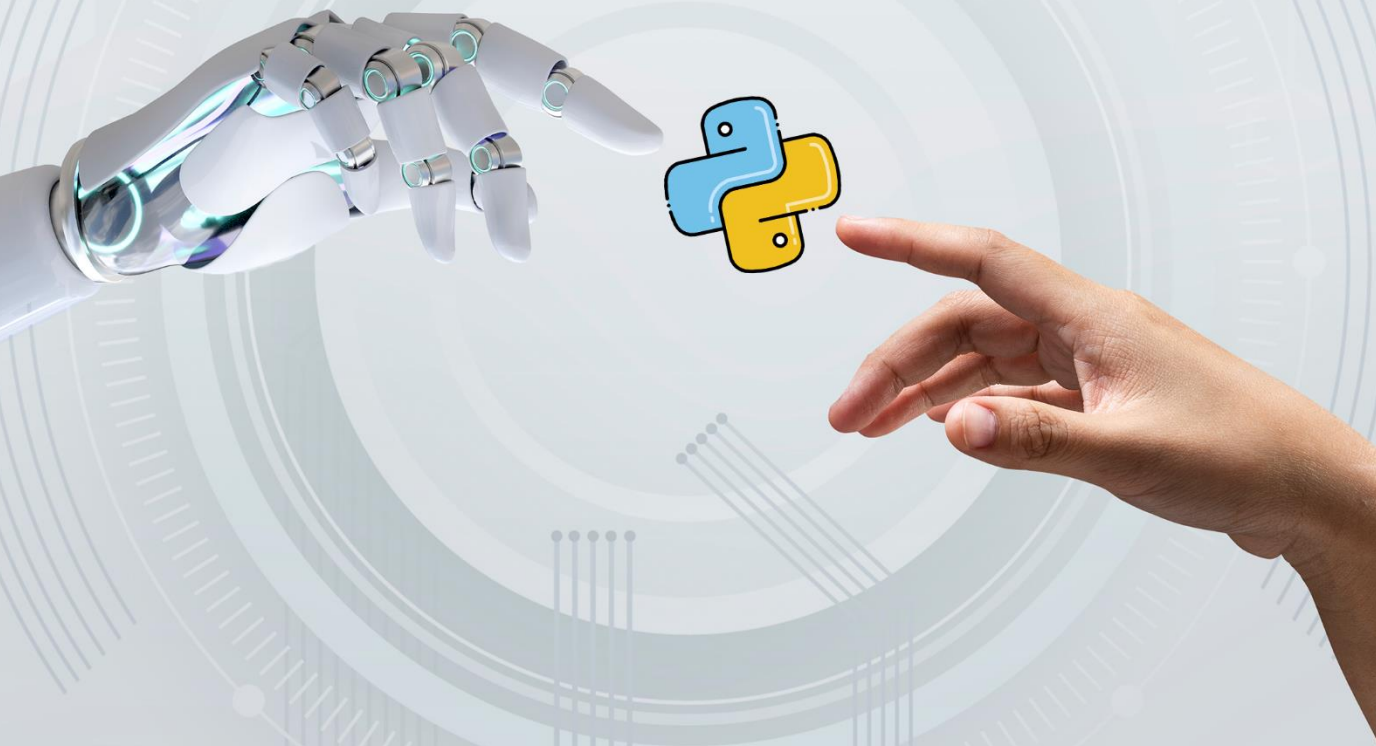
PT MEDIA PENERBIT INDONESIA

MACHINE LEARNING

untuk deteksi dan identifikasi object menggunakan python



Dwi Nurul Huda, S.T., M.Kom
Liza Safitri, S.T., M. Kom
Mochammad Rizki Romdoni, S.Kom., M.T
Ade Winarni, M.T
Abdur Rahman, S.Kom

Machine Learning

Untuk deteksi dan identifikasi object

Menggunakan python

Dwi Nurul Huda, S.T., M.Kom
Liza Safitri, S.T., M. Kom
Mochammad Rizki Romdoni, S.Kom., M.T
Ade Winarni, M.T
Abdur Rahman, S.Kom

PT MEDIA PENERBIT INDONESIA

MACHINE LEARNING UNTUK DETEKSI DAN IDENTIFIKASI OBJECT MENGGUNAKAN PYTHON

Ditulis oleh:

Dwi Nurul Huda, S.T., M.Kom
Liza Safitri, S.T., M. Kom
Mochammad Rizki Romdoni, S.Kom., M.T
Ade Winarni, M.T
Abdur Rahman, S.Kom

ISBN: 978-634-7012-16-6
IV + 119 hlm; 18,2x25,7 cm.
Cetakan I, November 2024

Desain Cover dan Tata Letak:
Melvin Mirsal

Diterbitkan, dicetak, dan didistribusikan oleh
PT Media Penerbit Indonesia
Royal Suite No. 6C, Jalan Sedap Malam IX, Sempakata
Kecamatan Medan Selayang, Kota Medan 20131
Telp: 081362150605
Email: ptmediapenerbitindonesia@gmail.com
Web: https://mediapenerbitindonesia.com
Anggota IKAPI No.088/SUT/2024

KATA PENGANTAR

Puji syukur Tim Penulis panjatkan atas kehadirat Allah SWT, karena atas Rahmat dan Hidayah-Nya, buku Machine Learning untuk Deteksi dan Identifikasi Object Menggunakan Python dapat terselesaikan. Machine learning merupakan cabang dari ilmu kecerdasan buatan yang memungkinkan mesin dapat belajar secara mandiri untuk dapat membuat keputusan berdasarkan pola yang telah diidentifikasi, sehingga mampu meningkatkan kinerja program tersebut. Cara kerja machine learning akan belajar berdasarkan data yang diberikan kemudian membentuk pola berdasarkan identifikasi dan akan menghasilkan suatu keputusan yang diharapkan dapat bermanfaat bagi pengguna.

