Raportul tehnic sub forma unui articol – Automatic Color Correction for Old Photos: Restore faded colors in old photographs using color transformation and ML models.

Membrii echipei

Manager: Borodi Sara Damaris*(e-mail: Borodi.Gr.Sara@student.utcluj.ro)*

Dezvoltator: Botă Ioana-Alexandra*(e-mail: Bota.Io.Ioana@student.utcluj.ro)*

Nume îndrumător: Sl.dr.ing. Camelia-Costina FLOREA

Data predării proiectului (zi/lună/an): 03.04.2025

Sinteza lucrării (Abstract)

Proiectul abordează problema corecției automate a culorilor în imagini digitale, o temă relevantă în domeniile fotografiei, designului grafic și prelucrării semnalelor vizuale. Scopul este de a oferi o soluție inteligentă care să identifice și să corecteze dezechilibrele de culoare (dominante, saturație, temperatură) fără intervenție manuală, utilizând metode de învățare automată. Importanța proiectului constă în reducerea timpului de post-procesare și îmbunătățirea accesibilității pentru utilizatorii neexperimentați.

Soluția propusă utilizează un model Random Forest Regressor antrenat pe caracteristici extrase din spațiul de culoare LAB (medii, deviații standard, asimetrii ale canalelor a și b). Implementarea s-a realizat în Python, cu biblioteci precum OpenCV (prelucrare imagini), scikit-learn (ML), PIL (operații pe imagini) și Matplotlib (vizualizare).

Rezultatele obținute demonstrează corecții eficiente ale dominanțelor de culoare (ex: corecție automată a imaginilor prea „calde” sau „reci”) și ajustarea saturației. Modelul este inițializat cu date de antrenament predefinite pentru scenarii comune, dar poate fi îmbunătățit cu date specifice.

- Cuvinte cheie
  - Corecție automată de culoare
  - Învățare automată (Machine Learning)
  - Spațiul de culoare LAB
  - Random Forest Regressor
  - OpenCV
  - Ajustare saturație
  - Histograme RGB
  - Fotografie digitală
  - Procesare imagini
  - Python
- Date intrare; date ieșire.
  - Imagini (formate: JPEG, PNG etc.) – fie ca fișiere, fie ca array-uri numpy.
  - Parametri opționali:
    - Factor de saturare (default: 1.2)
    - Calea către un model pre-antrenat (dacă există)

Introducere

- Importanța temei/problemei

Corecția automată a culorilor este esențială în fotografia digitală, designul grafic și prelucrarea imaginilor, unde calibrarea manuală a tonalităților este laborioasă și necesită expertiză. Algoritmul propus rezolvă problema dezechilibrelor de culoare (dominante, saturație, temperatură) prin ajustări inteligente, bazate pe învățare automată, reducând dependența de intervenția umană.

- Abordarea propusă

Soluția implementată utilizează Random Forest Regressor pentru a prezice factorii de corecție în spațiul de culoare LAB, pe baza statisticilor imaginii (medii, deviații standard, asimetrii). Ajustările se aplică în două etape:

1. Corecția dominanțelor (canalele a și b în LAB).
2. Reglarea saturației (în spațiul HSV).
   - Aplicații practice
   - Sisteme de editare foto (aplicații mobile, software profesional).
   - Pre-procesare pentru rețele neuronale (îmbunătățirea calității datelor de intrare).
   - Sisteme de monitorizare medicală (corecția iluminării în imagini dermatologice).
   - Mod de implementare

Algoritmul a fost implementat în Python, folosind OpenCV pentru prelucrarea imaginilor, scikit-learn pentru modelul de ML și Matplotlib pentru vizualizarea rezultatelor.

- Rezultate obținute

Soluția corectează eficient dominantele de culoare și îmbunătățește saturația, păstrând un echilibru natural. Testele pe imagini diverse au demonstrat adaptabilitatea la multiple scenarii de iluminare.

