MODUL 7
IMAGE PROCESSING LANJUTAN

1.1 Tujuan Praktikum

1. Memahami konsep dasar thresholding dan mengaplikasikannya untuk memisahkan objek dari latar belakang pada gambar digital.
2. Menerapkan berbagai jenis filter untuk memperbaiki kualitas gambar, seperti smoothing, sharpening, dan edge detection.
3. Merancang dan mengimplementasikan pipeline sederhana untuk pemrosesan citra mulai dari preprocessing hingga ekstraksi fitur.
4. Mengevaluasi hasil pemrosesan gambar dan membandingkan efektivitas berbagai teknik thresholding dan filtering yang digunakan.

1.2 Alat dan Bahan

1. Laptop/PC
2. Visual Code
3. Library

1.3 Pendahuluan

Segmentasi objek adalah salah satu tahap penting dalam pemrosesan citra (Image Processing) yang bertujuan untuk memisahkan objek yang menjadi focus analisis dari latar belakang (background). Tujuan utamanya adalah untuk memudahkan identifikasi dan analisis objek dalam sebuah gambar, seperti dalam aplikasi pengenalan wajah, deteksi objek atau klasifikasi citra. Pada modul ini metode Teknik dasar segmentasi yang akan dipelajari yaitu menggunakan metode thresholding dan filtering, yang digunakan untuk memisahkan objek dari latar belakang serta meningkatkan kualitas citra.Serta bagaimana mengekstraksi informasi objek yang tersegmentasi menggunakan region properties agar dapat lebih memahami karakteristik dari objek.

1.4 Thresholding (Ambang Batas Citra)

1. Definisi Thresholding

   Thresholding adalah teknik segmentasi gambar yang memisahkan objek dari background berdasarkan nilai intensitas piksel. Secara umum, piksel yang memiliki nilai intensitas di atas atau dibawah suatu ambang (threshold) tertentu akan

diklasifikasikan kedalam objek atau background.

- Objek (biasanya berwarna putih)
- Latar belakang (biasanya berwarna hitam)

Ini dilakukan dengan memilih nilai ambang tertentu, lalu:

- Piksel dengan intensitas lebih besar dari ambang → diubah menjadi satu nilai (contoh: 255 atau putih)
- Piksel dengan intensitas lebih kecil dari ambang → diubah menjadi nilai lain (contoh: 0 atau hitam)

Prinsip dasar:
Jika adalah intensitas piksel pada koordinat (x,y) , maka hasil thresholding T (x,y) dapat didefinisikan sebagai:

$$T(x,y) = \begin{cases} 1, Jika\ I(x,y) > Threshold \\ 0, Lainnya \end{cases}$$

- I(x,y) adalah intensitas piksel pada koordinat (x, y) dalam citra grayscale.
- Threshold adalah nilai ambang yang telah ditentukan untuk memisahkan objek dari latar belakang.
- T(x, y) adalah hasil thresholding pada koordinat (x, y) yang akan bernilai:
    - 1 jika intensitas piksel I(x,y)I(x, y)I(x,y) lebih besar dari nilai threshold. Biasanya ini berarti piksel tersebut termasuk dalam objek (misalnya putih pada citra biner).
    - 0 jika intensitas piksel I(x,y)I(x, y)I(x,y) lebih kecil atau sama dengan nilai threshold. Ini berarti piksel tersebut termasuk latar belakang (misalnya hitam pada citra biner).

2. Jenis-jenis Thresholding

    a. Global Thresholding

    Global thresholding adalah metode dimana satu nilai ambang digunakan untuk seluruh gambar. Artinya, setiap piksel yang memiliki nilai lebih besar dari nilai threshold akan diubah menjadi nilai tertentu (misalnya putih), sementara piksel yang lebih kecil akan diubah menjadi nilai lainnya (misalnya hitam). Global thresholding cocok untuk digunakan pada gambar dengan pencahayaan merata atau gambar yang tidak memiliki perbedaan intensitas cahaya yang signifikan antara area yang satu dengan area lainnya.

