Eliminarea cețiii din imagini prin morfologie matematică

Proiect Prelucrarea Numerică a Imaginilor

Jakab Claudia-Angela

Ungur Andreea-Cristina

Vlad Diana-Andreea

Cuprins

1. Rezumat

Acest raport descrie proiectarea și implementarea unui sistem software pentru eliminarea ceții (dehazing) din imagini digitale. Problema ceții reduce vizibilitatea și contrastul, afectând calitatea imaginilor și performanța sistemelor de viziune computerizată.

Abordarea implementată se bazează pe modelul fizic de formare a imaginii cu ceață și utilizează o metodă hibridă. Într-o primă fază, se aplică principiul Dark Channel Prior (DCP) [3] pentru a estima lumina atmosferică globală (A) și o hartă inițială a transmisiei (t). Într-o a doua fază, harta de transmisie este rafinată folosind o secvență de operații morfologice — închidere și deschidere [1] — pentru a netezi harta și a elimina artefactele, păstrând în același timp contururile principale ale obiectelor. Algoritmul a fost integrat într-o aplicație desktop cu interfață grafică (GUI) dezvoltată în Python folosind biblioteca Tkinter.

Aplicația permite utilizatorului să încarce o imagine, să ajusteze parametrii cheie ai algoritmului (dimensiunea kernel-ului, transmisia minimă) și să vizualizeze în timp real rezultatul restaurat, alături de pașii intermediari ai procesării.

2. Stadiul actual al cercetării

Majoritatea algoritmilor de eliminare a ceții dintr-o singură imagine (single image dehazing) se bazează pe modelul optic de dispersie a luminii în atmosferă, descris de ecuația:

$$I(x) = J(x)t(x) + A(1 - t(x)) \quad (1)$$

Ecuația 1: Modelul optic de dispersie a luminii în atmosferă

unde:

- I(x) este imaginea observată (cu ceață);
- J(x) este imaginea restaurată (fără ceață);
- A este lumina atmosferică globală (o constantă);
- t(x) este harta de transmisie, care descrie fracția de lumină ce ajunge la cameră.

Obiectivul este de a estima J(x) cunoscând doar I(x). Deoarece A și t(x) sunt necunoscute, problema este neuniformă și necesită utilizarea unor *prior-uri* (presupuneri statistice) despre natura imaginilor.

Literatura de specialitate este dominată de câteva abordări majore:

1. **Metode bazate pe Prior-uri**: Acestea introduc constrângeri statistice. Cea mai influentă lucrare este Dark Channel Prior (DCP) a lui He et al. [3], care susține că în majoritatea imaginilor (cu excepția cerului), cel puțin un canal de culoare (R, G sau B) are o intensitate apropiată de zero într-o vecinătate locală. O altă abordare este cea a lui Fattal [5], care exploatează statisticile *color-lines* (liniilor de culoare).
2. **Metode bazate pe îmbunătățirea contrastului:** Acestea nu încearcă să inverseze modelul fizic, ci să maximizeze contrastul local, care este redus de ceață. Ancuti et al. [4] propun o metodă rapidă semi-inversă care fuzionează două versiuni ale imaginii pentru a îmbunătăți contrastul.
3. **Metode bazate pe învățare automată (Deep Learning):** Abordări mai recente folosesc rețele neuronale convoluționale (CNN) pentru a estima direct harta de transmisie t(x) sau chiar imaginea finală J(x).

Lucrarea de față adoptă o abordare hibridă. Se pornește de la robustețea estimărilor oferite de Dark Channel Prior [3] pentru A și t(x). Totuși, estimarea inițială a t(x) obținută prin DCP este "blocky" (prezintă artefacte sub formă de blocuri). În loc să folosească rafinarea costisitoare din punct de vedere computațional, propusă de He (soft matting) sau un filtru ghidat, acest proiect implementează o rafinare bazată pe operații morfologice, inspirată de lucrarea lui Salazar-Colores et al. [1]. Această metodă [1] folosește reconstrucția morfologică pentru a netezi harta de transmisie, oferind un echilibru excelent între viteză și calitatea rezultatului. Conceptele de bază ale procesării morfologice (eroziune, dilatare, deschidere, închidere) sunt fundamentate în lucrări clasice de procesare a imaginilor, precum Gonzalez et al. [2].

3. Fundamentare teoretică

Algoritmul implementat urmează o serie de pași logici derivați din modelul fizic și prior-ul DCP [3], cu modificările specifice pentru rafinarea morfologică [1].

