matrice de acumulare A(m, c). Pentru fiecare punct din planul imaginii, vom crește valorile din această matrice de acumulare pentru pozițiile corespunzătoare de-a lungul dreptei asociate în planul dual:

$$A(m,c) = A(m,c) + 1 \text{ dacă (m,c) aparține dreptei din planul dual} \tag{5}$$

Prin urmare, punctul (sau punctele) de intersecție ale dreptelor în planul dual vor fi reflectate de celulele care primesc cele mai multe "voturi". Deoarece fiecare punct din planul dual are ca și corespondent o dreaptă în planul primal, putem astfel identifica liniile din imagini.

Singurul pas rămas este să identificăm punctele cu un număr semnificativ de voturi din matricea de acumulare pentru a găsi liniile din imagine. Din nou, va trebui să predefinim un prag astfel încât să excludem liniile candidat care intersectează doar câteva dintre punctele-muchie, și să le luăm în considerare pe cele care intersectează un număr semnificativ de puncte muchie.

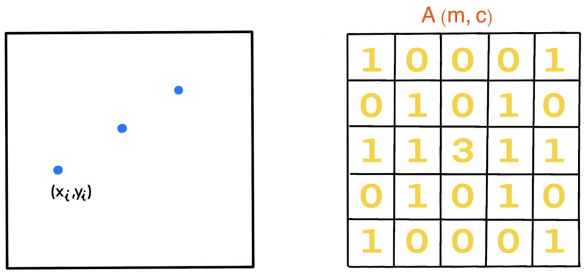


Figure 5: Incrementarea celulelor acumulatorului

4.3 Performanța reprezentării carteziene

Această modalitate de reprezentare a dreptelor prin intermediul pantei și a punctului de intersecție cu axa OY este ineficientă deoarece nu este mărginită în niciuna dintre dimensiuni. Pentru a acoperi toate direcțiile posibile ale dreptelor din imagine, atât panta cât și punctul de intersecție cu axa OY trebuie să poată varia între $-\infty$ și $+\infty$ (luând în considerare, de exemplu, cazul unei drepte verticale). Discretizarea planului dual în aceste condiții devine imposibilă.

4.4 Reprezentarea ρ - Θ

Putem rezolva problema parametrilor nemărginiți prin folosirea reprezentării polare a dreptelor, cunoscută și sub denumirea de reprezentare ρ - Θ. Aceasta implică descrierea dreptei prin intermediul unui vector care trece prin origine și este perpendicular pe dreaptă:

$x * cos\Theta + y * sin\Theta - \rho = 0$, unde:

Θ este unghiul format de dreaptă și axa absciselor

ρ este distanța față de origine

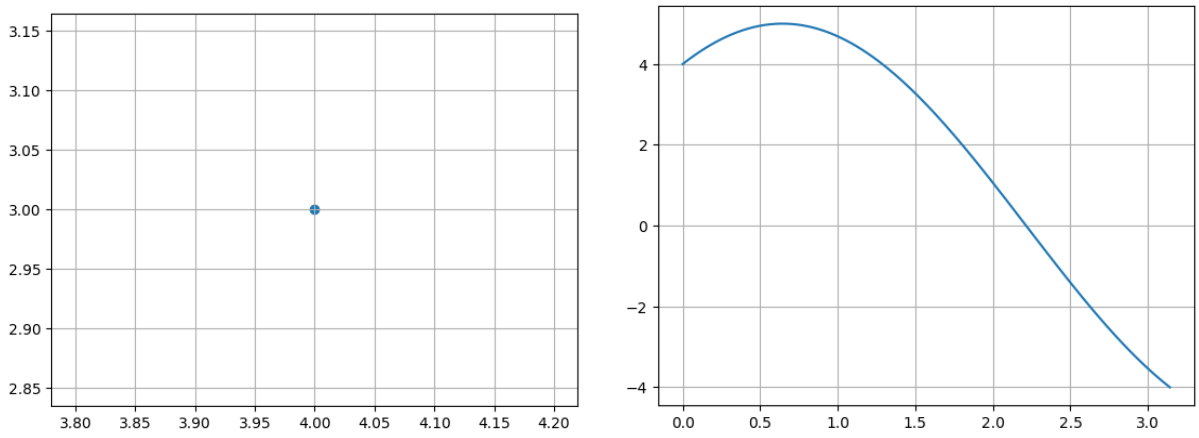


Figure 6: Planul primal si planul dual folosind reprezentarea Rho-Theta

În acest caz, orientarea Θ și distanța ρ au valori mărginite deoarece imaginea are dimensiuni finite, iar unghiul se află în intervalul $[0, \pi]$. Valoarea maximă pentru ρ este diagonala imaginii, notată ρ_{max}, iar valoarea minimă este $\rho_{min} = -\rho_{max}$.

4.5 Trecerea în planul dual folosind reprezentarea ρ - Θ

În planul dual, un punct va fi reprezentat de o sinusoidă determinată de variabilele Θ și ρ.

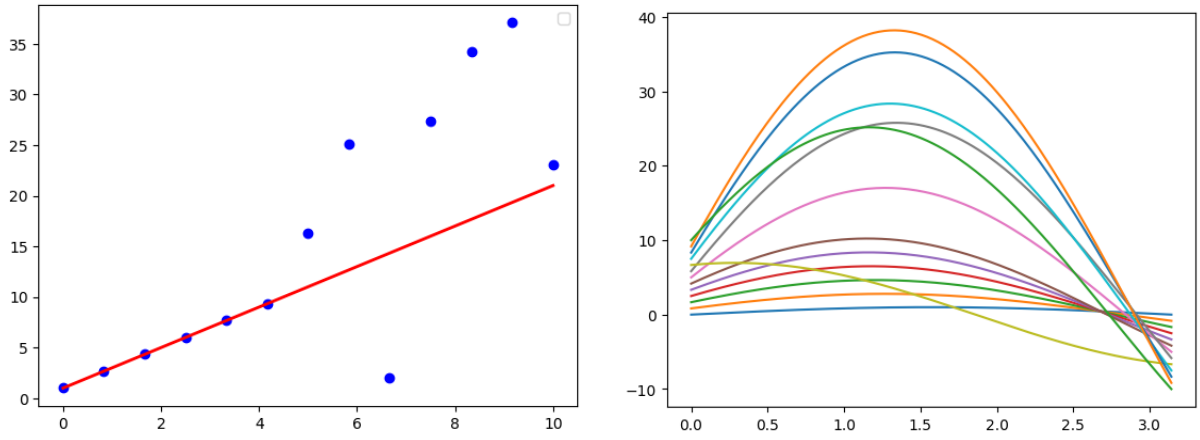
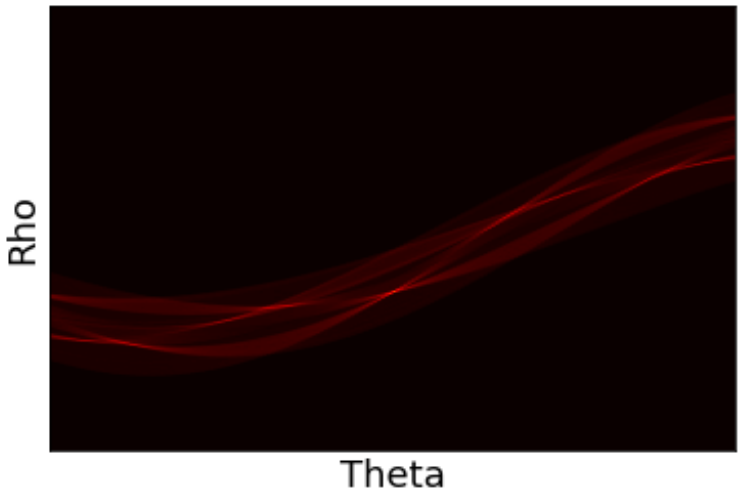


Figure 7: Planul primal și planul dual pentru puncte aparținând unei drepte (rho-theta)

Metoda de votare este identică cu cea utilizată în reprezentarea cartezianã. Vom genera o matrice de acumulare în funcție de Θ și ρ care discretizează planul dual. Punctele situate pe aceeași linie în planul imaginii vor genera sinusoide care se intersectează în mai multe puncte (funcția sin este periodică). Sistemul de votare va atribui valori maxime punctelor de pe abscisele $\Theta + k * \pi$.