Buku ini berisi pembahasan tentang deteksi dan identifikas Object yang biasanya dilakukan oleh machine learning. Buku ini pula dilengkapi dengan praktek menggunakan python agar pembaca dapat memahami langsung ketika diimplementasikan ke dalam bahasa pemrograman.

Tim Penulis menyadari bahwa dalam menulis dan menyusun buku ini masih sangat jauh dari kesempurnaan. Oleh karena itu, segala kritik dan saran yang bersifat membangun sangat kami harapkan demi perbaikan ke depannya. Akhir kata, semoga buku ini dapat memberikan manfaat bagi para pembaca.

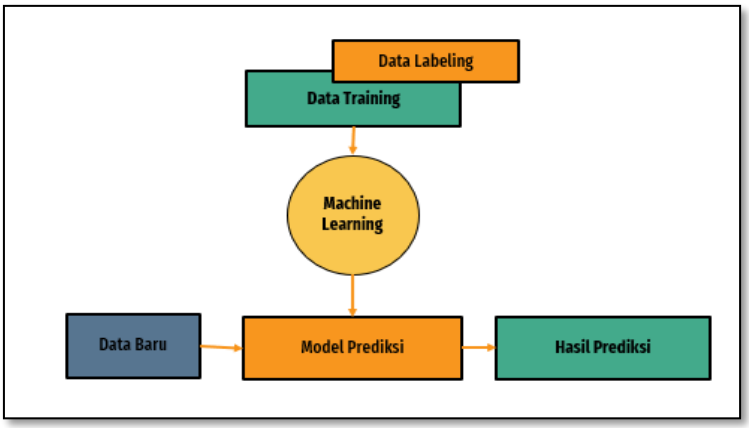
Tim Penulis

DAFTAR ISI

BAB I
DASAR PYTHON UNTUK MACHINE LEARNING

Machine Learning mengacu pada penerapan pemodelan statistik dan pengetahuan pengembangan algoritma pada sistem komputer, sehingga sistem komputer dapat melakukan aktivitas seperti analisis data dan penarikan kesimpulan yang diperlukan tanpa instruksi eksplisit.

Sistem komputer memeriksa data historis dalam jumlah besar dan mencari tren menggunakan algoritma pembelajaran mesin. Hal ini memungkinkan penentuan hasil yang lebih akurat tergantung pada kumpulan input yang diberikan.



Gambar 1. Sistem Kerja Machine Learning

Python merupakan bahasa pemrograman untuk *general-purpose programming* yang termasuk dalam kategori *high-level programming language*. Saat ini, Python merupakan salah satu bahasa pemrograman yang banyak digunakan di berbagai bidang, mulai dari

pengembangan web hingga data science. Keunggulan Python, terletak pada kesederhanaannya dan mudah dipelajari, fleksibilitas, komunitas yang besar, library yang lengkap dan platform independen.

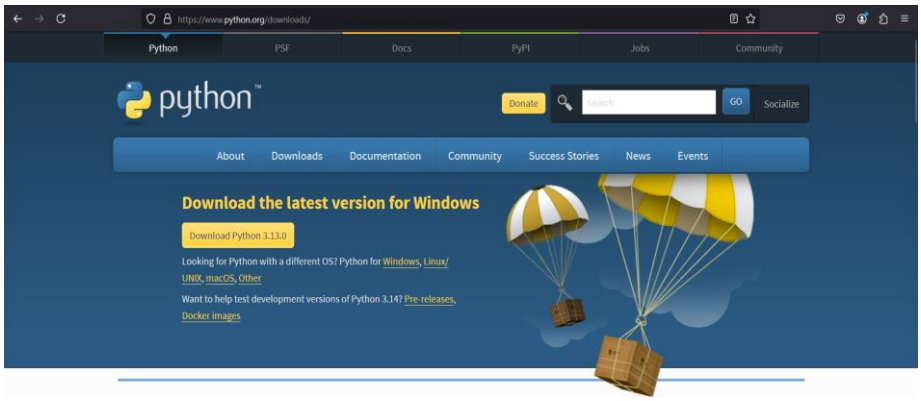
Python dikembangkan oleh Guido Van Rossum dan dirilis pada tahun 1991. Python dapat digunakan untuk mengembangkan aplikasi web (server), membuat aplikasi perangkat lunak, menyederhanakan penalaran matematis, membuat sistem skrip, dan pemrograman mikrokontroler (MicroPython).