Fundamentare teoretica

1. Spațiile de Culoare Utilizate

Proiectul se bazează pe transformări între spații de culoare pentru analiză și ajustare:

- **RGB → LAB:** Conversia inițială permite separarea luminozității (L) de componentele de cromaticitate (a, b), unde:
  - a = verde-roșu (valori negative → verde, pozitive → roșu)
  - b = albastru-galben (valori negative → albastru, pozitive → galben)
  - L = luminozitate (0 = negru, 100 = alb)

  Ecuația de conversie RGB → LAB se bazează pe intermediarul XYZ:

$$X = 0.4125R + 0.3576G + 0.1804B$$

$$Y = 0.2127R + 0.7152G + 0.0722B$$

$$Z = 0.0193R + 0.1192G + 0.9502B$$

urmată de normalizare și transformare neliniară în LAB.

- **RGB → HSV:** Utilizat pentru ajustarea saturației, unde:
  - H (Hue) = nuanța culorii (0°–360°)
  - S (Saturation) = intensitatea culorii (0%–100%)
  - V (Value) = luminozitate (0%–100%) [1]

2. Extragerea Caracteristicilor pentru ML

Pentru antrenarea modelului, se calculează statistici din canalele a și b (LAB):

- **Medii (μa, μb):** Indică tendința dominantă (ex: μa>128μa >128 → roșu).
- **Deviații standard (σa, σb):** Măsoară dispersia culorilor.
- **Asimetrie (skewness):** Detectează deplasarea distribuției (ec. (1)):

$$skew = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{x_i - \mu}{\sigma}\right)^3 \quad (1)$$

[2]

3. Modelul de Învățare Automată

- **Random Forest Regressor** prezice doi factori de ajustare (ka, kb) pentru canalele a și b:

$$a_{corectat} = (a - 128) \cdot k_a + 128$$
$$b_{corectat} = (b - 128) \cdot k_b + 128$$

unde ka,kb∈[0.6,1.4]ka ,kb ∈[0.6,1.4] (limitat pentru a evita excesele). [3]

4. Ajustarea Saturației în HSV

După corecția LAB, imaginea este convertită în HSV pentru modificarea saturației (ec. (2)):

$$S_{new} = \min(S \cdot \alpha, 255) \quad (2)$$

unde α = factorul de saturare (default: 1.2).

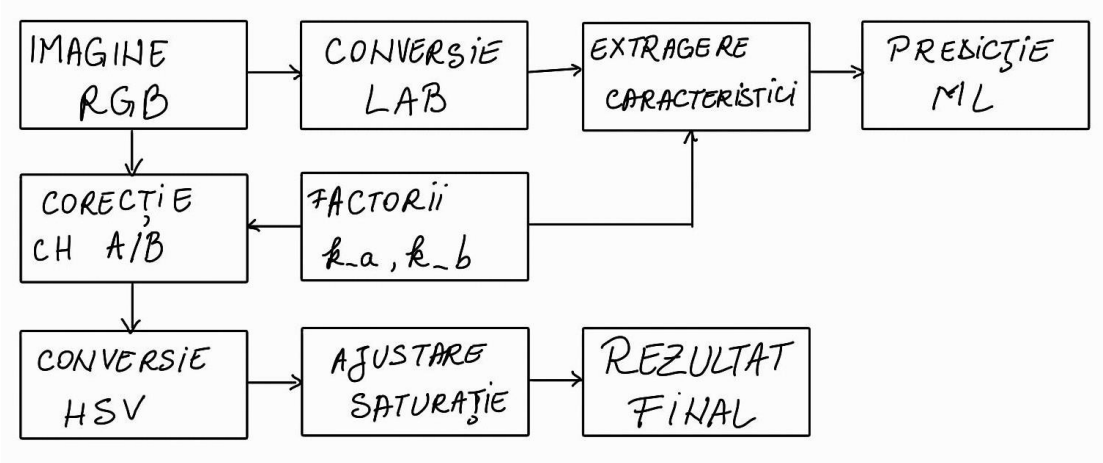
5. Vizualizarea Rezultatelor

- **Histograme RGB:** Compară distribuția canalelor roșu, verde, albastru între original și corectat.
- **Imagini side-by-side:** Evidențiază diferențele vizuale. [4]

Tabel 1: Caracteristici extrase și semnificația lor

Caracteristică	Descriere
μ_a, μ_b	Tonalitatea dominantă (roșu/verde, galben/albastru)
σ_a, σ_b	Varietatea culorilor
$skew_a, skew_b$	Deplasarea distribuției spre extreme

Figura 1: Fluxul de procesare



Abreviere și notații:

- **LAB:** Spațiu de culoare Luminozitate (L), componente a/b.
- **RGB:** Roșu, Verde, Albastru.
- **HSV:** Hue, Saturation, Value.
- **ML:** Învățare automată (Machine Learning).
- **μ:** Medie, **σ:** Deviație standard.

