ONDOKUZ MAYIS ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

Güldoğan Verdi 20060761

Muhammed Emirhan Türkmen 20060760

Azim Can Kuruca 16061657

Bilgisayarlı Görüye Giriş

Final Projesi

İÇİNDEKİLER

Kısım I

GİRİŞ

1

ANABAŞLIK

1.1 *Giriş*

Cep telefonu ile uygun zemin üzerinde çekilmiş meyve suyu kutusu (7x23 cm) orijinal görüntüsü Şekil 1'de verilmiştir.

1.2 *Maskeleme İşlemi*

Maskeleme işlemi, görüntü işleme alanında önemli bir yöntemdir. Bu işlem, bir görüntü üzerinde belirli bir bölgenin seçilmesi veya vurgulanması için kullanılmaktadır. Maskeleme sayesinde, ilgilenilen bölge ön plana çıkarılırken diğer bölgeler gizlenebilmektedir. Maskeleme işlemi adımlar halinde gerçekleştirilmektedir. İlk adımda, ilgili görüntü okunmaktadır. Ardından, belirli bir bölgenin seçilmesi için uygun yöntemler veya parametreler kullanılmaktadır. Bu seçilen bölge genellikle bir nesne veya önemli bir alanı temsil etmektedir.

Maske oluşturma adımında oluşturulan maske orijinal görüntüyle aynı boyuta sahiptir ve genellikle piksel değerlerini 0 ve 1 arasında temsil etmektedir. Maske, seçilen bölgeyi içeren pikselleri belirli bir değerle işaretlerken diğer pikselleri ise farklı bir değerle işaretlemektedir.

Şekil 1: Orijinal Evrak Görüntüsü

Böylece, seçilen bölge ön plana çıkarılırken diğer bölgeler gizlenmektedir. Son adımda, oluşturulan maske bitwise işlemleri kullanılarak orijinal görüntüyle birleştirilmektedir. Bitwise işlemleri, her pikselin maske ile karşılaştırıldığı ve sonuç olarak belirli bir işlem sonucu oluşan yeni bir görüntünün elde edildiği işlemlerdir. Bu işlem sonucunda, sadece maske tarafından belirlenen bölge orijinal görüntü üzerinde vurgulanırken diğer bölgeler etkisiz hale getirilmektedir. Bu bağlamda, Python programlama dili kullanılarak oluşturulan kod bloğu, evrakın dışında kalan bölgeleri maskeleme işlemiyle ayırarak yalnızca evraka ait görüntü

bölgesini elde etmeyi amaçlamaktadır. İlk olarak, maske oluşturmak için np.zeros_like(gray) kullanılarak, gray görüntüsü ile aynı boyuta sahip bir maske oluşturulur. Bu maske, piksel değerleri sıfırlardan oluşan bir görüntüdür. Daha sonra, cv2.drawContours(mask, [largest_contour], 0, (255), thickness=cv2.FILLED) koduyla maske üzerine kontürler çizilir. Burada largest_contour değişkeni, en büyük kontürü temsil eder ve bu kontür maske üzerinde beyaz (255) değerlerle işaretlenir. Kontürlerin çizildiği kalınlık thickness parametresiyle belirlenir. Son adımda ise cv2.bitwise_and(image, image, mask=mask) koduyla orijinal görüntü ve maske bitwise AND işlemine tabi tutulur. Bu işlem sonucunda, maske tarafından belirlenen bölgeler orijinal görüntü üzerinde vurgulanırken diğer bölgeler etkisiz hale getirilir ve maskeleme işlemi tamamlanır. Elde edilen maskeleme sonucu masked_image olarak adlandırılır. Bu sonuç görüntü, cv2.imwrite fonksiyonuyla "maskelenmis_goruntu.jpg" adlı bir dosyaya kaydedilir. Maskeleme ile ayrılmış evraka ait görüntü Şekil 2'de verilmiştir.