Pada Python terdapat library yang dapat di pergunakan untuk *Machine Learning* seperti :

1. Numerical Python(NumPy)
2. Matplotlib
3. Pandas
4. OpenCV
5. OS
6. IO
7. Pillow
8. Ultralytics
9. Dsb

A. Instalasi Python

Sebelum memulai berbagai latihan dalam buku ini, pastikan interpreter Python telah terinstal. Python adalah bahasa pemrograman interpretatif yang populer, di mana kode program dieksekusi baris demi baris secara langsung oleh interpreter. Python dapat diunduh secara gratis dari situs resminya: https://python.org/downloads.



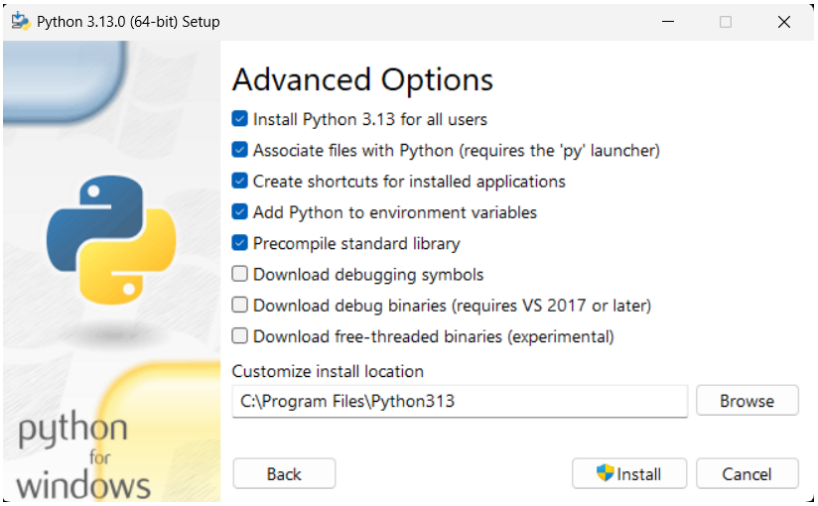
Gambar 2. Halaman Download Python

Pilih versi Python yang sesuai dengan sistem operasi. Perhatikan apakah ingin menginstal versi 32-bit atau 64-bit. Jika sistem operasi pada perangkat menggunakan windows 7, maka dapat memilih instalasi versi 3.8.7.

Klik tombol "Download" untuk memulai proses pengunduhan. Setelah unduhan selesai, cari file installer Python yang baru saja diunduh (biasanya berekstensi .exe untuk Windows). Klik dua kali pada file tersebut untuk memulai proses instalasi.

Gambar 3. Proses Instalasi Python

Klik tombol “Customize Installation”, maka akan muncul form pengaturan seperti gambar berikut :



Gambar 4. Konfigurasi Installer

Klik “Python 3.13 for all users” dan “Add Python to Environment Variables”. Selanjutnya klik tombol “Install”. Setelah selesai, jalankan Command Promp, ketikan perintah:

```
python --version
```

Gambar 5. Python Berhasil diinstall

B. Numerical Python (NumPy)

NumPy merupakan singkatan dari Numerical Python. NumPy merupakan salah satu library pada Python yang mendukung penggunaan array (termasuk array multidimensi) serta menyediakan koleksi fungsi matematika untuk mengoperasikan array tersebut (Harris et al., 2020).

Dalam dunia data science dan machine learning, NumPy telah menjadi pustaka yang tak tergantikan. Kemampuannya dalam memanipulasi array numerik secara cepat dan efisien menjadikan NumPy sebagai "tulang punggung" dalam berbagai tugas, mulai dari persiapan data hingga implementasi algoritma pembelajaran mesin.

Gambar 6. NumPy

Tata cara penggunaan Numerical Python :

- Cara install Numerical Python

```
pip install numpy
```

- Cara pemanggilan / import dari Numerical Python

  Numerical Python kita umpamakan menggunakan (as) untuk mempermudah dalam pemanggilan.

```
import numpy as np
```

Best Practice

Misalkan terdapat sebuah data stok dan data barang keluar yang telah dibuat dalam bentuk array

```
stok = np.array([30,60,50.5,90,70])
brg_keluar = np.array([6,7,11,8.5,7])
```

Data yang sudah di buat dalam bentuk array tersebut, dapat dilakukan perhitungan, misalnya untuk mengetahui sisa stok dapat digunakan perintah berikut :

```
sisa_stok = stok -brg_keluar
sisa_stok
```

Menghasilkan output sebagai berikut :

```
Out[4]: array([24. , 53. , 39.5, 81.5, 63. ])
```

- Bentuk Matriks / Array

Cara membuat dan merubah bentuk array matriks/array sangat mudah:

```
a = np.arange(30)   # Buat numpy array
print(a)
```

Menghasilkan output sebagai berikut:

[0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29]

Jika menggunakan perintah **shape** maka formatnya adalah seperti ini:

```
a.shape = (5,6)
print(a)
```

Menghasilkan output sebagai berikut:

```
[[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]
 [12 13 14 15 16 17]
 [18 19 20 21 22 23]]
```