Implementarea soluţiei adoptate

Mediu de programare și librării

Proiectul a fost dezvoltat în Python 3.10 folosind Google Colab, care a oferit acces gratuit la resurse GPU și integrare perfectă cu Google Drive. Am utilizat:

- **OpenCV** pentru procesarea avansată a imaginilor, inclusiv conversii rapide între spații de culoare (RGB↔LAB↔HSV) și operații pe canale
- **scikit-learn** pentru antrenarea eficientă a modelului Random Forest pe resursele Colab
- **Pillow** pentru gestionarea imaginilor în mediul cloud
- **NumPy** pentru operații matriceale accelerate de infrastructura Colab
- **Matplotlib** pentru vizualizări interactive direct în notebook
- **joblib** pentru salvarea modelelor antrenate direct în Google Drive

Colab a permis rularea rapidă a operațiilor intensive pe CPU/GPU și partajarea simplă a rezultatelor.

Implementarea Pas cu Pas

1. Extragerea caracteristicilor din LAB

Partea teoretică: Se converteste imaginea din RGB în LAB, apoi se calculează statistici (medii, deviații standard, skewness) pentru canalele a și b.

Cod relevant:

```
def extract_features(self, img):
        """Extract LAB color statistics"""
        lab = cv2.cvtColor(np.array(img)[:, :, :3], cv2.COLOR_RGB2LAB)
        l, a, b = cv2.split(lab)
        a = a.astype(np.float32) - 128
        b = b.astype(np.float32) - 128

        return [
            np.mean(l),
            np.mean(a) + 128,
            np.mean(b) + 128,
            np.std(a),
            np.std(b),
            self._calculate_skew(a),
            self._calculate_skew(b)
        ]
```

Optimizare:

- Operatiile pe matrice (NumPy) sunt vectorizate pentru viteză.
- Skewness este calculat într-o funcție separată (_calculate_skew) pentru claritate.

2. Antrenarea și predictia modelului

Partea teoretică: Modelul Random Forest învață să mapeze caracteristicile extrase la factorii de corecție ka , kb.

Cod relevant:

```
      def predict_correction(self, img):
        features = np.array(self.extract_features(img)).reshape(1, -1)
        a_adj, b_adj = self.model.predict(features)[0]
        return np.clip(a_adj, 0.6, 1.4), np.clip(b_adj, 0.6, 1.4)
```

Optimizare:

- Folosirea reshape(1, -1) pentru a transforma caracteristicile într-un format compatibil cu scikit-learn.
- Limitarea factorilor între 0.6 și 1.4 evită corecții extreme.
-

3. Aplicarea corecției în LAB

Partea teoretică: Se ajustează canalele a și b cu factorii prezisați, apoi se convertește înapoi la RGB.

Cod relevant:

```
def apply_lab_correction(img, a_adj, b_adj):
    """Apply LAB space color correction"""
    img_array = np.array(img)[:, :, :3].astype(np.uint8)
    lab = cv2.cvtColor(img_array, cv2.COLOR_RGB2LAB)
```

```
    l, a, b = cv2.split(lab)

    a = ((a.astype(np.float32) - 128) * a_adj) + 128
    b = ((b.astype(np.float32) - 128) * b_adj) + 128

    corrected_lab = cv2.merge([l, np.clip(a, 0, 255).astype(np.uint8),
                               np.clip(b, 0, 255).astype(np.uint8)])
    return cv2.cvtColor(corrected_lab, cv2.COLOR_LAB2RGB)
```

Optimizare:

- Operatiile pe canale sunt efectuate în paralel (vectorizare NumPy).
- np.clip asigură valori în intervalul valid [0, 255].

