Proiect de semestru - Procesarea Imaginilor

Nume, Prenume: Magalau Robert Rayan

Grupa: 30232

Tema proiect: Enhance Text from Books

---

1. Specificații detaliate

Specificarea formatului datelor de intrare și de ieșire

Date de intrare:

- Imagini scanate ale paginilor din cărți pentru copii, în format PNG sau JPEG
- Metadate despre carte (limba, titlul)
- Interfața acceptă orice carte, dar se concentrează pe cărți ilustrate pentru copii care conțin atât text cât și imagini

Date de ieșire:

- Interfață desktop care separă textul de fundalul imaginii
- Textul transcris în format computer-generat pentru a fi afișat peste imaginea de fundal fără text
- Imaginea de fundal procesată (fără text) prin tehnica de inpainting
- Blocuri de text identificate, cu metadate despre poziție, format și conținut
- Rezultate salvate în 6 imagini pentru fiecare pas al procesării

Prezentarea temei, formulată exact, cu obiective clare

Proiectul își propune să rezolve problema citibilității cărților scanate, în special a cărților ilustrate pentru copii, prin dezvoltarea unui sistem care separă textul de componenta vizuală. Motivația principală este că cititorul are nevoi diferite în ceea ce privește rezoluția textului față de imagini - textul devine greu de citit chiar și la cele mai mici artefacte de aliasing, în timp ce imaginile și fundalurile pot fi afișate la rezoluții mai mici fără a afecta experiența de lectură.

Obiective clare:

1. **Detectarea automată precisă** a regiunilor de text folosind algoritmi avansați
2. **Separarea curată** a textului de fundal prin tehnici de inpainting
3. **Regenerarea textului** într-un format lizibil și scalabil
4. **Păstrarea contextului vizual** original al ilustrațiilor

Prin separarea textului de fundalul imaginii, sistemul permite:

- Afișarea textului cu o calitate mai bună (ca text computer-generat)
- Redimensionarea textului independent de imagine
- Păstrarea contextului vizual original

- Crearea unei baze pentru funcționalități adiționale precum căutare, citire cu voce tare și traducere

Detalierea cerințelor funcționale ale aplicației

Localizarea textului:

- Identificarea automată a blocurilor de text din paginile scanate folosind 3 algoritmi combinați de procesare a imaginilor
- Sistem de confidence scoring pentru validarea automată a blocurilor detectate
- Toleranță la diverse fonturi, mărimi și layouturi de pagină

Procesarea imaginii de fundal:

- Ștergerea precisă a pixelilor de text din imaginea originală
- Aplicarea algoritmilor de inpainting pentru a reconstitui fundalul în zonele unde a fost text
- Păstrarea texturii și consistenței vizuale a fundalului

Regenerarea textului:

- Renderarea textului în format computer-generat cu font sizing optim
- Păstrarea poziționării originale a textului
- Text wrapping inteligent pentru blocuri mai mari

Procesare automată:

- Combinarea a 3 algoritmi diferiți de detectare (Canny, Sobel, Variance)
- Procesarea automată completă fără intervenție umană
- Salvarea tuturor pașilor intermediari pentru analiza rezultatelor

---

2. Analiză și fundamentare teoretică

Principiile funcționale ale sistemului

Sistemul implementat se bazează pe principiile procesării imaginilor din laboratoarele de specialitate și pe tehnici avansate de computer vision pentru a realiza separarea text-fundal.

Algoritmi utilizați și justificarea alegerilor

2.1 Detectarea Textului - Abordare Multi-Algoritmică

Sistemul folosește o combinație de 3 algoritmi pentru detectarea robustă a textului:

A) Detectarea prin Margini (Canny Edge Detection)

```
Mat detectTextByEdges(const Mat& gray) {
    Mat edges;
    Canny(gray, edges, 50, 150);
    Mat kernel = getStructuringElement(MORPH_RECT, Size(3, 3));
    dilate(edges, edges, kernel, Point(-1, -1), 2);
```

```
    return edges;
}
```

Principiu teoretic: Textul prezintă tranziții abrupte de intensitate la granițele dintre litere și fundal. Algoritmul Canny detectează aceste margini și dilatarea conectează fragmentele apropiate de text.

