

Computer Vision

Bengkel Koding

2023-07-28

Table of contents

| | |
|---|-----------|
| Computer Vision | 4 |
| A. Tentang Buku ini | 4 |
| B. Pengenalan Kursus Computer Vision dan Python | 4 |
| Modul 1. Pengenalan Computer Vision dan Python | 8 |
| A. Pengenalan Computer Vision | 8 |
| B. Pengenalan Python | 10 |
| C. Python untuk Computer Vision | 11 |
| D. Latihan Praktik | 12 |
| E. Sesi Tanya Jawab dan Diskusi | 16 |
| Modul 2. Python | 17 |
| A. Dasar-dasar Python | 17 |
| Apa itu Python? | 17 |
| Lingkungan Python | 18 |
| Menjalankan Program Python | 21 |
| Pernyataan Dasar Python dan Jenis Data | 22 |
| B. Komputasi Menggunakan Module Python | 29 |
| Modul Python | 30 |
| NumPy | 31 |
| SciPy | 32 |
| Matplotlib | 33 |
| Python Image Library (PIL) | 34 |
| Scikits | 34 |
| Modul Python OpenCV | 35 |
| Modul 3. Dasar-dasar Image Processing | 36 |
| A. Pengenalan Image Processing | 36 |
| B. Konsep Dasar Image Processing | 37 |
| C. Python Untuk Image Processing | 41 |
| Spatial Filter | 41 |
| Modul 4. Feature Extraction and Matching | 43 |
| A. Pengenalan Feature Extraction and Matching | 43 |
| Feature Descriptor | 43 |
| Feature Matching | 45 |

| | |
|--|-----------|
| B. HOG dan SIFT | 46 |
| Histogram of Oriented Gradient (HOG) | 46 |
| Scale Invariant Feature Transform (SIFT) | 47 |
| C. BoW dan BoVW | 54 |
| References | 57 |

Computer Vision

A. Tentang Buku ini

Buku ini dirancang sebagai panduan komprehensif untuk memahami dan menerapkan teknik computer vision menggunakan bahasa pemrograman Python. Buku ini ditujukan untuk individu yang berminat dalam bidang computer vision, baik mereka yang baru memulai perjalanan mereka atau mereka yang sudah memiliki pengalaman dasar dalam bidang ini dan ingin memperdalam pengetahuan mereka.

Buku ini mencakup berbagai topik mulai dari konsep dasar computer vision dan image processing, hingga teknik machine learning dan deep learning yang canggih yang digunakan dalam aplikasi computer vision. Selain itu, buku ini juga mencakup bagian penting tentang etika dalam computer vision, suatu topik yang sering diabaikan tetapi sangat penting.

Dalam buku ini, pembaca akan belajar melalui penjelasan konsep, contoh kode, gambar, diagram, latihan, dan proyek praktis. Tujuan utama buku ini adalah untuk memberikan pemahaman yang solid tentang konsep dasar dan lanjutan dalam computer vision dan bagaimana menerapkannya dalam situasi nyata menggunakan Python.

B. Pengenalan Kursus Computer Vision dan Python

Kursus ini bertujuan untuk memperkenalkan Anda pada konsep dan praktik Computer Vision menggunakan Python. Selama kursus ini, Anda akan mempelajari dasar-dasar Computer Vision dan mengapa Python menjadi bahasa pemrograman yang populer dalam bidang ini.

Hal yang akan dipelajari dalam kursus ini meliputi:

Module 1: Pengenalan Computer Vision dan Python Pada modul ini, kita akan mempelajari pengenalan dasar tentang Computer Vision dan Python. Kami akan memahami definisi dan sejarah Computer Vision, serta aplikasi dan contoh penggunaannya dalam kehidupan sehari-hari. Kami juga akan membahas perbedaan antara Computer Vision dan Image Processing. Selain itu, Anda akan diperkenalkan dengan Python, termasuk sejarah, keunggulan, cara instalasi, dan struktur bahasa pemrograman Python. Kami akan memperkenalkan beberapa library Python yang umum digunakan dalam Computer Vision, seperti OpenCV, TensorFlow, dan Keras, dan cara menginstal dan menggunakannya.

Module 2: Dasar-dasar Image Processing Pada modul ini, kami akan membahas dasar-dasar Image Processing. Anda akan mempelajari definisi dan sejarah Image Processing, serta aplikasi dan contoh penggunaannya. Kami akan memperkenalkan konsep dasar dalam pemrosesan citra, termasuk pengenalan citra digital dan operasi dasar pada citra seperti transformasi, filtering, dan edge detection. Anda juga akan belajar tentang penggunaan library Python seperti OpenCV, PIL/Pillow, dan scikit-image untuk operasi dasar pemrosesan citra. Modul ini akan mencakup latihan praktik untuk menerapkan operasi dasar tersebut pada dataset gambar menggunakan Python.

Module 3: Feature Extraction and Matching Modul ini akan membahas tentang Feature Extraction dan Matching dalam Computer Vision. Kami akan menjelaskan definisi dan kegunaan Feature Extraction, serta memperkenalkan beberapa metode Feature Extraction seperti SIFT dan HOG. Anda akan mempelajari prinsip kerja dan implementasi SIFT dan HOG menggunakan Python dan OpenCV. Selain itu, kami juga akan membahas tentang Feature Matching, termasuk pengenalan metode-metode Feature Matching dan implementasinya menggunakan Python dan OpenCV. Modul ini akan melibatkan praktik implementasi SIFT, HOG, dan Feature Matching pada dataset gambar menggunakan Python dan OpenCV.

Module 4: Computer Vision dan Machine Learning Pada modul ini, kami akan menjelaskan hubungan antara Computer Vision dan Machine Learning. Anda akan mempelajari definisi dan sejarah Machine Learning, serta aplikasinya dalam Computer Vision. Kami akan membahas Klasifikasi Gambar, termasuk prinsip kerja dan implementasinya dengan Machine Learning menggunakan Python, TensorFlow, dan Keras. Selain itu, kami juga akan membahas Object Detection, termasuk prinsip kerja dan implementasinya dengan Machine Learning menggunakan Python, TensorFlow, dan Keras. Modul ini akan mencakup praktik implementasi Klasifikasi Gambar dan Object Detection pada dataset gambar menggunakan Python, TensorFlow, dan Keras.

Module 5: Deep Learning untuk Computer Vision Pada modul ini, kami akan memperkenalkan Deep Learning dalam Computer Vision. Anda akan mempelajari definisi dan sejarah Deep Learning, perbedaan antara Machine Learning dan Deep Learning, serta aplikasinya dalam Computer Vision. Kami akan menjelaskan Neural Networks dan Convolutional Neural Networks (CNN), serta prinsip kerja CNN dalam Computer Vision. Anda akan belajar menggunakan library Python seperti TensorFlow dan Keras untuk membangun dan melatih model CNN. Modul ini akan mencakup praktik membangun dan melatih model CNN dengan Python, TensorFlow, dan Keras, serta mengaplikasikannya pada dataset gambar untuk tugas klasifikasi.

Module 6: Object Detection dan Segmentation Pada modul ini, kami akan membahas tentang Object Detection dan Image Segmentation dalam Computer Vision. Anda akan mempelajari definisi, metode, dan teknik dalam Object Detection, serta pengenalan Image Segmentation dan perbedaannya dengan Object Detection. Kami akan memperkenalkan model-state-of-the-art seperti YOLO dan Mask R-CNN, dan penggunaan library Python seperti TensorFlow dan

Keras untuk mengimplementasikan model tersebut. Modul ini akan melibatkan praktik implementasi teknik Object Detection dan Image Segmentation pada dataset gambar menggunakan Python, TensorFlow, dan Keras.

Module 7: Computer Vision untuk Video Pada modul ini, kami akan menjelaskan penggunaan Computer Vision untuk video. Anda akan mempelajari perbedaan antara Computer Vision untuk gambar dan video, serta aplikasi Computer Vision dalam video. Kami akan membahas teknik tracking dalam video, termasuk definisi, metode, dan implementasinya. Selain itu, kami juga akan membahas activity recognition, termasuk definisi, metode, dan implementasinya. Anda akan menggunakan library Python seperti OpenCV, TensorFlow, dan Keras untuk mengimplementasikan teknik tracking dan activity recognition pada video. Modul ini akan melibatkan praktik mengimplementasikan teknik tersebut dengan menggunakan Python, OpenCV, TensorFlow, dan Keras.

Module 8: Transfer Learning dan Fine-tuning dalam Deep Learning Pada modul ini, kami akan memperkenalkan Transfer Learning dan Fine-tuning dalam Deep Learning. Anda akan mempelajari definisi, prinsip kerja, manfaat, dan aplikasi Transfer Learning dan Fine-tuning dalam Computer Vision. Kami akan menjelaskan cara menggunakan model pre-trained seperti VGG16 dan ResNet untuk tugas Computer Vision, serta melakukan fine-tuning pada model pre-trained tersebut. Modul ini akan melibatkan praktik mengimplementasikan Transfer Learning dan Fine-tuning pada model Deep Learning dengan menggunakan Python, TensorFlow, dan Keras.

Module 9: Computer Vision di Edge Devices Pada modul ini, kami akan membahas penggunaan Computer Vision di Edge Devices. Anda akan mempelajari definisi dan sejarah Edge Computing, perbedaan antara Cloud Computing dan Edge Computing, serta manfaat dan tantangan dalam menerapkan Computer Vision di edge devices. Kami akan menjelaskan teknik optimisasi model untuk edge devices, seperti quantization, pruning, dan knowledge distillation. Selain itu, Anda akan belajar tentang platform deployment seperti TensorFlow Lite dan ONNX, serta cara melakukan deployment model ke edge devices. Modul ini akan melibatkan praktik mengoptimalkan model Computer Vision untuk edge devices, dan melakukan deployment model ke edge devices menggunakan platform seperti TensorFlow Lite atau ONNX.

Module 10: Ethics in Computer Vision Pada modul ini, kami akan membahas etika dalam Computer Vision. Anda akan mempelajari peran etika dalam pengembangan dan penerapan teknologi Computer Vision, serta isu-isu utama dalam etika Computer Vision seperti privasi dan bias. Kami akan mendiskusikan strategi untuk mitigasi isu privasi dan bias. Modul ini akan melibatkan diskusi tentang kasus-kasus nyata yang berkaitan dengan isu etika dalam Computer Vision, serta bagaimana Anda dapat menerapkan prinsip etika dalam pekerjaan Anda sebagai praktisi Computer Vision.

Module 11: Capaian dan Masa Depan Computer Vision Pada modul terakhir ini, kami akan melakukan ringkasan materi yang telah dipelajari sepanjang kursus. Kami akan membahas capaian terkini dalam bidang Computer Vision, serta tantangan dan prospek masa depan dalam Computer Vision. Modul ini akan melibatkan diskusi tentang bagaimana Anda dapat

menerapkan pengetahuan dan keterampilan Anda dalam Computer Vision di masa depan, serta peluang karir dan pengembangan profesional dalam bidang Computer Vision.

Proyek-proyek: Selama kursus, Anda akan mengikuti serangkaian proyek yang melibatkan penerapan konsep dan teknik yang telah dipelajari. Proyek-proyek ini mencakup pembuatan sistem klasifikasi gambar, sistem deteksi dan pengenalan wajah, sistem deteksi objek dengan transfer learning, dan sistem klasifikasi kendaraan berbasis video tracking. Anda akan melalui tahap desain sistem, implementasi, evaluasi, dan praktik dalam setiap proyek untuk menguji pemahaman dan keterampilan Anda dalam Computer Vision.

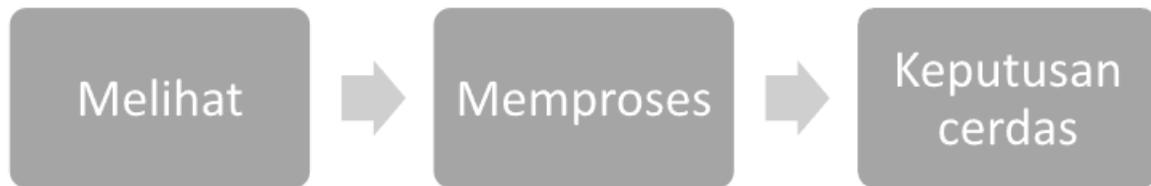
Dengan menyelesaikan kursus ini, Anda akan memperoleh pengetahuan dan keterampilan praktis yang diperlukan untuk mengembangkan dan menerapkan solusi Computer Vision. Anda akan mampu menggunakan Python dan berbagai library terkait untuk memproses gambar dan video, melakukan ekstraksi fitur, membangun dan melatih model Deep Learning, serta menerapkan teknik Computer Vision pada berbagai aplikasi.

Modul 1. Pengenalan Computer Vision dan Python

Chapter ini membahas pengenalan Computer Vision dan Python dalam konteks visi komputer. Bagian pertama menjelaskan konsep dasar Computer Vision, tantangan yang dihadapinya, dan manfaatnya dalam berbagai bidang industri. Bagian kedua memperkenalkan Python sebagai bahasa pemrograman yang kuat dan populer dalam visi komputer, dengan pustaka seperti NumPy dan OpenCV yang mendukung pengembangan solusi analisis gambar dan video. Bagian selanjutnya membahas penggunaan Python untuk visi komputer, termasuk membaca gambar, melakukan operasi pada gambar, dan menggunakan algoritma untuk mengenali objek. Dilanjutkan dengan latihan praktik untuk mengaplikasikan konsep yang telah dipelajari. Terakhir, terdapat sesi tanya jawab dan diskusi untuk berinteraksi, mengatasi tantangan, dan berbagi wawasan tentang visi komputer dan penggunaan Python dalam bidang tersebut.

A. Pengenalan Computer Vision

Visi komputer adalah bidang ilmu komputer yang bertujuan untuk memungkinkan komputer memproses dan mengidentifikasi gambar dan video dengan cara yang sama seperti penglihatan manusia. Visi komputer bertujuan untuk meniru sistem visual manusia. Tujuan utamanya adalah membangun sistem buatan yang dapat mengekstrak informasi dari gambar, yaitu membuat komputer memahami gambar dan video. Data gambar dapat berupa urutan video, gambar kedalaman, pandangan dari beberapa kamera, atau data multidimensional dari sensor gambar. Tujuan utama visi komputer adalah menggambarkan sebuah scene dunia nyata dalam satu atau lebih gambar, dan mengidentifikasi serta merekonstruksi propertinya, seperti karakteristik warna, informasi bentuk, karakteristik tekstur, pencahayaan scene, dan sebagainya.



Kita dapat melihat kesamaan antara sistem penglihatan manusia dan sistem visi komputer. Seperti yang diilustrasikan dalam Gambar 1, prinsip dasar dari kedua sistem ini hampir sama,



Figure 1: Gambar 1: Sistem penglihatan manusia (a) vs. visi komputer (b)

yaitu konversi cahaya menjadi sinyal/informasi yang berguna untuk membangun model akurat dari dunia fisik. Demikian pula, jika dilihat secara keseluruhan, struktur visi manusia dan visi komputer agak mirip, yaitu keduanya memiliki sensor cahaya yang mengubah foton menjadi sinyal (gambar), langkah pemrosesan, dan akhirnya mekanisme untuk menginterpretasi sinyal (pengenalan objek).

Perbedaan antara visi komputer, pemrosesan gambar, dan grafika komputer dapat disimpulkan sebagai berikut:

Dalam Visi Komputer (analisis gambar, interpretasi gambar, pemahaman scene), inputnya adalah gambar dan outputnya adalah interpretasi sebuah scene. Analisis gambar berkaitan dengan pengukuran kuantitatif dari sebuah gambar untuk memberikan deskripsi tentang gambar tersebut. Dalam Pemrosesan Gambar (pemulihan gambar, rekonstruksi, penyaringan, kompresi, visualisasi), inputnya adalah gambar dan outputnya juga berupa gambar. Terakhir, dalam Grafika Komputer, inputnya adalah sebuah scene dunia nyata dan outputnya adalah sebuah gambar. Visi komputer membuat model dari gambar (analisis), sedangkan grafika komputer menggunakan model sebagai input dan mengonversinya menjadi gambar (synthesis). Dalam perspektif Hukum Bayes, konsep ini dapat dijelaskan sebagai berikut:

$$P(\text{Gambar}) = \frac{P(\text{Dunia}) \times P(\text{Dunia})}{P(\text{Gambar})}$$

Dalam persamaan ini, $P(\text{Dunia}|\text{Gambar})$ merupakan Visi Komputer, $P(\text{Dunia})$ mengacu pada pemodelan objek dalam dunia nyata, dan $P(\text{Gambar}|\text{Dunia})$ adalah Grafika Komputer. Perspektif ini mendorong pendekatan pembelajaran statistik. Oleh karena itu, visi komputer berkaitan dengan pengembangan mesin-mesin yang dapat melihat dan berinteraksi dengan dunia.