Fungsi **ravel()** berguna untuk untuk menjadikan matriks dalam bentuk(ukuran) apapun menjadi *array* satu dimensi. Fungsi ravel() pada NumPy digunakan untuk meratakan atau mengubah bentuk (shape) dari suatu array multidimensi (seperti matriks) menjadi sebuah array satu dimensi. Dengan kata lain, ravel() akan menggabungkan semua elemen dalam array multidimensi menjadi satu baris panjang.

```
print(a.ravel())
```

Menghasilkan output sebagai berikut:

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29]
```

Fungsi fungsi **resize()** sama dengan kata kunci **shape** yang membedakan hanyalah cara penulisannya saja. Fungsi resize() dan atribut shape pada array NumPy memiliki tujuan yang berbeda, meskipun keduanya berkaitan dengan dimensi dari sebuah array. resize() digunakan untuk mengubah ukuran sebuah array secara langsung, baik memperbesar maupun memperkecil. Sementara itu, shape adalah atribut yang memberikan informasi tentang dimensi saat ini dari sebuah array,

dan dapat digunakan untuk mengubah dimensi tanpa mengubah data yang sudah ada.

```
a.resize(3,10)
print(a)
```

Menghasilkan output sebagai berikut:

[[0 1 2 3 4 5 6 7 8 9]
[10 11 12 13 14 15 16 17 18 19]
[20 21 22 23 24 25 26 27 28 29]]

Perbedaan utama antara keduanya terletak pada:

- Modifikasi data: resize() dapat menyebabkan data hilang atau diulang jika ukuran baru tidak sesuai dengan ukuran asli. Sedangkan, mengubah shape tidak akan mengubah data yang sudah ada, melainkan hanya mengubah cara menginterpretasikan data tersebut dalam dimensi yang berbeda.**
- Kembalian: resize() umumnya mengubah array asli secara *in-place*, artinya array asli akan dimodifikasi langsung. Sedangkan, mengubah shape tidak mengubah array asli, tetapi mengembalikan sebuah view dari array tersebut dengan dimensi yang baru.

Dalam library NumPy, terdapat operasi transpose yang digunakan untuk mengubah orientasi elemen-elemen dalam sebuah matriks. Jika kita mentransposekan matriks A, maka baris-baris pada A akan menjadi kolom-kolom pada matriks hasil transposenya, dan

sebaliknya. Operasi ini seringkali digunakan dalam berbagai perhitungan matriks, seperti dalam aljabar linear dan pemrosesan data.

Untuk melakukan transpose pada sebuah matriks di NumPy, kita dapat menggunakan atribut .T. Misalnya, jika kita memiliki matriks A, maka transposenya dapat dihitung dengan A.T

```
print(a.T)
```

Menghasilkan output sebagai berikut:

[[0 3 6 9 12 15 18 21 24 27]
[1 4 7 10 13 16 19 22 25 28]
[2 5 8 11 14 17 20 23 26 29]]

- Menggabungkan array/matriks

Buat matriks baru **b**:

```
b = np.array(([1,2],[3,4]))
print(b)
```

Buat matriks baru **c**:

```
c = np.array(([5,6],[7,8]))
print(c)
```

Dalam Library NumPy terdapat funsgi vstack yang merupakan kepanjangan dari 'vertical stack', yang berarti tumpukan vertikal. Vstack bukanlah sebuah tuple, melainkan sebuah fungsi yang disediakan oleh pustaka NumPy dalam bahasa pemrograman Python. Fungsi vstack()

digunakan untuk menggabungkan atau menumpuk beberapa array secara vertikal menjadi satu array baru.

Array-array yang akan digabungkan menggunakan vstack() harus memiliki jumlah kolom yang sama. Hasil dari operasi vstack() adalah sebuah array baru yang memiliki jumlah baris sama dengan jumlah total baris dari semua array input, dan jumlah kolom sama dengan jumlah kolom dari salah satu array input.

Contoh penggunaan vstack :

```
print(np.vstack((b,c)))
```

Menghasilkan output sebagai berikut:

```
[[1 2]
 [3 4]
 [5 6]
 [7 8]]
```

Selain vstack, NumPy juga menyediakan fungsi hstack yang merupakan singkatan dari *horizontal stack*. Fungsi ini digunakan untuk menggabungkan atau menumpuk beberapa array atau matriks secara horizontal menjadi satu array baru. Hasil penggabungan ini akan membentuk sebuah array yang lebih lebar dengan jumlah baris yang sama dengan array aslinya.

```
print(np.hstack((b,c)))
```

Menghasilkan output sebagai berikut:

```
[[1 2 5 6]
 [3 4 7 8]]
```

Perbandingan dengan vstack:

- vstack digunakan untuk menggabungkan array secara vertikal (ke bawah), sehingga menghasilkan array yang lebih tinggi dengan jumlah kolom yang sama.
- hstack digunakan untuk menggabungkan array secara horizontal (ke samping), sehingga menghasilkan array yang lebih lebar dengan jumlah baris yang sama.

Kapan menggunakan hstack:

- Ketika ingin menggabungkan beberapa fitur atau atribut dari data yang memiliki dimensi yang sama (misalnya, menggabungkan data numerik dan kategorikal).
- Ketika ingin membuat matriks yang lebih besar dengan menambahkan kolom-kolom baru.

Fungsi column_stack pada NumPy digunakan untuk menggabungkan beberapa array satu dimensi secara horizontal menjadi sebuah matriks dua dimensi. Setiap array satu dimensi yang digabungkan akan menjadi satu kolom pada matriks yang dihasilkan. Dan merupakan salah satu pustaka utama untuk komputasi numerik dalam Python.

Menggabungkan secara horizontal**:** Artinya, array-array satu dimensi yang kita miliki akan disusun secara berdampingan (samping menyamping) untuk membentuk sebuah matriks yang lebih besar.

Menjadi satu kolom**:** Setiap array satu dimensi yang kita gabungkan akan menempati satu kolom pada matriks hasil gabungan. Ini berarti, jika kita memiliki 3 buah array satu dimensi, maka matriks yang dihasilkan akan memiliki 3 kolom.

Contoh penggunaan column_stack:

```
d = np.array(([1,2,3]))
e = np.array(([4,5,6]))
f = np.array(([7,8,9]))
print("Array d :", d)
print("Array e :", e)
print("Array f :", f)
```

Array d : [1 2 3]

Array e : [4 5 6]

Array f : [7 8 9]

Jika menggunakan fungsi column_stack dari NumPy untuk menggabungkan ketiga array tersebut secara kolom, maka akan dihasilkan sebuah matriks baru. Setiap array yang kita gabungkan akan menjadi satu kolom pada matriks yang baru ini. Jika hasilnya dapat dilihat pada matriks berikut:

```
print(np.column_stack((d, e, f)))
```

Menghasilkan output sebagai berikut:

[[1 4 7]

[2 5 8]

[3 6 9]]

Selanjutnya proses pembagian sebuah array atau matriks menjadi beberapa array atau matriks berukuran lebih kecil. Proses ini sering disebut *partitioning* atau *splitting* dan umumnya melibatkan pembuatan array atau matriks baru untuk menyimpan hasil pembagian.

Buat matrik baru terlebih dahulu

```
g = np.arange(16)*2
h = g.reshape(4,4)
print("Array g :", g,"\n")
print("Matriks h :\n", h)
```

Array g : [0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30]

Menghasilkan output sebagai berikut:

Matriks h :

[[0 2 4 6]

[8 10 12 14]

[16 18 20 22]

[24 26 28 30]]

Proses pemisahan data (splitting) adalah teknik untuk membagi sebuah kumpulan data yang besar (seperti array atau matriks) menjadi potongan-potongan data yang lebih kecil. Potongan-potongan data yang dihasilkan ini kemudian disimpan dalam struktur data yang terpisah, seperti array atau matriks baru.

```
print(np.split(g, 2), "\n")
print(np.split(h, 2))
```

Proses pembagian ini bertujuan untuk partisi sebuah array atau matriks sehingga menghasilkan sub-array atau sub-matriks dengan ukuran yang sama.

array([0, 2, 4, 6, 8, 10, 12, 14]), array([16, 18, 20, 22, 24, 26, 28, 30])]
[array([[0, 2, 4, 6],

 [8, 10, 12, 14]]), array([[16, 18, 20, 22],
 [24, 26, 28, 30]])]

Apabila kita memisah dengan jumlah yang tidak sama :

```
print(np.split(g, 3))
```

Menghasilkan output sebagai berikut:

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~/.local/lib/python3.6/site-packages/numpy/lib/shape_base.py in split(ary, indices_or_sections, axis)
    842     try:
--> 843         len(indices_or_sections)
    844     except TypeError:

TypeError: object of type 'int' has no len()

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
<ipython-input-18-3cec0be61a82> in <module>
----> 1 print(np.split(g, 3))

~/.local/lib/python3.6/site-packages/numpy/lib/shape_base.py in split(ary, indices_or_sections, axis)
    847         if N % sections:
    848             raise ValueError(
--> 849                 'array split does not result in an equal division')
    850     res = array_split(ary, indices_or_sections, axis)
    851     return res

ValueError: array split does not result in an equal division
```

Fungsi hsplit digunakan untuk membagi sebuah array atau matriks menjadi beberapa bagian yang terpisah secara horizontal, sehingga setiap bagian akan menjadi baris baru, Dengan menggunakan fungsi ini, kita dapat memisahkan data dalam suatu array atau matriks

menjadi beberapa baris yang lebih kecil, berdasarkan kriteria pemisahan tertentu.

Dalam Python, fungsi np.hsplit() dari library NumPy umumnya digunakan untuk membagi sebuah array secara horizontal. Fungsi ini menerima array sebagai input dan mengembalikan tuple yang berisi array-array hasil pembagian

```
print(np.hsplit(g,4))
```

Menghasilkan output sebagai berikut:

[array([0, 2, 4, 6]), array([8, 10, 12, 14]), array([16, 18, 20, 22]), array([24, 26, 28, 30])]

```
print(np.hsplit(h,2))
```

Menghasilkan output sebagai berikut:

[array([[0, 2],
[8, 10],
[16, 18],
[24, 26]]), array([[4, 6],
[12, 14],
[20, 22],
[28, 30]])]

Fungsi vsplit dalam pemrosesan data, khususnya dalam operasi matriks, digunakan untuk membagi sebuah matriks menjadi beberapa sub-matriks yang lebih kecil secara vertikal. Proses pembagian ini dilakukan dengan memisahkan baris-baris dari matriks asal menjadi

matriks-matriks baru yang memiliki jumlah kolom yang sama dengan matriks asal, namun dengan jumlah baris yang berbeda sesuai dengan parameter yang ditentukan. Fungsi vsplit adalah alat yang sangat berguna dalam manipulasi matriks. Dengan memahami cara kerjanya, kita dapat melakukan berbagai operasi pada data yang disusun dalam bentuk matriks dengan lebih efisien.