    b. Adaptive Thresholding

    Adaptive thresholding berbeda dengan global thresholding karena nilai ambang dihitung untuk setiap bagian kecil gambar. Dengan kata lain, threshold untuk setiap piksel dihitung berdasarkan area sekitarnya. Metode ini lebih cocok digunakan Ketika pencahayaan pada gambar tidak merata seperti memiliki bagian

gelap dan terang. Dalam adaptive thresholding terdapat dua pendekatan umum:

- Mean Adaptive Thresholding, yaitu nilai ambang untuk setiap piksel dihitung berdasarkan rata-rata nilai piksel di sekitar area tersebut.
- Gaussian Adaptive Thresholding, yaitu nilai ambang dihitung dengan cara yang sama seperti mean adaptive thresholding. Namun menggunakan Gaussian untuk menghitung rata-rata nilai disekitar piksel. Ini memberikan hasil yang lebih harus karna bobot lebih pada nilai piksel lebih dekat dengan pusat.

```
thresh = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, 11, 2)
```

c. Otsu's Method

Otsu's method adalah Teknik thresholding otomatis yang tidak memerlukan nilai threshold yang ditentukan secara manual, sebaliknya otsu menghitung nilai ambang optimal dengan cara meminimalkan variasi dalam dua kelompok piksel (foreground dan background). Hal ini dilakukan dengan cara mencari nilai ambang yang menghasilkan pemisahan terbaik antara du akelas (objek dan latar belakang), berdasarkan histogram gambar. Otsu's method bekerja dengan baik pada gambar dengan histogram bimodal, yaitu gambar yang memiliki dua puncak (satu untuk objek dan satu untuk latar belakang), yang biasanya terjadi pada gambar dengan kontras tinggi antara latar belakang dan objek.

```
ret2, th2 = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)
```

3. Aplikasi Tresholding

Thresholding memiliki banyak aplikasi dalam berbagai bidang pemrosesan citra diantaranya:

- Segmentasi Objek

  Thresholding sering digunakan untuk memisahkan objek dar latar belakang, contoh aplikasi objek yaitu:

  - Misalnya memisahkan sel dalam gambar mikroskop
  - Memisahkan koin dari latar belakang agar lebih mudah dihitung jumlahnya
  - Memisahkan plat nomomr dari latar belakang kendaraan.
- Pra-pemrosesan untuk Analisis Lanjutan

  Setelah thresholding, gambar lebih sederhana karena hanya terdiri dari hitam dan

putih. Hal ini sangat berguna untuk:

- Kontur objek bisa dideteksi lebih mudah
- Fitur (seperti bentuk, ukuran) lebih mudah diambil karena terpisah jelas dari latar belakangnya
- Gambar yang telah disegmentasi dan disederhanakan dengan thresholding bisa digunakan sebagai input untuk algoritma machine learning untuk klasifikasi atau deteksi objek.

- OCR (Optical Character Recognition)

  Dalam aplikais OCR, thresholding membantu dalam mengekstrak teks dari gambar, Dengan thresholding, teks yang ada dalam gambar dapat dipisahkan dari latar belakang, sehingga mesin OCR dapat lebih mudah mengenali karakter yang ada di dalam gambar dan mengonversinya menjadi teks digital.
- Pengolahan Citra Medis

  Dalam pengolahan citra medis, thresholding digunakan untuk menandai area-area penting dalam gambar, seperti area tumor dalam citra medis (misalnya CT scan atau MRI). Dengan menggunakan thresholding, area yang mencurigakan atau abnormal bisa lebih mudah dikenali untuk analisis lebih lanjut.

1.5 Pipeline Pengolahan Citra

1. Definisi Pipeline Pengolahan Citra

   Pipeline pengolahan citra adalah serangkaian langkah yang diatur secara berurutan dan trstruktur untuk memproses citra atau gambar. Tujuan utamanya adalah untuk mengotomatiskan proses-proses pengolahan citra, menyederhanakan analisis data citra (terutama bila bekerja dengan dataset besar), dan memastikan bahwa hasil pengolahan gambar tetap konsisten. Pipeline dapat dianalogikan sebagai sebuah pabrik, gambar atau citra masuk di satu ujung, melewati beberapa tahapan pemrosesan (mesin), dan keluar di ujung lain sebagai hasil yang sudah siap.
2. Contoh Tahapan Pipeline (Langkah-langkah Umum)
   a. Input: Membaca Gambar
      - Membuka file gambar dari disk, kamera, atau sumber lain.
      - Bisa dalam berbagai format (JPG, PNG, TIFF, dll).