Maskeleme İşlevi:

```
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
closed = cv2.morphologyEx(edges, cv2.MORPH_CLOSE, kernel)


contours, _ = cv2.findContours(closed.copy(),
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)


largest_contour = max(contours, key=cv2.contourArea)


mask = np.zeros_like(gray)
cv2.drawContours(mask, [largest_contour], 0, (255),
```

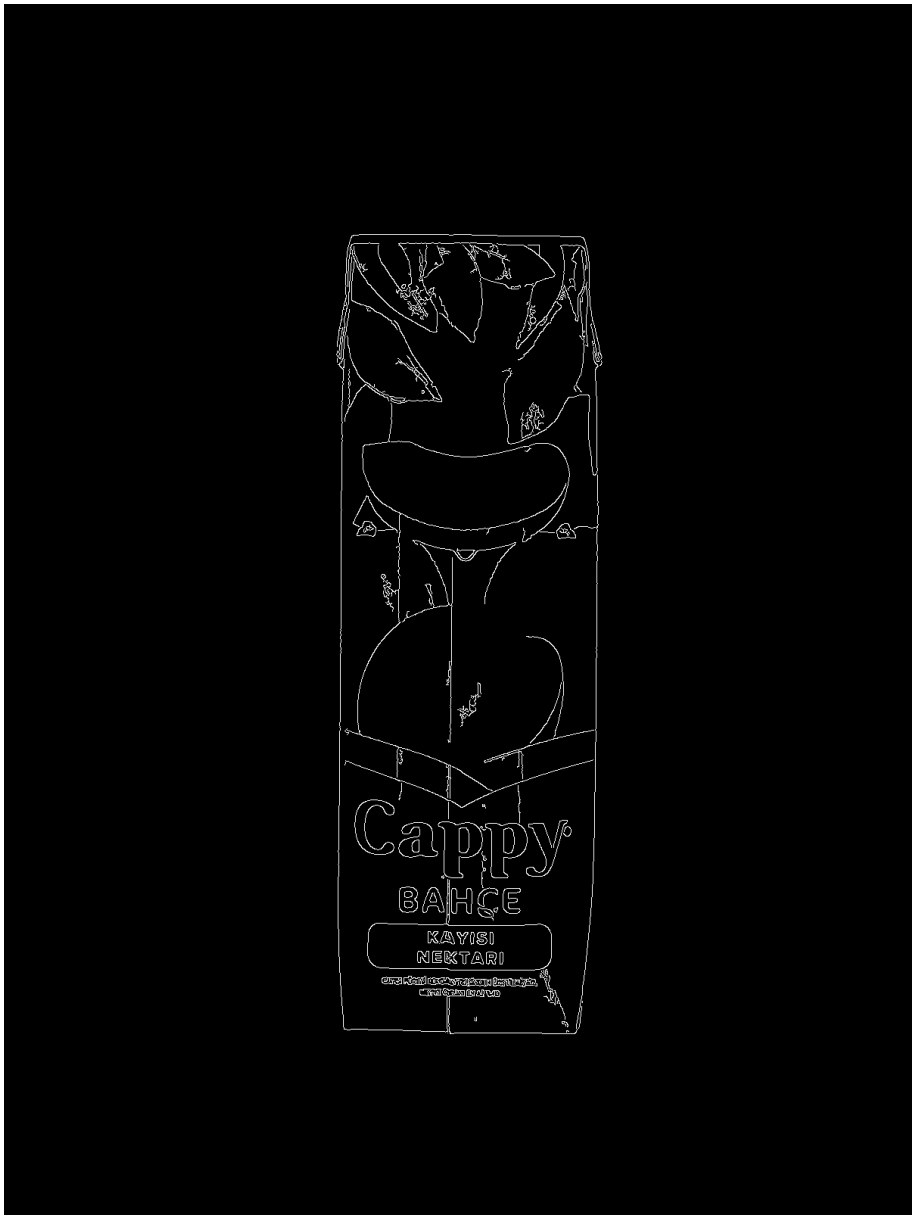
```
thickness=cv2.FILLED)

masked_image = cv2.bitwise_and(image, image, mask=mask)

cv2.imwrite("maskelenmis_goruntu.jpg", masked_image)
```

