Dijital Görüntü İşleme Ödev Raporu

1. S-Curve Kontrast Güçlendirme (Ödev 3)

Implementasyon

`algorithms/sigmoid_functions.py` modülünde 4 farklı sigmoid fonksiyonu implemente edilmiştir:

1. **Standard Sigmoid**
    - `standard_sigmoid(image, alpha=1.0)`
    - Temel sigmoid fonksiyonu: f(x) = 1 / (1 + e^(-alpha * x))
    - Görüntüyü [0,1] aralığına normalize eder
    - Kontrastı alpha parametresi ile ayarlar
2. **Shifted Sigmoid**
    - `shifted_sigmoid(image, alpha=1.0, shift=0.0)`
    - Yatay kaydırılmış sigmoid: f(x) = 1 / (1 + e^(-alpha * (x - shift)))
    - Kontrast ve parlaklık ayarı için shift parametresi
3. **Sloped Sigmoid**
    - `sloped_sigmoid(image, alpha=1.0, beta=1.0)`
    - Eğimli sigmoid: f(x) = beta * (1 / (1 + e^(-alpha * x)))
    - Beta parametresi ile eğim kontrolü
4. **Custom Sigmoid**
    - `custom_sigmoid(image, alpha=1.0, beta=1.0, gamma=1.0)`
    - Özel sigmoid: f(x) = beta * (1 / (1 + e^(-alpha * (x^gamma - 0.5))))
    - En esnek kontrast ayarı

2. Hough Transform Uygulamaları (Ödev 4)

Implementasyon

`algorithms/hough_transform.py` modülünde 3 farklı tespit fonksiyonu:

1. **Lane Line Detection**
    - `detect_lane_lines(image, rho=1, theta=np.pi/180, threshold=50, min_line_length=100, max_line_gap=10)`
    - Görüntüyü gri tonlamaya çevirir
    - Gaussian blur uygular
    - Canny edge detection ile kenarları tespit eder

- HoughLinesP ile çizgileri tespit eder

2. **Eye Detection**
   - `detect_eyes(image, scale_factor=1.1, min_neighbors=5, min_size=(30, 30))`
   - Haar Cascade Classifier kullanır
   - Gözleri tespit eder ve dikdörtgen çizer
3. **Circle Detection**
   - `detect_circles(image, dp=1.2, min_dist=100, param1=50, param2=30, min_radius=0, max_radius=0)`
   - Hough Circle Transform kullanır
   - Daireleri tespit eder ve çizer

3. Deblurring Algoritması (Ödev 5)

Implementasyon

`algorithms/deblurring.py` modülünde:

1. **Motion Blur Kernel Estimation**
   - `estimate_motion_blur_kernel(image, angle=0, length=10)`
   - Hareket bulanıklığı çekirdeğini oluşturur
   - Açı ve uzunluk parametreleri ile ayarlanabilir
2. **Wiener Deconvolution**
   - `wiener_deblur(image, kernel, K=0.01)`
   - FFT kullanarak Wiener dekonvolüsyonu uygular
   - K parametresi ile gürültü kontrolü
3. **Richardson-Lucy Deconvolution**
   - `richardson_lucy_deblur(image, kernel, iterations=30)`
   - İteratif dekonvolüsyon algoritması
   - Daha hassas sonuçlar için çoklu iterasyon

4. Nesne Sayma ve Özellik Çıkarma (Ödev 6)

Implementasyon

`algorithms/object_analysis.py` modülünde:

1. **Dark Green Region Detection**
   - `detect_dark_green_regions(image, hsv_lower=(35, 50, 50), hsv_upper=(85, 255, 255))`

- HSV renk uzayında koyu yeşil bölgeleri tespit eder
- Kontur analizi yapar

2. **Feature Extraction**
   - `calculate_region_features(image, contour)`
   - Her bölge için özellikler hesaplar:
     - Merkez koordinatları
     - Uzunluk/Genişlik/Çapraz
     - Enerji/Entropi
     - Ortalama/Medyan değerler

3. **Excel Export**
   - `export_to_excel(df, filename='analysis_results.xlsx')`
   - Analiz sonuçlarını Excel formatında kaydeder
   - Özelleştirilmiş başlık ve sütun formatlaması

Kullanıcı Arayüzü

Tüm bu fonksiyonlar PyQt5 tabanlı bir GUI ile entegre edilmiştir:

- Her ödev için ayrı sayfa
- Görüntü yükleme/kaydetme
- Parametre ayarlama kontrolleri
- Gerçek zamanlı görüntü işleme
- Sonuçları görselleştirme

Sonuç

Tüm ödev fonksiyonları başarıyla implemente edilmiş ve test edilmiştir. Kod modüler bir yapıda organize edilmiş ve her bir algoritma ayrı bir modülde bulunmaktadır. Kullanıcı arayüzü, tüm fonksiyonlara kolay erişim sağlamakta ve parametreleri dinamik olarak ayarlamaya olanak tanımaktadır.