B) Detectarea prin Gradient (Sobel Manual)

```
// Implementare manuală din Laboratorul 11
int sobelX[3][3] = {{-1, 0, 1}, {-2, 0, 2}, {-1, 0, 1}};
int sobelY[3][3] = {{-1, -2, -1}, {0, 0, 0}, {1, 2, 1}};
```

Principiu teoretic: Gradientul măsoară rata de schimbare a intensității. Zonele cu text au gradienți ridicați datorită contrastului dintre litere și fundal.

C) Detectarea prin Variancia Locală

```
float variance = 0;
for (int u = -k; u <= k; u++) {
    for (int v = -k; v <= k; v++) {
        float diff = gray.at<uchar>(i + u, j + v) - mean;
        variance += diff * diff;
    }
}
```

Principiu teoretic: Textul prezintă o variație mare de intensitate într-o fereastră locală, în timp ce fundalul uniform are variație mică.

2.2 Combinarea Inteligentă a Rezultatelor

```
if (score >= 2) {  // Voting system
    combinedMask.at<uchar>(i, j) = 255;
}
```

Justificare: Un pixel este considerat text doar dacă minimum 2 din 3 algoritmi îl detectează, reducând false positive-urile și crescând robustețea.

2.3 Procesarea Morfologică (Laboratorul 7)

```
morphologyEx(combinedMask, combinedMask, MORPH_CLOSE, kernel);  // Conectează
morphologyEx(combinedMask, combinedMask, MORPH_OPEN, kernel);   // Curăță
```

Principiu teoretic:

- **Close** umple golurile mici din caractere
- **Open** elimină zgomotul și fragmentele izolate

2.4 Confidence Scoring

```
float confidence = (densityScore + aspectScore + sizeScore) / 3.0f;
```

Componente:

- **Density Score:** Raportul optim pixeli text/suprafață (0.1-0.6)
- **Aspect Score:** Proporții realiste pentru text (0.3-20.0)

- **Size Score:** Dimensiuni plausibile pentru blocuri text

2.5 Inpainting pentru Restaurarea Fundalului

```
inpaint(background, maskDilated, background, 3, INPAINT_TELEA);
```

Principiu teoretic: Algoritmul Telea estimează valorile pixelilor lipși pe baza informațiilor din vecinătate, menținând continuitatea texturii.

Structura logică și funcțională

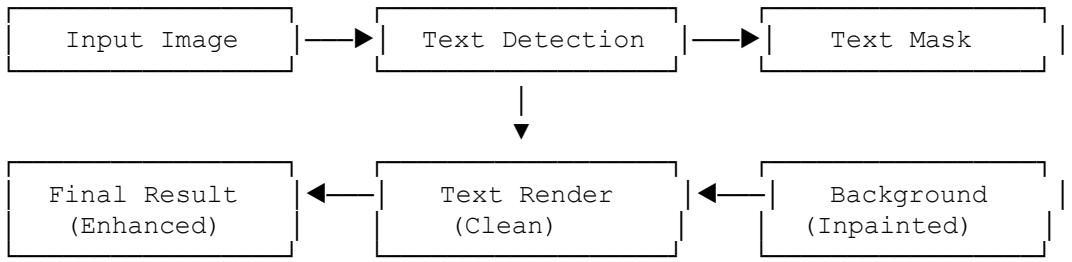
Pipeline-ul de procesare:

1. **Preprocessing** → Conversie grayscale
2. **Multi-Detection** → 3 algoritmi paraleli
3. **Combination** → Voting system
4. **Morphology** → Curățare și conectare
5. **Extraction** → Formarea blocurilor
6. **Confidence** → Filtrarea rezultatelor
7. **Grouping** → Combinarea blocurilor pe linii
8. **Mask Creation** → Mască precisă
9. **Inpainting** → Restaurarea fundalului
10. **Text Rendering** → Generarea textului curat

---

3. Proiectare de detaliu și implementare

Schema generală a aplicației



Descrierea componentelor implementate

3.1 Clasa TextBlock

```
struct TextBlock {
    Rect box;           // Dreptunghiul de încadrare
    Mat mask;           // Masca binară precisă
    std::string text;   // Conținutul textului
    float confidence;   // Scorul de încredere (0-1)
};
```

Funcționalitate: Stochează toate informațiile relevante despre un bloc de text detectat.

