VERİ BİLİMİ DERSİ

OpenCV ile Görüntü İşleme

Hafta 11 · Modül 11
Dijital Görüntü, Filtreler, Kenar Tespiti ve Konturlar

Dr. Murat Altun

Veri Bilimi ve Yapay Zekâ Eğitimi · 2026

6
Saat
(Teori + Uygulama)

3
Notebook
(Temel, Filtre, Şekil)

10+
OpenCV Fonk.
(imread, Canny, vb.)

İçindekiler

Dijital Görüntü Nedir?

Piksel, Kanal ve Renk Derinliği

Dijital görüntü, satır ve sütunlardan oluşan bir piksel matrisidir. Her piksel bir renk değeri taşır. Gri tonlamalı görüntüde 1 kanal (0–255), renkli görüntüde 3 kanal (BGR veya RGB) bulunur. 8-bit derinlikte her kanal 256 farklı ton alabilir; 3 kanalla 16.7 milyon renk elde edilir.

Renk Kanalları

- **B (Mavi)** — Kanal 0 — Mavi tonlar
- **G (Yeşil)** — Kanal 1 — Yeşil tonlar
- **R (Kırmızı)** — Kanal 2 — Kırmızı tonlar

H×W×C	0–255	3	16.7M
Görüntü Boyutları	Piksel Değer Aralığı	Kanal Sayısı (BGR)	Renk Kombinasyonu

OpenCV Tanıtımı

Open Source Computer Vision Library

2000 yılında Intel tarafından başlatılan OpenCV, 2500+ optimize algoritma içerir. Python, C++, Java desteği ile bilgisayarlı görme alanının en yaygın kütüphanesidir. Kurulum: pip install opencv-python

Otonom Araçlar

Şerit takibi, nesne algılama, trafik işareti tanıma

Tıbbi Görüntüleme

X-ray, MRI analizi, tümör segmentasyonu, hücre sayımı

Güvenlik Sistemleri

Yüz tanıma, hareket algılama, plaka okuma sistemi

Endüstriyel Üretim

Kalite kontrol, ürün sayma, kusur tespiti, ölçüm

Görüntü Okuma ve Yazma


```python
import cv2
import matplotlib.pyplot as plt

# Görüntü okuma (BGR formatında)
img = cv2.imread('resim.jpg')

# Boyut bilgisi
print(img.shape)  # (yükseklik, genişlik, kanal)

# BGR → RGB dönüşümü ve gösterim
rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.imshow(rgb)
plt.axis(
'off')
plt.show()

# Görüntü kaydetme
cv2.imwrite('cikti.png', img)
```

cv2.imread()

Görüntüyü NumPy dizisi olarak yükler. Varsayılan format BGR'dir (RGB değil!). Dosya bulunamazsa None döner.

plt.imshow()

Matplotlib RGB bekler. Önce cvtColor ile dönüştürün.

cv2.imwrite()

PNG, JPG, BMP formatlarında kayıt.

Renk Uzayı Dönüşümleri

BÖLÜM 2

1 BGR → RGB

```
cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

Matplotlib ve diğer kütüphaneler RGB kullanır. Görselleştirme öncesi zorunlu dönüşüm.

2 BGR → Grayscale

```
cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Tek kanala düşürür (0–255). Eşikleme, kenar tespiti gibi işlemlerden önce gereklidir.

3 BGR → HSV

```
cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

Hue (renk tonu), Saturation (doygunluk), Value (parlaklık). Renk bazlı maskeleme için idealdir.

Görüntü Boyutlandırma


```
# Sabit boyuta yeniden boyutlandırma
resized = cv2.resize(img, (300, 200))