Contoh penggunaan vspilt :

```
print(np.vsplit(h, 2))
```

Menghasilkan output sebagai berikut:

```
[array([[ 0,  2,  4,  6],
        [ 8, 10, 12, 14]]), array([[16, 18, 20, 22],
        [24, 26, 28, 30]])]
```

Fungsi vsplit tidak dapat diterapkan pada array satu dimensi karena vsplit dirancang khusus untuk membagi array multidimensi (dua dimensi atau lebih) menjadi beberapa sub-array secara vertikal.

```
print(np.vsplit(g,4))
```

Menghasilkan output sebagai berikut:

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-22-d480b9e79103> in <module>
----> 1 print(np.vsplit(g,4))

~/.local/lib/python3.6/site-packages/numpy/lib/shape_base.py in vsplit(ary, indices_or_sections)
    969     """
    970     if _nx.ndim(ary) < 2:
--> 971         raise ValueError('vsplit only works on arrays of 2 or more dimensions')
    972     return split(ary, indices_or_sections, 0)
    973

ValueError: vsplit only works on arrays of 2 or more dimensions
```

Fungsi array_split dan split sama-sama digunakan untuk membagi data menjadi bagian-bagian yang lebih kecil. Namun, terdapat perbedaan signifikan dalam cara keduanya bekerja:

- Fungsi split: Membagi sebuah string atau list menjadi beberapa sub-string atau sub-list berdasarkan pemisah yang ditentukan. Jumlah elemen pada setiap sub-bagian akan sama, kecuali jika ada pemisah berulang di akhir data.
- Fungsi array_split: Khusus digunakan untuk membagi array atau matriks menjadi beberapa sub-array atau sub-matriks. Fleksibilitasnya terletak pada kemampuan untuk menentukan jumlah sub-bagian yang diinginkan, sehingga ukuran setiap sub-bagian tidak harus sama.

Kapan menggunakan array_split:

- Saat Anda ingin membagi array atau matriks menjadi potongan-potongan dengan ukuran yang tidak harus sama.
- Ketika Anda ingin membagi data berdasarkan kriteria tertentu yang tidak hanya bergantung pada pemisah.
- Jika Anda bekerja dengan data numerik atau array multi-dimensi.

Contoh penggunaan array_split:

- Membagi array data menjadi bagian-bagian yang sama besar, kecuali bagian terakhir yang mungkin lebih kecil.
- Membagi matriks menjadi beberapa baris atau kolom dengan jumlah yang tidak sama.
- Membagi data menjadi kelompok-kelompok berdasarkan nilai tertentu.

```
print(np.array_split(g, 5))
```

Menghasilkan output sebagai berikut:

[array([0, 2, 4, 6]), array([8, 10, 12]), array([14, 16, 18]), array([20, 22, 24]), array([26, 28, 30])]

C. Matplotlib

Matplotlib adalah library yang digunakan dalam bahasa pemrograman Python untuk menciptakan visualisasi data dengan grafis yang menarik dan informatif (Rougier et al., 2021). Matplotlib Python muncul sebagai jembatan antara angka mentah dan pemahaman manusia, mengubah statistik yang awalnya rumit menjadi visualisasi menarik dan mudah dipahami.

Gambar 7. Matplotlib

Matplotlib menawarkan “palet” yang kaya akan grafik, diagram, dan visualisasi interaktif, membuat data terasa lebih “hidup”.

- Cara install Matplotlib

```
1 pip install matplotlib
```

- Cara pemanggilan / import Matplotlib
  Matplotlib kita umpamakan menggunakan (as) untuk mempermudah dalam pemanggilan.

```
1 import matplotlib.pyplot as plt
```

Berikut adalah contoh penggunaan dari Matplotlib dalam bentuk penyajian grafik seperti :

1. Grafik Plot
2. Grafik Scatter
3. Grafik Bar

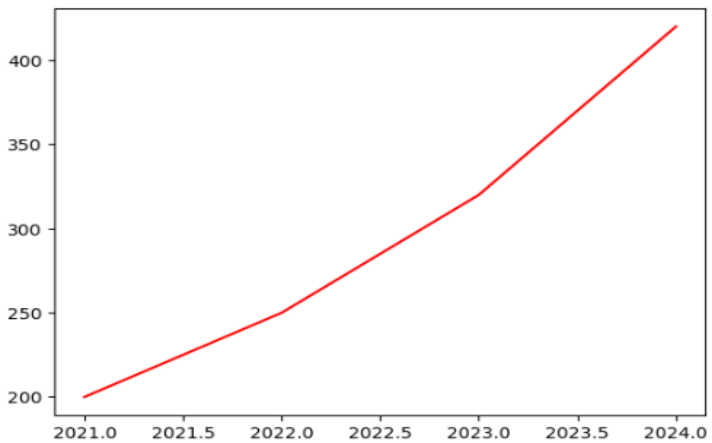
STUDI KASUS : PENYAJIAN GRAFIK PLOT

Terdapat sebuah data tahun dan pencapaian dalam bentuk list, kemudian akan dibuatkan grafik dalam betuk plot.