Konsep ini diilustrasikan dalam Gambar 2.

Visi komputer saat ini digunakan dalam berbagai aplikasi dunia nyata, termasuk inspeksi mesin, pengenalan karakter optik (OCR), pembangunan model 3D (fotogrametri), analisis gambar medis, pengawasan video otomatis, biometrik, fusi dan penyambungan gambar, morphing, pemodelan 3D, dan lain-lain. Seperti yang ditunjukkan dalam Gambar 3, visi komputer terkait dengan banyak bidang penelitian penting.

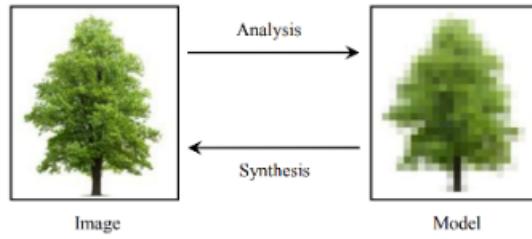


Figure 2: Gambar 2: Visi Komputer vs Grafika Komputer

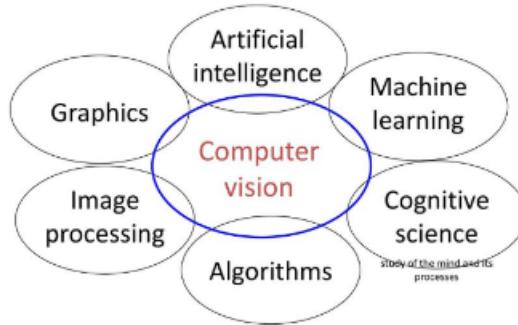


Figure 3: Gambar 3: Computer vision dan disiplin terkait

Selain itu, visi komputer juga menghadapi tantangan etika dan privasi. Penggunaan visi komputer dalam pengawasan dan pengenalan wajah telah menimbulkan pertanyaan tentang privasi dan hak individu. Bagaimana data visual dikumpulkan, digunakan, dan disimpan perlu diatur dengan hati-hati untuk melindungi privasi individu dan mencegah penyalahgunaan teknologi.

Namun, dengan penelitian dan pengembangan yang berkelanjutan, visi komputer, sebagai subbidang ilmu komputer dengan tujuan membangun mesin agar bisa memproses dan menginterpretasikan gambar dan video seperti yang dilakukan sistem visual manusia, terus berkembang dan memperbaiki keterbatasannya, menawarkan prospek yang menjanjikan untuk masa depan.

B. Pengenalan Python

Python adalah sebuah bahasa pemrograman yang telah matang dan menawarkan sifat open source, yang berarti kode sumbernya tersedia secara bebas dan dapat diubah atau dikembangkan lebih lanjut oleh siapa saja. Sebagai bahasa yang mudah dipelajari, Python telah berhasil mengumpulkan basis pengguna yang sangat luas. Komunitas ini, yang terdiri dari jutaan pengguna di seluruh dunia, siap membantu Anda mengembangkan keterampilan dan

pemahaman Anda tentang Python, baik Anda seorang pemula atau programmer berpengalaman.

Basis pengguna Python, yang beragam dan aktif, telah berkontribusi pada pengembangan berbagai alat dan perpustakaan pendukung. Perpustakaan ini adalah kumpulan rutinitas yang telah dikompilasi sebelumnya, dan dirancang untuk memfasilitasi berbagai upaya ilmiah dan teknis. Mulai dari data science, machine learning, pemrosesan bahasa, robotika, hingga visi komputer, Python telah memberikan solusi yang kuat dan fleksibel. Itulah sebabnya Python telah menjadi salah satu bahasa komputasi ilmiah yang paling penting, baik dalam lingkungan akademisi maupun industri.

Popularitas Python tentu saja datang dengan tantangannya sendiri. Seperti hutan belantara, ekosistem Python telah berkembang menjadi sesuatu yang tampak rumit dan sulit ditembus. Beberapa ilmuwan dan profesional baru dalam dunia Python mungkin merasa frustrasi dan stres saat berhadapan dengan banyaknya pilihan yang tersedia. Mereka harus membuat keputusan penting seperti perpustakaan mana yang harus digunakan untuk menggambar grafik atau editor teks mana yang harus digunakan untuk menulis program mereka.

Fleksibilitas Python datang sebagai solusi. Python mampu menangani berbagai format data, menjalankan peralatan ilmiah, dan berintegrasi dengan bahasa tingkat rendah seperti C, C++, dan FORTRAN. Python dapat digunakan sebagai “bahasa perekat”, memungkinkan integrasi berbagai skrip atau sistem yang berbeda. Jadi, meskipun ada banyak pilihan, Python menawarkan kemampuan untuk memilih dan menyesuaikan sesuai dengan kebutuhan spesifik Anda, yang menjadikannya lebih mudah bagi pengguna baru untuk memulai dan berkembang.

C. Python untuk Computer Vision

Computer Vision adalah sebuah cabang dari ilmu komputer yang memungkinkan mesin untuk memahami dan memanipulasi konten visual. Dengan adanya teknologi ini, kita bisa mendapatkan banyak keuntungan seperti dalam pengenalan wajah, deteksi objek, hingga analisis gambar dan video dalam bidang medis. Dalam bidang ini, Python telah menjadi bahasa pemrograman yang sangat berharga dan penting.

Python adalah bahasa pemrograman yang telah matang dan terbuka (open-source), membuatnya menjadi pilihan yang sangat baik untuk computer vision. Python menyediakan sintaks yang mudah dibaca dan dipahami, sehingga memudahkan pengguna baru untuk memahami dan memanfaatkan berbagai perpustakaan pendukung yang tersedia untuk computer vision. Di antara perpustakaan tersebut, ada OpenCV, TensorFlow, dan PyTorch, yang semuanya penting untuk pengembangan dan implementasi solusi computer vision.

OpenCV (Open Source Computer Vision Library) adalah perpustakaan yang sangat populer dalam bidang pengolahan gambar dan computer vision. Dikembangkan oleh Intel dan dise-

barkan dengan lisensi BSD, OpenCV memungkinkan pengembangan solusi computer vision dengan cepat dan efisien, baik untuk aplikasi real-time maupun offline.

Sementara itu, TensorFlow dan PyTorch adalah kerangka kerja pembelajaran mesin yang mendukung operasi yang diperlukan untuk pekerjaan computer vision. Keduanya mendukung teknik pembelajaran mesin canggih seperti jaringan saraf dan pembelajaran mendalam (deep learning) yang sering digunakan dalam aplikasi computer vision modern.

Python juga memiliki kelebihan dalam penanganan data visual. Dengan Python, pengguna dapat dengan mudah membaca, menulis, dan memanipulasi gambar dan video dalam berbagai format. Python juga mendukung operasi pra-pemrosesan yang diperlukan untuk data visual, seperti perubahan ukuran gambar, normalisasi, dan augmentasi data.

Terakhir, Python juga menawarkan kemudahan dalam visualisasi data dan hasil. Dengan menggunakan perpustakaan seperti Matplotlib dan Seaborn, pengguna dapat dengan mudah memvisualisasikan data dan hasil dalam berbagai format. Dengan kata lain, dengan Python, computer vision menjadi lebih mudah diakses dan dipahami, baik oleh profesional maupun pemula. Python telah membuktikan dirinya sebagai bahasa pemrograman yang kuat dan fleksibel yang dapat mendukung berbagai tugas dan proyek dalam bidang computer vision.

D. Latihan Praktik

Python merupakan bahasa pemrograman yang ideal untuk belajar dan menerapkan visi komputer. Berikut adalah beberapa latihan praktik yang dapat Anda lakukan untuk meningkatkan keterampilan visi komputer Anda menggunakan Python.

Latihan 1: Membaca dan Menampilkan Gambar

Tujuan dari latihan ini adalah untuk memahami cara membaca dan menampilkan gambar menggunakan Python dan OpenCV, sebuah pustaka yang sering digunakan dalam visi komputer.

```
import cv2

# Membaca gambar
gambar = cv2.imread('namofile.jpg')

# Menampilkan gambar
cv2.imshow('Gambar', gambar)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Berikut adalah fungsi dari setiap baris kode:

- import cv2: Ini adalah perintah untuk mengimpor library OpenCV (cv2) ke dalam program Python. OpenCV adalah library populer yang digunakan untuk memanipulasi gambar dan video.
- gambar = cv2.imread('namafile.jpg'): Baris ini membaca gambar dengan nama file ‘namafile.jpg’ menggunakan fungsi imread() dari OpenCV. Fungsi ini mengembalikan matriks NumPy yang mewakili gambar.
- cv2.imshow('Gambar', gambar): Ini adalah perintah untuk menampilkan gambar ke jendela dengan judul ‘Gambar’. Fungsi imshow() dari OpenCV digunakan untuk menampilkan gambar dalam jendela.
- cv2.waitKey(0): Baris ini menunggu pengguna menekan tombol apa pun untuk melanjutkan eksekusi program. Nilai argumen 0 menunjukkan bahwa program akan tetap berjalan sampai tombol ditekan.
- cv2.destroyAllWindows(): Ini adalah perintah untuk menutup semua jendela yang dibuka oleh program. Fungsi destroyAllWindows() digunakan untuk membersihkan semua jendela tampilan.

Latihan 2: Mengubah Gambar ke Grayscale

Banyak operasi dalam visi komputer dijalankan pada gambar grayscale. Latihan ini bertujuan untuk mengubah gambar berwarna menjadi grayscale.

```
import cv2

# Membaca gambar
gambar = cv2.imread('namafile.jpg')

# Mengubah gambar ke grayscale
gray = cv2.cvtColor(gambar, cv2.COLOR_BGR2GRAY)

# Menampilkan gambar grayscale
cv2.imshow('Gambar Grayscale', gray)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Berikut adalah fungsi dari setiap baris kode:

- gray = cv2.cvtColor(gambar, cv2.COLOR_BGR2GRAY): Baris ini mengubah gambar dari warna (BGR) ke skala keabuan (grayscale) menggunakan fungsi cvtColor() dari OpenCV. Fungsi ini menerima dua argumen, yaitu gambar yang ingin diubah (variabel gambar) dan konversi warna yang ingin dilakukan (dalam hal ini, dari BGR ke grayscale). Hasil konversi disimpan dalam variabel gray.

Latihan 3: Deteksi Tepi

Deteksi tepi adalah teknik penting dalam visi komputer. Latihan ini bertujuan untuk menerapkan deteksi tepi pada gambar.

```
import cv2
import numpy as np

# Membaca gambar
gambar = cv2.imread('namafile.jpg')

# Mengubah gambar ke grayscale
gray = cv2.cvtColor(gambar, cv2.COLOR_BGR2GRAY)

# Mengaplikasikan deteksi tepi
edges = cv2.Canny(gray, 30, 100)

# Menampilkan gambar dengan tepi yang terdeteksi
cv2.imshow('Deteksi Tepi', edges)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Berikut adalah fungsi dari setiap baris kode:

- edges = cv2.Canny(gray, 30, 100): Baris ini menerapkan deteksi tepi pada gambar skala keabuan (grayscale) menggunakan metode Canny dengan parameter threshold lower dan upper sebesar 30 dan 100. Fungsi Canny() dari OpenCV menghasilkan gambar dengan tepi yang terdeteksi, yang disimpan dalam variabel edges.

Latihan 4: Deteksi Wajah

Pustaka OpenCV menyediakan pretrained cascade classifiers yang dapat digunakan untuk deteksi wajah dan fitur wajah lainnya.

```
import cv2

# Membaca gambar
gambar = cv2.imread('namafile.jpg')

# Mengubah gambar ke grayscale
gray = cv2.cvtColor(gambar, cv2.COLOR_BGR2GRAY)

# Membuat objek face cascade
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_defa
```

```

# Mendeteksi wajah
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

# Menggambar kotak pada setiap wajah yang terdeteksi
for (x, y, w, h) in faces:
    cv2.rectangle(gambar, (x, y), (x+w, y+h), (0, 255, 0), 2)