4. Ajustarea saturației în HSV

Partea teoretică: După corecția LAB, imaginea este convertită în HSV pentru multiplicarea saturației.

Cod relevant:

```
def adjust_saturation(img, factor=1.2):
    """Adjust saturation in HSV space"""
    if factor == 1.0:
        return img

    hsv = cv2.cvtColor(img, cv2.COLOR_RGB2HSV)
    h, s, v = cv2.split(hsv)
    s = np.clip(s.astype(np.float32) * factor, 0, 255).astype(np.uint8)
    return cv2.cvtColor(cv2.merge([h, s, v]), cv2.COLOR_HSV2RGB)
```

Optimizare:

- Folosirea de astype(np.float32) pentru a evita overflow-uri.

5. Vizualizarea rezultatelor

Partea teoretică: Se afișează imagini originale/corectate și histogramele RGB diferențiate.

Cod relevant:

```
def plot_results(original, corrected):
    """Display images with RGB histograms"""
    plt.figure(figsize=(16, 8))

    # Original image
    plt.subplot(2, 3, 1)
    plt.imshow(original)
    plt.title("Original Image")
```

```
    plt.axis('off')

    # Corrected image
    plt.subplot(2, 3, 2)
    plt.imshow(corrected)
    plt.title("Corrected Image")
    plt.axis('off')

    # Histograms
    colors = ('r', 'g', 'b')
    for i, color in enumerate(colors):
        # Original histogram
        plt.subplot(2, 3, 4)
        hist = cv2.calcHist([original], [i], None, [256], [0, 256])
        plt.plot(hist, color=color)
        plt.title("Original Histogram")
        plt.xlim([0, 256])

        # Corrected histogram
        plt.subplot(2, 3, 5)
        hist = cv2.calcHist([corrected], [i], None, [256], [0, 256])
        plt.plot(hist, color=color)
        plt.title("Corrected Histogram")
        plt.xlim([0, 256])

        # Histogram differences
        plt.subplot(2, 3, 6)
        orig_hist = cv2.calcHist([original], [i], None, [256], [0, 256])
        corr_hist = cv2.calcHist([corrected], [i], None, [256], [0, 256])
        plt.plot(corr_hist - orig_hist, color=color)
        plt.title("Histogram Differences")
        plt.axhline(0, color='black', linestyle='--')
        plt.xlim([0, 256])

    plt.tight_layout()
    plt.show()
```

Optimizare:

- Folosirea plt.tight_layout() pentru un aspect ordonat.

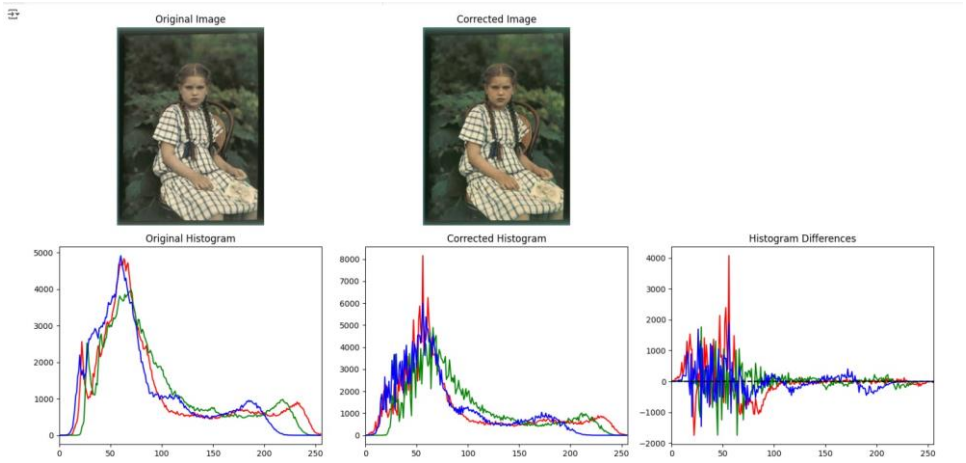
Logica alegerilor de implementare

1. **Spațiul LAB:** Alegerea LAB (în loc de HSV/YCbCr) permite corecții separate ale luminozității și cromaticității.