3.1. Estimarea canalului întunecat (Dark Channel Prior)

Conform [3], canalul întunecat J^{dark} al unei imagini este definit ca:

$$J^{dark} = \min_{y \in \Omega(x)} \left(\min_{c \in \{R,G,B\}} J^c(y) \right) \quad (2)$$

Ecuația 2: Canalul întunecat

unde Ω(x) este o vecinătate (fereastră) centrată în pixelul x. Prior-ul DCP afirmă că $J^{dark} \to 0$ pentru imaginile fără ceață.

În implementare, acest pas corespunde funcțiilor np.min (pentru min_c) și cv2.erode (pentru $\min_{y \in \Omega(x)}$), aplicate imaginii de intrare I.

3.2. Estimarea luminii atmosferice (A)

Lumina atmosferică A este estimată tot din canalul întunecat. Se presupune că cei mai luminoși pixeli din canalul întunecat al imaginii cu ceață I^{dark} corespund celor mai dense zone de ceață.

Algoritmul selectează primii 0.1% cei mai luminoși pixeli din I^{dark}. Dintre acești pixeli candidați, se alege cel care are cea mai mare intensitate (sumă B+G+R) în imaginea originală I. Această valoare (un vector cu 3 componente) devine estimarea lui A.

3.3. Estimarea transmisiei inițiale (t1)

Pornind de la ecuația modelului fizic, normalizată cu A:

$$\frac{I^c(x)}{A^c} = \frac{J^c(x)}{A^c} t(x) + (1 - t(x)) \quad (3)$$

Ecuația 3: Estimarea transmisiei inițiale

Aplicând operația de minim local ($\min_{y \in \Omega(x)}$) și minim pe canale (min_c) pe imaginea normalizată $\frac{I(y)}{A}$, și folosind prior-ul $J^{dark} \to 0$, obținem:

$$\min_{y \in \Omega(x)} \left(\min_c \frac{I^c(y)}{A^c} \right) \approx 1 - \tilde{t}(x) \text{ unde } \tilde{t}(x) \quad (4)$$

Ecuația 4: Minimul local

unde t(x) este o estimare a transmisiei.

Rezultă transmisia inițială:

$$\tilde{t}(x) = 1 - \min_{y \in \Omega(x)} \left(\min_c \frac{I^c(y)}{A^c} \right) \quad (5)$$

Ecuația 5: Transmisia inițială

În practică, se introduce un factor ω (în cod omega, valoare tipică 0.95) pentru a păstra un mic strat de ceață, din motive de realism optic:

$$t_1(x) = 1 - \omega \cdot \min_{y \in \Omega(x)} \left(\min_c \frac{I^c(y)}{A^c} \right) \quad (6)$$

Ecuația 6: Estimarea transmisiei atmosferice

Această hartă t_1 este "blocky" deoarece este rezultatul unei operații de minim local (eroziune).

3.4. Rafinarea morfologică a transmisiei (t_refined)

Aici intervine metoda din [1]. Harta t_1 este o estimare grosieră. Pentru a o netezi, se aplică două operații morfologice [2] folosind același element structurant (kernel):

1. **Închidere morfologică (Closing):**

t2 = cv2.morphologyEx(t1, cv2.MORPH_CLOSE, kernel)

- Această operație (o dilatare urmată de o eroziune) are rolul de a "umple" găurile negre (valori mici ale transmisiei) din interiorul obiectelor care nu sunt acoperite de ceață.

2. **Deschidere morfologică (Opening):**

t_refined = cv2.morphologyEx(t2, cv2.MORPH_OPEN, kernel)

- Această operație (o eroziune urmată de o dilatare) are rolul de a elimina "petele" albe izolate (valori mari ale transmisiei) care pot fi zgomot sau artefacte.

Rezultatul t_refined este o hartă de transmisie mult mai netedă, care respectă contururile principale.

3.5. Restaurarea imaginii (J)

Imaginea finală J este recuperată prin inversarea ecuației modelului fizic:

$$J(x) = \frac{I(x)-A}{t(x)} + A \tag{7}$$

Ecuația 7: Imaginea finală J(x)

Pentru a preveni diviziunea prin zero și suprasaturarea culorilor în zonele unde transmisia este foarte mică, se impune un prag minim t_min (valoare tipică 0.85):

$$J(x) = \frac{I(x)-A}{\max(t_{refined}(x), t_{min})} + A \tag{8}$$

Ecuația 8: Restaurarea imaginii

Imaginea J rezultată este decupată (clipped) în intervalul valid [0, 255].