# Menampilkan gambar dengan wajah yang terdeteksi
cv2.imshow('Deteksi Wajah', gambar)
cv2.waitKey(0)
cv2.destroyAllWindows()

```

Berikut adalah fungsi dari setiap baris kode:

- `face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")`: Baris ini membuat objek cascade classifier untuk mendeteksi wajah. Cascade classifier adalah algoritma yang digunakan untuk mendeteksi objek dalam gambar berdasarkan fitur-fitur yang telah ditraining sebelumnya. Dalam hal ini, digunakan cascade classifier untuk mendeteksi wajah dengan menggunakan file XML yang disebut "haarcascade_frontalface_default.xml". File XML ini berisi informasi tentang fitur-fitur yang relevan untuk mendeteksi wajah.
- `faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))`: Baris ini mendeteksi wajah dalam gambar menggunakan metode `detectMultiScale()` dari cascade classifier. Metode ini menerima gambar skala keabuan (`gray`) sebagai input dan mengembalikan array yang berisi koordinat wajah yang terdeteksi. Parameter-parameter yang digunakan adalah `scaleFactor` (faktor skala untuk deteksi multi-skala), `minNeighbors` (jumlah minimum tetangga yang harus ada agar wajah dianggap valid), dan `minSize` (ukuran minimum wajah yang diterima).
- `for (x, y, w, h) in faces: cv2.rectangle(gambar, (x, y), (x+w, y+h), (0, 255, 0), 2)`: Baris ini menggunakan loop untuk menggambar kotak pada setiap wajah yang terdeteksi. Koordinat dan ukuran wajah yang terdeteksi (`x, y, w, h`) diperoleh dari array wajah yang ditemukan sebelumnya. Kotak tersebut digambar menggunakan fungsi `rectangle()` dari OpenCV pada gambar asli (variabel `gambar`).

Harap diingat bahwa setiap latihan ini hanyalah awal. Visi komputer adalah bidang yang sangat luas dengan banyak teknik dan algoritma yang berbeda. Untuk benar-benar mahir, Anda perlu memahami teori di balik teknik ini dan bagaimana menerapkannya dalam situasi nyata.

E. Sesi Tanya Jawab dan Diskusi

Q: Apa itu computer vision dan mengapa itu penting?

A: Computer vision adalah bidang teknologi yang memungkinkan komputer dan mesin untuk ‘melihat’ dan memahami konten visual, seperti gambar dan video. Computer vision penting karena memungkinkan automasi dan analisis tingkat lanjut dalam berbagai bidang, seperti keamanan, kesehatan, manufaktur, dan banyak lagi.

Q: Mengapa Python sering digunakan dalam computer vision?

A: Python sering digunakan dalam computer vision karena mudah dipelajari, memiliki sintaks yang jelas dan bersih, dan didukung oleh banyak library dan framework yang kuat seperti OpenCV, TensorFlow, dan PyTorch. Python juga open-source, yang berarti kode sumbernya tersedia secara bebas dan dapat dimodifikasi atau diperluas oleh komunitas.

Q: Bagaimana saya bisa belajar lebih banyak tentang computer vision dan Python?

A: Anda bisa memulai dengan belajar dasar-dasar Python dan lalu belajar tentang perpustakaan seperti OpenCV, TensorFlow, dan PyTorch. Anda juga bisa mengikuti and 1 attachments

Q: Saya baru belajar Python, apakah saya bisa belajar Computer Vision?

A: Ya, Anda bisa belajar Computer Vision meski baru belajar Python. Sebenarnya, Python adalah bahasa pemrograman yang baik untuk dipelajari jika Anda tertarik dengan Computer Vision karena memiliki banyak library dan framework, seperti OpenCV, TensorFlow, dan PyTorch, yang dirancang khusus untuk visi komputer dan pembelajaran mesin.

Q: Apa yang dimaksud dengan Object Detection dalam Computer Vision?

A: Object Detection adalah teknologi dalam visi komputer yang mengidentifikasi atau mendeteksi objek dari berbagai kelas (seperti manusia, kendaraan, atau hewan) dalam gambar atau video. Teknologi ini biasanya digunakan dalam aplikasi seperti pengawasan video, sistem navigasi untuk kendaraan otonom, dan banyak lagi.

Q: Apakah memungkinkan untuk melakukan Computer Vision tanpa Machine Learning?

A: Ya, memang memungkinkan untuk melakukan tugas-tugas visi komputer tertentu tanpa menggunakan Machine Learning. Misalnya, teknik seperti pengolahan gambar, deteksi tepi, dan thresholding bisa digunakan untuk ekstraksi fitur dan pemrosesan gambar dasar. Namun, untuk tugas yang lebih kompleks seperti deteksi objek, pengenalan wajah, dan analisis video, biasanya diperlukan teknik Machine Learning atau Deep Learning.

Modul 2. Python

A. Dasar-dasar Python

Apa itu Python?

Python adalah bahasa pemrograman tingkat tinggi yang populer. Bahasa ini dapat menangani berbagai tugas pemrograman seperti komputasi numerik, pengembangan web, pemrograman basis data, pemrograman jaringan, pemrosesan paralel, dan lainnya.

Python populer karena berbagai alasan, termasuk:

- Bahasa ini gratis.
- Tersedia di semua sistem operasi populer seperti Windows, Mac, atau Linux.
- Python adalah bahasa yang diinterpretasikan. Oleh karena itu, pemrogram dapat menguji bagian kode di baris perintah sebelum menggabungkannya ke dalam program mereka. Tidak ada kebutuhan untuk kompilasi atau penghubungan. Python memungkinkan pemrograman yang lebih cepat.
- Python lebih sederhana secara sintaksis dibandingkan dengan C/C++/Fortran. Oleh karena itu, Python sangat mudah dibaca dan lebih mudah untuk debug.
- Python datang dengan berbagai modul yang standar atau dapat diinstal dalam instalasi Python yang ada. Modul-modul ini dapat melakukan berbagai tugas seperti membaca dan menulis berbagai file, komputasi ilmiah, visualisasi data, dan lainnya.
- Program yang ditulis dalam Python dapat dijalankan di berbagai sistem operasi atau platform dengan sedikit atau tanpa perubahan.
- Python adalah bahasa yang dinamis dalam pengetikannya. Oleh karena itu, tipe data dari variabel tidak harus dinyatakan sebelum penggunaannya, membuatnya lebih mudah untuk orang dengan pengalaman coding yang kurang.
- Python memiliki komunitas pengembang dan pengguna yang berdedikasi dan selalu diperbarui.

Meskipun Python memiliki banyak keunggulan yang membuatnya menjadi salah satu bahasa yang diinterpretasikan paling populer, Python memiliki beberapa kelemahan yang dibahas di bawah ini:

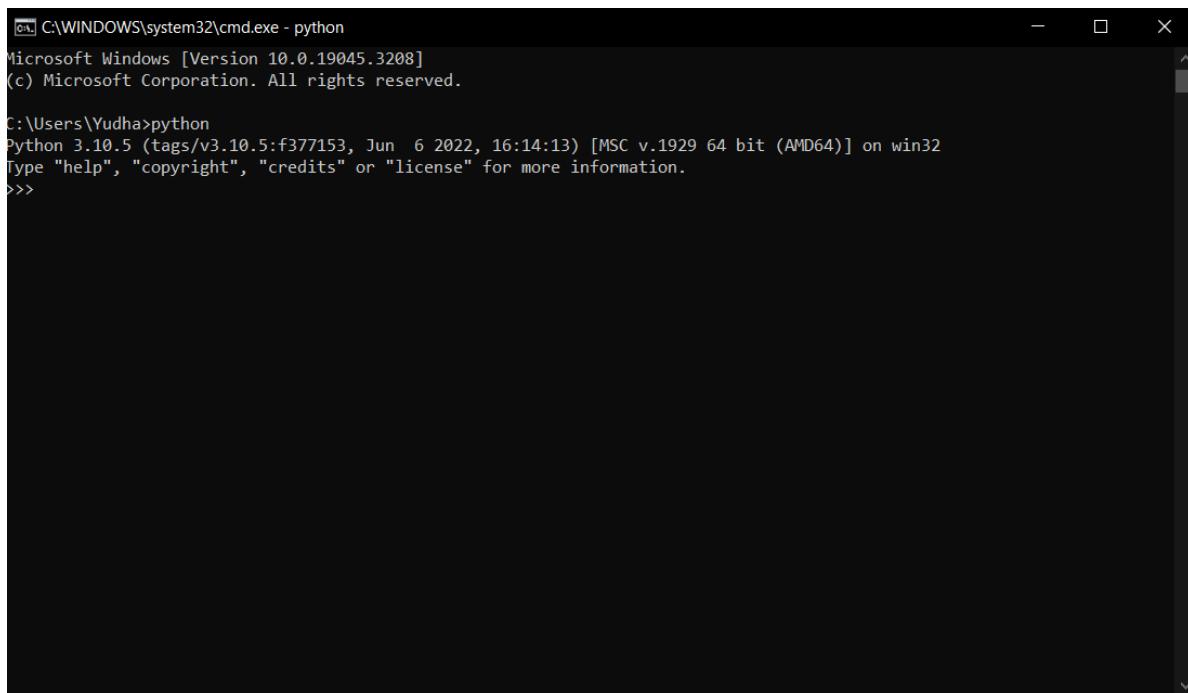
- Karena fokus Python adalah pada kemampuan untuk pemrograman yang lebih cepat, kecepatan eksekusi menderita. Program Python mungkin 10 kali atau lebih lambat (misalnya) dibandingkan dengan program C yang setara, tetapi program Python akan berisi lebih sedikit baris kode dan dapat diprogram untuk menangani berbagai jenis data dengan mudah. Kelemahan ini dalam kode Python dapat diatasi dengan mengubah bagian kode yang intensif secara komputasi ke C/C++ atau dengan penggunaan struktur data dan modul yang tepat.
- Indentasi kode tidak opsional. Ini membuat kode mudah dibaca. Namun, kode dengan loop dan konstruk lainnya akan diindentasi ke kanan, membuatnya sulit untuk membaca kode.

Lingkungan Python

Terdapat beberapa lingkungan Python yang dapat dipilih. Beberapa sistem operasi seperti Mac, Linux, Unix, dan lainnya memiliki interpreter bawaan. Interpreter tersebut mungkin mengandung semua modul tetapi tidak siap pakai untuk komputasi ilmiah. Distribusi khusus telah dibuat dan dijual kepada komunitas ilmiah, dibangun sebelumnya dengan berbagai modul ilmiah Python. Saat menggunakan distribusi ini, pengguna tidak perlu menginstal modul ilmiah secara individual. Jika modul tertentu yang diminati tidak tersedia dalam distribusi, modul tersebut dapat diinstal. Salah satu distribusi paling populer adalah Anaconda. Instruksi untuk menginstal distribusi Anaconda dapat ditemukan di www.anaconda.com

Interpreter Python

Interpreter Python yang terintegrasi dalam sebagian besar sistem operasi dapat dimulai dengan hanya mengetik python di jendela terminal. Ketika interpreter dimulai, prompt perintah (»>) muncul. Perintah Python dapat dimasukkan di prompt untuk diproses. Misalnya, di Windows, ketika interpreter Python bawaan dimulai, output yang mirip dengan yang ditunjukkan di bawah ini muncul:

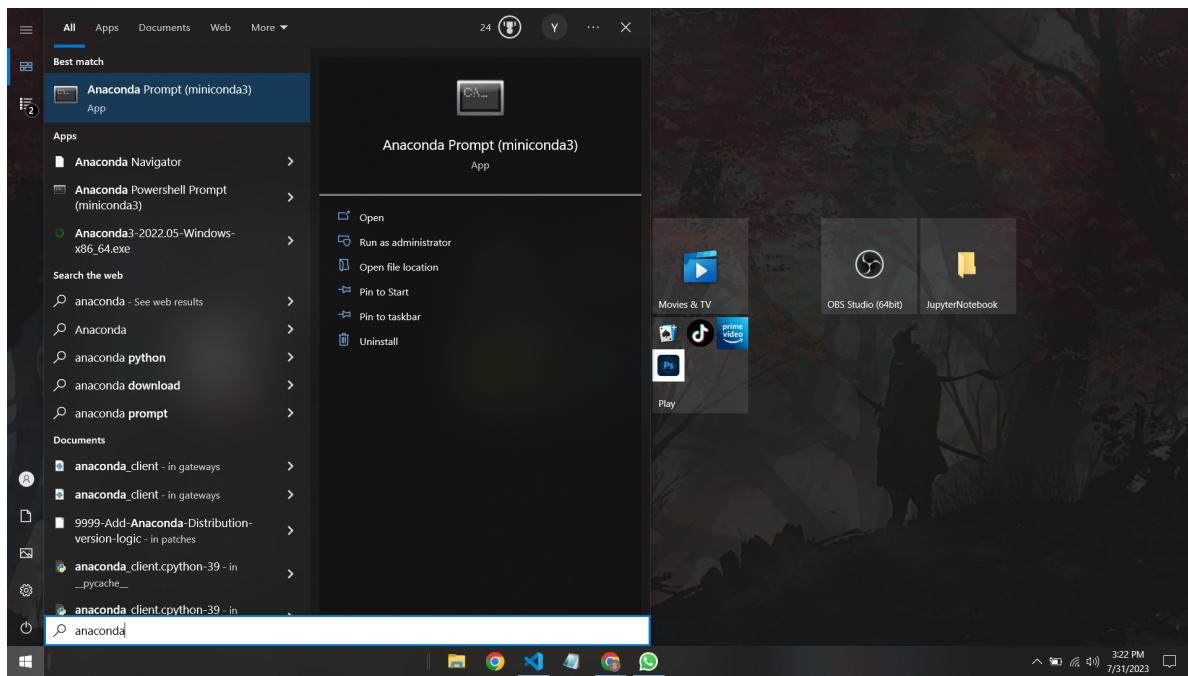


C:\WINDOWS\system32\cmd.exe - python
Microsoft Windows [Version 10.0.19045.3208]
(c) Microsoft Corporation. All rights reserved.
C:\Users\Yudha>python
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>

Perhatikan bahwa dalam contoh di atas, interpreter Python adalah versi 3.10.5. Kemungkinan Anda mungkin memiliki versi yang berbeda.

Distribusi Python

Distribusi Python Anaconda menyediakan hampir 100 modul Python ilmiah paling populer seperti perhitungan ilmiah, aljabar linear, komputasi simbolik, pemrosesan gambar, pemrosesan sinyal, visualisasi, integrasi program C/C++ ke Python, dll. Ini didistribusikan dan dikelola oleh Continuum Analytics. Ini tersedia secara gratis untuk akademisi dan tersedia dengan harga untuk semua orang lainnya. Selain berbagai modul yang dibangun ke dalam Anaconda, pemrogram dapat menginstal modul lain menggunakan manajer paket conda [Ana20b], tanpa mempengaruhi distribusi utama. Untuk mengakses Python dari baris perintah, mulailah ‘Anaconda Prompt’ yang dapat dieksekusi



dan kemudian ketik python.

A screenshot of the Anaconda Prompt window titled "Anaconda Prompt (miniconda3) - python". The window shows the Python command-line interface (CLI) running. The output in the terminal window is:

```
(base) C:\Users\Yudha>python
Python 3.9.12 (main, Apr  4 2022, 05:22:27) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

The window has a standard Windows-style title bar and a scroll bar on the right side.

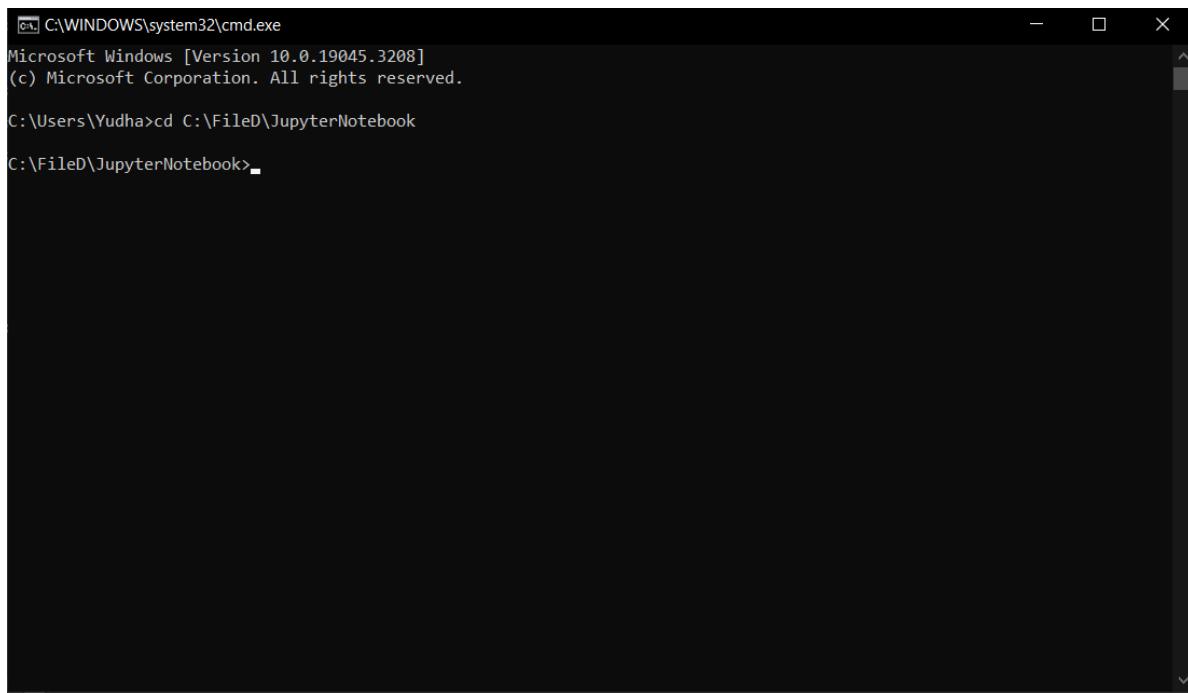
Menjalankan Program Python

Menggunakan interpreter Python apa pun (bawaan atau dari distribusi), Anda dapat menjalankan program Anda menggunakan perintah di sistem operasi (OS) prompt perintah. Jika file firstprog.py adalah file Python yang perlu dieksekusi, kemudian ketik perintah berikut ini di OS prompt perintah.

```
python latihanPython.py
```

Simbol » adalah prompt terminal dan »> mewakili prompt Python. Pendekatan terbaik untuk menjalankan program Python di bawah sistem operasi apa pun adalah dengan menggunakan Lingkungan Pengembangan Terpadu seperti IDLE atau Spyder karena memberikan kemampuan untuk mengedit file dan juga menjalankannya di bawah antarmuka yang sama.

