Görüntü Filtreleme

- **Bulanıklaştırma**
    - **Gaussian Bullanıklaştırma:** Bu yöntem, her bir pikseli çevresindeki piksellerin medyan değeri ile değiştirerek çalışır. Görüntüyü yumuşatır ve görüntü üzerinde düzleştirme işlemi uygular. Uygulamak için ise "cv2.GaussianBlur()" metodu kullanılır.
    - **Medyan Bulanıklaştırma:** Görüntüdeki gürültüyü azaltmak ve kenarları korumak için kullanılan bir görüntü işleme tekniğidir. Uygulamak için ise "cv2.medianBlur()" metodu kullanılır.
    - **Bilateral bulanıklaştırma:** Görüntüdeki kenarları keskin tutarak gürültüyü azaltmak için kullanılan etkili bir görüntü işleme tekniğidir. Diğer bulanıklaştırma tekniklerinden farklı olarak, bu yöntem sadece mekansal yakınlığa değil, aynı zamanda renk benzerliğine de dayalıdır. Uygulamak için ise "cv2.bilateralFilter()" metodu kullanılır.
- **Kenar Algılama**
    - **Sobel Kenar Algılama:** Görüntüdeki yoğunluk gradyanlarını tespit ederek kenarları belirleyen bir tekniktir. Uygulamak için ise "cv2.Sobel()" fonksiyonu kullanılır.
    - **Canny Kenar Algılama:** Canny algoritması, görüntüdeki keskin kenarları belirlemek için birden fazla adım içeren bir süreçtir. Bu adımlar arasında gürültü azaltma, gradyan hesaplama, non-maximum suppression ve çift eşikli kenar takibi bulunur. Uygulamak için ise "cv2.Canny()" fonksiyonu kullanılır. Bu fonksiyon üç farklı temel parametre alır:

      'image': girdi görüntüsü, 'threshold1': alt eşik, 'threshold2': üst eşik
- **Morfolojik İşlemler**
    - **Erozyon:** Nesnelerin etrafındaki gürültüyü azaltmak ve küçük nesneleri ortadan kaldırmak için kullanılır. Uygulamak için ise "cv2.erode()" fonksiyonu kullanılır. Bu fonksiyon üç farklı temel parametre alır:

      'src': girdi görüntüsü, 'kernel': yapı elemanı, 'iterations': erozyon işleminin döngü sayısı
    - **Genişleme:** Genişleme erozyonun tam tersidir. Bu işlem, nesneleri daha büyük hale getirmek veya nesneleri birleştirmek için kullanılır.

Eşikleme Teknikleri

- **İkili Eşikleme:** Bu teknik, nesneleri arka plandan ayırmak ve daha basit bir yapı elde etmek için yaygın olarak kullanılır. Uygulamak için ise “cv2.threshold()” fonksiyonu kullanılır. Bu fonksiyon aşağıdaki temel parametreleri alır:

  src: Girdi görüntüsü, thresh: eşik değeri, maxval: eşik değeri aşıldığında atanacak maksimum değer, type: eşikleme türü

- **Adaptif Eşikleme:** Görüntüyü küçük bölgelere böler ve her bölge için ayrı bir eşik değeri hesaplar. Bu sayede farklı bölgelerdeki aydınlatma değişikliklerini daha iyi ele alır. Bu teknik, homojen olmayan arka planlarda veya yüksek kontrastlı görüntülerde daha iyi sonuçlar verir. Uygulamak için ise “cv2.adaptiveThreshold()” fonksiyonu kullanılır. Bu fonksiyonun parametreleri şunlardır:

  src: girdi görüntüsü, maxValue: maksimum piksel değeri, adaptiveMethod: adaptif eşikleme yöntemi, thresholdType: eşikleme tipi, blockSize: blok boyutu, C: sabit terim

- **Otsu's Eşikleme:** Görüntüdeki piksel yoğunluk histogramını analiz ederek, bu histogramı iki sınıfa bölen en uygun eşik değerini otomatik olarak belirlemeye çalışan bir tekniktir.

Kontur Algılama ve Analizi

- **Kontur Bulma:** Nesnelerin sınırlarını belirlemek için kullanılan bir yöntemdir. Kontur bulma işlemi “cv2.findContours()” fonksiyonu kullanılarak gerçekleştirilir.

- **Kontur Çizimi:** Bir görüntüde bulunan konturların görsel olarak vurgulanmasını sağlar. Bu işlem, nesnelerin sınırlarını belirlemek veya görüntüdeki belirli alanları işaretlemek için kullanılır. OpenCV kütüphanesi, konturları çizmek için “cv2.drawContours()“ fonksiyonunu sağlar.

- **Kontur Özellikleri:** Bir görüntüdeki konturların geometrik ve topolojik niteliklerini analiz etmek için kullanılır. Bu özellikler, konturun şeklini, boyutunu, konumunu ve diğer önemli bilgilerini içerebilir. Temel kontur özellikleri:

  - **Kontur Alanı:** Konturun iç kısmının kaç piksel kapladığını gösterir.

    “area = cv2.contourArea(contour)“ şeklinde ifade edilir.

- **Kontur Çevresi:** Konturun çevresini hesaplar. Bu, konturun etrafındaki uzunluğu temsil eder. “perimeter = cv2.arcLength(contour, True)” şeklinde ifade edilir.
- **Kontur Merkezi:** Konturun ağırlık merkezini hesaplar.

- **Şekil Analizi:** Nesnelerin veya şekillerin özelliklerini belirlemek ve tanımlamak için kullanılan bir tekniktir. Şekil analizi, genellikle kontur analizine dayanır.

  Şekil Analizi Adımları:

  1. **Görüntü Ön İşleme:**
     a. Gri Tonlamalı Dönüştürme: Renkli görüntüleri gri tonlamalıya çevirir.
     b. Eşikleme: Görüntüyü ikili hale getirir.
  2. **Kontur Bulma**:
     a. Kontur tespiti yapılır, böylece görüntüdeki nesnelerin sınırları belirlenir.

        “ cv2.findContours() “ fonksiyonu ile konturlar bulunabilir.
  3. **Şekil Özelliklerinin Çıkartılması**:
     a. **Alan**: Konturun içindeki piksel sayısı.
     b. **Çevre**: Şeklin çevresi.
     c. **Bağlayıcı Dikdörtgen**: Şekli çevreleyen en küçük dikdörtgen.
     d. **Minimum Çember:** Şekli çevreleyen en küçük çember.
     e. **Eğiklik**: Şeklin en ve boy oranı.
     f. **Konvekslik**: Şeklin konveks olup olmadığı durumu.
     g. **Yaklaşan Kontur**: Şeklin basitleştirilmiş hali.
     h. **Solidity**: Şeklin konveks hali ile alanı arasındaki oran.
  4. **Şekil Tanıma ve Karşılaştırma**:
     a. Şekil özellikleri kullanılarak nesneler tanımlanabilir veya karşılaştırılabilir. Örneğin, alan ve çevre değerlerini kullanarak üçgen, dikdörtgen veya daire gibi şekiller ayırt edilebilir.