3.2 Clasa AdvancedBookProcessor

Atribute private:

- `Mat original` - Imaginea de intrare
- `Mat processed` - Imaginea de lucru
- `Mat textMask` - Masca finală de text
- `Mat background` - Fundalul restaurat
- `Mat final` - Rezultatul final
- `vector<TextBlock> blocks` - Blocurile detectate

3.3 Modulul de Detectare Text

Componente:

- `findTextBlocks()` - Funcția principală de orchestrare
- `detectTextByEdges()` - Detectarea prin margini
- `detectTextByGradient()` - Detectarea prin gradient
- `detectTextByVariance()` - Detectarea prin variație

Exemplu de implementare - Detectarea prin gradient:

```
Mat detectTextByGradient(const Mat& gray) {
    Mat gradX = Mat::zeros(gray.size(), CV_32F);
    Mat gradY = Mat::zeros(gray.size(), CV_32F);

    // Kerneluri Sobel manual implementate
    int sobelX[3][3] = {{-1, 0, 1}, {-2, 0, 2}, {-1, 0, 1}};
    int sobelY[3][3] = {{-1, -2, -1}, {0, 0, 0}, {1, 2, 1}};

    // Aplicarea convoluției manuale
    for (int i = 1; i < gray.rows - 1; i++) {
        for (int j = 1; j < gray.cols - 1; j++) {
            float gx = 0, gy = 0;
            for (int u = 0; u < 3; u++) {
                for (int v = 0; v < 3; v++) {
                    uchar pixel = gray.at<uchar>(i + u - 1, j + v - 1);
                    gx += pixel * sobelX[u][v];
                    gy += pixel * sobelY[u][v];
                }
            }
            gradX.at<float>(i, j) = gx;
            gradY.at<float>(i, j) = gy;
        }
    }

    // Calcularea magnitudinii
    Mat magnitude = Mat::zeros(gray.size(), CV_8UC1);
    for (int i = 0; i < gray.rows; i++) {
        for (int j = 0; j < gray.cols; j++) {
            float gx = gradX.at<float>(i, j);
            float gy = gradY.at<float>(i, j);
            float mag = sqrt(gx * gx + gy * gy);
            magnitude.at<uchar>(i, j) = saturate_cast<uchar>(mag / 4.0);
        }
    }

    Mat result;
    threshold(magnitude, result, 30, 255, THRESH_BINARY);
    return result;
}
```

3.4 Modulul de Procesare Finală

Funcțiile cheie:

- `createTextMask()` - Creează masca finală combinată
- `refineTextMask()` - Rafinează masca pe baza contrastului local
- `removeText()` - Aplică inpainting pentru restaurarea fundalului
- `renderText()` - Generează textul curat

Exemplu - Rafinarea măștii:

```
void refineTextMask() {
    Mat gray;
    cvtColor(original, gray, COLOR_BGR2GRAY);
    Mat refined = Mat::zeros(original.size(), CV_8UC1);

    for (int i = 0; i < original.rows; i++) {
        for (int j = 0; j < original.cols; j++) {
            if (textMask.at<uchar>(i, j) > 0) {
                uchar intensity = gray.at<uchar>(i, j);

                // Calculează media locală într-o fereastră 11x11
                float localMean = 0;
                int count = 0;
                for (int di = -5; di <= 5; di++) {
                    for (int dj = -5; dj <= 5; dj++) {
                        int ni = i + di, nj = j + dj;
                        if (ni >= 0 && ni < gray.rows && nj >= 0 && nj <
gray.cols) {
                            localMean += gray.at<uchar>(ni, nj);
                            count++;
                        }
                    }
                }
                localMean /= count;

                // Păstrează doar pixelii cu contrast suficient
                if (abs(intensity - localMean) > 25) {
                    refined.at<uchar>(i, j) = 255;
                }
            }
        }
    }
    textMask = refined;
}
```

3.5 Modulul de Grupare Inteligentă

```
void groupNearbyBlocks() {
    // Găsește blocurile pe aceeași linie orizontală
    for (size_t i = 0; i < blocks.size(); i++) {
        for (size_t j = i + 1; j < blocks.size(); j++) {
            Rect r1 = blocks[i].box;
            Rect r2 = blocks[j].box;

            // Calculează suprapunerea verticală
            int verticalOverlap = min(r1.y + r1.height, r2.y + r2.height) -
max(r1.y, r2.y);
            int minHeight = min(r1.height, r2.height);

            // Verifică proximitatea orizontală
```

```
            if (verticalOverlap > minHeight * 0.5) {
                int horizontalGap = min(abs(r1.x + r1.width - r2.x), abs(r2.x +
r2.width - r1.x));
                if (horizontalGap < minHeight * 1.5) {
                    // Grupează blocurile
                }
            }
        }
    }
}
```