Buka CMD lalu pindah Directiory menggunakan command cd

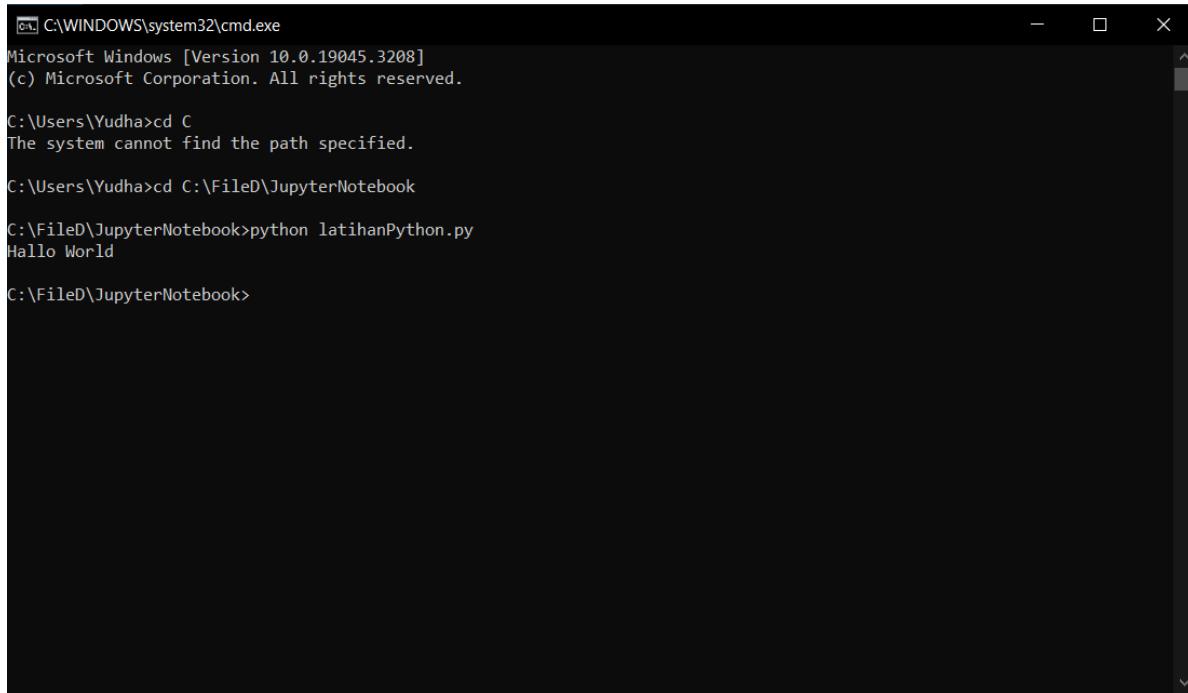


The screenshot shows a Windows Command Prompt window titled 'cmd' with the path 'C:\WINDOWS\system32\cmd.exe'. The window displays the following text:

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.3208]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Yudha>cd C:\FileD\JupyterNotebook
C:\FileD\JupyterNotebook>
```

Jalankan file dengan ekstensi .py



C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.3208]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Yudha>cd C
The system cannot find the path specified.

C:\Users\Yudha>cd C:\FileD\JupyterNotebook

C:\FileD\JupyterNotebook>python latihanPython.py
Hello World

C:\FileD\JupyterNotebook>

Pernyataan Dasar Python dan Jenis Data

Indentasi

Dalam Python, blok kode ditunjukkan dengan indentasi. Misalnya dalam kode di bawah ini, kita pertama-tama mencetak pesan, 'Kami sedang menghitung kuadrat dari angka antara 0 dan 9'. Kemudian kita melakukan loop melalui nilai-nilai dalam rentang 0 hingga 9 dan menyimpannya dalam variabel 'i' dan juga mencetak kuadrat dari 'i'. Akhirnya kita mencetak pesan, 'Kami menyelesaikan tugas di akhir'.

Dalam bahasa lain, blok kode di bawah for-loop akan diidentifikasi dengan pasangan kurung kurawal {}. Namun, dalam Python kita tidak menggunakan kurung kurawal. Blok kode diidentifikasi dengan menggeser baris `print(i*i)` empat spasi ke kanan. Anda juga bisa memilih untuk menggunakan tab sebagai gantinya.

```
print('Menghitung kuadrat dari angka antara 0 dan 9')  
for i in range(10):  
    print(i*i)  
print('Menyelesaikan tugas ...')
```

Ada kelemahan signifikan dalam indentasi, terutama bagi pemrogram Python baru. Sebuah kode yang berisi beberapa for-loop dan if-statements akan diindentasi lebih jauh ke kanan membuat kode tidak dapat dibaca. Masalah ini dapat diredukan dengan mengurangi jumlah for-loop dan if-statements. Ini tidak hanya membuat kode dapat dibaca tetapi juga mengurangi

waktu komputasi. Ini dapat dicapai dengan pemrograman menggunakan struktur data seperti daftar, kamus, dan set secara tepat.

Komentar

Komentar adalah bagian penting dari setiap bahasa pemrograman. Dalam Python, komentar baris tunggal ditandai dengan hash # di awal baris. Beberapa baris dapat dikomentari dengan menggunakan string tanda kutip tiga (tanda kutip tunggal tiga kali atau tanda kutip ganda tiga kali) di awal dan di akhir blok.

```
# Ini adalah komentar baris tunggal
"""
Ini adalah komentar multiline
"""
# Komentar adalah cara yang baik untuk menjelaskan kode.
```

Variabel

Python adalah bahasa dinamis dan oleh karena itu Anda tidak perlu menentukan jenis variabel seperti dalam C/C++. Variabel bisa dianggap sebagai wadah nilai. Nilai tersebut bisa berupa bilangan bulat, float, string, daftar, tuple, kamus, set, dll.

```
a = 1
a = 10.0
a = 'hello'
```

Dalam contoh di atas nilai bilangan bulat 1, nilai float 10.0, dan nilai string hello untuk semua kasus disimpan dalam variabel yang sama. Namun, hanya nilai yang terakhir ditugaskan yang merupakan nilai saat ini untuk a.

Operator

Python mendukung semua operator aritmatika umum seperti +, —, *, /. Juga mendukung operator perbandingan umum seperti >, <, ==, !=, >=, <=, dll. Selain itu, melalui berbagai modul Python menyediakan banyak operator untuk melakukan operasi trigonometri, matematika, geometri, dll.

Loop

Konstruksi looping yang paling umum dalam Python adalah pernyataan for-loop, yang memungkinkan iterasi melalui kumpulan objek. Berikut ini adalah contohnya:

```
for i in range(1,5):
    print(i)
```

Dalam contoh di atas output dari for-loop adalah angka dari 1 hingga 5. Fungsi range memungkinkan kita untuk membuat nilai mulai dari 1 dan berakhir dengan 5. Konsep semacam ini

mirip dengan for-loop yang biasanya ditemukan dalam C/C++ atau sebagian besar bahasa pemrograman.

Kekuatan sebenarnya dari for-loop terletak pada kemampuannya untuk melakukan iterasi melalui objek Python lainnya seperti daftar, kamus, set, string, dll. Kami akan membahas objek Python ini secara lebih detail nanti.

```
a = ['python', 'scipy']
for i in a:
    print(i)
```

Dalam program di atas, for-loop melakukan iterasi melalui setiap elemen dari daftar dan mencetaknya.

Dalam program berikutnya, isi dari kamus dicetak menggunakan for-loop. Kamus dengan dua kunci lang dan ver didefinisikan. Kemudian, menggunakan for-loop, berbagai kunci diiterasi dan nilai yang sesuai dicetak.

```
a = {
'lang':'python',
'ver': '3.11.3'
}
for key in a:
    print(a[key])
```

Diskusi tentang penggunaan for-loop untuk melakukan iterasi melalui berbagai baris dalam file teks, seperti file nilai yang dipisahkan koma, ditunda hingga bagian berikutnya.

Pernyataan if-else

If-else adalah pernyataan kondisional yang populer dalam semua bahasa pemrograman termasuk Python. Pernyataan if-else tidak harus menggunakan operator kondisional seperti <, >, ==, dll. Contoh pernyataan if-elif-else ditunjukkan di bawah ini.

```
if a<10:
    print('a kurang dari 10')
elif a<20:
    print('a antara 10 dan 20')
else:
    print('a lebih dari 20')
```

Misalnya, pernyataan if berikut ini legal dalam Python. Pernyataan if ini memeriksa kondisi bahwa daftar d tidak kosong.

```

d = [ ]
if d:
    print('d tidak kosong')
else:
    print('d kosong')

```

Dalam kode di atas, karena d kosong, klausa else benar dan kita memasuki blok else dan mencetak d kosong.

Struktur Data

Kekuatan nyata Python terletak pada penggunaan liberal struktur datanya. Kritik umum terhadap Python adalah bahwa itu lambat dibandingkan dengan C/C++. Hal ini terutama benar jika for-loop digunakan dalam pemrograman Python. Ini dapat direduksi dengan penggunaan yang tepat dari struktur data seperti daftar, tuple, kamus dan set. Kami mendeskripsikan masing-masing struktur data ini dalam bagian ini.

Daftar (lists)

Daftar mirip dengan array di C/C++. Tetapi, tidak seperti array di C/C++, daftar dalam Python dapat menampung objek dari jenis apa pun seperti int, float, string dan termasuk daftar lainnya. Daftar dapat diubah ukurannya, karena ukurannya dapat diubah dengan menambahkan atau menghapus elemen. Contoh berikut akan membantu menunjukkan kekuatan dan fleksibilitas daftar.

```

a = ['python', 'scipy', 3.6]
a.pop(-1)
print(a)
# Output: ['python', 'scipy']

a.append('numpy')
print(a)
# Output: ['python', 'scipy', 'numpy']

print(a[0])
# Output: python

print(a[-1])
# Output: numpy

print(a[0:2])
# Output: ['python', 'scipy']

```

Di baris pertama, daftar baru dibuat. Daftar ini berisi dua string dan satu angka float. Di baris kedua, kita menggunakan fungsi pop untuk menghapus elemen terakhir (indeks = —1). Elemen yang di-pop dicetak ke terminal. Setelah pop, daftar hanya berisi dua elemen daripada tiga asli. Kami menggunakan append, dan memasukkan elemen baru, “numpy” ke akhir daftar. Akhirnya, dalam dua perintah berikutnya kita mencetak nilai daftar di indeks 0 dan posisi terakhir yang ditunjukkan dengan menggunakan “—1” sebagai indeks. Dalam perintah terakhir, kami memperkenalkan slicing dan mendapatkan daftar baru yang hanya berisi dua nilai pertama dari daftar. Hal ini menunjukkan bahwa kita dapat mengoperasikan daftar menggunakan metode seperti pop, insert, atau remove dan juga menggunakan operator seperti slicing.

Daftar dapat berisi daftar lain. Berikut adalah contohnya. Kami akan mempertimbangkan kasus daftar yang berisi empat angka dan diatur untuk terlihat seperti matriks.

```
a = [[1,2] , [3,4]]  
print(a[0])  
# Output: [1,2]  
  
print(a[1])  
# Output: [3,4]  
  
print(a[0][0])  
# Output: 1
```

Di baris 1, kami mendefinisikan daftar dari daftar. Nilai [1,2] ada dalam daftar pertama dan nilai [3,4] ada dalam daftar kedua. Kedua daftar dikombinasikan untuk membentuk daftar 2D. Di baris kedua, kami mencetak nilai elemen pertama dari daftar. Perhatikan bahwa ini mencetak baris pertama atau daftar pertama dan bukan hanya sel pertama. Di baris keempat, kami mencetak nilai baris kedua atau daftar kedua. Untuk mendapatkan nilai elemen pertama dalam daftar pertama, kita perlu mengindeks daftar seperti yang diberikan pada baris 6. Seperti yang Anda lihat, pengindeksan berbagai elemen dari daftar seolah-olah memanggil lokasi elemen dalam daftar.

Meskipun elemen daftar dapat dioperasikan secara individual, kekuatan Python terletak pada kemampuannya untuk mengoperasikan seluruh daftar sekaligus menggunakan metode daftar dan pemahaman daftar.

Fungsi/Metode Daftar

Mari kita pertimbangkan daftar yang kami buat di bagian sebelumnya. Kami dapat mengurutkan daftar menggunakan metode sort seperti yang ditunjukkan di baris 2. Metode sort tidak mengembalikan daftar; sebaliknya, itu memodifikasi daftar saat ini. Oleh karena itu daftar yang ada akan berisi elemen dalam urutan yang diurutkan. Anda dapat melihat bahwa dalam kode berikut.

```
a = ['python', 'numpy', 'scipy']
a.sort()
print(a)
# Output: ['numpy', 'python', 'scipy']
```

Dalam kode di atas, metode sort adalah cara mengurutkan daftar. Karena metode ini adalah metode inplace, daftar yang ada diubah, dan tidak ada nilai yang dikembalikan. Jadi, setelah perintah sort, a diubah menjadi daftar urutan.

Pemahaman daftar(List Comprehention)

Pemahaman daftar adalah fitur Python yang sangat kuat dan merupakan cara yang efisien untuk mengoperasikan daftar. Anda dapat membuat daftar baru dari daftar yang ada dengan pemahaman daftar. Sebagai contoh, mari kita ambil daftar dan kita ingin menghasilkan daftar baru yang mengandung semua angka dari daftar asli yang lebih besar dari 5. Dalam bahasa pemrograman lainnya kita akan menggunakan for-loop dan memeriksa setiap elemen satu per satu untuk melihat apakah itu lebih besar dari 5. Namun, dalam Python kita bisa menggunakan pemahaman daftar dan mendapatkan daftar baru dalam satu baris kode.

```
a = [1, 2, 3, 4, 5, 6, 7, 8, 9]
b = [i for i in a if i > 5]
print(b)
# Output: [6, 7, 8, 9]
```

Dalam kode di atas, variabel b merupakan daftar baru yang dibuat dari daftar a. Nilai i dalam daftar a ditambahkan ke b hanya jika i lebih besar dari 5. Jadi, Python adalah bahasa yang sangat kuat dan memiliki banyak fitur yang memungkinkan pengguna untuk menulis kode yang ringkas dan efisien. Python memudahkan pembacaan dan pemahaman kode dengan perintah sederhana dan jelas yang menggunakan sintaks alami. Struktur data Python, seperti daftar, memungkinkan manipulasi data yang efisien dan fleksibel.

Tuples

Tuples sangat mirip dengan list kecuali bahwa mereka tidak dapat diubah, yaitu, panjang dan isi tuple tidak dapat diubah saat runtime. Secara sintaksis, list menggunakan [] sedangkan tuples menggunakan (). Sama seperti list, tuple mungkin berisi jenis data apa pun termasuk tuple lain. Berikut adalah beberapa contoh:

```
a = (1,2,3,4)
print(a)
# Output: (1,2,3,4)

b = (3,)
c = ((1,2),(3,4))
```

Sets

Set adalah kumpulan objek unik yang tidak berurutan. Untuk membuat set, kita perlu menggunakan fungsi set atau operator {} . Berikut beberapa contohnya:

```
s1 = set([1,2,3,4])
s2 = set((1,1,3,4))
print(s2)
# Output: {1,3,4}
```

Dictionaries

Dictionaries menyimpan pasangan kunci-nilai. Sebuah kamus dibuat dengan mengapit pasangan kunci-nilai di dalam {} .

```
a = {
    'lang' : 'python',
    'ver': '3.11.3'
}
```

Penanganan File

Python menyediakan kemampuan untuk membaca dan menulis file. Ia juga memiliki fungsi, metode, dan modul untuk membaca format khusus seperti file nilai yang dipisahkan dengan koma (csv), format Microsoft Excel (xls), dll. Kami akan melihat setiap metode dalam bagian ini.

Membaca file CSV

Berikut adalah kode yang membaca file csv sebagai file teks.

```
fo = open('myfile.csv')
for i in fo.readlines():
    print(i)
fo.close()
```

Sebagai alternatif dari membaca file csv sebagai file teks, kita dapat menggunakan modul csv.

```
import csv
for i in csv.reader(open('myfile.csv')):
    print(i)
```

Membaca file Excel File Microsoft Excel dapat dibaca dan ditulis menggunakan modul openpyxl.

```
from openpyxl import load_workbook
wb = load_workbook('myfile.xlsx')
for sheet in wb:
    for row in sheet.values:
        for col in row:
            print(col, end=' | ')
print()
```

Fungsi yang Ditentukan Pengguna

Fungsi adalah bagian kode yang dapat digunakan kembali yang mungkin mengambil input dan mungkin atau tidak mengembalikan output. Berikut adalah contoh:

```
import math
def circleproperties(r):
    area = math.pi*r*r
    circumference = 2*math.pi*r
    return area, circumference

a, c = circleproperties(5) # Radius of the circle is 5
print('Area and Circumference of the circle are', a, c)
```

Fungsi circleproperties menerima satu argumen input, radius (r). Pernyataan return pada akhir definisi fungsi mengembalikan nilai yang dihitung (dalam hal ini, area dan keliling) ke fungsi pemanggil. Untuk memanggil fungsi, gunakan nama fungsi dan berikan nilai radius sebagai argumen yang dibungkus dalam tanda kurung. Akhirnya, area dan keliling lingkaran ditampilkan menggunakan panggilan fungsi print.

B. Komputasi Menggunakan Module Python

Diketahui bahwa Python dilengkapi dengan berbagai modul bawaan. Modul-modul ini melakukan berbagai operasi khusus, mulai dari komputasi, manajemen database, hingga fungsi server web. Mengingat fokus buku ini adalah pembuatan aplikasi ilmiah, pembahasan dibatasi pada modul Python yang memungkinkan komputasi seperti scipy, numpy, matplotlib, Python Imaging Library (PIL), dan paket scikit. Relevansi masing-masing modul ini dijelaskan dan ditunjukkan penggunaannya dengan contoh. Pembahasan juga mencakup pembuatan modul Python baru.

Modul Python

Ada sejumlah modul Python ilmiah yang telah dibuat dan tersedia dalam distribusi Python. Beberapa modul paling populer yang relevan adalah:

- Numpy: Sebuah perpustakaan yang kuat untuk manipulasi array dan matriks.
- Scipy: Menyediakan fungsi untuk melakukan operasi matematika tingkat tinggi seperti filtering, analisis statistik, pemrosesan gambar, dll.
- Matplotlib: Menyediakan fungsi untuk plotting dan bentuk visualisasi lainnya.
- Python Imaging Library: Menyediakan fungsi untuk pembacaan gambar dasar, penulisan dan pemrosesan.
- Scikits: Sebuah paket tambahan untuk scipy. Modul dalam scikit dimaksudkan untuk ditambahkan ke scipy setelah pengembangan.

Membuat Modul

Modul adalah file Python yang berisi beberapa fungsi atau kelas dan komponen opsional lainnya. Semua fungsi dan kelas ini berbagi namespace yang sama, yaitu, nama file modul. Sebagai contoh, program berikut adalah modul Python yang valid.