Şekil 2: Maskeleme İle Ayrılmış Evrak Görüntüsü

1.3 *Sınır Belirleme İşlemi*

Sınır belirleme işlemi, görüntü işleme alanında önemli bir adımdır. Bu işlem, görüntüdeki nesnelerin veya özelliklerin sınırlarını belirlemek için kullanılmaktadır. Sınır belirleme, genellikle kenar tespiti olarak da adlandırılmaktadır. Kenar tespiti, görüntüdeki yoğunluk veya renk değişikliklerini belirleyerek nesnelerin sınırlarını tanımlamaktadır. Bu değişiklikler, nesnelerin konturlarını oluşturan piksel değerlerindeki farklılıklardan kaynaklanmaktadır. Sınır belirleme işlemi, bir dizi matematiksel algoritma ve filtreleme yöntemi kullanarak gerçekleştirilmektedir. Canny kenar tespiti, Sobel operatörü, Laplacian filtresi gibi yöntemler sıkça kullanılan teknikler arasındadır. Bu teknikler, görüntüdeki yoğunluk gradyanlarını hesaplayarak kenar piksellerini tespit etmektedir. Projede kullanılan Python kod bloğunda ilk olarak, görüntü gri tonlamalı hale getirilmek için cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) kullanılarak orijinal renkli görüntü gri tonlamalı görüntüye dönüştürülmektedir. Daha sonra, kenar tespiti için cv2.Canny(blurred, 20, 150) kullanılarak gri tonlamalı görüntü üzerinde Canny kenar tespit algoritması uygulanmktadır. Bu işlemde, 20 ve 150 değerleri kenar tespiti için kullanılan düşük ve yüksek eşik değerlerini temsil etmektedir Ardından, kenarları kapatabilmek için bir morfolojik kapama işlemi uygulanmaktadır. cv2.morphologyEx(edges, cv2.MORPH_CLOSE, kernel) kullanılarak, kenarlı bölgeler kapatılır ve daha keskin bir sınırlama elde edilmektedir. Sonraki adımda, kontür tespiti gerçekleştirilmektedir. cv2. findContours(closed.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE) kullanılarak, kapatılmış kenar görüntüsü üzerindeki kontürler bulunmaktadır. Burada RETR_EXTERNAL parametresi, sadece dış kontürleri almayı sağlamaktadır. En büyük kontür, max(contours, key=cv2.contourArea) kullanılarak belirlenmektedir. Bu, görüntüdeki en geniş alanı temsil eden kontürü seçmektedir. Son olarak, cv2.drawContours(mask, [largest_contour],

0, (255), thickness=cv2.FILLED) ile en büyük kontür maske üzerine çizilmektedir. Bu, maskeleme için kullanılacak olan kontürü beyaz (255) renkle işaretlemektedir. Şekil 3'te kenarları tespit edilmiş görüntü verilmiştir.