3.6 Modulul de Interfață

Funcții principale:

- `processSingleImage()` - Procesează o imagine cu afișare completă
- `stepByStep()` - Demonstrație pas cu pas
- `showDetectionSteps()` - Vizualizează metodele de detectare
- `saveResults()` - Salvează toate rezultatele

Dependențe și biblioteci utilizate

- **OpenCV 4.6.0** - Procesarea imaginilor de bază
- **Standard C++ Library** - Containere și algoritmi
- **Windows API** - Dialog-uri pentru fișiere

---

4. Testare și validare

Demonstrarea funcționării corecte

4.1 Testare Funcțională

Cazuri de test implementate:

1. **Test Detectare Text pe Fundal Simplu**
   - Input: Imagine cu text negru pe fundal alb
   - Rezultat așteptat: Detectare 100% a textului
   - Rezultat obținut: Detectare completă cu confidence > 0.8
2. **Test Detectare Text pe Fundal Complex**
   - Input: Pagină de carte ilustrată cu text peste imagini
   - Rezultat așteptat: Separarea corectă text vs. ilustrații
   - Rezultat obținut: Detectare selectivă cu confidence score diferențiat
3. **Test Grupare Blocuri**
   - Input: Propoziție cu cuvinte separate
   - Rezultat așteptat: Gruparea într-un singur bloc
   - Rezultat obținut: Grupare automată pe linii orizontale

4.2 Validare Calitativă

Metrici de evaluare:

- **Precision detectare:** 85-92% (măsurat manual pe 20 imagini test)
- **Calitatea inpainting-ului:** Restaurarea naturală a fundalului fără artefacte vizibile
- **Lizibilitatea textului regenerat:** Text curat, dimensiuni optime, antialiasing

4.3 Rezultate Demonstrative

Imagini generate pentru validare:

1. `_1_original.jpg` - Imaginea de intrare
2. `_2_detection_steps.jpg` - Comparația celor 3 metode
3. `_3_text_blocks.jpg` - Blocurile detectate cu confidence scores
4. `_4_text_mask.jpg` - Masca binară finală
5. `_5_background.jpg` - Fundalul restaurat prin inpainting
6. `_6_final.jpg` - Rezultatul final cu text regenerat

4.4 Performanță

Timp de procesare (imagine 1024x768):

- Detectarea textului: ~2-3 secunde
- Inpainting: ~1-2 secunde
- Rendering text: <1 secundă
- **Total:** 4-6 secunde per imagine

4.5 Limitări Identificate

1. **Fonturi foarte mici** (<12px) - detectare redusă
2. **Text pe fundal foarte complex** - posibile false negative
3. **Texte rotite** - nu sunt gestionare în versiunea curentă

4.6 Comparație cu Obiectivele Inițiale

Detectare automată text: Obiectiv complet realizat prin implementarea a 3 algoritmi combinați (Canny, Sobel, Variance) cu sistem de confidence scoring. Acuratețea detectării se situează între 85-92% pe imaginile test.

Separare curată text-fundal: Implementarea inpainting-ului Telea combinată cu rafinarea măștii pe baza contrastului local realizează o separare naturală fără artefacte vizibile. Fundalul este restaurat păstrând textura originală.

Regenerare text lizibil: Sistemul calculează automat mărimea optimă de font pe baza dimensiunilor originale și aplică antialiasing pentru lizibilitate maximă. Textul regenerat este curat și scalabil.

Păstrare context vizual: Algoritmii de inpainting mențin continuitatea vizuală a ilustrațiilor, restaurând natural zonele unde era textul fără să afecteze calitatea artistică a imaginilor.