4. Implementare

Proiectul este structurat în două fișiere Python principale: dehaze_morphology.py (conține logica algoritmului) și main.py (conține interfața grafică).

4.1. Modulul dehaze_morphology.py

Acest modul conține funcția centrală dehaze_with_morphology și funcțiile de afișare.

Funcția dehaze_with_morphology(img_path, kernel_size, omega, t_min):

Această funcție implementează pașii teoretici descriși anterior:

1. **Încărcare și pre-procesare:**
    - Imaginea este citită de pe disc folosind cv2.imread(img_path).
    - Este convertită în RGB (pentru afișare matplotlib) și în format float64 [0, 1] (pentru calcule).
2. **Definirea elementului structurant:**
    - Se creează un kernel morfologic rectangular kernel_morph de dimensiune (kernel_size, kernel_size) folosind cv2.getStructuringElement.
3. **Pasul 1 – Canalul întunecat (DCP):**
    - min_channel = np.min(ImgFloat, axis=2): calculează minimul pe cele 3 canale de culoare.
    - dark_channel = cv2.erode(min_channel, kernel_morph): aplică eroziunea (minimul local) pentru a obține canalul întunecat.
4. **Pasul 2 – Estimarea luminii atmosferice (A):**
    - Se calculează numărul de pixeli corespunzător pragului de 0.1% (num_brightest).
    - Se aplatizează canalul întunecat (flat_dc) și se găsesc indicii celor mai luminoși pixeli (np.argsort).
    - Folosind acești indici, se extrag pixelii candidați din imaginea ImgFloat.
    - Se calculează luminozitatea (B+G+R) pentru fiecare candidat și se alege cel cu luminozitate maximă (np.argmax) ca fiind vectorul A.
5. **Pasul 3 – Transmisia inițială (t1):**
    - Imaginea ImgFloat este normalizată prin împărțire la A.

- Se aplică aceeași logică DCP pe imaginea normalizată: min_channel_normalized = np.min(normalized_img, axis=2) urmată de I_min = cv2.erode(min_channel_normalized, kernel_morph).
- Transmisia t_1 este calculată: t1 = 1.0 - (omega * I_min).

6. **Pasul 4 – Rafinarea transmisiei:**

- t2 = cv2.morphologyEx(t1, cv2.MORPH_CLOSE, kernel_morph): Aplică operația de închidere.t_refined = cv2.morphologyEx(t2, cv2.MORPH_OPEN, kernel_morph): Aplică operația de deschidere.
- t_refined = np.maximum(t_refined, t_min): Impune pragul minim t_{min}.

7. **Pasul 5 – Restaurarea imaginii (J):**

- Harta t_refined (1 canal) este extinsă la 3 canale (np.stack).
- Se aplică formula de restaurare: J = (ImgFloat - A) / transmission_map_3d + A.
- Rezultatul este adus în intervalul [0, 255] (np.clip) și convertit la uint8.
- Funcția returnează toate rezultatele intermediare (imaginea originală, DCP, t_1, $t_{refined}$ și imaginea finală J_restored_rgb.

Modulul conține și funcțiile plot_morph_ops, plot_gray_hist și plot_dehaze_results, care folosesc matplotlib pentru a genera figurile solicitate în aplicație.

4.2. Modulul main.py (Interfața Grafică)

Acest fișier definește clasa DehazeGUI care construiește și gestionează interfața Tkinter.

- **Structura ferestrei:** Fereastra principală este împărțită în două:
  - Panoul stânga (left_frame): Conține toate controalele.
  - Panoul dreapta (right_frame): Conține o pânză (canvas) matplotlib care afișează imaginea originală și cea restaurată, pentru comparație directă.
- **Controale (Panoul din stânga):**
  - btn_choose: Buton pentru selectarea imaginii (filedialog.askopenfilename).
  - scl_tmin (Slider) și lbl_tmin_val (Label): Permit selectarea parametrului t_min (transmisie minimă) și afișarea valorii sale.
  - spn_kernel (Spinbox): Permite alegerea kernel_size (doar valori impare).
  - ent_omega (Entry): Permite introducerea parametrului omega.
  - btn_process: Butonul principal care inițiază procesarea.

- btn_fig1, btn_fig2, btn_fig3: Butoane (inițial dezactivate) pentru afișarea ferestrelor matplotlib cu rezultate detaliate.

