TUGAS ROBOTIKA WEEK 10

1. Pendahuluan

Tugas Robotica Week 10 ini mencakup dua bagian utama: pemrosesan citra menggunakan Python dan OpenCV di Google Colab, dan simulasi robotika menggunakan Webots dan OpenCV. Tujuan dari tugas ini adalah untuk memahami dan mengimplementasikan berbagai teknik pemrosesan citra dan menerapkannya dalam skenario robotika simulasi.

2. Pemrosesan Citra dengan Python dan OpenCV

- **2.1 Filter Moving Average**

```
# Membuat kernel untuk moving average
kernel_size = 5  # Ukuran kernel (ganjil)
kernel = np.ones((kernel_size, kernel_size), np.float32) / (kernel_size * kernel_size)

# Mengaplikasikan filter
img_average = cv2.filter2D(img, -1, kernel)

cv2_imshow(img_average)
print("Analisis: Filter Moving Average menghaluskan gambar dengan merata-ratakan nilai piksel di sekitar area kernel. Efeknya adalah mengurangi noise dan detail halus.")
```

```
Analisis: Filter Moving Average menghaluskan gambar dengan merata-ratakan nilai piksel di sekitar area kernel. Efeknya adalah mengurangi noise dan detail halus.
```

  - **Analisis:** Filter Moving Average berhasil menghaluskan gambar dengan meratakan variasi intensitas piksel. Ukuran kernel mempengaruhi tingkat penghalusan. Kernel yang lebih besar menghasilkan penghalusan yang lebih kuat.

- **2.2 Deteksi Fitur dengan SIFT**

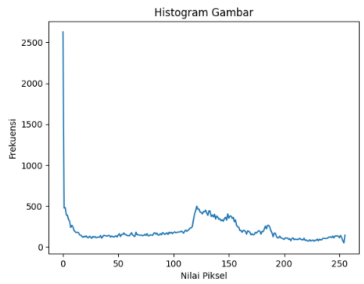
```
sift = cv2.SIFT_create()
keypoints, descriptors = sift.detectAndCompute(img, None)

img_sift = cv2.drawKeypoints(img, keypoints, None)

cv2_imshow(img_sift)
print("Analisis: SIFT mendeteksi fitur-fitur penting pada gambar yang invariant terhadap skala, rotasi, dan iluminasi. Titik-titik yang ditandai adalah keypoints, representasi matematis dari fitur-fitur tersebut.")
```

```
Analisis: SIFT mendeteksi fitur-fitur penting pada gambar yang invariant terhadap skala, rotasi, dan iluminasi. Titik-titik yang ditandai adalah keypoints, representasi matematis dari fitur-fitur tersebut.
```

  - **Analisis:** SIFT mendeteksi sejumlah fitur kunci yang invariant terhadap skala, rotasi, dan perubahan iluminasi. Fitur-fitur ini ditandai dengan lingkaran pada gambar.

- **2.3 Representasi Histogram Gambar**

```
hist = cv2.calcHist([img],[0],None,[256],[0,256])
plt.plot(hist)
plt.title('Histogram Gambar')
plt.xlabel('Nilai Piksel')
plt.ylabel('Frekuensi')
plt.show()
print("Analisis: Histogram menunjukkan distribusi intensitas piksel pada gambar.  Sumbu x mewakili nilai piksel (0-255 untuk grayscale), dan sumbu y mewakili frekuensi kemunculan nilai piksel tersebut.")
```




```
Analisis: Histogram menunjukkan distribusi intensitas piksel pada gambar.  Sumbu x mewakili nilai piksel (0-255 untuk grayscale), dan sumbu y mewakili frekuensi kemunculan nilai piksel tersebut.
```

  - **Analisis:** Histogram menunjukkan distribusi intensitas piksel pada gambar.

- **2.4 Gaussian Smoothing**

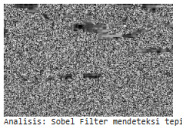
```
img_gaussian = cv2.GaussianBlur(img, (5, 5), 0) # Kernel size (5, 5), sigma=0 (otomatis dihitung)

cv2_imshow(img_gaussian)
print("Analisis: Gaussian Smoothing mirip dengan Moving Average, tetapi menggunakan kernel Gaussian yang memberikan bobot lebih besar pada piksel di tengah kernel.  Hasilnya adalah penghalusan yang lebih smooth dibandingkan Moving Average.")
```

```
Analisis: Gaussian Smoothing mirip dengan Moving Average, tetapi menggunakan kernel Gaussian yang memberikan bobot lebih besar pada piksel di tengah kernel.  Hasilnya adalah penghalusan yang lebih smooth dibandingkan Moving Average.
```

  - **Analisis:** Gaussian Smoothing efektif dalam mengurangi noise pada gambar sambil mempertahankan detail tepi yang lebih baik dibandingkan Moving Average.