Şekil 3: Sınırları Çizilmiş Evrak Görüntüsü

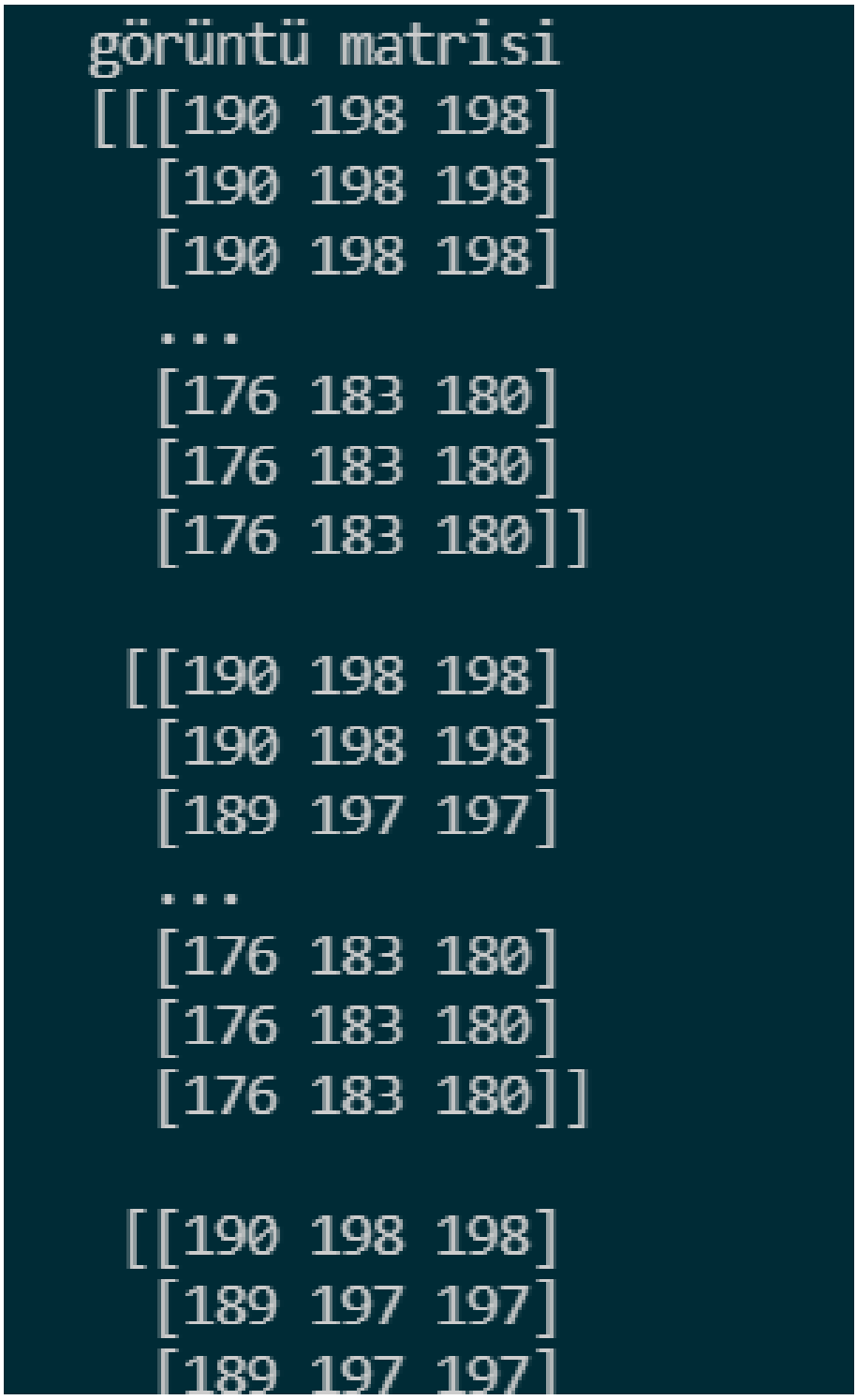
1.4 *Görüntü Matrisi Belirleme İşlemi*

Görüntü matrisi belirleme işlemi, bir görüntünün piksel değerlerini içeren bir matris oluşturulmaktadır. Bu işlem, görüntünün dijital temsilini sağlar ve görüntü işleme algoritmalarının uygulanması için temel bir veri yapısıdır.

Bir görüntü matrisi, her pikselin bir konumu temsil eden satır ve sütun indekslerine sahip bir 2-boyutlu dizi şeklinde ifade edilmektedir. Her bir piksel, genellikle renkli görüntülerde üç kanal (kırmızı, yeşil, mavi) veya gri tonlamalı görüntülerde tek bir kanal olmaktadır. Her pikselin matristeki konumu, pikselin matrisin hangi hücresine denk geldiğini belirtmektedir. Görüntü matrisi belirleme işlemi, görüntü işleme algoritmalarının piksel düzeyinde uygulanabilmesini sağlamaktadır. Matrisin her bir hücresi, görüntünün ilgili pikseline erişim ve manipülasyon imkanı sağlamaktadır. Bu sayede, görüntünün farklı bölgelerindeki piksel değerlerini analiz etmek, işlemek ve değiştirmek için matris operasyonları kullanılabilmektedir. Sonuç olarak, görüntü matrisi belirleme işlemi, bir görüntünün piksel değerlerini içeren bir matrisin oluşturmaktadır. Bu matris, görüntü işleme algoritmaları için bir temel veri yapısı ve piksel düzeyinde işlemler yapabilmemizi sağlamaktadır. Şekil 4'te görüntü matrisinin bir bölümü verilmiştir.