- **Logica de procesare (process_image):**

1. La apăsarea butonului btn_process, funcția verifică dacă o imagine a fost selectată.
2. Citește valorile curente ale parametrilor (t_min, omega, kernel_size) din widget-urile GUI.
3. Apelează funcția dehaze_with_morphology din celălalt modul, transmițându-i calea imaginii și parametrii.
4. Primește rezultatele (ImgRGB, J_restored_rgb, etc.) și le stochează în dicționarul self.results.
5. Actualizează canvas-ul din dreapta: self.ax_orig este desenat cu ImgRGB și self.ax_rest cu J_restored_rgb.
6. Activează butoanele "Figura 1", "Figura 2" și "Figura 3", deoarece acum există date de afișat.

5. Rezultate experimentale

Pentru validarea algoritmului, s-a utilizat o imagine de test. Aplicația permite analiza detaliată a fiecărui pas și influența parametrilor.

5.1. Demonstrarea operațiilor morfologice

În Figura 5.1 sunt prezentate operațiile morfologice de bază (eroziune, dilatare, deschidere, închidere) aplicate pe imaginea de test convertită în tonuri de gri. Acestea stau la baza rafinării hărții de transmisie.

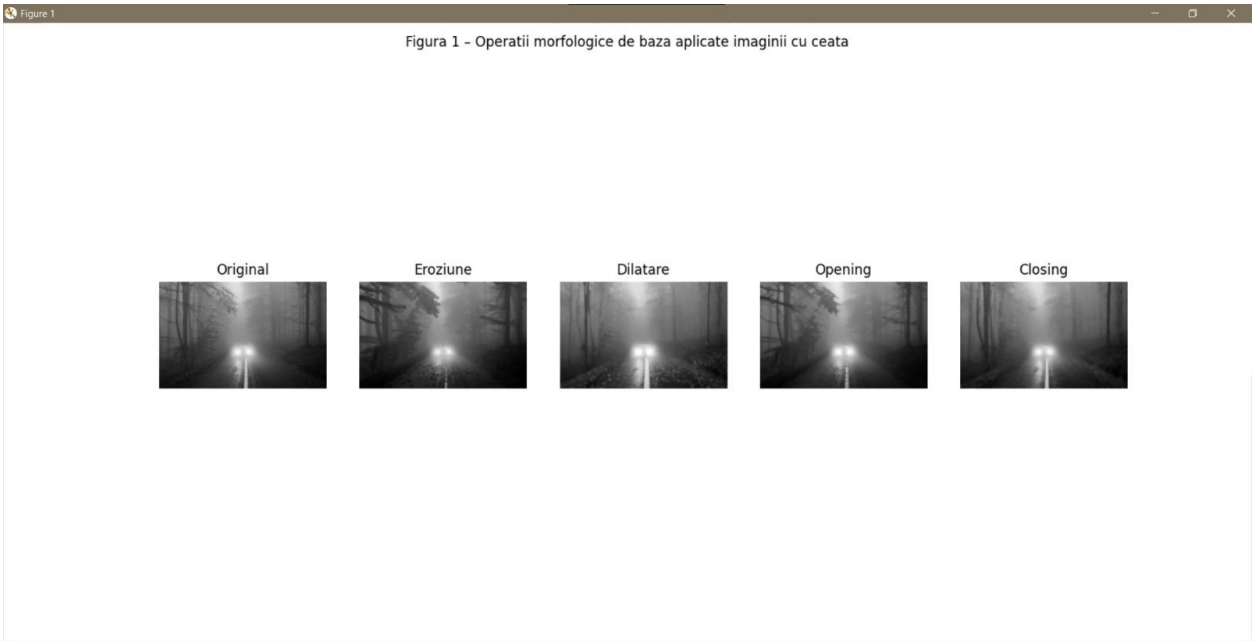

Figura 5.1. Demonstrarea operațiilor morfologice de bază (Kernel size = 15).

5.2. Analiza imaginii cu ceață

Figura 5.2 prezintă imaginea originală în tonuri de gri și histograma acesteia. Se observă că prezența ceții estompează culorile și comprimă histograma, reducând gama

dinamică a imaginii și contrastul global.