# Oran koruyarak küçültme
half = cv2.resize(img, None,
    fx=0.5, fy=0.5)

# Interpolasyon belirterek
up = cv2.resize(img, (800, 600),
    interpolation=cv2.
INTER_CUBIC)
```

Interpolasyon Yöntemleri

`INTER_NEAREST`	En hızlı, düşük kalite
`INTER_LINEAR`	Varsayılan, dengeli
`INTER_CUBIC`	Büyütmede yüksek kalite
`INTER_AREA`	Küçültmede en iyi sonuç

💡 Pratik İpucu

cv2.resize() genişlik × yükseklik (W, H) sırasıyla alır, ancak img.shape yükseklik × genişlik (H, W) döner. Bu fark sık karıştırılır! Modele beslenmeden önce tüm görüntülerin aynı boyutta olması gerekir (ör. 224×224).

Temel Bulanıklaştırma Filtreleri

Filtre	OpenCV Fonksiyonu	Kullanım Alanı	Kernel Boyutu
Box Blur	`cv2.blur(img, (5,5))`	Genel yumuşatma	3×3, 5×5, 7×7
Gaussian Blur	`cv2.GaussianBlur(img, (5,5), 0)`	Gürültü azaltma	Tek sayı: 3,5,7
Median Blur	`cv2.medianBlur(img, 5)`	Tuz-biber gürültüsü	Tek sayı: 3,5,7
Bilateral	`cv2.bilateralFilter(img, 9, 75, 75)`	Kenar koruyarak blur	d=5,7,9

Kernel (Çekirdek) Nedir?

Kernel, görüntü üzerinde kayan küçük bir matristir. Her pikselin yeni değeri, komşu piksellerin ağırlıklı ortalaması ile hesaplanır. Büyük kernel = daha fazla bulanıklaştırma. Gaussian kernel merkeze daha çok ağırlık verir.

Gaussian Blur — Kod Örneği


```python
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('foto.jpg')

# Farklı kernel boyutları ile karşılaştırma
kernels = [(3,3), (7,7), (15,15), (31,31)]

fig, axes = plt.subplots(1, 4, figsize=(16,4))
for ax, k in zip(axes, kernels):
    blur = cv2.GaussianBlur(img, k, 0)
    ax.imshow(cv2.cvtColor(blur,
        cv2.COLOR_BGR2RGB))
    ax.set_title(f'Kernel {k}')
plt.tight_layout()
plt.show()
```

3×3 — Hafif yumuşatma

7×7 — Orta bulanıklık

15×15 — Güçlü blur

31×31 — Çok güçlü blur

Kenar Tespiti: Canny Algoritması

Canny Algoritma Adımları

1	**Gürültü Azaltma**	Gaussian blur ile yumuşatma
2	**Gradyan Hesaplama**	Sobel filtresi ile x ve y yönlü türevler
3	**Non-Maximum Suppression**	İnce kenarlar elde etme
4	**Çift Eşikleme**	Güçlü ve zayıf kenarları ayırma
5	**Hysteresis**	Zayıf kenarları güçlülere bağlama

Düşük eşik → daha fazla kenar (gürültülü). Yüksek eşik → daha az ama keskin kenarlar. Oran genelde 1:2 veya 1:3 olarak seçilir.

```python
# Canny kenar tespiti
gray = cv2.cvtColor(img,
    cv2.COLOR_BGR2GRAY)

# threshold1=50, threshold2=150
edges = cv2.Canny(gray, 50, 150)

plt.imshow(edges, cmap='gray')
plt.show()
```

50
Alt Eşik
(threshold1)

150
Üst Eşik
(threshold2)

Görüntü Eşikleme (Thresholding)

1 Binary Threshold

```
ret, th = cv2.threshold(
    gray, 127, 255,
    cv2.THRESH_BINARY)
```

Piksel > eşik → beyaz, değilse → siyah. En temel eşikleme yöntemi.

2 Adaptive Threshold

```
th = cv2.adaptiveThreshold(
    gray, 255,
    cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
    cv2.THRESH_BINARY, 11, 2)
```

Bölgesel eşik hesaplar. Farklı aydınlatma koşullarında çok daha iyi sonuç verir.

3 Otsu Threshold

```
ret, th = cv2.threshold(
    gray, 0, 255,
    cv2.THRESH_BINARY +
cv2.THRESH_OTSU)
```

Optimum eşik değerini otomatik bulur. Bimodal histogramlarda mükemmel çalışır.

Görüntü Üzerine Şekil Çizme