        Contoh (Python OpenCV):

        ```
        img = cv2.imread('gambar.jpg')
        ```

   b. Preprocessing: Membersihkan Gambar
      - Filter noise: mengurangi gangguan/kerusakan pada gambar.

- Contohnya: Median Filter, Gaussian Blur.

- Konversi ke Grayscale: dari gambar berwarna ke hitam-putih (1 channel), supaya lebih sederhana untuk tahap berikutnya.

  Contoh:

  ```
  gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
  blurred = cv2.GaussianBlur(gray, (5,5), 0)
  ```

c. Thresholding: Segmentasi

- Memisahkan objek dari latar belakang berdasarkan ambang tertentu.
- Hasilnya gambar biner (hitam-putih).

Contoh:

```
ret, binary = cv2.threshold(blurred, 127, 255, cv2.THRESH_BINARY)
```

d. Feature Extraction: Mengambil Informasi Penting

- Setelah objek terpisah, kita bisa menghitung:
  - Luas objek
  - Bentuk/konturnya
  - Panjang lebar
  - Posisi
  - dll.

  Misal: menemukan kontur dan menghitung area:

  ```
  contours, _ = cv2.findContours(binary, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
  for cnt in contours:
      area = cv2.contourArea(cnt)
      print("Luas objek:", area)
  ```

e. Output: Menyimpan atau Analisis Hasil

- Menyimpan gambar hasil pemrosesan.
- Menyimpan data fitur ke file (CSV, database).
- Menampilkan hasil analisis, grafik, laporan, dll.

Contoh:

```
cv2.imwrite('hasil.jpg', binary)
```

3. Keuntungan Menggunakan Pipeline
    - Standarisasi

      Pipeline memastikan bahwa semua gambar diproses dengan aturan yang sama. Tidak bergantung siapa yang memprosesnya, sehingga hasil pengolahan citra akan konsisten.
    - Reproduksibilitas

      Jika pipeline sudah dirancang dan diterapkan, maka hasil pengolahan citra akan selalu konsisten setiap kali pipeline tersebut dijalankan, bahkan jika dijalankan berulang kali. Ini memastikan hasil yang reproduksi atau bisa diulang dengan hasil yang sama.
    - Efisiensi

      Dengan pipeline, kita bisa memproses banyak gambar secara batch atau sekaligus dalam waktu yang lebih efisien dibandingkan dengan cara manual. Ini sangat berguna terutama bila bekerja dengan dataset besar, karena menghemat waktu dan usaha.
    - Mudah Dikembangkan

      Pipeline yang sudah ada bisa dengan mudah dikembangkan untuk menambahkan tahapan baru. Misalnya, setelah tahap segmentasi, kita bisa menambahkan langkah untuk deteksi tepi atau klasifikasi objek tanpa mengubah alur dasar pipeline yang sudah ada.

1.6 Filtering (Penyaringan Gambar)

1. Definisi Filtering

   Filtering dalam pengolahan citra adalah proses untuk memodifikasi gambar dengan tujuan memperbaiki kualitas atau mengambil informasi tertentu. Proses ini dilakukan dengan menggeser sebuah kernel atau filter di seluruh gambar, kemudian menghitung nilai baru untuk setiap piksel berdasarkan aturan yang ada di dalam kernel. Sederhananya, setiap piksel baru pada gambar adalah hasil dari kombinasi piksel-piksel sekitarnya sesuai dengan aturan yang ada dalam kernel.
2. Apa itu Kernel?

   Kernel adalah matriks kecil yang berisi nilai-nilai numerik yang menentukan

bagaimana kontribusi piksel-piksel di sekitarnya pada piksel baru. Kernel biasanya memiliki ukuran seperti 3×3, 5×5, atau ukuran lainnya.