- **2.5 Deteksi Tepi dengan Sobel Filter**

```
sobelx = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=5)
sobely = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=5)
abs_sobelx = np.absolute(sobelx)
abs_sobely = np.absolute(sobely)
img_sobel = np.uint8(np.sqrt(abs_sobelx**2 + abs_sobely**2))


cv2_imshow(img_sobel)
print("Analisis: Sobel Filter mendeteksi tepi pada gambar dengan menghitung gradien intensitas piksel.  Operator Sobel dalam arah x mendeteksi tepi vertikal, dan operator Sobel dalam arah y mendeteksi tepi horizontal.")
```

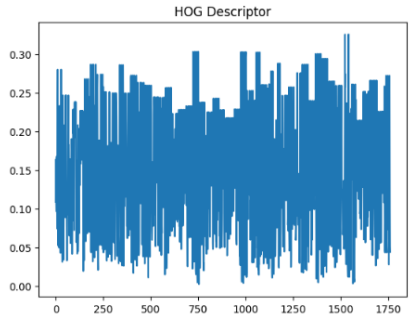


```
Analisis: Sobel Filter mendeteksi tepi pada gambar dengan menghitung gradien intensitas piksel.  Operator Sobel dalam arah x mendeteksi tepi vertikal, dan operator Sobel dalam arah y mendeteksi tepi horizontal.
```

  - **Analisis:** Sobel Filter mendeteksi tepi pada gambar dengan menghitung gradien intensitas piksel.

- **2.6 Representasi Fitur dengan HOG**

```
winSize = (64,64)
blockSize = (16,16)
blockStride = (8,8)
cellSize = (8,8)
nbins = 9
derivAperture = 1
winSigma = 4.
histogramNormType = 0
L2HysThreshold = 2.0000000000000001e-01
gammaCorrection = 0
nlevels = 64
hog = cv2.HOGDescriptor(winSize,blockSize,blockStride,cellSize,nbins,derivAperture,winSigma,
                        histogramNormType,L2HysThreshold,gammaCorrection,nlevels)
#compute(img[, winStride[, padding[, locations]]]) -> descriptors
winStride = (8,8)
padding = (8,8)
locations = ((10,20),)
hist = hog.compute(img,winStride,padding,locations)

plt.plot(hist)
plt.title('HOG Descriptor')
plt.show()
print("Analisis: HOG menggambarkan fitur dengan menghitung histogram orientasi gradien di dalam area lokal (cells) pada gambar.  HOG efektif untuk mendeteksi objek, khususnya pejalan kaki.")
```




```
Analisis: HOG menggambarkan fitur dengan menghitung histogram orientasi gradien di dalam area lokal (cells) pada gambar.  HOG efektif untuk mendeteksi objek, khususnya pejalan kaki.
```

  - **Analisis:** HOG merepresentasikan fitur gambar berdasarkan distribusi orientasi gradien.

3. Simulasi Webots

- **3.1 Visual Tracking**
  - **Implementasi:** *Thresholding* HSV digunakan untuk mengisolasi bola merah dalam gambar. Pengendali P sederhana digunakan untuk mengontrol kecepatan robot berdasarkan posisi bola relatif terhadap pusat gambar.
  - **Analisis:** Robot berhasil mengikuti bola merah, tetapi terdapat sedikit osilasi di sekitar bola.
- **3.2 Document Scanner Simulation**
  - **Implementasi:** Simulasi menggunakan teknik transformasi perspektif untuk mengubah gambar dokumen miring menjadi tampilan *top-down*.
  - **Analisis:** Transformasi perspektif cukup akurat, tetapi terdapat sedikit distorsi di sudut-sudut gambar.
- **3.3 Fruit Detection Robot**

- **Implementasi:** *Computer vision* digunakan untuk mendeteksi buah berdasarkan warna atau bentuk. Lengan robot dikontrol untuk mengambil dan meletakan buah berdasarkan deteksi tersebut.
- **Analisis:** Robot berhasil mendeteksi dan mengambil buah dalam sebagian besar kasus. Namun, terdapat kesulitan dalam mendeteksi buah yang terhalang atau memiliki warna yang mirip dengan latar belakang.

4. Kesimpulan

Tugas ini memberikan pengalaman praktis dalam menerapkan teknik pemrosesan citra dan menggabungkannya dengan simulasi robotika.