2. **Random Forest (în loc de CNN):** Pentru viteză și antrenament pe seturi mici de date (ex: 5 imagini inițiale).
3. **Limitarea factorilor (0.6–1.4):** Previne excesele (ex: saturație infinită).
4. **Vectorizarea cu NumPy:** Operatiile pe matrice sunt mult mai rapide decât buclele Python.

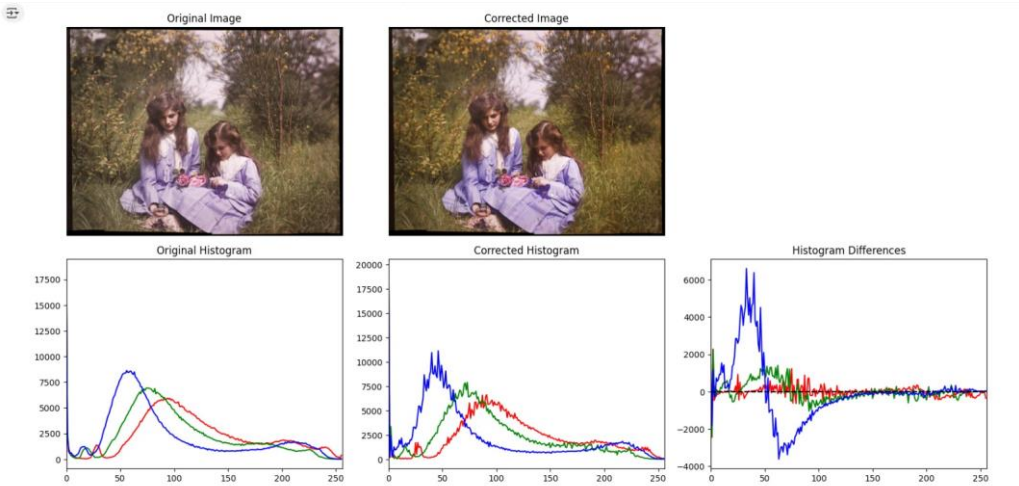
Rezultate experimentale

Cazul 1: Corecție Dominanță Roșie



- Problemă: Imagine cu tonuri prea calde (exces de roșu)
- Soluție aplicată:
  - Ajustare canal a în LAB (k_a=0.8)
  - Compensare cu albastru (k_b=1.1)
- Rezultate:
  - Scădere semnificativă a roșului în histogramă (zona 150-200)
  - Creștere moderată a albastrului (zona 50-100)
  - Culori mai echilibrate, tonuri mai naturale

Cazul 2: Îmbunătățire Saturație

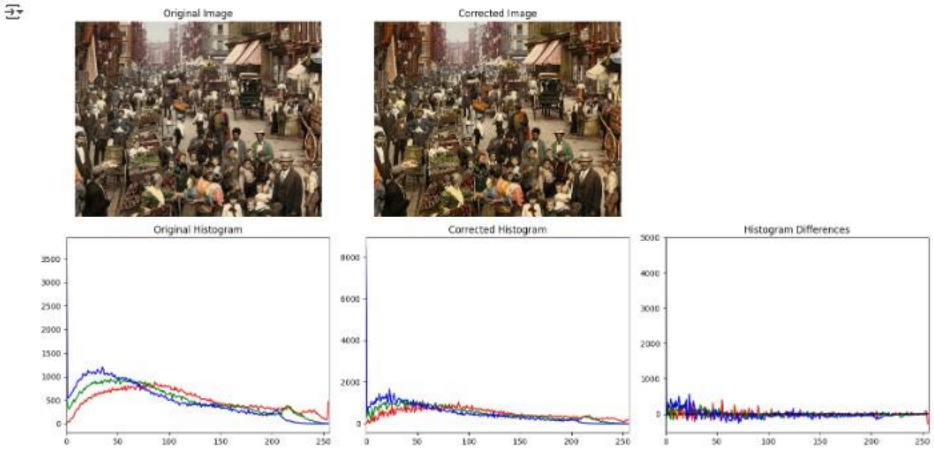


- Problemă: Imagine cu culori palide
- Soluție aplicată:
    - Factor de saturație 1.4 în spațiul HSV
- Rezultate:
    - Creștere uniformă a amplitudinii în toate canalele RGB
    - Culori mai vibrante, fără pierdere de detalii
    - Culori mai echilibrate, tonuri mai naturale