Contoh kernel 3×3:

[1 1 1]

[1 1 1]

[1 1 1]

Ini adalah contoh sederhana untuk mean filter yang digunakan untuk meratakan gambar, di mana setiap piksel baru adalah rata-rata dari piksel-piksel sekitarnya. Kernel digeser ke seluruh gambar, biasanya dari kiri atas ke kanan bawah, dan setiap kali kernel berada di suatu posisi, proses perhitungan akan dilakukan.

3. Jenis-jenis Filtering
   a. Smoothing Filters (Filter Perata)

      Tujuan: Mengurangi noise atau membuat gambar tampak lebih halus.

      - Mean Filter (Average Filter)
        - Setiap piksel baru dihasilkan dari rata-rata nilai piksel di sekitarnya. Filter ini efektif untuk menghilangkan noise kecil pada gambar.
      - Gaussian Filter
        - Mirip dengan mean filter, namun memiliki bobot lebih besar pada piksel yang lebih dekat dengan pusatnya, sehingga menghasilkan hasil yang lebih halus daripada mean filter biasa.

      Contoh penerapan:

      ```
      blur = cv2.GaussianBlur(img, (5,5), 0)
      ```

   b. Sharpening Filters (Filter Penajam)

      Tujuan: Membuat tepi objek lebih tegas, memperjelas detail gambar.

      - Laplacian Filter
        - Filter ini digunakan untuk menyorot perubahan cepat dalam intensitas, seperti pada titik-titik tepi objek dalam gambar. Hasilnya akan lebih tajam dan berkontras tinggi

          Contoh:

          ```
          laplacian = cv2.Laplacian(img, cv2.CV_64F)
          ```

   c. Edge Detection Filters (Filter Deteksi Tepi)

      Tujuan: Mendeteksi batas/tepi antar objek dalam gambar.

      - Sobel Filter

- Menghitung perubahan intensitas dalam arah horizontal dan vertikal untuk mendeteksi tepi. Biasanya diterapkan pada dua arah (X dan Y)

  Contoh:

  ```
  sobelx = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=5)  # arah X
  sobely = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=5)  # arah Y
  ```

- Canny Edge Detector
  - Filter ini lebih kompleks dan melalui beberapa tahap untuk menghasilkan tepi yang bersih dan tipis. Tahapan tersebut meliputi smoothing, perhitungan gradien, non-maximum suppression, dan thresholding.

    Contoh: `edges = cv2.Canny(img, 100, 200)`

4. Konsep Penting dalam Filtering

   a. Convolution

   Convolution adalah proses dasar dalam filtering. Proses ini dilakukan dengan menggeser kernel di atas gambar, lalu untuk setiap posisi kernel, kalikan nilai kernel dengan nilai piksel gambar yang berada di bawahnya, dan jumlahkan hasilnya untuk menghasilkan nilai piksel baru. Secara matematis:

   $$output(x,y) = \Sigma \Sigma [kernel(i,j) \times input(x+i, y+j)]$$

   b. Kernel Size (Ukuran Kernel)

   Ukuran kernel sangat mempengaruhi hasil filtering:

   - Semakin besar ukuran kernel, semakin kuat efeknya.
     - Pada smoothing, gambar akan menjadi lebih blur, karena lebih banyak noise yang dihilangkan.
     - Pada deteksi tepi, tepi objek bisa menjadi kurang tajam jika kernel terlalu besar.
   - Biasanya, kernel yang digunakan berukuran ganjil (misalnya 3×3, 5×5, 7×7) agar ada titik tengah yang menjadi referensi. Perbandingan kernel size:
     - 3×3: Efek smoothing ringan, detail gambar masih terlihat.
     - 7×7: Efek smoothing lebih kuat, banyak detail gambar hilang.

Ringkasan Cepat

Filter	Tujuan	Contoh
Smoothing (Mean, Gaussian)	Mengurangi noise	Blur, denoise
Sharpening (Laplacian)	Menegaskan detail	Tajamkan gambar

Filter	Tujuan	Contoh
Edge Detection (Sobel, Canny)	Mendeteksi tepi objek	Segmentasi, kontur