```
1 tahun = [2021, 2022, 2023, 2024]
2 Pencapaian = [200, 250, 320, 420]
3 plt.plot(tahun,Pencapaian, 'r')
```

Output yang di hasilkan adalah sebagai berikut :

```
Out[6]: [<matplotlib.lines.Line2D at 0x1b7885b8fd0>]
```



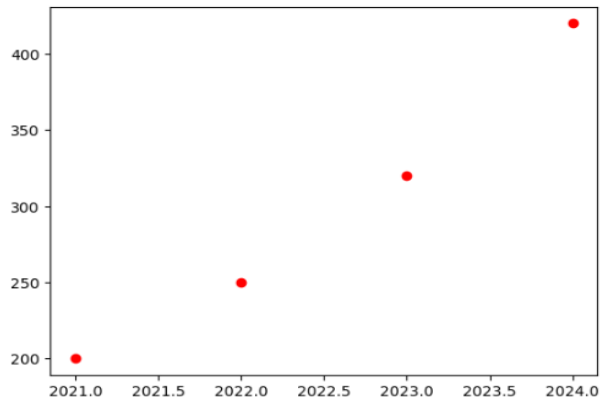
STUDI KASUS : PENYAJIAN GRAFIK SCATTER

Terdapat sebuah data tahun dan pencapaian dalam bentuk list, kemudian akan dibuatkan grafik dalam betuk plot.

```
1 tahun = [2021, 2022, 2023, 2024]
2 Pencapaian = [200, 250, 320, 420]
3 plt.scatter(tahun,Pencapaian, c='r')
```

Output yang di hasilkan adalah sebagai berikut :

```
Out[7]: <matplotlib.collections.PathCollection at 0x1b7888fc290>
```



STUDI KASUS : PENYAJIAN GRAFIK BAR

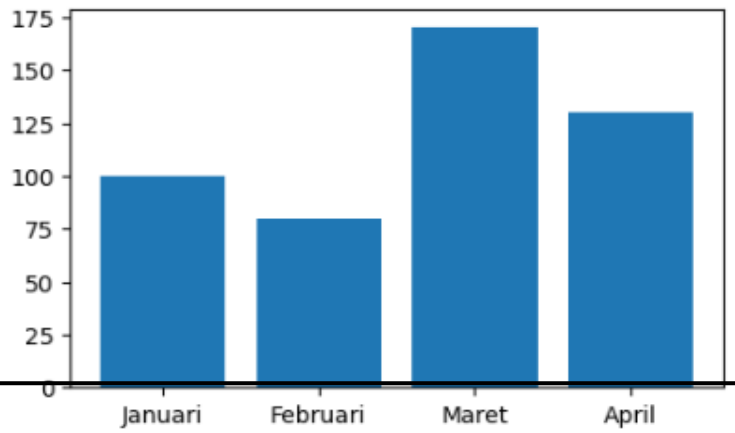
Terdapat dua data yaitu data bulan dan pendapatan dalam betuk list data

Bulan = ['Januari','Februari','Maret','April']

Pendapatan = [100,80,170,130] data akan di buat kedalam grafik bar.

```
1 Bulan = ['Januari','Februari','Maret','April']
2 Pendapatan = [100,80,170,130]
3 plt.figure(figsize=(5,3), dpi=100)
4 bars =plt.bar(Bulan, Pendapatan)
5 plt.show()
```

Output yang di hasilkan sebagai berikut :



D. Pandas

Pandas merupakan salah satu library dalam python yang berguna untuk melakukan analisis dan pengolahan data dari menengah sampai besar. Pandas memiliki dua struktur data, yaitu series dan dataframe (McKinney, 2022).

Pandas python memiliki beberapa keunggulan seperti :

1. Dapat digunakan secara bersamaan dengan library lain dalam data science,
2. Dapat dijalankan menggunakan berbagai text editor, namun lebih disarankan untuk menggunakan Jupyter Notebook.

- Cara install

```
pip install pandas
```

- Cara pemanggilan / import Pandas

  Pandas kita umpamakan menggunakan (as) untuk mempermudah dalam pemanggilan.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

- Upload data CSV

  Pada contoh kali ini kita akan menggunakan data yang sebelumnya kita buat di excel dengan ekstensen *.CSV. Contoh data sebagai berikut:

Setelah data CSV di buat, kemudian kita upload pada folder yang sama dengan file source code nya. Seperti berikut :

- Pemanggilan CSV pada Jupyter Notebook

```
1 d_stok = pd.read_csv('BahanPokok.csv', index_col=0)
```

- Untuk menampilkan data bisa dengan memanggil nama variablenya.