```
# nama file: examplemodules.py
version = '1.0'
def printpi():
    print('Nilai pi adalah 3.1415')
```

Sebuah fungsi bernama ‘printpi’ dan variabel yang disebut ‘version’ dibuat dalam modul ini. Fungsi ini melakukan operasi sederhana untuk mencetak nilai pi.

Memuat Modul

Untuk memuat modul ini, gunakan perintah berikut di baris perintah Python atau dalam program Python. Kata “examplemodules” adalah nama file modul.

```
import examplemodules
```

Setelah modul dimuat, fungsi dapat dijalankan menggunakan perintah di bawah. Perintah pertama mencetak nilai pi bersama dengan label, sementara perintah kedua mencetak nomor versi.

```
examplemodules.printpi()
# Nilai pi adalah 3.1415
```

```
examplemodules.version  
# 1.0
```

Modul contoh yang ditunjukkan di atas hanya memiliki satu fungsi. Sebuah modul mungkin berisi beberapa fungsi atau kelas. Dalam contoh pertama, modul datetime dimuat. Namun dalam contoh ini, hanya tertarik untuk mendapatkan tanggal saat ini menggunakan date.today().

```
import datetime  
print(datetime.date.today()) # 2023-07-31
```

Dalam contoh kedua, hanya fungsi yang diperlukan (date) dalam modul datetime yang dimuat. Untuk modul besar, disarankan untuk mengimpor hanya fungsi yang diperlukan agar kode lebih mudah dibaca.

```
from datetime import date  
print (date.today()) # 2023-07-31
```

Dalam contoh ketiga, mengimpor semua fungsi dalam modul yang diberikan menggunakan *. Setelah diimpor, nama file (dalam hal ini “date”) yang berisi fungsi (dalam hal ini “today()”) perlu ditentukan. Metode impor ini biasanya tidak disarankan, karena dapat menghasilkan tabrakan namespace. Misalnya, menjadi ambigu jika fungsi date ada di modul datetime atau dari pernyataan impor lainnya.

```
from datetime import *\nprint(date.today()) # 2023-07-31
```

Dalam contoh keempat, mengimpor modul (dalam hal ini numpy) dan menggantinya dengan sesuatu yang lebih pendek seperti np. Ini dikenal sebagai aliasing. Ini akan mengurangi jumlah karakter yang perlu diketik dan akibatnya baris kode yang perlu dipertahankan.

```
import numpy as np >> np.ones( [3,3] )  
array([[ 1.,  1. ,  1.],  
       [ 1.,  1. ,  1.],  
       [ 1.,  1. ,  1.]])
```

NumPy

Modul numpy menambahkan kemampuan untuk memanipulasi array dan matriks menggunakan kumpulan fungsi matematika. Numpy berasal dari modul yang sudah tidak digunakan

lagi, yaitu Numeric dan Numarray. Numeric adalah upaya pertama untuk menyediakan kemampuan memanipulasi array, tetapi sangat lambat dalam melakukan komputasi pada array yang besar. Numarray, di sisi lain, terlalu lambat dalam mengolah array yang kecil. Kode dasar kedua modul ini digabungkan untuk membuat numpy. Numpy memiliki fungsi dan rutinitas untuk melakukan aljabar linear, pengambilan sampel acak, polinomial, fungsi keuangan, operasi himpunan, dan lain-lain. Karena buku ini berfokus pada pemrosesan gambar dan karena gambar merupakan array, kita akan menggunakan kemampuan manipulasi matriks numpy.

Array atau Matriks Numpy?

Numpy memanipulasi matriks matematika dan vektor, sehingga menghitung lebih cepat dari pada penggunaan loop tradisional yang memanipulasi skalar. Dalam numpy, terdapat dua jenis kelas matriks matematika: array dan matriks. Kedua kelas tersebut dirancang untuk tujuan serupa, tetapi array lebih umum dan memiliki dimensi n, sedangkan matriks memfasilitasi perhitungan aljabar linear yang lebih cepat. Beberapa perbedaan antara array dan matriks dijelaskan di bawah ini:

- Objek matriks memiliki rank 2, sedangkan array memiliki rank > 2 .
- Objek matriks dapat dikalikan menggunakan operator `*`, sedangkan operator yang sama pada array melakukan perkalian elemen-demi-elemen. Fungsi `dot()` harus digunakan untuk melakukan perkalian pada array.
- Array adalah tipe data default di numpy.
Array lebih sering digunakan dalam numpy dan modul-modul lain yang menggunakan numpy untuk komputasi mereka. Matriks dan array dapat saling dipertukarkan, tetapi disarankan untuk menggunakan array.

Instalasi NumPy

```
! pip install numpy
```

Menggunakan Library Numpy

```
import numpy as np
```

SciPy

Scipy adalah sebuah perpustakaan fungsi, program, dan alat matematika untuk pemrograman ilmiah dalam bahasa Python. Scipy menggunakan numpy untuk komputasi internalnya. Scipy adalah perpustakaan yang luas yang memungkinkan pemrograman berbagai aplikasi matematika seperti integrasi, optimisasi, transformasi Fourier, pemrosesan sinyal, statistik,

pemrosesan gambar multidimensi, dan lain-lain. Travis Oliphant, Eric Jones, dan Pearu Peterson menggabungkan modul-modul mereka untuk membentuk scipy pada tahun 2001. Sejak saat itu, banyak sukarelawan di seluruh dunia telah berpartisipasi dalam pemeliharaan scipy. Scipy memuat modul dapat memakan banyak sumber daya CPU dan memori. Hal ini terutama berlaku untuk paket-paket besar seperti scipy yang mengandung banyak submodul. Dalam kasus seperti itu, muat hanya submodul yang spesifik.

Installasi SciPy

```
! pip install scipy
```

Menggunakan Library Scipy

```
from scipy import ndimage  
import scipy.ndimage as im
```

Pada perintah pertama, hanya submodul ndimage yang dimuat. Pada perintah kedua, modul ndimage dimuat sebagai im.

Matplotlib

Matplotlib adalah perpustakaan plot 2D/3D untuk Python. Ia dirancang untuk menggunakan tipe data numpy. Matplotlib dapat digunakan untuk menghasilkan plot di dalam program Python. Contoh yang menunjukkan fitur-fitur matplotlib ditunjukkan dalam Gambar 2.1.

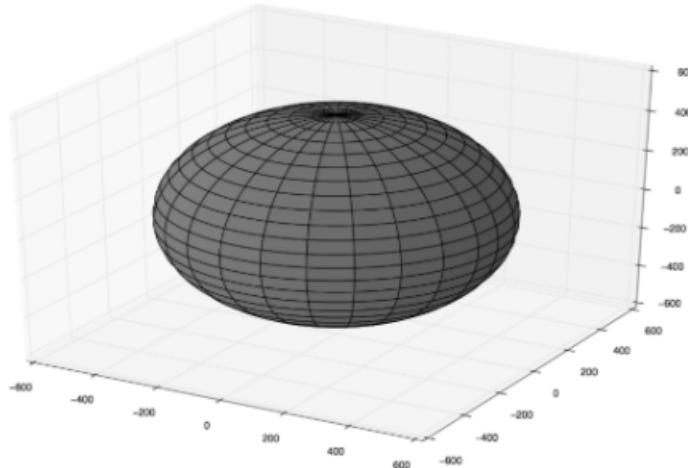


Figure 4: Gambar 2.1 Ilustrasi Matplotlib

Instalasi Matplotlib

```
! pip install matplotlib
```

Menggunakan Library Matplotlib

```
import matplotlib
```

Python Image Library (PIL)

Python Imaging Library (PIL) adalah modul untuk membaca, menulis, dan memproses file gambar. Modul ini mendukung sebagian besar format gambar umum seperti JPEG, PNG, TIFF, dll. **Namun, sejak versi Python 3.0, PIL tidak lagi dikembangkan dan digantikan oleh library yang disebut Pillow.**

Installasi Pillow

```
! pip install Pillow
```

Menggunakan Library Pillow

```
from PIL import image
```

Scikits

Scikits adalah singkatan dari scipy toolkits. Ini adalah paket tambahan yang dapat digunakan bersama dengan alat-alat scipy. Sebuah algoritma diprogram dalam scikits jika:

- Algoritma tersebut masih dalam pengembangan dan belum siap untuk digunakan secara luas dalam scipy.
- Paket tersebut memiliki lisensi yang tidak kompatibel dengan scipy.
- Scipy adalah paket ilmiah umum dalam bahasa Python. Oleh karena itu, dirancang agar dapat diterapkan dalam berbagai bidang. Jika sebuah paket dianggap khusus untuk bidang tertentu, maka paket tersebut tetap menjadi bagian dari scikits.

Scikits terdiri dari modul-modul dari berbagai bidang seperti ilmu lingkungan, analisis statistik, pemrosesan gambar, rekayasa mikrowave, pemrosesan audio, masalah nilai batas, penyesuaian kurva, komputasi kuantum, dll.

Dalam buku ini, kita akan fokus hanya pada rutinitas pemrosesan gambar dalam scikits yang disebut scikit-image. Rutinitas scikit-image ini berisi algoritma-algoritma untuk input/output, morfologi, deteksi dan analisis objek, dll.

Installasi Skimage

```
! pip install scikit-image
```

Penggunaan Library Skimage

```
from skimage import filters  
import skimage.filters as fi
```

Pada perintah pertama, hanya submodul filters yang dimuat. Pada perintah kedua, modul filters dimuat sebagai fi.

Modul Python OpenCV

Open Source Computer Vision Library (OpenCV) [Ope20a] adalah perpustakaan perangkat lunak untuk pemrosesan gambar, penglihatan komputer, dan pembelajaran mesin. Ini memiliki lebih dari 2000 algoritma untuk memproses data gambar. OpenCV memiliki basis pengguna yang besar dan digunakan secara luas di lembaga akademik, organisasi komersial, dan lembaga pemerintah. Perpustakaan ini menyediakan ikatan (binding) untuk bahasa pemrograman umum seperti C, C++, Python, dll.

Installasi Opencv