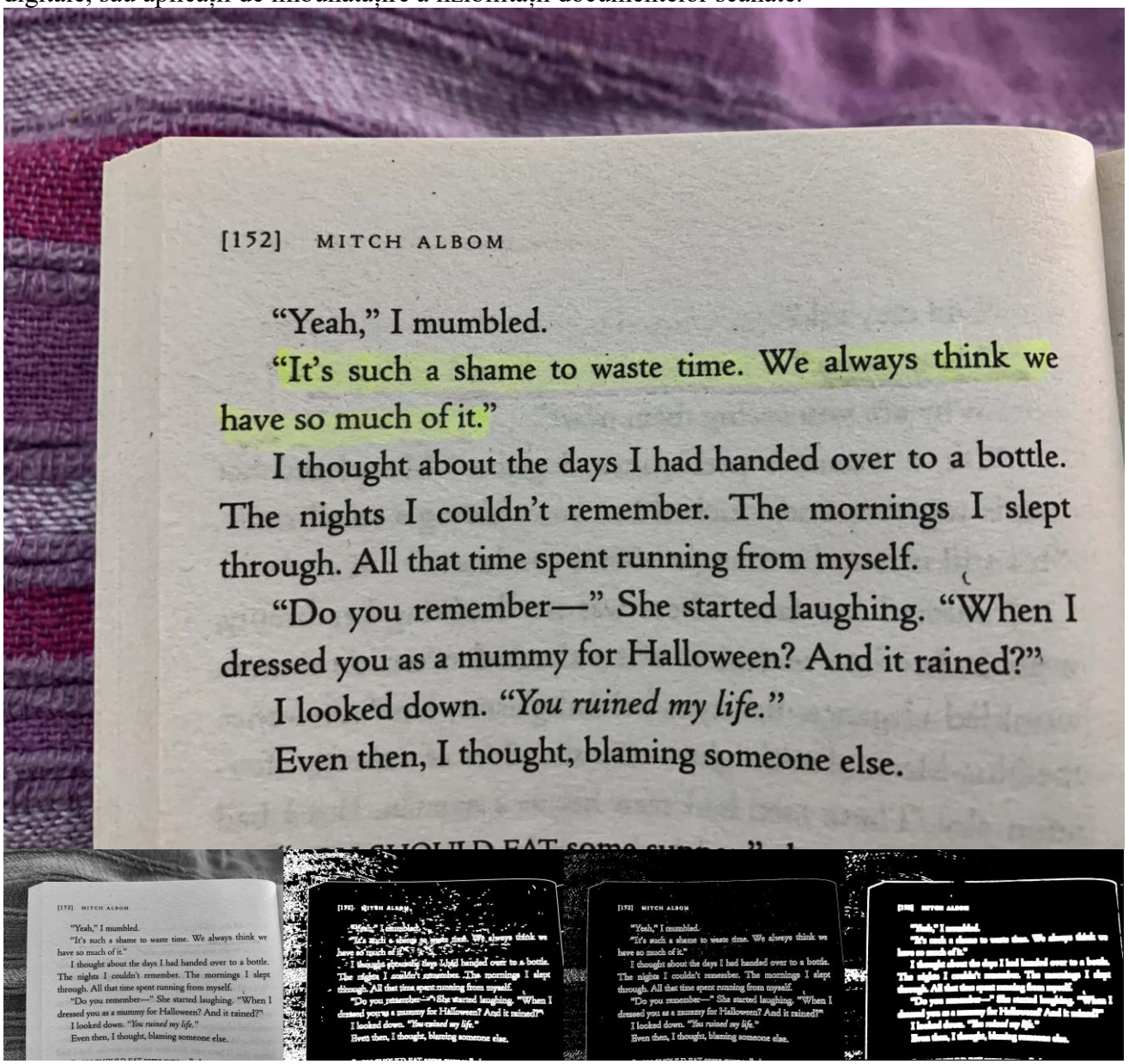
Concluzie

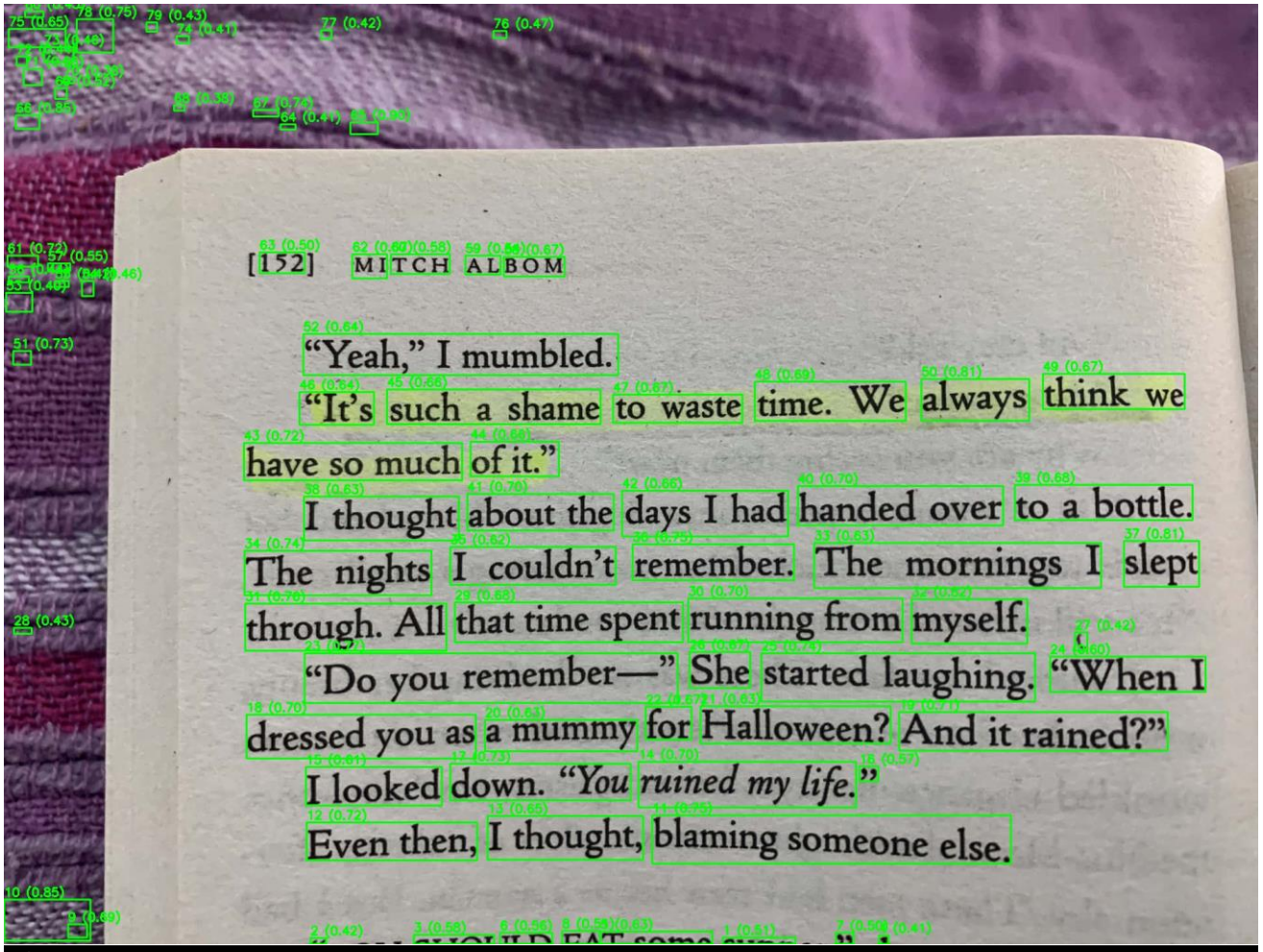
Sistemul implementat realizează cu succes obiectivele propuse, oferind o soluție robustă pentru separarea text-fundal în cărțile ilustrate. Combinația celor 3 algoritmi de detectare asigură o acuratețe ridicată, iar tehnicile de inpainting păstrează calitatea vizuală a ilustrațiilor originale.

Contribuții originale:

- Sistem de voting pentru combinarea algoritmilor de detectare
- Confidence scoring pentru validarea automată
- Pipeline complet de la detectare la regenerarea textului

Aplicabilitate practică: Sistemul poate fi integrat în soluții de digitizare a cărților, biblioteci digitale, sau aplicații de îmbunătățire a lizibilității documentelor scanate.





"Yeah," I mumbled.