4.6 Performanța reprezentării ρ - Θ

Cuantificarea parametrilor este importantă pentru reducerea complexității computaționale și pentru nivelul de finețe al reprezentării duale și este strâns legată de dimensiunea acumulatorului Hough.

Atunci când dimensiunea unei celule din matricea de acumulare este prea mare, acest lucru poate conduce la "contopirea" unor drepte diferite, rezultând într-o reprezentare mai puțin detaliată. Pe de altă parte, o dimensiune prea mică a celulei poate genera un vot dispersat, făcând dificilă identificarea unei celule reprezentative din cauza zgomotului.

Prin urmare, pentru ambii parametri se definește un nivel de cuantificare în funcție de precizia dorită. Presupunem că acumulatorul Hough, notat cu H, reprezintă spațiul parametrilor cuantizați ai dreptei. Pașii de cuantificare pentru Θ și ρ sunt, respectiv, $\Delta\Theta$ și $\Delta\rho$, iar valorile lor maxime sunt Θ_{max} și ρ_{max}. Astfel, dimensiunea acumulatorului va fi (ρ_{max}/ $\Delta\rho$ x Θ_{max}/ $\Delta\Theta$).

5 Algoritm pentru detecția liniilor dintr-o imagine

Parcurgând toți pașii descriși mai sus, obținem următorul algoritm pentru detecția liniior dintr-o hartă a muchiilor:

Algorithm 2 Algoritm pentru detecția liniilor (având harta muchiilor)

Data:
    EdgeMap E de dimensiune $W \times H$
    Parametru de discretizare pentru theta $ThetaCount$
    Threshold pentru numărul de voturi V
Result: Listă a liniilor din imagine L

$diag \leftarrow \sqrt{W^2 + H^2}$
$RhoCount \leftarrow 2 \times \text{diag} + 1$
$angles \leftarrow \text{linspace}(0, pi, ThetaCount)$
$acumulator \leftarrow$ new int[ThetaCount][RhoCount] = {0}

for i in range H **do**
    for j in range W **do**
        if $E[i][j]$ **then**
            for k in range(theta_count) **do** ▷ Un pixel-muchie votează o sinusoidă
                $\rho \leftarrow i \times \cos(angles(k)) + j \times \sin(angles(k))$
                $acumulator[\theta][\rho + diag]$ += 1

$L \leftarrow []$
for i in range H **do**
    for j in range W **do**
        if acumulator$[i][j] \geq V$ **then** ▷ Celulă cu suficiente voturi, declarăm linie
            L.append(get_cartesian_coords(i, j))
return L

Algoritmul complet pentru determinarea liniilor dintr-o imagine juxtapune cei doi algoritmi explicați mai sus.

Algorithm 3 Algoritm pentru detecția liniilor dintr-o imagine

Data:
    Imagine I de dimensiune $W \times H$ sau $W \times H \times 3$
Result: Listă a liniilor din imagine L

$edgeMap \leftarrow algorithm\_1(I)$
$lines \leftarrow algorithm\_2(edgeMap)$
return *lines*

6 Tunarea hiper-parametrilor

Observăm în cei doi algoritmi descriși mai sus (cel pentru detecția muchiilor și cel pentru detecția liniilor) existența anumitor parametri care pot influența acuratețea metodei:

- Pentru detectarea muchiilor, trebuie să stabilim un prag minim pentru norma gradientului, pentru a ignora pixelii-muchie candidați cu intensitate slabă (și în cazul detectorului Canny, avem două praguri, W și S)
- În cazul detectării liniilor, trebuie să stabilim nivelul de discretizare al planului dual și un prag pentru numărul de voturi pe care o celulă a acumulatorului trebuie să îl atingă pentru a semnala o linie

Putem crea o serie de imagini de test pentru a identifica valori convenabile ale hiper-parametrilor. Astfel, putem genera programatic imagini cu fundal monocrom, pe care adăugăm muchii sau linii cu diverse nivele de intensitate (unele mai ușor de distins față de fundal, altele mai dificil). Apoi stabilim în care imagini ar trebui să fie detectate liniile și în care nu. Experimentăm cu diferite valori ale hiperparametrilor până la obținerea unui set care satisface testele.