```
! pip install opencv-python
```

Menggunakan Library OpenCV

```
import cv2
```

Modul 3. Dasar-dasar Image Processing

Pada Modul ini membahas dasar-dasar image processing membahas konsep dasar dalam Image Processing, yang merupakan bagian penting dari Computer Vision. Anda akan mempelajari definisi Image Processing, aplikasinya dalam kehidupan sehari-hari, dan perbedaannya dengan Computer Vision. Selain itu, Anda akan memahami konsep dasar citra digital, termasuk operasi transformasi, filtering, dan edge detection. Penggunaan Python untuk Image Processing juga akan diajarkan, termasuk instalasi dan penggunaan library seperti OpenCV, TensorFlow, dan Keras. Modul ini juga mencakup latihan praktik untuk mengaplikasikan konsep dan teknik yang telah dipelajari. Dengan menyelesaikan Modul 2, Anda akan memiliki pemahaman yang kuat tentang dasar-dasar Image Processing dan keterampilan untuk mengimplementasikannya menggunakan Python.

A. Pengenalan Image Processing

Manusia mengandalkan penglihatan mereka untuk tugas-tugas mulai dari pengenalan pola hingga naluri bertahan hidup. Kemampuan manusia untuk melakukan analisis yang kompleks dan rinci terhadap suatu karya seni berdasarkan input visual adalah sesuatu yang luar biasa. Namun, sejauh mana manusia dapat melakukan apa yang komputer lakukan dengan sangat cepat masih perlu diteliti.

Kebutuhan untuk mengekstrak informasi dari citra dan menginterpretasikan isinya telah menjadi salah satu faktor pendorong dalam perkembangan pemrosesan citra dan visi komputer selama beberapa dekade terakhir. Aplikasi pemrosesan citra meliputi berbagai aktivitas manusia, antara lain:

- Aplikasi medis: Modalitas pencitraan diagnostik seperti radiografi digital, PET (tomografi emisi positron), CT (tomografi aksial komputer), MRI (pemindaian resonansi magnetik), dan fMRI (pemindaian resonansi magnetik fungsional) telah diadopsi secara luas oleh komunitas medis.
- Aplikasi industri: Sistem pemrosesan citra telah berhasil digunakan dalam sistem manufaktur untuk berbagai tugas, seperti sistem keamanan, kontrol kualitas, dan pengendalian kendaraan berpemandu otomatis (AGVs).

- Aplikasi militer: Skenario yang paling menantang dan kritis dalam hal kinerja pemrosesan citra adalah untuk mendukung tugas militer, mulai dari deteksi tentara atau kendaraan hingga panduan rudal dan pengenalan objek dan tugas pengintaian menggunakan kendaraan udara tak berawak atau UAV. Selain itu, aplikasi militer sering kali membutuhkan penggunaan sensor pendekripsi yang khusus, seperti kamera jarak dan kamera inframerah yang melihat ke depan.
- Penegakan hukum dan keamanan: Pengawasan adalah salah satu bidang yang banyak diteliti dalam komunitas pemrosesan video.
- Teknologi biometrik (seperti pengenalan sidik jari, wajah, iris, dan telapak tangan) telah menjadi subjek penelitian dalam pemrosesan citra selama lebih dari satu dekade dan kini telah digunakan secara komersial.
- Elektronik konsumen: Kamera digital dan camcorder, dengan kemampuan pemrosesan yang canggih, telah membuat film dan teknologi pita analog menjadi usang. Paket perangkat lunak untuk meningkatkan, mengedit, mengatur, dan mempublikasikan citra dan video telah maju pesat dalam kompleksitasnya sambil tetap menjaga antarmuka yang ramah pengguna. TV berdefinisi tinggi, monitor, pemutar DVD, dan pemutar video pribadi (PVR) semakin meningkat popularitasnya karena harga yang terjangkau. Perkembangan jaringan dan distribusi juga telah berhasil membuat terobosan dalam perangkat lain, seperti personal digital assistants (PDA), ponsel, dan pemutar musik portabel (MP3).
- Internet, khususnya World Wide Web: Ada banyak informasi visual yang tersedia di web. Kolaborasi dalam mengunggah, berbagi, dan memberi anotasi (tagging) pada video semakin populer. Menemukan dan mengambil citra dan video di web berdasarkan isinya tetap menjadi tantangan terbuka dalam penelitian.

B. Konsep Dasar Image Processing

Sebuah gambar digital merupakan larik 2D dari angka-angka yang mewakili versi sampel dari sebuah gambar. Gambar didefinisikan dalam bentuk grid, setiap lokasi grid disebut piksel. Sebuah gambar direpresentasikan oleh grid yang terbatas dan setiap nilai intensitas direpresentasikan oleh sejumlah bit yang terbatas. $M \times N$ pada gambar $f(x, y)$ didefinisikan sebagai:

| | | | | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 197 | 193 | 174 | 168 | 150 | 152 | 129 | 151 | 172 | 161 | 155 | 156 |
| 166 | 182 | 163 | 74 | 75 | 62 | 33 | 17 | 110 | 210 | 180 | 154 |
| 180 | 180 | 40 | 14 | 54 | 6 | 10 | 33 | 45 | 106 | 159 | 181 |
| 206 | 199 | 5 | 124 | 131 | 111 | 120 | 204 | 166 | 15 | 56 | 180 |
| 194 | 68 | 137 | 251 | 237 | 239 | 239 | 228 | 227 | 87 | 71 | 201 |
| 172 | 106 | 207 | 233 | 233 | 214 | 220 | 239 | 228 | 98 | 74 | 206 |
| 188 | 88 | 179 | 209 | 186 | 215 | 211 | 158 | 139 | 75 | 20 | 169 |
| 189 | 97 | 165 | 84 | 10 | 168 | 134 | 11 | 31 | 62 | 22 | 148 |
| 199 | 168 | 191 | 193 | 158 | 227 | 178 | 143 | 182 | 104 | 36 | 190 |
| 205 | 174 | 155 | 252 | 236 | 231 | 149 | 178 | 228 | 43 | 95 | 234 |
| 190 | 216 | 136 | 149 | 236 | 187 | 86 | 150 | 79 | 36 | 218 | 241 |
| 190 | 234 | 147 | 108 | 227 | 210 | 127 | 102 | 36 | 101 | 255 | 224 |
| 190 | 214 | 173 | 46 | 103 | 143 | 96 | 50 | 2 | 109 | 249 | 218 |
| 187 | 196 | 235 | 75 | 1 | 81 | 47 | 0 | 6 | 217 | 258 | 211 |
| 183 | 202 | 237 | 145 | 0 | 0 | 12 | 108 | 200 | 138 | 243 | 236 |
| 195 | 206 | 123 | 297 | 177 | 121 | 123 | 200 | 175 | 13 | 96 | 218 |

$$f(x, y) = \begin{bmatrix} f(0,0) \\ f(1,0) \\ \vdots \\ f(N-1,0) \end{bmatrix} f($$

Dalam representasi ini, digunakan $[0, L - 1]$ jumlah tingkat intensitas untuk mewakili semua nilai piksel grayscale, dan k jumlah bit digunakan untuk mewakili setiap tingkat intensitas, yaitu, $L = 2^k$. Jadi, jumlah bit yang diperlukan untuk menyimpan gambar $M \times N$ adalah $M \times N \times k$. Kecerahan sebuah gambar merujuk pada tingkat keseluruhan cahaya atau kegelapan gambar, sementara kontras adalah perbedaan antara intensitas piksel maksimum dan minimum dalam sebuah gambar. Kecerahan dapat meningkat atau dikurangi dengan penambahan atau pengurangan sederhana pada nilai piksel.

Sebuah gambar biner direpresentasikan oleh hanya satu bit. Di sisi lain, gambar grayscale direpresentasikan oleh 8 bit. Sebuah gambar raster adalah kumpulan titik-titik, yang disebut piksel. Sebuah gambar vektor adalah kumpulan garis dan kurva yang terhubung, dan digunakan untuk menghasilkan objek.

Sebuah gambar adalah fungsi f , dari ruang R^2 ke ruang R . Sebuah gambar direpresentasikan oleh $f(x,y)$, dan itu menunjukkan intensitas di posisi (x,y) . Oleh karena itu, sebuah gambar hanya didefinisikan pada sebuah persegipanjang, dengan rentang yang terbatas, yaitu,

$$f : [a, b] \times [c, d] \rightarrow [0, 1]$$

Gambar berwarna memiliki komponen Merah (R), Hijau (G), dan Biru (B). Masing-masing dari ketiga komponen R, G, B biasanya direpresentasikan oleh 8 bit, dan oleh karena itu dibutuhkan 24-bit untuk sebuah gambar berwarna. Tiga warna primer ini dicampur dalam proporsi yang berbeda untuk mendapatkan warna-warna yang berbeda. Untuk berbagai aplikasi pengolahan gambar, format RGB, HIS, YIQ, YCbCr, dll. digunakan. Sebuah gambar berwarna adalah fungsi tiga komponen, yang merupakan fungsi “nilai-vektor”, dan direpresentasikan sebagai berikut:

$$f(x, y) = [r(x, y) \ g(x, y) \ b(x, y)]$$

Gambar terindeks memiliki peta warna yang terkait, yang merupakan daftar dari semua warna yang digunakan dalam gambar tersebut. Contoh dari format ini adalah gambar PNG dan GIF. Jadi, berbagai jenis gambar digital dapat dijelaskan sebagai berikut.

- Gambar biner - 1 bit/pixel
- Gambar grayscale - 8 bit/pixel
- Gambar warna asli atau RGB - 24 bit/pixel
- Gambar terindeks - 8 bit/pixel

Resolusi spasial sebuah gambar mendefinisikan jumlah piksel yang digunakan untuk mencakup ruang visual yang ditangkap oleh gambar tersebut. Resolusi intensitas sebuah gambar bergantung pada jumlah bit yang digunakan untuk mewakili nilai intensitas yang berbeda. Jumlah gambar atau frame video yang ditangkap oleh kamera dalam waktu tertentu menentukan resolusi temporal. Kurangnya jumlah tingkat intensitas (resolusi intensitas rendah) di area halus sebuah gambar menghasilkan “efek kontur palsu”, yaitu, menciptakan tepi atau garis palsu di mana aslinya tidak ada. Juga “efek kotak-kotak” terjadi ketika resolusi spasial sebuah gambar sangat rendah.

Pengolahan gambar digital berurusan dengan manipulasi dan analisis gambar digital oleh sistem digital. Sebuah operasi pengolahan gambar biasanya mendefinisikan gambar baru g dalam hal gambar masukan f . Seperti yang ditunjukkan dalam Gambar 2.1, kita dapat mengubah rentang f menjadi $g(x, y) = t(f(x, y))$, atau kita dapat mengubah domain f menjadi $g(x, y) = f(tx(x, y), ty(x, y))$. Nilai piksel dimodifikasi dalam transformasi pertama (Gambar 2.1(a)); sedangkan posisi piksel spasial berubah dalam transformasi kedua (Gambar 2.1(b)). Domain sebuah gambar dapat diubah dengan memutar dan menyesuaikan skala gambar seperti yang diilustrasikan dalam Gambar 2.2.



Figure 5: Mengubah Rentang Gambar

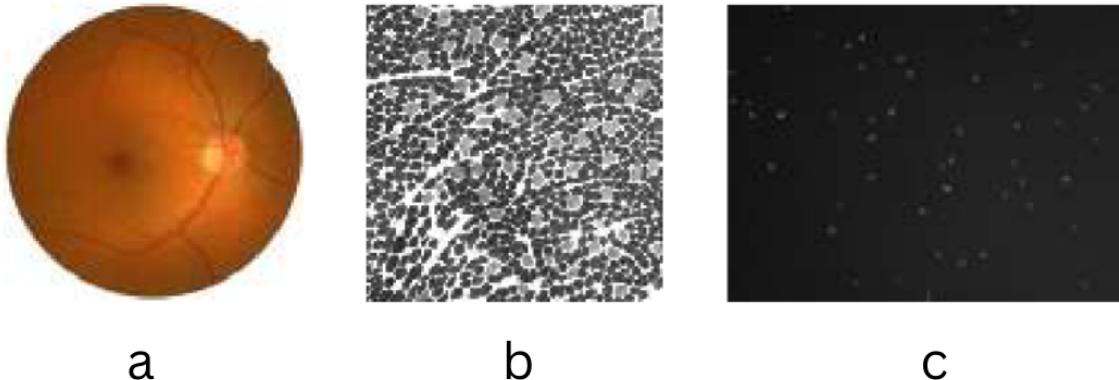


Figure 6: Mengubah Domain dari Suatu Gambar

C. Python Untuk Image Processing

Dalam tutorial ini, Anda akan menemukan fungsi dasar untuk memuat, memanipulasi, dan menampilkan gambar. Informasi utama dari gambar akan diperoleh, seperti ukuran, jumlah saluran, kelas penyimpanan, dan lain-lain. Setelah itu, Anda akan dapat melakukan filter klasik pertama Anda.

Proses yang berbeda akan dilakukan pada gambar-gambar berikut ini:



Spatial Filter

Filtering

Seperti halnya filter air yang menghilangkan kotoran, filter pemrosesan gambar menghapus fitur yang tidak diinginkan (seperti noise) dari sebuah gambar. Setiap filter memiliki utilitas khusus dan dirancang untuk menghilangkan jenis noise tertentu atau meningkatkan aspek tertentu dari gambar. Kami akan membahas banyak filter beserta tujuan dan efek mereka pada gambar.

Untuk melakukan filtering, digunakan filter atau masker. Biasanya, ini berupa jendela persegi dua dimensi yang bergerak melintasi gambar hanya mempengaruhi satu piksel pada satu waktu. Setiap angka dalam filter dikenal sebagai koefisien. Koefisien dalam filter menentukan efek dari filter dan akibatnya gambar keluaran. Mari kita pertimbangkan filter 3×3 , F , yang diberikan dalam Tabel 4.1.

Jika (i, j) adalah piksel dalam gambar, maka sub-gambar di sekitar (i, j) dengan dimensi yang sama dengan filter akan dipertimbangkan untuk filtering. Pusat filter ditempatkan agar tumpang tindih dengan (i, j) . Piksel-piksel dalam sub-gambar dikalikan dengan koefisien yang sesuai dalam filter. Hal ini menghasilkan matriks dengan ukuran yang sama dengan filter. Matriks ini disederhanakan menggunakan persamaan matematika untuk mendapatkan satu nilai yang akan menggantikan nilai piksel pada (i, j) dalam gambar. Persamaan matematika yang tepat tergantung pada jenis filter. Misalnya, dalam kasus filter rata-rata, nilai F adalah

-7, di mana N adalah jumlah elemen dalam filter. Gambar yang difilter diperoleh dengan mengulangi proses penempatan filter pada setiap piksel dalam gambar, memperoleh satu nilai, dan menggantikan nilai piksel dalam gambar asli. Proses ini memindahkan jendela filter melintasi gambar disebut konvolusi dalam domain spasial.

Mari kita pertimbangkan sub-gambar berikut dari gambar I, yang berpusat pada (i, j) .

Konvolusi filter yang diberikan dalam Tabel 4.1 dengan sub-gambar dalam Tabel 4.2 diberikan sebagai berikut:

Di mana $I_{new}(i, j)$ adalah nilai keluaran pada lokasi (i, j) . Proses ini harus diulang untuk setiap piksel dalam gambar. Karena filter memainkan peran penting dalam proses konvolusi, filter juga dikenal sebagai kernel konvolusi. Operasi konvolusi harus dilakukan pada setiap piksel dalam gambar, termasuk piksel di batas gambar. Ketika filter ditempatkan pada piksel batas, sebagian filter akan berada di luar batas gambar. Karena nilai piksel di luar batas tidak ada, nilai-nilai baru harus dibuat sebelum konvolusi. Proses ini untuk menciptakan nilai piksel di luar batas disebut padding. Piksel yang dipadatkan dapat diasumsikan nol atau nilai tetap. Pilihan padding lainnya seperti nearest neighbor atau reflect menciptakan piksel yang dipadatkan menggunakan nilai piksel dalam gambar. Dalam kasus nol, piksel yang dipadatkan semua bernilai nol. Dalam kasus konstan, piksel yang dipadatkan mengambil nilai spesifik. Dalam kasus reflect, piksel yang dipadatkan mengambil nilai dari baris atau kolom terakhir. Piksel yang dipadatkan hanya dipertimbangkan untuk konvolusi dan akan dibuang setelah konvolusi.

Mari kita pertimbangkan contoh untuk menunjukkan berbagai pilihan padding. Gambar 4.1(a) adalah gambar input berukuran 7x7 yang akan dikonvolusi menggunakan filter 3x5 dengan pusat filter di $(1,2)$. Untuk memasukkan piksel batas dalam konvolusi, kita memasukkan gambar dengan satu baris di atas dan satu baris di bawah serta dua kolom di sebelah kiri dan dua kolom di sebelah kanan. Secara umum, ukuran filter menentukan jumlah baris dan kolom yang akan dipadatkan ke gambar.

- Padding dengan nol: Semua piksel yang dipadatkan diberi nilai nol (Gambar 4.1(b)).
- Padding dengan konstanta: Nilai konstan 5 digunakan untuk semua piksel yang dipadatkan (Gambar 4.1(c)). Nilai konstan dapat dipilih berdasarkan jenis gambar yang sedang diproses.
- Nearest neighbor: Nilai dari baris atau kolom terakhir (Gambar 4.1(d)) digunakan untuk padding.
- Reflect: Nilai dari baris atau kolom terakhir (Gambar 4.1(e)) dipantulkan melintasi batas gambar.
- Wrap: Dalam opsi wrap seperti yang ditunjukkan dalam Gambar 4.1(f), baris pertama (atau kolom) setelah batas mengambil nilai yang sama dengan baris pertama (atau kolom) dalam gambar, dan seterusnya.

Modul 4. Feature Extraction and Matching

A. Pengenalan Feature Extraction and Matching

Ekstraksi dan Pencocokan Fitur adalah tugas penting dalam visi komputer, seperti struktur dari gerak, pengambilan gambar, dan deteksi objek.

Fitur adalah bagian dari informasi yang relevan untuk menyelesaikan tugas komputasi yang terkait dengan aplikasi tertentu. Fitur mungkin struktur tertentu dalam gambar seperti titik, tepi atau objek. Fitur mungkin juga merupakan hasil dari operasi lingkungan umum atau deteksi fitur yang diterapkan pada gambar. Fitur dapat diklasifikasikan menjadi dua kategori utama:

- Fitur yang ada di lokasi tertentu dari gambar, seperti puncak gunung, sudut bangunan, pintu masuk, atau petak salju yang berbentuk menarik. Jenis fitur yang dilokalkan ini sering disebut fitur titik kunci (atau bahkan sudut) dan sering digambarkan dengan tampilan tambalan piksel yang mengelilingi lokasi titik.
- Fitur yang dapat dicocokkan berdasarkan orientasi dan kenampakan lokalnya (profil tepi) disebut tepi dan juga dapat menjadi indikator yang baik untuk batasan objek dan kejadian oklusi dalam urutan citra.

Komponen utama Deteksi dan Pencocokan Fitur:

Deteksi : Identifikasi Feature Point.

Deskripsi: Penampilan lokal di sekitar setiap titik fitur dijelaskan dalam beberapa cara yang (idealnya) tidak berubah di bawah perubahan iluminasi, translasi, skala, dan rotasi dalam bidang. Kami biasanya berakhir dengan vektor deskriptor untuk setiap titik fitur.

Pencocokan: Deskriptor dibandingkan di seluruh gambar, untuk mengidentifikasi fitur serupa. Untuk dua gambar kita mungkin mendapatkan satu set pasangan (X_i, Y_i) (X'_i, Y'_i), di mana (X_i, Y_i) adalah fitur dalam satu gambar dan (X'_i, Y'_i) fitur pencocokannya di gambar lainnya gambar.

Feature Descriptor

Deskriptor fitur adalah algoritme yang mengambil gambar dan menampilkan deskriptor fitur/vektor fitur. Deskriptor fitur menyandikan informasi menarik ke dalam rangkaian angka

dan bertindak sebagai semacam “sidik jari” numerik yang dapat digunakan untuk membedakan satu fitur dari fitur lainnya.

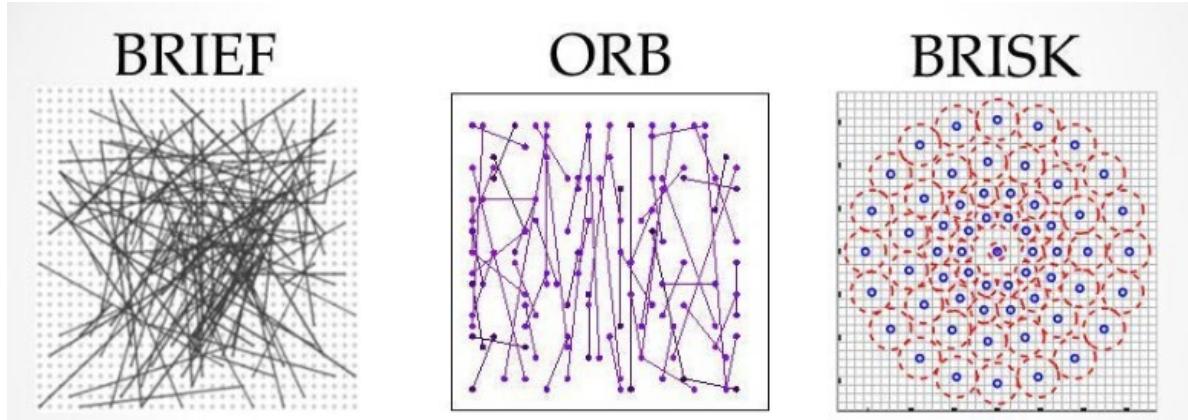


Figure 7: Gambar... Macam-macam Feature Descriptor

Idealnya, informasi ini akan menjadi invarian di bawah transformasi citra, sehingga kita dapat menemukan kembali fitur tersebut bahkan jika citra diubah dalam beberapa cara. Setelah mendekripsi feature point, kami melanjutkan untuk menghitung deskriptor untuk masing-masingnya. Deskriptor dapat dikategorikan menjadi dua kelas:

- Deskriptor Lokal: Ini adalah representasi kompak dari lingkungan lokal suatu titik. Deskriptor lokal mencoba untuk menyerupai bentuk dan penampilan hanya di lingkungan lokal sekitar titik dan dengan demikian sangat cocok untuk merepresentasikannya dalam hal pencocokan.
- Deskriptor Global : Deskriptor global menjelaskan keseluruhan gambar. Mereka umumnya tidak terlalu kuat karena perubahan sebagian gambar dapat menyebabkannya gagal karena akan memengaruhi deskriptor yang dihasilkan.

Macam - macam Algoritma Descriptor :

- SIFT(Scale Invariant Feature Transform)
- SURF(Speed Up Robust Feature)
- ORB(Oriented FAST and Rotate BRIEF)
- BRISK(Binary Robust Invariant Scalable Keypoints)
- BRIEF(Binary Robust Independent Elementary Feature)

Feature Matching

Pencocokan fitur atau umumnya pencocokan gambar, bagian dari banyak aplikasi visi komputer seperti pendaftaran gambar, kalibrasi kamera dan pengenalan objek, adalah tugas membangun korespondensi antara dua gambar dari pemandangan/objek yang sama. Pendekatan umum untuk pencocokan citra terdiri dari pendekripsi sekumpulan poin penting yang masing-masing terkait dengan deskriptor citra dari data citra. Setelah fitur dan deskriptornya diekstraksi dari dua atau lebih gambar, langkah selanjutnya adalah membuat beberapa pencocokan fitur awal antara gambar-gambar ini.

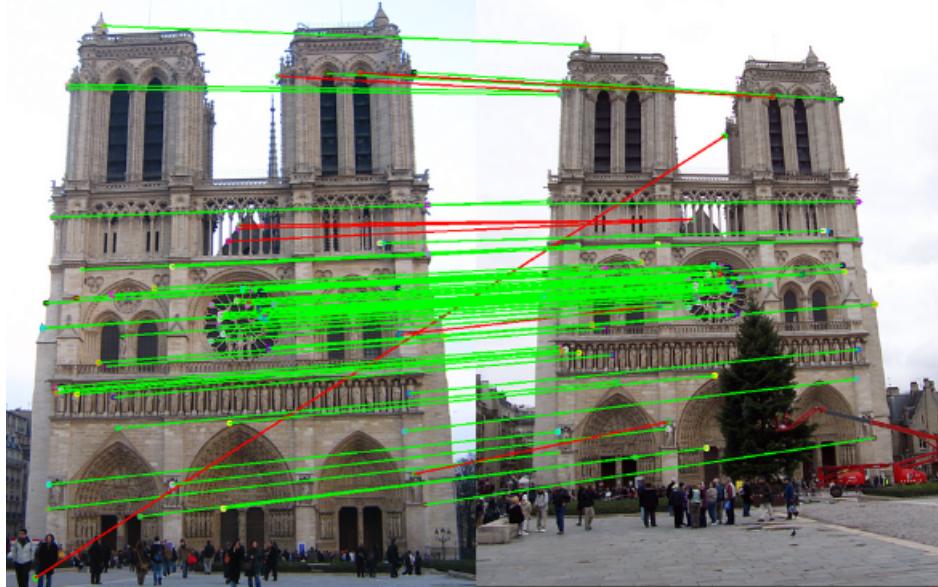