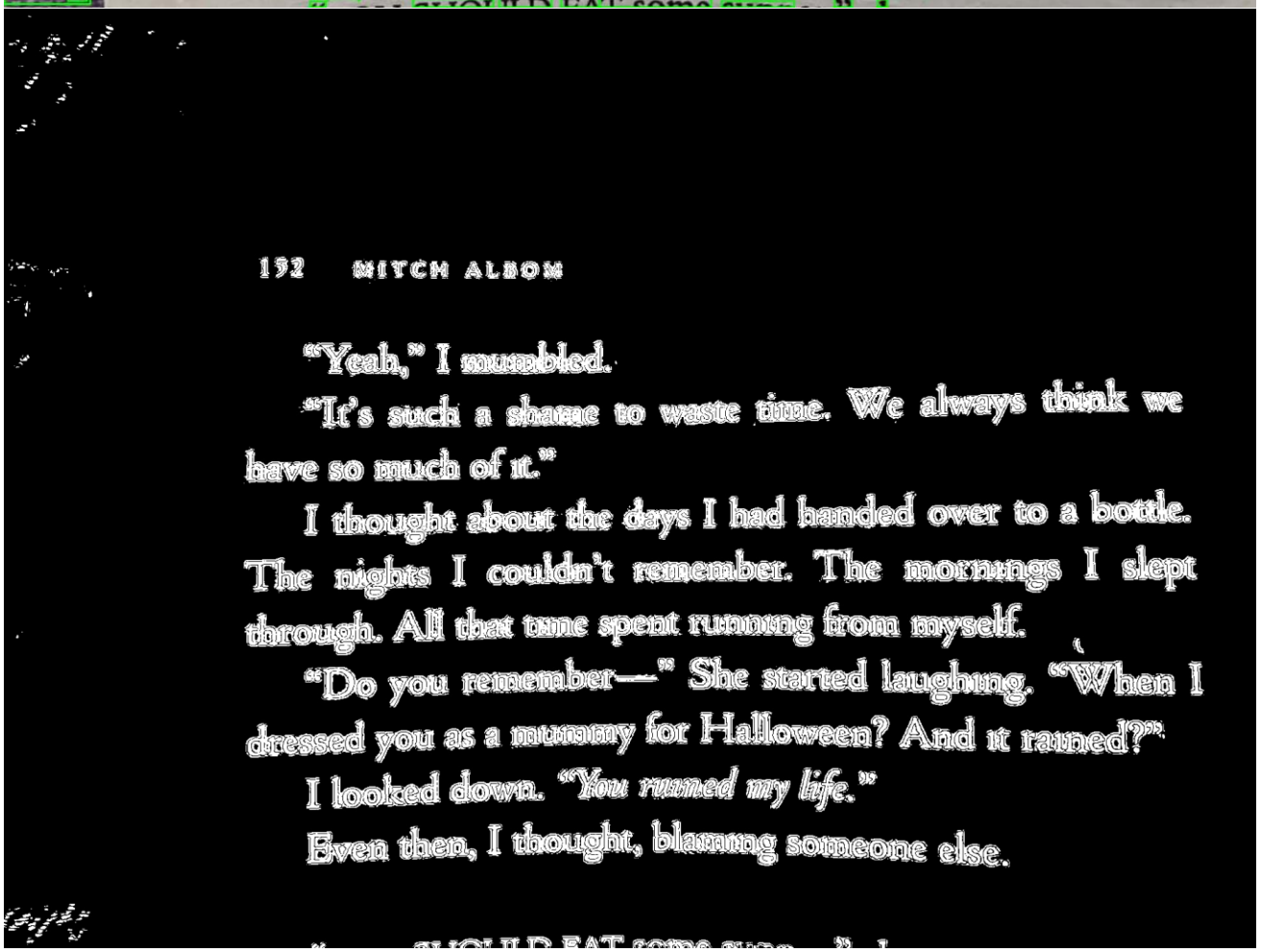
"It's such a shame to waste time. We always think we have so much of it."

I thought about the days I had handed over to a bottle. The nights I couldn't remember. The mornings I slept through. All that time spent running from myself.

"Do you remember—" She started laughing. "When I dressed you as a mummy for Halloween? And it rained?"

I looked down. "*You ruined my life.*"

Even then, I thought, blaming someone else.



"Yeah," I mumbled.

"It's such a shame to waste time. We always think we have so much of it."

I thought about the days I had handed over to a bottle. The nights I couldn't remember. The mornings I slept through. All that time spent running from myself.

"Do you remember—" She started laughing. "When I dressed you as a mummy for Halloween? And it rained?"

I looked down. "*You ruined my life.*"

Even then, I thought, blaming someone else.