Pentru tunarea parametrilor vom folosi o functie de scor în funcție de numărul de pixeli corect identificați, numărul de pixeli nedetectați și numărul de pixeli incorect identificați. În algoritmul nostru am folosit următoarea formulă:

$$\text{scor} = \frac{\text{corect}}{\text{lipsă} \cdot \text{greșit}} \tag{6}$$

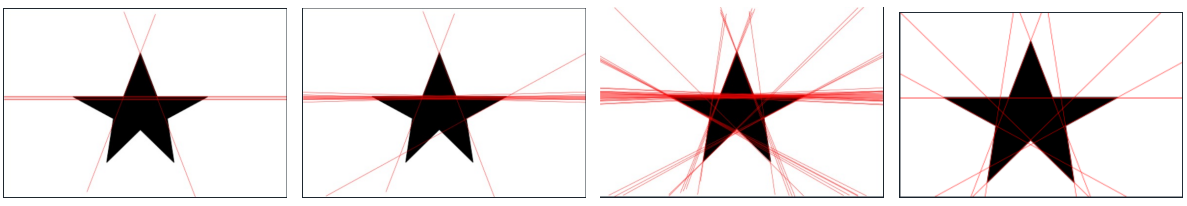


Figure 8: Diferiți pași din tunarea hiperparametrilor

Un posibil algoritm pentru tunarea parametrilor este prezentat mai jos:

Algorithm 4 Algoritm pentru tunarea parametrilor

Data:
    Set de imagini $Images$
    Set de imagini target $Targets$
    Set de thresholduri pentru gradient $ThreshGrad$
    Set de thresholduri pentru votare $ThreshVot$
Result: Parametrii optimi pentru setul de imagini GradOptim și VotOptim

for $crtgrad$ in $ThreshGrad$ **do**
    for $crtvot$ in $ThreshVot$ **do**
        for $crtimg$ in $Images.size()$ **do**
            $scor \leftarrow 0$
            $lines \leftarrow algorithm\_2(Images[crtimg], crtgrad, crtvot)$
            $scor \leftarrow scoreFunction(Targets[crtimg], lines)$
            if $scor \geq maxScor$ **then**
                $maxScor \leftarrow scor$
                $GradOptim \leftarrow crtgrad$
                $VotOptim \leftarrow crtvot$
return $GradOptim$, $VotOptim$

7 Concluzii

În acest proiect, am explorat problema recunoașterii liniilor în imagini, prezentând o tehnică în două etape: recunoașterea muchiilor dintr-o imagine, apoi identificarea liniilor pe care se așază aceste muchii. Am văzut, de asemenea, o posibilă metodă de tunare a hiper-parametrilor pentru a găsi un set de valori convenabile.

Figure 9: Set de mobilier

Transformata Hough poate fi aplicată și pentru identificarea cercurilor din imagini. Dacă ne sunt cunoscute razele cercurilor pe care dorim să le identificăm, punctele-muchii din imagine vor vota pentru centrul cercului, astfel încât, în planul dual, vom identifica intersecții de cercuri (centrul unui cerc de rază R pe care se află un punct are ca loc geometric un alt cerc de rază R).

În situația în care nu cunoaștem razele cercurilor pe care vrem să le identificăm, spațiul parametric al acumulatorului va fi tridimensional, iar în planul dual vom detecta intersecții de conuri.

O versiune generală a transformatei Hough poate identifica forme arbitrare [1].

Referințe

[1] D.H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[2] Canny Edge Detector. Canny edge detector — Wikipedia, the free encyclopedia. [Online; accessed 27-January-2024].