```
import numpy as np

# Boş tuval oluştur (siyah, 400x600, 3 kanal)
canvas = np.zeros((400, 600, 3), dtype=np.uint8)

# Dikdörtgen: sol-üst (50,50), sağ-alt (200,150), yeşil, kalınlık 2
cv2.rectangle(canvas, (50,50), (200,150), (0,255,0), 2)

# Daire: merkez (300,200), yarıçap 60, mavi, dolu (-1)
cv2.circle(canvas, (300,200), 60, (255,0,0), -1)

# Çizgi: başlangıç (0,0), bitiş (600,400), kırmızı, kalınlık 3
cv2.line(canvas, (0,0), (600,400), (0,0,255), 3)

# Elips: merkez (450,300), eksenler (80,50), açı 30
cv2.ellipse(canvas, (450,300), (80,50), 30, 0, 360, (0,255,255), 2)
```

Görüntüye Metin Ekleme


```
# Metin ekleme
cv2.putText(
    img,
    'Merhaba OpenCV!',        # metin
    (50, 100),                # konum (x,y)
    cv2.FONT_HERSHEY_SIMPLEX, # font
    1.5,                      # ölçek
    (255, 255, 255),          # renk (beyaz)
    2,                        # kalınlık
    cv2.LINE_AA               # anti-alias
)
```

⚠️ OpenCV Türkçe karakter desteklemez (ö, ş, ğ, ü, ç, ı). Türkçe metin için PIL/Pillow kütüphanesini kullanın: ImageDraw.text() ile TTF font yükleyebilirsiniz.

Font Seçenekleri

`HERSHEY_SIMPLEX`	Sans-serif, düz
`HERSHEY_PLAIN`	Küçük, sade
`HERSHEY_DUPLEX`	Sans-serif, kalın
`HERSHEY_COMPLEX`	Serif, normal
`HERSHEY_TRIPLEX`	Serif, kalın
`HERSHEY_SCRIPT_SIMPLEX`	El yazısı
`ITALIC`	Herhangi biri + italik

Kontur Tespiti


```python
# Gri tonlama ve eşikleme
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
_, thresh = cv2.threshold(gray, 127, 255,
    cv2.THRESH_BINARY)

# Konturları bul
contours, hierarchy = cv2.findContours(
    thresh, cv2.RETR_TREE,
    cv2.
CHAIN_APPROX_SIMPLE)

# Konturları çiz (tümü, yeşil, kalınlık 2)
cv2.drawContours(img, contours, -1,
    (0,255,0), 2)
```

Kullanım Alanları

1. Nesne sayma (hücre, ürün)
2. Şekil tanıma ve sınıflandırma
3. Bölge seçme (ROI) ve kırpma
4. Belge tarama ve düzeltme
5. Hareket takibi (tracking)

Retrieval Modları

RETR_EXTERNAL

Sadece dış konturlar

RETR_LIST

Tüm konturlar (hiyerarşisiz)

RETR_TREE

Tam hiyerarşi ağacı

HSV ile Renk Tespiti


```python
# BGR → HSV dönüşümü
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

# Mavi renk aralığı tanımla
lower_blue = np.array([100, 50, 50])
upper_blue = np.array([130, 255, 255])

# Maske oluştur
mask = cv2.inRange(hsv,
    lower_blue, upper_blue)

# Maskeyi uygula
result = cv2.bitwise_and(
    img, img, mask=mask)
```

HSV Renk Aralıkları

Kırmızı	H: 0–10, 170–180
Turuncu	H: 10–25
Sarı	H: 25–35
Yeşil	H: 35–85
Mavi	H: 100–130
Mor	H: 130–170

🌍 Gerçek Dünya Örneği

Tarımda yeşil yaprak segmentasyonu, fabrikada renkli ürün ayırma, trafik ışığı tanıma — hepsi HSV maskeleme ile başlar.

Morfolojik İşlemler

1 Erosion (Aşındırma)

```
cv2.erode(img, kernel, iterations=1)
```

Beyaz bölgeleri küçültür. Gürültü noktalarını yok eder, nesneleri inceltir.

2 Dilation (Genişletme)