Figure 8: Gambar... Pencocokan gambar

Secara umum, kinerja metode pencocokan berdasarkan interest point atau feature point bergantung pada properti dari feature point yang mendasarinya dan pilihan deskriptor gambar terkait. Dengan demikian, detektor dan deskriptor yang sesuai untuk konten gambar harus digunakan dalam aplikasi. Misalnya, jika gambar mengandung sel bakteri, detektor blob harus digunakan daripada detektor sudut. Tapi, jika gambar tersebut adalah pemandangan kota dari udara, detektor sudut cocok untuk menemukan struktur buatan manusia. Selain itu, pemilihan detektor dan deskriptor yang mengatasi degradasi citra sangatlah penting. Macam - macam Algoritma Pencocokan:

- Brute-Force Matcher
- FLANN(Fast Library for Approximate Nearest Neighbors) Matcher

B. HOG dan SIFT

Histogram of Oriented Gradient (HOG)

Histogram of Oriented Gradients (HOG) adalah deskriptor fitur citra yang dapat digunakan untuk deteksi objek [52]. Untuk mengekstrak fitur ini, frekuensi orientasi gradien dalam bagian-bagian lokal citra dihitung. Penampilan dan bentuk objek lokal dalam sebuah citra dapat dijelaskan melalui distribusi gradien intensitas atau arah tepi. Langkah-langkah berikut perlu diimplementasikan untuk mengekstrak fitur HOG:

- Perhitungan gradien: Langkah pertama adalah menghitung gradien horizontal dan vertikal yang terpusat (G_x dan G_y) tanpa melakukan smoothing pada citra. Untuk tujuan ini, dapat digunakan operator Sobel atau operator deteksi tepi lainnya untuk mendapatkan gradien. Untuk citra berwarna, saluran warna yang memberikan magnitudo gradien tertinggi untuk setiap piksel dapat dipilih. Kemudian, magnitudo gradien dan orientasi gradien dihitung sebagai berikut:

$$\text{Magnitudo: } |\Delta f| = \sqrt{G_x^2 + G_y^2}$$

$$\text{Orientation: } \theta = \arctan \left(\frac{G_y}{G_x} \right)$$

- Orientasi pengalamatan: Langkah kedua adalah pembuatan histogram sel. Untuk ini, orientasi gradien diquantisasi ke dalam bin. Setiap bin akan mendapatkan voting berdasarkan magnitudo gradien. Voting juga dapat diberi bobot dengan filter Gaussian untuk mengurangi bobot piksel-piksel di dekat tepi blok.

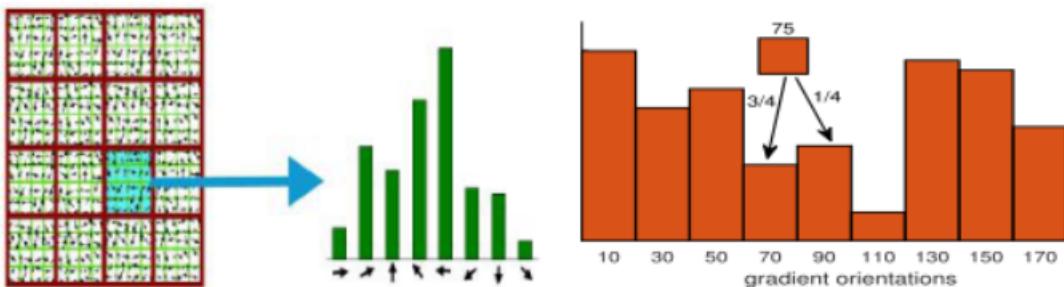


Figure 9: Histogram of oriented gradients: (a) cell histogram and (b) orientation binning

Mari kita ambil contoh ekstraksi deskriptor HOG. Untuk ini, kita menggunakan citra berukuran 64×128 . Pertama, citra dibagi menjadi blok 16×16 dengan tumpang tindih 50%. Jadi, totalnya akan ada $7 \times 15 = 105$ blok. Setiap blok harus terdiri dari 2×2 sel dengan ukuran 8×8

piksel. Sekarang, orientasi gradien diquantisasi menjadi 9 bin. Di sini, voting adalah magnitudo gradien. Sekarang, voting diinterpolasi secara bilinear antara pusat bin yang berdekatan. Misalnya, misalkan orientasi adalah 75° . Kemudian, jarak antara pusat bin bin 70 dan bin 90 adalah 5° dan 15° , masing-masing. Oleh karena itu, rasio kepemilikan adalah $15/20$ atau $3/4$ dan $5/20$ atau $1/4$. Hal ini ditunjukkan secara diagrammatik pada Gambar 3.24(a) dan (b). Histogram dari gradien yang diarahkan ditunjukkan pada Gambar 3.25.

- Penggabungan blok deskriptor: Histogram sel kemudian digabungkan untuk membentuk vektor fitur seperti yang ditunjukkan pada Gambar 3.26. Pada contoh kita, histogram yang diperoleh dari blok yang tumpang tindih 2×2 sel digabungkan menjadi vektor fitur 1-D dengan dimensi $105 \times 2 \times 2 \times 9 = 3780$.
- Normalisasi blok: Setiap blok dapat dinormalisasi dengan faktor normalisasi yang berbeda, seperti L2-norm, Zq-norm, Li-squared root norm, dll. Normalisasi blok membuat deskriptor invariant terhadap variasi pencahayaan dan fotometri.
- Deskriptor final: Deskriptor HOG akhir dapat digunakan untuk pengenalan objek. Deskriptor ini merupakan fitur untuk algoritma pembelajaran mesin, seperti Support Vector Machine (SVM).



Figure 10: Hasil HOG

Scale Invariant Feature Transform (SIFT)

Transformasi fitur skala invarian (SIFT) adalah algoritma deteksi fitur untuk mendeteksi dan menggambarkan fitur lokal pada gambar untuk pengenalan objek. Detektor sudut Harris invarian terhadap translasi dan rotasi, tetapi tidak terhadap skala. Namun, algoritma SIFT dapat mendeteksi dan menggambarkan fitur lokal pada gambar. Fitur-fitur ini invarian terhadap

translasi, penskalaan, dan rotasi gambar, serta sebagian invariant terhadap perubahan pencahayaan [55]. Fitur-fitur gambar yang diekstraksi dari gambar latihan harus tetap terdeteksi bahkan dengan perubahan skala gambar, noise, dan pencahayaan. Konsep skala memainkan peran penting dalam analisis gambar. Dalam analisis gambar, kita perlu mengekstraksi fitur gambar yang sesuai dengan menganalisis struktur gambar yang berbeda. Struktur-struktur ini mungkin ada pada skala yang berbeda. Jadi, jumlah informasi yang disampaikan oleh struktur gambar tertentu tergantung pada skala. SIFT mempertimbangkan masalah ini, yaitu fitur-fitur yang invariant terhadap perubahan skala. Langkah-langkah utama dari algoritma SIFT adalah sebagai berikut:

- Estimasi ekstremum skala-ruang: Ini sesuai dengan ekstremum DoG dan memastikan ekstraksi wilayah invariant skala. Tentukan lokasi perkiraan dan skala titik fitur yang menonjol (juga disebut “keypoints”).
- Lokalisasi dan penyaringan titik fitur: Perbaiki lokasi dan skala mereka, yaitu pilih titik fitur yang asli dan buang yang buruk.
- Pemberian orientasi: Tentukan orientasi untuk setiap titik fitur, yaitu kurangi efek rotasi.
- Membuat deskriptor: Menggunakan histogram deskriptor orientasi untuk setiap titik kunci.

Deskriptor ini dibuat dengan menghitung histogram orientasi untuk setiap titik kunci. Deskriptor ini menangkap informasi gradien lokal di sekitar titik kunci dan memberikan representasi yang khas dari wilayah lokal. Langkah-langkah dalam membuat deskriptor adalah sebagai berikut:

- Bagi wilayah di sekitar titik kunci menjadi sub-wilayah atau bin yang lebih kecil.
- Untuk setiap sub-wilayah, hitung magnitudo dan orientasi gradien piksel.
- Tetapkan setiap piksel ke salah satu bin berdasarkan orientasi gradiennya.
- Untuk setiap bin, akumulasikan magnitudo gradien dari piksel yang ditugaskan.
- Buat histogram orientasi dengan mempertimbangkan magnitudo gradien yang terakumulasi sebagai bobot untuk setiap bin.
- Normalisasi histogram untuk membuatnya invariant terhadap perubahan pencahayaan.
- Deskriptor akhir terbentuk dengan menggabungkan histogram orientasi yang telah dinormalisasi.

Setelah deskriptor dihitung untuk semua titik kunci, mereka dapat digunakan untuk berbagai aplikasi seperti pengenalan objek, penyatuan gambar, dan pencarian gambar. Pemadanan titik kunci di gambar-gambar yang berbeda berdasarkan deskriptornya memungkinkan pencocokan fitur yang kuat melintasi perubahan skala, rotasi, dan pencahayaan.

Untuk mengimplementasikan algoritma SIFT dalam Python, dapat menggunakan pustaka OpenCV. OpenCV menyediakan fungsi-fungsi untuk deteksi titik kunci, perhitungan deskriptor, dan pencocokan fitur. Dapat merujuk pada dokumentasi dan tutorial OpenCV untuk informasi detail tentang cara menggunakan algoritma SIFT dalam Python.C. Feature Matching.

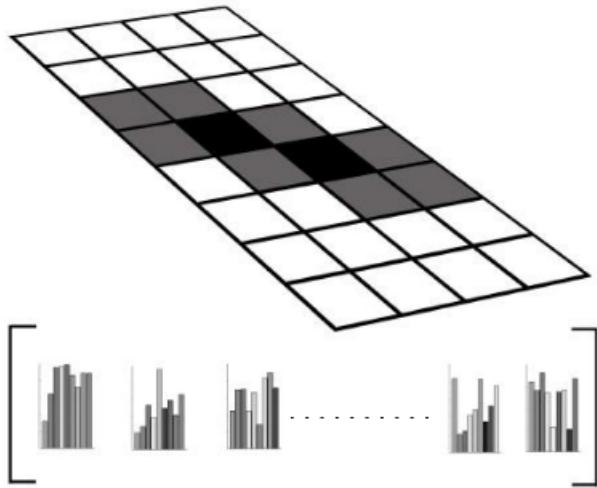


FIGURE 3.26: Concatenated feature vector.

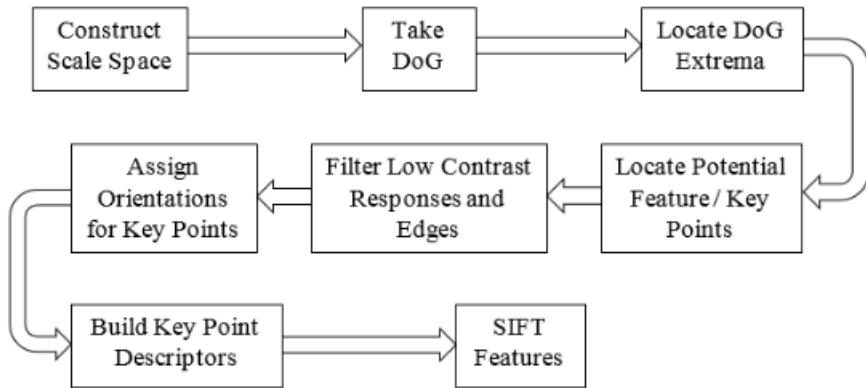


Figure 11: Gambar 3.27 Implementasi Langkah algoritma SIFT

Gambar 3.27 menunjukkan semua langkah implementasi dari algoritma SIFT. Langkah-langkah ini sekarang akan dijelaskan secara detail di bawah ini.

- **Deteksi fitur skala-invarian:** Langkah pertama adalah mendeteksi titik-titik unik (kunci) yang dapat dipilih kembali secara berulang dengan perubahan lokasi/skala. Un-

tuk tujuan ini, seperti yang ditunjukkan di Gambar 3.28, digunakan representasi skala dengan menghitung piramida Laplacian yang dinormalisasi skala menggunakan difference of Gaussian (DoG) multiskala. Secara khusus, DoG dari citra $D(x, y, \sigma)$ diberikan oleh:

$$D(x, y, \sigma) = L(x, y, k_i) - L(x, y, k_j)$$

Di mana, $L(x, y, k)$ adalah hasil konvolusi dari citra asli $f(x, y)$ dengan blur Gaussian $G(x, y, k)$ pada skala k , dengan kata lain,

$$L(x, y, k_i) = G(x, y, k_i) * I(x, y)$$

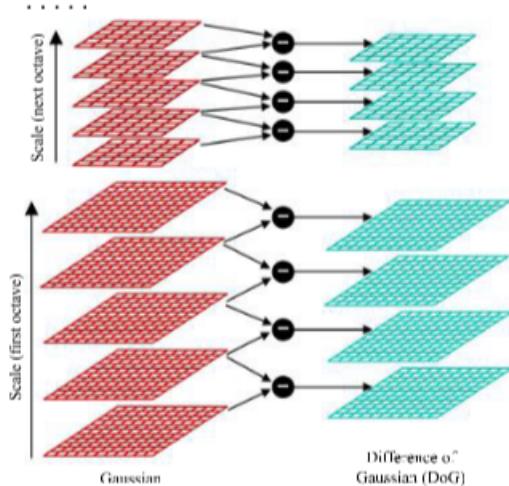


Figure 12: Gambar 3.28 Formasi Laplacian Pyramid

- **Deteksi puncak dalam skala-ruang:** Pada tahap ini, titik-titik ekstrem lokal dideteksi dengan mempertimbangkan baik ruang maupun skala. Tujuannya adalah untuk mengidentifikasi lokasi dan skala yang dapat diberikan secara berulang dalam pandangan yang berbeda dari adegan atau objek yang sama. Pada kasus diskrit, hal ini ditentukan dengan membandingkan dengan 26 tetangga terdekat seperti yang ditunjukkan di Gambar 3.29.
- **Lokalisasi titik kunci dan penolakan outlier:** Selanjutnya, skala yang memberikan ekstremum dalam perbedaan Gaussian ditetapkan sebagai skala untuk titik kunci. Namun, deteksi ekstremum dalam skala-ruang menghasilkan banyak calon titik kunci. Namun demikian, beberapa titik kunci tidak stabil. Oleh karena itu, langkah berikutnya dari algoritma ini adalah menolak beberapa titik kunci yang memiliki kontras rendah atau yang terlokalisasi buruk di sepanjang tepi. Titik kunci dengan kontras rendah sensitif terhadap noise.
- **Penghilangan titik kunci kontras rendah:** Titik kunci dengan kontras rendah dan terlokalisasi buruk dihilangkan dengan menggunakan interpolasi subpixel/subskala

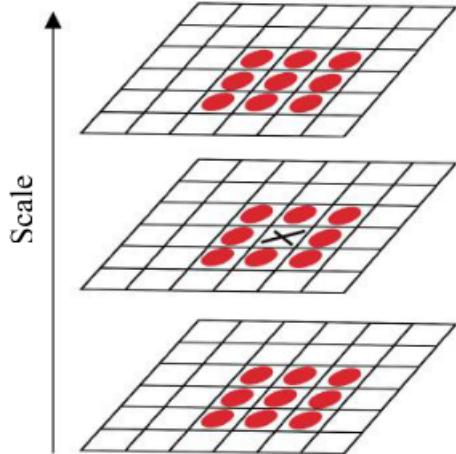


Figure 13: 3.29. Scale-space peak detection

menggunakan ekspansi Taylor, yang diberikan oleh:

$$D(x) = D + \frac{\partial D^T}{\partial x} x + \frac{\frac{1}{2} x^T (\partial^2 D)}{\partial x^2} x$$

Di mana, D dan turunannya dihitung di titik kunci calon dan $x = (x, y,)$ adalah offset dari titik ini. Lokasi puncak x diperkirakan dengan mempertimbangkan turunan fungsi ini terhadap x dan mengatur nilainya menjadi nol. Jika offset x lebih besar dari ambang batas yang telah ditentukan dalam dimensi manapun, maka hal ini menunjukkan bahwa puncak berada lebih dekat dengan titik kunci calon lainnya. Dalam hal ini, titik kunci calon harus diubah dan interpolasi dilakukan di sekitar titik tersebut. Jika tidak, offset ditambahkan ke titik kunci calon tersebut. Hal ini dilakukan untuk mendapatkan perkiraan yang diinterpolasi untuk lokasi puncak.

Penghilangan respons tepi: Titik-titik tepi sesuai dengan kontras tinggi dalam satu arah dan rendah dalam arah lainnya. Fungsi DoG memiliki respons kuat di sepanjang tepi gambar. Titik kunci yang memiliki lokasi yang sangat tidak terdefinisi tetapi memiliki respons tepi tinggi dihilangkan. Langkah ini meningkatkan stabilitas. Puncak yang tidak terdefinisi dengan baik dalam fungsi DoG menunjukkan kelengkungan tinggi di sepanjang tepi dan nilai rendah dalam arah tegak lurus. Kelengkungan utama dapat dihitung dengan mengevaluasi matriks Hessian. Perlu dicatat bahwa untuk puncak yang tidak terdefinisi dengan baik dalam fungsi DoG, kelengkungan utama di sepanjang tepi jauh lebih besar daripada kelengkungan utama sepanjangnya. Untuk menemukan kelengkungan utama, kita perlu mencari solusi untuk eigenvalue dari matriks Hessian orde kedua H sebagai berikut:

$$H = D_{xx} D_{xy} D_{xy} D_{yy}$$

Diketahui,

$$\text{Trace}(H) = D_{xx} + D_{yy} = \lambda_1 + \lambda_2$$

$$\text{Det}(H) = D_{xx} D_{yy} - D_{xy}^2 = \lambda_1 \lambda_2$$

$$R = \left(\frac{\text{Trace}(H^2)}{\text{Det}(H)} \right) = \frac{(r+1)^2}{r}$$

Dalam persamaan di atas, rasio R hanya bergantung pada rasio eigenvalue $r = A1/A2$, dan R minimum ketika eigenvalue sama satu sama lain. Jika perbedaan absolut antara dua eigenvalue lebih tinggi, maka perbedaan absolut antara dua kelengkungan utama D juga akan lebih tinggi. Ini sesuai dengan nilai tinggi dari R. Oleh karena itu, eliminasi titik kunci dilakukan jika:

$$\left(\frac{\text{Trace}(H^2)}{\text{Det}(H)} \right) > \frac{(r+1)^2}{r}$$

- **Penugasan orientasi:** Pada langkah ini, setiap titik kunci diberikan satu atau lebih orientasi. Orientasi ditentukan berdasarkan arah gradien citra lokal. Deskriptor titik kunci dapat direpresentasikan relatif terhadap orientasi ini. Itulah sebabnya, mereka invariant terhadap rotasi citra. Untuk ini, magnitudo dan orientasi pada citra yang telah dihaluskan dengan Gaussian (pada skala yang sesuai dengan titik kunci) dihitung. Pertama, citra yang telah dihaluskan dengan Gaussian $L(x,y,a)$ pada skala titik kunci a diambil agar semua perhitungan terkait dilakukan dalam cara yang invariant terhadap skala. Untuk sampel citra $L(x,y)$ pada skala a , magnitudo gradien $m(x,y)$ dan orientasi (x,y) dihitung menggunakan perbedaan piksel sebagai berikut:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

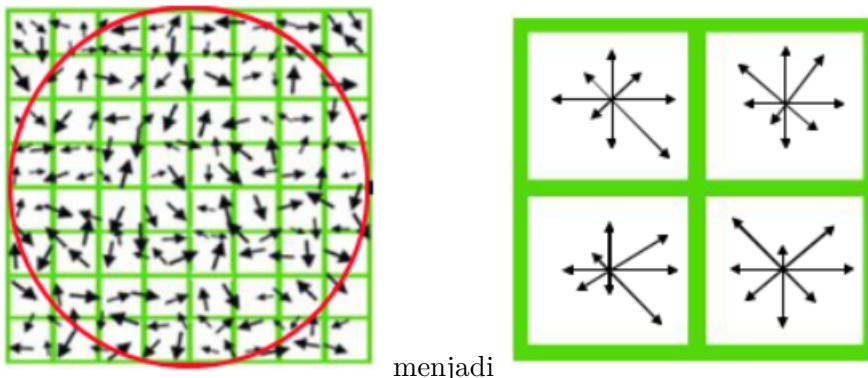
$$\theta(x,y) = \arctan\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right)$$

Perhitungan magnitudo dan arah untuk gradien harus dilakukan untuk setiap piksel tetangga di sekitar titik kunci dalam citra L yang telah dihaluskan dengan Gaussian. Selanjutnya, histogram orientasi dibentuk dari orientasi gradien titik sampel dalam sebuah wilayah di sekitar titik kunci. Histogram orientasi memiliki 36 bin yang mencakup rentang 360 derajat orientasi. Setiap sampel yang ditambahkan ke histogram diberi bobot berdasarkan magnitudo gradiennya dan oleh jendela lingkaran berbobot Gaussian. Puncak-puncak dalam histogram ini sesuai dengan orientasi dominan dari patch citra. Pada skala dan lokasi yang sama, dapat ada beberapa titik kunci dengan orientasi yang berbeda. Jika terdapat penugasan beberapa orientasi, titik kunci tambahan harus dibuat. Titik kunci tambahan tersebut harus memiliki lokasi dan skala yang sama dengan titik kunci asli untuk setiap orientasi tambahan. Jadi, setiap titik

kunci memiliki parameter $(x, y, 2, \dots)$.

- **Deskriptor titik kunci:** Pada langkah-langkah sebelumnya, lokasi titik kunci pada skala tertentu telah ditemukan. Setelah itu, orientasi ditetapkan untuk titik kunci tersebut. Penugasan orientasi menjamin invariansi terhadap lokasi citra, skala, dan rotasi. Langkah selanjutnya adalah menghitung vektor deskriptor untuk masing-masing titik kunci. Tujuannya adalah membuat deskriptor menjadi sangat khas. Selain itu, deskriptor juga seharusnya sebagian invariant terhadap pencahayaan, sudut pandang 3D, dll. Langkah ini harus dilakukan pada citra yang paling mendekati skala titik kunci. Sebelum menghitung deskriptor, penting untuk memutar wilayah dengan nilai orientasi negatif (minus) yang berkaitan dengan titik kunci.

Pertama, sejumlah histogram orientasi dibuat pada subwilayah (lingkungan) piksel 4×4 , dan 8 bin dialokasikan untuk setiap subwilayah tersebut. Histogram ini berasal dari magnitudo dan nilai orientasi dari sampel dalam wilayah 16×16 di sekitar titik kunci. Jadi, setiap histogram berisi sampel dari subwilayah 4×4 dari wilayah lingkungan asli. Selanjutnya, magnitudo diberi bobot dengan fungsi Gaussian. Deskriptor kemudian menjadi vektor yang berisi semua nilai dari histogram-histogram ini. Karena terdapat $4 \times 4 = 16$ histogram, masing-masing dengan 8 bin, vektor tersebut akan memiliki 128 elemen. Jadi, dimensi deskriptor atau vektor fitur akan menjadi 128. Vektor ini kemudian dinormalisasi menjadi panjang satuan untuk mengurangi efek variasi pencahayaan. Gambar 3.30 menunjukkan gradien gambar dan deskriptor titik kunci yang sesuai.



Untuk aplikasi seperti pencarian gambar berbasis konten, fitur SIFT dapat dihitung untuk sekumpulan gambar dalam basis data, dan deskriptor-fiturnya disimpan dalam basis data tersebut. Begitu juga untuk sebuah gambar query, fitur SIFT dapat dihitung. Untuk pencocokan, deskriptor terdekat dalam basis data yang sesuai dengan deskriptor gambar query dapat ditemukan dengan menggunakan metrik jarak yang sesuai, dan gambar query tersebut dapat diambil kembali dari basis data.

Proses pencarian dapat dilakukan dengan membandingkan deskriptor-fitur gambar query dengan deskriptor-fitur gambar dalam basis data menggunakan metrik jarak seperti jarak Euclidean atau jarak cosine. Metrik jarak digunakan untuk mengukur sejauh mana kedua deskriptor-fitur tersebut mirip satu sama lain. Jika terdapat kemiripan yang signifikan an-

tara deskriptor-fitur gambar query dengan deskriptor-fitur gambar dalam basis data, gambar-gambar tersebut dapat dianggap sebagai pencocokan potensial.

Setelah pencocokan dilakukan, gambar-gambar dalam basis data dapat diurutkan berdasarkan tingkat kemiripan dengan gambar query. Gambar-gambar yang memiliki deskriptor-fitur yang paling mirip dengan gambar query akan muncul sebagai hasil teratas dalam hasil pencarian. Dengan demikian, gambar-gambar dalam basis data yang memiliki fitur yang mirip dengan gambar query dapat ditemukan dan dipulihkan dengan menggunakan fitur-fitur SIFT. Penggunaan fitur-fitur SIFT dalam aplikasi pencarian gambar berbasis konten memungkinkan pencarian yang lebih akurat dan efisien dengan mengabaikan perubahan skala, rotasi, dan pencahayaan dalam gambar.

C. BoW dan BoVW

Bag of Word (BOW) pada awalnya tidak digunakan untuk visi komputer namun digunakan dalam bidang Text-Processin. Terkadang dalam konteks visi komputer Bag of Word disebut juga Bag of Visual Word. Namun Kita tetap akan menggunakan istilah BOW karena ini adalah istilah yang digunakan pada OpenCV.

BoW adalah teknik yang digunakan untuk memberikan bobot atau hitungan untuk setiap kata dalam rangkaian dokumen; kita kemudian merepresentasikan dokumen-dokumen ini dengan vektor-vektor dari jumlah ini. Mari kita lihat pada sebuah contoh, sebagai berikut:

Dokumen 1: Saya suka Python dan Saya suka Java **Dokumen 2:** Saya suka Python dan C# **Dokumen 3:** Saya tidak suka pemrograman

Dari ketiga dokumen diatas dapat dibuat kamus atau kosakata, dengan nilai nilai sebagai berikut:

```
{  
    Saya : 4  
    suka : 4  
    Python : 2  
    dan : 2  
    Java : 1  
    C# : 1  
    tidak : 1  
    pemrograman : 1  
}
```

Kita memiliki 8 entri atau fitur yang nantinya akan direpresentasikan untuk dokumen 1,2, dan 3. Setiap vektor berisi nilai yang mewakili jumlah semua kata dalam kamus secara berurutan, untuk dokumen tertentu. Representasi vektor dari tiga kalimat sebelumnya

adalah sebagai berikut:

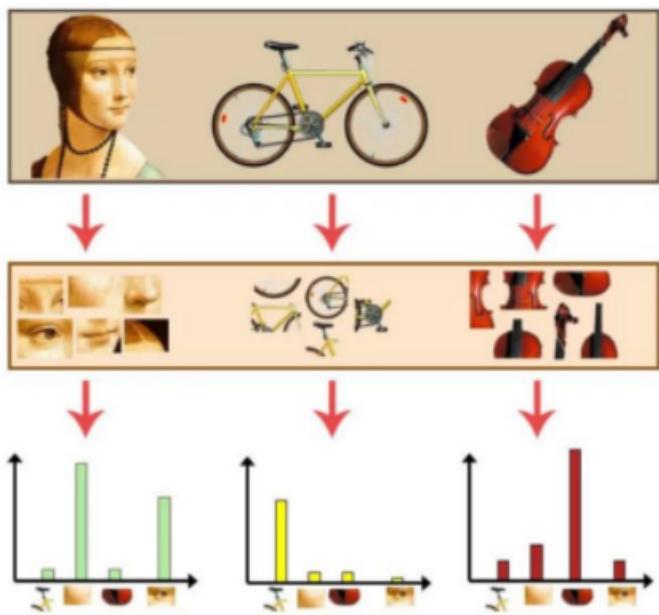
| | Saya | suka | Python | dan | Java | C# | tidak | pemrograman |
|-----------|------|------|--------|-----|------|----|-------|-------------|
| Dokumen 1 | 2 | 2 | 1 | 1 | 1 | 0 | 0 | 0 |
| Dokumen 2 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| Dokumen 3 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |

Vektor-vektor ini dapat dikonseptualisasikan sebagai representasi histogram dari sebuah dokument atau sebagai vektor deskriptor yang dapat digunakan untuk pengklasifikasian.

Konsep BOVW(Bag of Visual Word) diadaptasi dari BOW(Bag of Word) serta Information Retrieval yang ada pada NLP(Natural language Processing). BOW(Bag of Wors) bekerja dengan memakai frekuensi tiap kata-kata supaya mengetahui keyword dari dokumen dengan menghitung berapa jumlah setiap katta yang muncul di dokumen (Davida, 2018).

Yang membedakan BOVW dengan BOW yaitu pada BOW yang digunakan adalah kata-kata, sedangkan BOVW menggunakan fitur-fitur gambar atau bisa dikatakan pola unik pada gambar yang berperan sebagai “kata-kata” yang ada di BOW. Setelah menghitung frekuensi, maka dari perhitungan frekuensi tersebut setiap fitur-fitur gambar dibuat histogram. Fitur itu sendiri tersusun dari deskriptor dan keypoints (Davida, 2018).

Sesuai dengan namanya, keypoints merupakan titik-titik yang menunjukan bagian-bagian gambar yang “menonjol”. Sedangkan deskriptor adalah gambaran dari keypoint. Dengan begitu, walaupun gambar dikecilkan, diperluas, diputar, letak titik-titik keypoint-nya akan tetap sama. Histogram frekuensi yang terbentuk dapat digunakan untuk memprediksi kategori citra serta menemukan citra lainnya yang mirip atau serupa (Davida, 2018).



References