Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko

Osnove Računalniškega Vida
Segmentacija Slik

Avtor: Tilen Gašparič

Univerza v Mariboru
4. maj 2025

Kazalo

Slike

1 koda

1.1 kmeans

```
def kmeans(image, k=8, iterations=10, choice='r', T=10):
    centers = np.float32(izracunaj_centre(
        image,
        choice=choice,
        centerDimension=3,
        T=T
    ))

    h, w, c = image.shape
    pixelValues = np.float32(image.reshape((-1, 3)))

    for _ in range(iterations):
        distances = np.sqrt(
            (pixelValues[:, np.newaxis] - centers)**2)
        .sum(axis=2)
        labels = np.argmin(distances, axis=1)

        newCenters = np.zeros_like(centers)
        for i in range(k):
            clusterPixels = pixelValues[labels == i]
            if len(clusterPixels) > 0:
                newCenters[i] = clusterPixels.mean(axis=0)

        if np.allclose(centers, newCenters):
            break

        centers = newCenters

    segmentedCenters = centers.astype(np.uint8)
    segmentedImage = segmentedCenters[labels].reshape((h, w, c))

    return(segmentedImage)
```

1.2 izracunaj_centre

```
def izracunaj_centre(image, choice, centerDimension, T):
    centers = []
    if centerDimension == 3:
        features = image.reshape((-1, 3))
    elif centerDimension == 5:
        h, w = image.shape[:2]
        Y, X = np.mgrid[0:h, 0:w]
        features = np.hstack((
            image.reshape((-1, 3)),
            np.dstack((X, Y)
        ).reshape((-1, 2))))

    features = np.float32(features)

    if choice == 'r':
```

```
        displayImg = image.copy()

        def selectCenter(event, x, y, flags, param):
            if event == cv.EVENT_LBUTTONDOWN:
                if centerDimension == 3:
                    center = image[y, x]
                else:
                    center = np.append(image[y, x], [x, y])
                centers.append(center)
                cv.circle(displayImg, (x, y), 5, (0, 0, 255), -1)
                cv.imshow('center select', displayImg)

        cv.imshow('center select', displayImg)
        cv.setMouseCallback('center select', selectCenter)

        while True:
            key = cv.waitKey(1) & 0xFF
            if key == 27:  # ESC key
                break

        cv.destroyAllWindows()

    elif choice == 'n':
        maxAttempts = 1000
        attempts = 0

        while len(centers) < centerDimension and attempts < maxAttempts:
            candidate = features[random.randint(0, len(features) - 1)]

            valid = True
            for center in centers:
                if np.linalg.norm(candidate[:3] - center[:3]) < T:
                    valid = False
                    break

            if valid:
                centers.append(candidate)
                attempts = 0
            else:
                attempts += 1

        if attempts >= maxAttempts:
            print("took too many attempts (1000+) to find centers")

    return np.array(centers)
```

1.3 gaussian_kernel

```
def gaussian_kernel(distance, bandwidth):
    return np.exp(-0.5 * (distance / bandwidth) ** 2)
```

1.4 meanshift

```
def meanshift(
        image,
        velikost_okna=30,
        dimenzija=3,
        maxIterations=10,
        min_cd=5):
    h, w, _ = image.shape

    #define features
    if dimenzija == 3:
        features = image.reshape(-1, 3).astype(np.float32) / 255.0
    elif dimenzija == 5:
        Y, X = np.mgrid[0:h, 0:w]
        spatial = np.dstack((X/w, Y/h)).reshape(-1, 2)
        features = np.hstack((image.reshape(-1, 3)/255.0, spatial))

    def kernel(squaredDistance, windowSize):
        return np.exp(-squaredDistance / (2 * windowSize))

    visited = np.zeros(len(features), dtype=bool)
    finalCenters = []

    # spacial tree za buls neighbor search
    tree = BallTree(
        features[:, 3:] if dimenzija == 5 else features
    )

    #loop cez vse tocke
    for i in range(len(features)):
        if visited[i]:
            continue

        X = features[i].copy()
        converged = False

        for _ in range(maxIterations):
            if dimenzija == 5:
                indices = tree.query_radius(
                    [X[3:]],
                    r=velikost_okna
                )[0]
            else:
                indices = tree.query_radius(
                    [X],
                    r=velikost_okna
                )[0]

            neighbors = features[indices]

            squaredDistances = np.sum((neighbors - X)**2, axis=1)
            weights = kernel(squaredDistances, velikost_okna ** 2)
            sumOfWeights = np.sum(weights)

            if sumOfWeights == 0:
                break

```

```
            newX = np.sum(
                neighbors * weights[:, np.newaxis], axis=0
            ) / sumOfWeights

            #check if converge
            if np.linalg.norm(newX - X) < min_cd:
                converged = True
                break

            X = newX

        if converged:
            merged = False
            for j, center in enumerate(finalCenters):
                if np.linalg.norm(center - X) < min_cd:
                    finalCenters[j] = (center + X) / 2
                    merged = True
                    break

            if not merged:
                finalCenters.append(X)

            if dimenzija == 5:
                indices = tree.query_radius(
                    [X[3:]],
                    r=velikost_okna/2
                )[0]
            else:
                indices = tree.query_radius(
                    [X],
                    r=velikost_okna/2
                )[0]
            visited[indices] = True

    # assign points to clusters
    if not finalCenters:
        return image.copy()

    centers = np.array(finalCenters)
    distances = np.sqrt(
        (
            (features[:, np.newaxis] - centers) ** 2
        ).sum(axis=2)
    )
    labels = np.argmin(distances, axis=1)

    # create image
    segmented = centers[labels, :3]
    segmented = (segmented * 255).clip(0, 255).astype(np.uint8)
    return segmented.reshape((h, w, 3))
```

2 Primeri

2.1 Paprike

Slika 1: Originalna slika paprik

Slika 2: Izbor centrov za K-means algoritem

Slika 3: Rezultat K-means algoritma

Slika 4: Rezultat mean-shift algoritma v 3 dimenzijah

Slika 5: Rezultat mean-shift algoritma v 5 dimenzijah

2.2 Jack Black

Slika 6: Originalna slika Jack Blacka

Slika 7: Izbor centrov za K-means algoritem

Slika 8: Rezultat K-means algoritma

Slika 9: Rezultat mean-shift algoritma v 3 dimenzijah

Slika 10: Rezultat mean-shift algoritma v 5 dimenzijah

3 Vprašanja

3.1 Vprašanje 1

Demonstrirajte na praktičnem primeru (pripravite sliko), kdaj je smiselno v prostoru značilnic uporabiti tudi lokacije.

3D (BGR) se uporablja v primeru, ko je barva edina smiselna značilnica, po kateri lahko ločimo objekte (konfeti, predmeti pred uniformnim ozadjem, ...). Je tudi hitrejše računanje saj se težke komputacije opravijo samo 3x.

5D (BGRXY) se uporabi, ko hočemo narediti segmentacijo realne slike, kjer je v sliki več objektov s podobnimi barvami, ki jih želimo ločiti

3.2 Vprašanje 2

Kaj so prednosti enega in drugega algoritma? Kaj so njune slabosti?

Prednosti K-means algoritma so zelo hitra komputacija in izbor želene količine klustrov, slabosti pa to, da moramo v naprej vedet, koliko različnih predmetov je v sceni.

Prednosti meanshift algoritma so adaptabilnost (se samodejno odloči koliko objektov je v sceni) in natančnost (je bolj točen kot kmeans), slabosti pa zelo zahtevno računanje razdalj med vsemi piksli.