```
cv2.dilate(img, kernel, iterations=1)
```

Beyaz bölgeleri büyütür. Boşlukları doldurur, nesneleri kalınlaştırır.

3 Opening (Açma)

```
cv2.morphologyEx(img, cv2.MORPH_OPEN, k)
```

Erosion + Dilation. Küçük gürültü noktalarını temizler, nesne boyutunu korur.

4 Closing (Kapama)

```
cv2.morphologyEx(img, cv2.MORPH_CLOSE, k)
```

Dilation + Erosion. Küçük delikleri kapatır, nesne içi boşlukları doldurur.

Webcam ile Canlı Görüntü İşleme


```
# Webcam bağlantısı
cap = cv2.VideoCapture(0)  # 0 = varsayılan kamera

while True:
    ret, frame = cap.read()
    if not ret: break

    # Gerçek zamanlı filtre uygula
    gray = cv2.cvtColor(frame,
        cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray, 50, 150)

    cv2.imshow('Kenarlar', edges)

    if cv2.waitKey(1) & 0xFF == ord('q'):

break

cap.release()
cv2.destroyAllWindows()
```

İşlem Akışı

1	`VideoCapture(0)`	Kamerayı aç
2	`cap.read()`	Kare oku (frame)
3	`İşlem uygula`	Filtre, kenar, renk
4	`imshow()`	Pencerede göster
5	`waitKey(1)`	Tuş bekle (1ms)
6	`release()`	Kaynakları serbest bırak

Haftalık Notebook'lar

BÖLÜM 4

opencv_temel.ipynb

Görüntü okuma/yazma, BGR ↔ RGB dönüşümü, boyutlandırma, piksel manipülasyonu, renk uzayları

`cv2.imread` `cv2.resize` `cvtColor`

filtreler_kenar.ipynb

Blur filtreleri karşılaştırma, Canny kenar tespiti, eşikleme yöntemleri, morfolojik işlemler

`GaussianBlur` `Canny` `threshold`

sekil_metin.ipynb

Geometrik şekil çizme, metin ekleme, kontur tespiti, HSV maskeleme, webcam örneği

`drawContours` `putText` `inRange`

Tüm notebook'lar Google Colab'da çalışacak şekilde hazırlanmıştır. pip install opencv-python-headless komutu ile kurulum yapabilirsiniz.

Ödev ve Kaynaklar

1 5 Filtre Kolajı

Bir fotoğrafa 5 farklı filtre uygulayın (orijinal, blur, gaussian, median, bilateral). plt.subplots ile 2×3 grid oluşturup her birinin altına filtre adını yazın. Karşılaştırma yorumlarınızı markdown hücresine ekleyin.

2 HSV Renk Tespiti Projesi

Bir görüntüden belirli bir rengi (kırmızı veya mavi) HSV maskeleme ile izole edin. Orijinal, maske ve sonuç görüntülerini yan yana gösterin. Farklı renk aralıklarını deneyip en iyi sonucu belgeleyin.

📚 Kaynaklar

1. **OpenCV Resmi Dokümantasyon** — `docs.opencv.org`
2. **OpenCV-Python Tutorials** — `docs.opencv.org/4.x/d6/d00/tutorial_py_root.html`
3. **PyImageSearch Blog** — `pyimagesearch.com`
4. **Computer Vision: Algorithms and Applications** — `Szeliski (2022) — Ücretsiz PDF`

Hafta 11 — Önemli Çıkarımlar

1. Dijital görüntü = piksel matrisi. OpenCV ile NumPy dizisi olarak manipüle edilir.
2. BGR ↔ RGB ↔ HSV dönüşümleri, her görüntü işleme projesinin temelidir.
3. Filtreler gürültüyü azaltır, Canny kenarları tespit eder — ikisi birlikte güçlüdür.
4. Kontur tespiti ile nesneleri sayabilir, şekillerini analiz edebilirsiniz.
5. HSV maskeleme, renk bazlı nesne tespitinin en basit ve etkili yoludur.

“Bir bilgisayarın görmesini sağlamak, ona dünyayı piksel piksel öğretmekle başlar.”

Dr. Murat Altun · Veri Bilimi ve Yapay Zekâ Eğitimi · 2026