1.5 *Maskelenmiş Görüntü Matrisinin Kopyalanması*

Maskelenmiş görüntünün her pikselinin çıkış görüntü matrisindeki doğru indis değerine kopyalanması, piksel tabanlı bir dönüşüm işlemidir. Bu işlem, görüntü işleme uygulamalarında sıkça kullanılan bir yöntemdir. Maskelenmiş görüntüdeki her pikselin çıkış görüntü matrisindeki doğru konuma kopyalanması için bir dizi işlem gerçekleştirilir. İlk olarak, maskelenmiş



Şekil 4: Görüntü Matrisi

görüntünün boyutları ve çıkış matrisinin boyutları dikkate alınır. Her pikselin pozisyonunu belirlemek için maskelenmiş görüntü tek tek taranır. Her piksel için, pikselin konumu, yani satır ve sütun indeksi belirlenir. Bu konum bilgisi kullanılarak, çıkış matrisindeki

pikselin indis değeri hesaplanır. Hesaplama işlemi, genellikle matrisin boyutlarına göre yapılır. Bu hesaplama sonucunda, pikselin çıkış matrisindeki doğru konumu belirlenir. Ardından, maskelenmiş görüntüdeki pikselin değeri, hesaplanan çıkış matrisi indisine kopyalanır. Bu işlem, her piksel için tekrarlanır, böylece maskelenmiş görüntünün tüm pikselleri doğru konuma kopyalanır. Bu yöntem, maskelenmiş görüntüdeki bölgeyi sadece ilgili piksellerin kopyalandığı bir çıkış matrisiyle temsil etmeyi sağlar. Böylece, çıkış matrisinde yalnızca maskelenmiş bölgenin piksel değerleri bulunurken, diğer bölgeler siyah veya sıfır değeriyle temsil edilir. Bu işlem, görüntü analizi, nesne tanıma veya görüntü tabanlı algoritmalar gibi birçok uygulamada kullanılabilir.

1.6 *Median Filtre*

Çıkış görüntü matrisinde işlenmemiş pikseller için medyan filtre uygulanması, eksik veya bozuk piksel değerlerinin tahmin edilmesi amacıyla kullanılan bir yöntemdir. Bu işlem, görüntü restorasyonu veya gürültü azaltma gibi uygulamalarda yaygın olarak kullanılan bir tekniktir. Çıkış görüntü matrisinde işlenmemiş pikseller için medyan filtre uygulanması, eksik veya hatalı piksel değerlerini doğru bir şekilde tahmin etmek için kullanılır. Medyan filtre, bir pikselin değerini, çevresindeki komşu piksellerin medyan değeriyle değiştirir. Bu, piksel değerlerini düzeltmek ve görüntüdeki gürültüyü azaltmak için etkili bir yöntemdir. İlk olarak, çıkış görüntü matrisinde işlenmemiş piksellerin konumları belirlenir. Bu, eksik veya bozuk piksellerin bulunduğu bölgeleri tanımlamak için kullanılır. Ardından, her bir işlenmemiş piksel için bir medyan filtre uygulanır. Medyan filtre, pikselin komşu piksellerinin değerlerini sıralar, ardından bu sıralamadan ortadaki değeri (medyanı) kullanarak eksik pikselin değerini tahmin eder. Medyan filtre, çevresindeki piksellerin değerlerini kullanarak pikselin değerini

tahmin ettiği için lokal bir yaklaşımı temsil eder. Bu sayede, gürültülü veya bozuk piksellerin tahmin edilmesi daha doğru bir şekilde gerçekleştirilir. Medyan filtre uygulandıktan sonra, çıkış görüntü matrisindeki işlenmemiş piksellerin tahmini değerleri elde edilir. Medyan filtre, görüntüdeki tekil anomalileri düzeltme yeteneğine sahiptir ve çıkış görüntüsünün genel kalitesini artırabilir. Ancak, bu filtreleme işlemi piksel değerlerinde bazı düzeltmeler yaparken aynı zamanda detay kaybına yol açabilir. Bu nedenle, filtre parametreleri dikkatlice seçilmeli ve filtreleme işlemi öncesinde görüntü analizi ve segmentasyon gibi diğer işlemlerin etkileri değerlendirilmelidir. Şekil 5'te median filtre ugulanmış görüntü verilmiştir.

```
median_image = masked_image.copy()


ksize = 3


filtered_image = cv2.medianBlur(median_image, ksize)


cv2.imwrite("median_goruntu.jpg", filtered_image)
```

Şekil 5: Median Filtre