ANALISIS

Google Colab

1. **Moving Average Filter**
   Filter Moving Average digunakan untuk mengurangi noise dalam gambar. Filter ini bekerja dengan mengambil rata-rata pixel di sekitar tiap pixel gambar.

   Penjelasan:

- **cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE)**: Membaca gambar dengan format grayscale (hanya intensitas pixel, tanpa warna).
- **np.ones((5, 5), np.float32) / 25**: Membuat kernel filter moving average 5x5 dengan setiap elemen bernilai 1/25.
- **cv2.filter2D(image, -1, kernel)**: Mengaplikasikan filter moving average pada gambar menggunakan kernel yang telah dibuat.
- **plt.subplot() dan plt.imshow()**: Menampilkan gambar asli dan gambar yang sudah difilter di samping satu sama lain.

   Analisis Hasil:
   Gambar yang sudah diterapkan filter moving average akan lebih halus dan mengurangi noise. Namun, detail gambar juga bisa sedikit hilang karena rata-rata pixel di sekitar area tersebut.

2. **Deteksi Fitur dengan SIFT (Scale-Invariant Feature Transform)**
   SIFT (Scale-Invariant Feature Transform) digunakan untuk mendeteksi dan menggambarkan fitur-fitur penting dalam gambar, yang dapat dikenali meskipun terjadi perubahan skala, rotasi, atau pencahayaan.

   Penjelasan:
- **cv2.SIFT_create()**: Membuat objek detektor SIFT yang akan digunakan untuk mendeteksi fitur dalam gambar.
- **sift.detectAndCompute(image, None)**: Mendapatkan keypoints (titik-titik fitur penting) dan deskriptor yang menggambarkan fitur tersebut.
- **cv2.drawKeypoints()**: Menggambar keypoints yang terdeteksi di atas gambar.
- **plt.imshow()**: Menampilkan gambar dengan keypoints yang terdeteksi.

   Analisis Hasil:
   Titik-titik yang ditunjukkan oleh keypoints adalah lokasi fitur penting dalam gambar. Fitur ini tahan terhadap perubahan ukuran dan rotasi, serta berguna dalam aplikasi seperti pencocokan gambar dan pelacakan objek.

3. **Representasi Histogram Gambar**
   Histogram gambar menggambarkan distribusi intensitas pixel dalam gambar. Ini berguna untuk menganalisis kontras dan pencahayaan gambar.

   Penjelasan:
- **cv2.calcHist([image], [0], None, [256], [0, 256])**: Menghitung histogram gambar grayscale. Ini menghasilkan distribusi frekuensi pixel berdasarkan intensitas dari 0 hingga 255.
- **plt.plot(histogram)**: Menampilkan histogram dalam bentuk plot dengan sumbu X sebagai intensitas pixel dan sumbu Y sebagai frekuensi.
- **plt.show()**: Menampilkan histogram.

   Analisis Hasil:
   Histogram menunjukkan sebaran intensitas pixel. Jika gambar gelap, histogram akan lebih terkonsentrasi di sisi kiri, sementara gambar terang lebih condong ke kanan. Histogram juga memberikan informasi tentang kontras gambar.

4. **Gaussian Smoothing**
   Gaussian Smoothing digunakan untuk menghaluskan gambar dengan cara memberikan bobot lebih pada pixel yang lebih dekat dengan pusat. Ini mengurangi noise secara lebih efektif dibandingkan filter rata-rata.

   Penjelasan:
- **cv2.GaussianBlur(image, (5, 5), 0)**: Menerapkan filter Gaussian blur pada gambar dengan kernel berukuran 5x5 untuk mengurangi noise.
- **plt.subplot() dan plt.imshow()**: Menampilkan gambar asli dan gambar yang telah di-blur (dengan Gaussian Smoothing).

   Analisis Hasil:
   Gambar yang telah diproses dengan Gaussian Smoothing akan lebih halus dan lebih sedikit noise, namun dapat mengurangi ketajaman gambar. Perbedaan antara gambar asli dan yang dihaluskan akan lebih terlihat pada detail kecil.

5. **Deteksi Tepi dengan Sobel Filter**
   Sobel Filter digunakan untuk mendeteksi tepi-tepi dalam gambar dengan cara mengukur perubahan intensitas antara pixel yang berdekatan.

   Penjelasan:
- **cv2.Sobel()**: Menggunakan Sobel filter untuk mendeteksi tepi pada gambar. sobel_x dan sobel_y mengukur gradien horizontal dan vertikal, masing-masing.
- **cv2.magnitude()**: Menghitung magnitude gradien berdasarkan tepi horizontal dan vertikal untuk menghasilkan gambar tepi.
- **plt.imshow()**: Menampilkan gambar tepi yang terdeteksi.

   Analisis Hasil:
   Hasil Sobel Filter menunjukkan tepi gambar. Area dengan perubahan intensitas yang besar akan muncul lebih terang, menunjukkan adanya tepi objek atau batas antar area.

6. **Representasi Fitur dengan Histogram of Oriented Gradients (HOG)**
   HOG adalah teknik ekstraksi fitur yang digunakan dalam deteksi objek, terutama dalam pengenalan bentuk dan objek dalam citra.

   Penjelasan:

- **hog(): Menghitung Histogram of Oriented Gradients (HOG) untuk gambar. Ini digunakan untuk menggambarkan tekstur atau fitur berbasis gradien dalam gambar.**
- **exposure.rescale_intensity(): Meningkatkan kontras gambar HOG untuk visualisasi yang lebih baik.**
- **plt.imshow(): Menampilkan gambar hasil HOG yang telah ditingkatkan kontrasnya.**

   Analisis Hasil:
   Gambar hasil HOG menunjukkan fitur berbasis gradien yang digunakan untuk mengenali pola atau objek dalam gambar. Gambar ini dapat digunakan untuk tugas pengenalan objek atau klasifikasi citra.