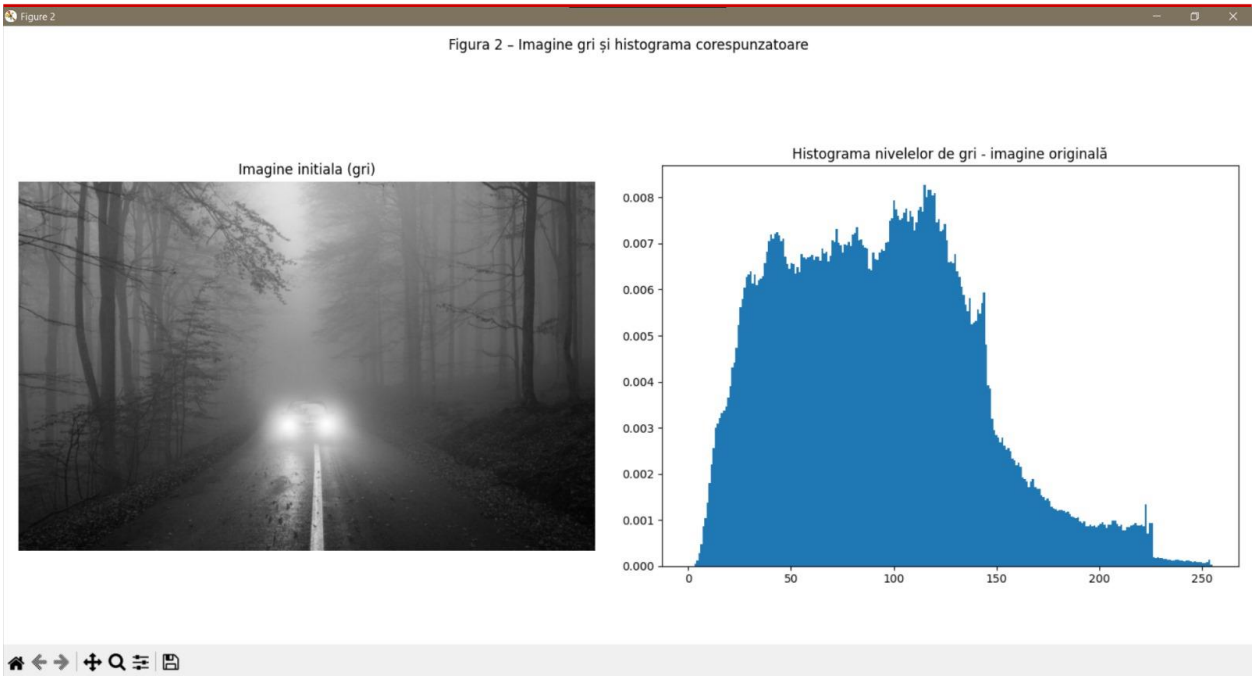

Figura 5.2. Imaginea originală (gri) și histograma corespunzătoare.

5.3. Pașii algoritmului de Dehazing

Figura 5.3 este cea mai relevantă, ilustrând pașii cheie ai algoritmului implementat.

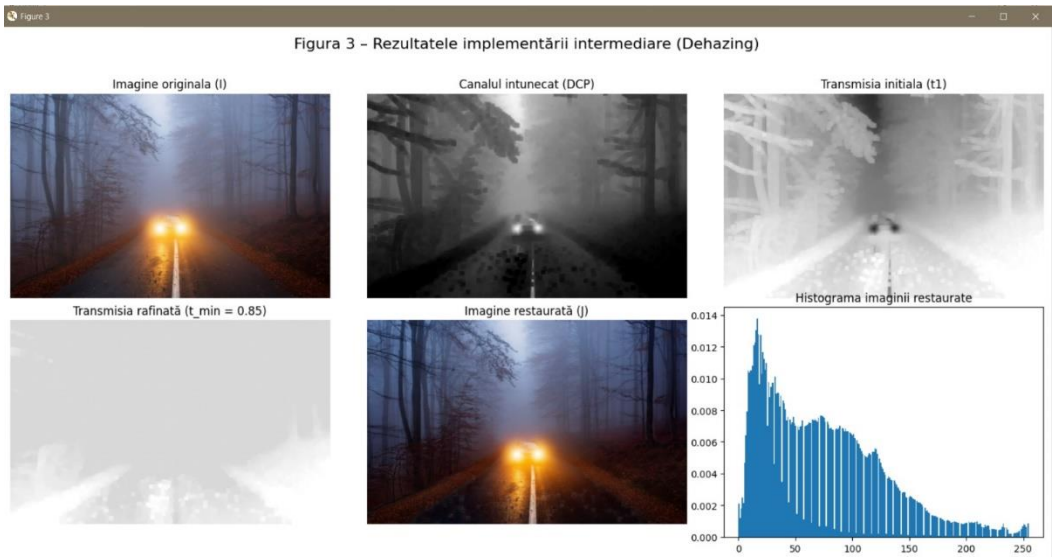

Figura 5.3. Pașii intermediari și rezultatul final al algoritmului de dehazing (Parametri: kernel=15, omega=0.95, t_min=0.85).