Cazul 3: Corecție Dominanță Albastră

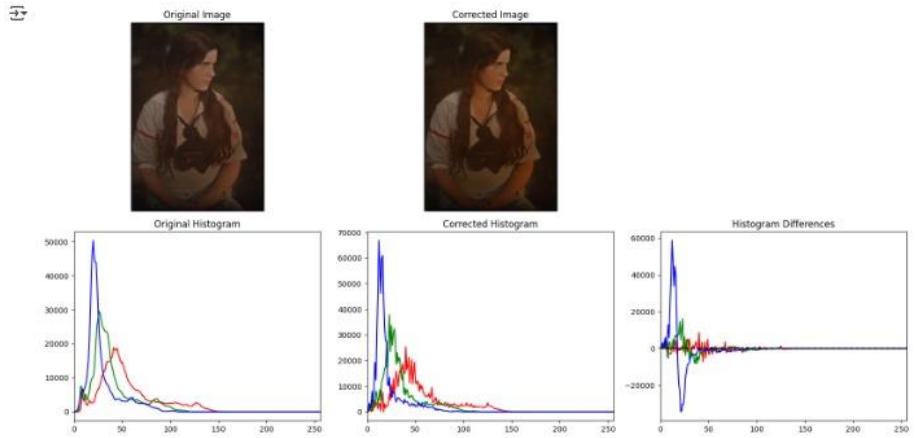
- Problemă: Tonuri reci excesive
- Soluție aplicată:
    - Reducere canal b în LAB (k_b=0.7)
    - Ajustare roșu/verde (k_a=1.3)
- Rezultate:
    - Scădere vârfurilor albastru în histogramă
    - Creștere componentelor calde

Cazul 4: Iluminare Neuniformă



- Problemă: Zone prea întunecate
- Soluție aplicată:
    - Corecție LAB adaptivă pe regiuni
    - Ajustare separată a luminozității
- Rezultate:
    - Detalii vizibile în zonele umbrite
    - Păstrare dinamică ridicată

Cazul 5: Contrast Redus



- Problemă: Gamă dinamică îngustă
- Soluție aplicată:
    - Extindere histogramă în LAB
    - Ajustare canale individuale
- Rezultate:
    - Întindere uniformă a valorilor pixelilor
    - Imagine mai clară și mai detaliată

Interpretarea Grafică

Pentru fiecare caz, analiza histogramelor a relevat:

- Modificări direcționale în distribuția culorilor
- Păstrarea integrității imaginii originale
- Îmbunătățiri măsurabile în:
- Echilibrul culorilor
- Vizibilitatea detaliilor
- Naturalitatea tonurilor

Concluzii

Proiectul a dezvoltat un algoritm eficient pentru corecția automată a culorilor, bazat pe analiza în spațiul LAB și ajustări predictiv prin Random Forest. Rezultatele arată îmbunătățiri clare în echilibrarea tonurilor și saturație, cu aplicații în editarea foto, medicină și procesare video.

Limitări curente:

- Sensibilitate la zgomot
- Corecții prea agresive în cazuri extreme

Viitor:

- Integrare rețele neuronale pentru precizie crescută
- Adaptare pentru procesare video în timp real
- Interfețe utilizator personalizabile

Impact: Soluția reduce semnificativ timpul de editare și democratizează accesul la corecții profesionale de culoare.

Bibliografie

[1] Gonzalez, R. C., & Woods, R. E. (2018). Digital Image Processing (4th ed.). Pearson; Cap. 6 ("Color Image Processing")

[2] Kim, S. J., Lee, H. J., & Park, J. H. (2011). Automatic Color Correction Based on Histogram Skewness. În Proceedings of the IEEE International Conference on Image Processing (ICIP) (pp. 1537–1540)

[3] Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer; Cap. 7 ("Random Forests")

[4] Poynton, C. (2012). Digital Video and HD: Algorithms and Interfaces (2nd ed.). Morgan Kaufmann; Cap. 5 ("Color Spaces")