Webots

1. **Visual Tracking dengan OpenCV**
   Tujuan: Simulasi robot untuk mengikuti bola merah menggunakan thresholding HSV dan pengendali P sederhana.

   Langkah-langkah Analisis

- Inisialisasi Kamera dan Gambar:
  Robot menggunakan kamera untuk menangkap gambar secara real-time. Kamera disetel pada resolusi tertentu, dan gambar yang diperoleh diubah ke format yang sesuai untuk pemrosesan.
- Thresholding HSV:
  Proses thresholding HSV digunakan untuk mendeteksi bola merah. HSV (Hue, Saturation, Value) lebih efektif dibandingkan RGB untuk mendeteksi objek berdasarkan warna tertentu. Misalnya, warna merah pada gambar diidentifikasi dengan menggunakan rentang nilai untuk Hue, Saturation, dan Value.

```
hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
lower_red = np.array([0, 120, 70])
upper_red = np.array([10, 255, 255])
mask = cv2.inRange(hsv, lower_red, upper_red)
```

   Di sini, lower_red dan upper_red menentukan batas bawah dan atas untuk warna merah pada ruang warna HSV.

- Deteksi Kontur dan Pelacakan:
  Setelah objek merah terdeteksi, OpenCV mencari kontur yang mewakili objek. Kontur digunakan untuk menentukan posisi bola pada gambar.

```
contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
for contour in contours:
  if cv2.contourArea(contour) > 1000:
    (x, y, w, h) = cv2.boundingRect(contour)
    cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
```

Kontur yang memiliki area lebih besar dari ambang batas (1000 piksel) dianggap sebagai bola yang perlu dilacak.

- Pengendali P (Proportional Controller):
  Pengendali P digunakan untuk menentukan gerakan robot berdasarkan kesalahan posisi bola relatif terhadap pusat gambar. Pengendali ini mengatur kecepatan robot berdasarkan perbedaan antara posisi bola dan posisi pusat.

```
error = x_center - frame_width / 2
control_signal = Kp * error
```

Di sini, Kp adalah konstanta proporsional, dan error adalah perbedaan posisi bola dengan posisi pusat layar.

2. Document Scanner Simulation

Tujuan: Simulasi pemindai dokumen untuk mengubah gambar dokumen menjadi tampilan top-down menggunakan teknik image processing.

Langkah-langkah Analisis

- Segmentasi Berdasarkan Warna:
  Proses pertama adalah segmentasi gambar untuk memisahkan area dokumen dari latar belakang menggunakan teknik thresholding pada ruang warna HSV.

```
mask = segment_by_color(numpy_im, HSV_LOW_RANGE, HSV_UP_RANGE)
```

Fungsi segment_by_color() menggunakan rentang nilai HSV untuk mendeteksi area dokumen yang diinginkan.

- Deteksi dan Identifikasi Sudut Dokumen:
  Kontur ditemukan pada gambar dan dihitung panjang dan lebar area yang terdeteksi untuk memastikan bahwa area tersebut benar-benar berbentuk persegi panjang.

```
contours = sorted(contours, key=cv2.contourArea, reverse=True)[:3]
```

Fungsi ini memfilter kontur berdasarkan area terbesar yang ditemukan.

- Transformasi Perspektif:
  Setelah sudut dokumen terdeteksi, gambar diubah menggunakan transformasi perspektif agar tampilan dokumen menjadi tampak lurus dari atas.

M = cv2.getPerspectiveTransform(src_points, dst_points)
warped = cv2.warpPerspective(image, M, (width, height))

Fungsi cv2.getPerspectiveTransform() menghitung matriks transformasi, dan cv2.warpPerspective() digunakan untuk mengubah perspektif gambar.

- Penyesuaian Ukuran dan Letterbox:
  Setelah gambar ditekuk, gambar tersebut disesuaikan dengan ukuran tampilan yang diinginkan, menggunakan teknik letterboxing.

letter_box = np.zeros((int(rows), int(cols), int(channels)), dtype=np.uint8)

3. Fruit Detection Robot

Tujuan: Simulasi deteksi buah menggunakan computer vision dan kontrol lengan robot untuk mengambil dan meletakkan buah.

Langkah-langkah Analisis

- Deteksi Buah dengan OpenCV:
  Algoritma ini menggunakan teknik thresholding untuk mendeteksi objek buah dalam gambar berdasarkan warna atau fitur tekstur. Salah satu teknik yang umum digunakan adalah konversi gambar ke ruang warna HSV dan kemudian thresholding.

hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
mask = cv2.inRange(hsv, lower_fruit_range, upper_fruit_range)

Di sini, lower_fruit_range dan upper_fruit_range adalah rentang nilai HSV yang digunakan untuk mendeteksi warna buah tertentu.

- Pencocokan Kontur dan Posisi Buah:
  Setelah mendeteksi mask buah, kontur dihitung untuk mengidentifikasi lokasi buah dan mengukur ukuran objek untuk memastikan bahwa itu adalah objek yang relevan untuk diambil.

contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

- Kontrol Lengan Robot:
  Setelah posisi buah terdeteksi, koordinat objek digunakan untuk mengarahkan lengan robot menuju objek tersebut. Posisi buah diberikan ke kontroler untuk menggerakkan lengan robot ke posisi yang sesuai.

robot_arm.move_to_position(x, y, z)

robot_arm.move_to_position() akan mengarahkan lengan robot ke posisi x, y, dan z yang telah dihitung berdasarkan deteksi objek.