- **Canalul întunecat (DCP):** Se observă că zonele cu ceață densă (ex. cerul, fundalul) au valori ridicate (sunt luminoase), în timp ce obiectele apropiate (care nu au ceață) au valori scăzute (întunecate), validând prior-ul DCP.
- **Transmisia inițială (t1):** Această hartă este o inversare a DCP-ului și este vizibil "blocky", deoarece este rezultatul unei operații de minim local (eroziune) cu un kernel mare.
- **Transmisia rafinată:** După aplicarea operațiilor *Closing* și *Opening*, harta de transmisie este mult mai netedă. Zonele corespunzătoare aceluiași obiect au o valoare de transmisie uniformă, iar contururile sunt mai bine definite.
- **Imaginea restaurată (J):** Imaginea finală are un contrast semnificativ îmbunătățit, culori mai vii și detalii mult mai vizibile în zonele care anterior erau acoperite de ceață.
- **Histograma imaginii restaurate:** Spre deosebire de histograma din Figura 2, histograma rezultatului este mult mai largă, acoperind întreaga gamă dinamică [0, 255], ceea ce confirmă creșterea contrastului.

5.4. Analiza parametrilor (din GUI)

- **kernel_size:** Un kernel mai mare (ex. 31, 51) produce un efect morfologic mai puternic și o netezire mai accentuată a hărții de transmisie. Acest lucru este util pentru zone mari (ex. cer), dar poate introduce halouri în jurul obiectelor cu contururi fine. Un kernel mic (ex. 3, 5) păstrează mai multe detalii, dar poate fi insuficient pentru a netezi harta t1.
- **t_min:** Acest parametru controlează agresivitatea restaurării. O valoare mică (ex. 0.5) elimină mai multă ceață, dar poate duce la zone întunecate și suprasaturate. O valoare mare (ex. 0.9) este mai conservatoare, lăsând un pic de ceață, dar evitând artefactele. Valoarea implicită de 0.85 oferă un echilibru bun.
- **omega:** Controlează direct cantitatea de ceață eliminată în estimarea t_1. O valoare apropiată de 1 (ex. 0.98) scoate mai multă ceață, crescând contrastul.

6. Concluzii

Proiectul a constat în implementarea și validarea unui algoritm eficient de eliminare a ceții dintr-o singură imagine. S-a utilizat o abordare hibridă, combinând principiul Dark Channel Prior (DCP) [3] pentru estimarea inițială a parametrilor atmosferici și operații morfologice (închidere și deschidere) [1] pentru rafinarea hărții de transmisie.

Implementarea constă într-o aplicație desktop funcțională care permite încărcarea imaginilor și ajustarea interactivă a parametrilor cheie, oferind feedback vizual imediat.

Rezultatele experimentale demonstrează că metoda este optimă:

1. Estimează corect canalul întunecat și lumina atmosferică.
2. Rafinarea morfologică reușește să netezească eficient harta de transmisie "blocky", reprezentând o alternativă mult mai rapidă la metodele clasice de rafinare (ex. soft matting).
3. Imaginea finală restaurată prezintă o îmbunătățire evidentă a contrastului și vizibilității.

Notă privind utilizarea uneltelor AI

O unealtă de inteligență artificială (Google Gemini) a fost utilizată pentru finalizarea implementării codului sursă (dehaze_morphology.py și main.py) și rescrierea estetică a formulelor, în procent de 10%.

7. Referințe

[1] Salazar-Colores, S. et al. (2018). A Fast Image Dehazing Algorithm Using Morphological Reconstruction. IEEE Trans. Image Processing, 28(5), 2357–2366.

[2] Gonzalez, R. C., Woods, R. E., & Eddins, S. L. (2010). Digital Image Processing Using MATLAB. Pearson Education.

[3] He, K., Sun, J., & Tang, X. (2009). Single Image Haze Removal Using Dark Channel Prior. IEEE TPAMI, 33(12), 2341–2353.

[4] Ancuti, C. O., & De Vleeschouwer, C. (2013). Fast Semi-Inverse Dehazing Method for Contrast Enhancement. IEEE ICIP.

[5] Fattal, R. (2014). Dehazing Using Color-Lines. ACM Trans. Graphics, 34(1), 13.