```
1 d_stok
```

Hasil output sebagai berikut :

Out[9]:

	stok	penjualan	sisa_stok
Beras	120	20	100
Minyak	80	10	70
Garam	68	10	58
Kecap	80	50	30
Kacang	30	12	18

STUDI KASUS

CONTOH 1. Penggunaan loc (berdasarkan nama kolom) dapat di contohkan dengan memanggil salah satu kolom.
Misal akan menampilkan berdasarkan kolom “stok”, maka perintah yang diketikan :

```
X = d_stok['stok']
X
```

Hasil Output sebagai berikut :

```
Out[11]: Beras     120
         Minyak     80
         Garam      68
         Kecap      80
         Kacang     30
         Name: stok, dtype: int64
```

Berikutnya akan menampilkan data barang yaitu “beras” :

```
d_stok.loc['Beras'] #stok barang
```

Hasil output sebagai berikut :

```
Out[13]: stok         120
         penjualan     20
         sisa_stok    100
         Name: Beras, dtype: int64
```

CONTOH 2. Penggunaan loc (berdasarkan nama index).

Misal tampilkan data kecap, minyak dan kacang, maka perintah yang diketikan :

```
d_stok.loc[["Kecap", "Minyak", "Kacang"]]
```

Hasil Output yang di peroleh sebagai berikut :

Out[17]:

	stok	penjualan	sisa_stok
Kecap	80	50	30
Minyak	80	10	70
Kacang	30	12	18

CONTOH 3. Menampilkan "Kecap", "Minyak", "Kacang" berdasarkan "stok","sisa_stok", maka perintah yang diketikan :

```
1 d_stok.loc[["Kecap", "Minyak", "Kacang"],["stok","sisa_stok"]]
```

Hasil Output yang di peroleh sebagai berikut :

Out[18]:

	stok	sisa_stok
Kecap	80	30
Minyak	80	70
Kacang	30	18

CONTOH 4. Penggunaan iloc (berdasarkan indeks kolom dan/atau index). Akan di tampilkan data “minyak”, dimana minyak akan berada pada baris ke-1, maka bisa melakukan pemanggilan seperti berikut :

```
1 d_stok.iloc[1]
```

Hasil Output sebagai berikut :

```
Out[19]: stok         80
         penjualan    10
         sisa_stok    70
         Name: Minyak, dtype: int64
```

Kemudian kita akan menampilkan data pada urutan 1,2 dan 3, maka perintah yang diketikan adalah sebagai berikut :

```
1 d_stok.iloc[[1,2,3]]
```

Hasil Output yang di peroleh sebagai berikut:

Out[20]:

	stok	penjualan	sisa_stok
Minyak	80	10	70
Garam	68	10	58
Kecap	80	50	30

dengan menggunakan Numerical Python, akan dilakukan perhitungan data stok "penjualan" yang lebih dari 5 dan data stok "penjualan" yang kurang dari 30

```
1 d_stok[np.logical_and(d_stok["penjualan"] > 5, d_stok["penjualan"] < 30)]
```

Hasil Output yang di peroleh sebagai berikut:

Out[22]:

	stok	penjualan	sisa_stok
Beras	120	20	100
Minyak	80	10	70
Garam	68	10	58
Kacang	30	12	18

Contoh lain penggunaan pandas, adalah sebagai berikut:

1. Menghapus kolom, misalnya “nama_variabel_turunan” dari objek panda df_aps:

```
del df_aps["nama_variabel_turunan"]
```

2. Menghapus kolom sementara, menggunakan perintah drop, parameter axis: bernilai 0 atau index apabila ingin menghapus label dari indeks; bernilai 1 apabila ingin menghapus kolom.

```
df_aps.drop("nama_turunan_tahun", axis=1).head()
```

3. Menghapus baris, bisa menggunakan perintah drop berdasarkan indeks.

```
df_aps.drop([0, 1])
```

4. Menghapus baris berdasarkan nilai di sebuah kolom:

```
df_aps[df_aps.nama_item_vertical_variabel != 'SUMATERA BARAT']
```

5. Mengubah nama kolom.

File display

```
df_aps = df_aps.rename(columns = {"data_content": "nilai",
                                  "nama_item_vertical_variabel": "provinsi"})
```

a. OpenCV

OpenCV (*Open Source Computer Vision*) merupakan sebuah pustaka bahasa pemograman yang ditujukan untuk penggunaan pengolahan citra secara waktu nyata. Secara original, OpenCV dikembangkan oleh Intel, kemudian didukung oleh Willow Garage dan kemudian Itseez (yang diakuisisi oleh Intel).

Gambar 8. OpenCV

Pustaka OpenCV sudah mendukung lintas platform dan dapat digunakan dengan secara bebas di bawah lisensi BSD Open-source. Dalam perkembangannnya, OpenCV sudah mendukung beberapa framework untuk deep-learning seperti TensorFlow, YOLO, Torch/PyTorch dan Caffee.

- Cara install OpenCV

```
pip install opencv-python
```

- Cara pemanggilan / import OpenCV

  Pemanggilan OpenCV bisa langsung menyebutkan cv2 atau bisa juga menggunakan alias (as)

```
import cv2
```

STUDI KASUS

Ambil gambar yang akan dijadikan sample (pada contoh studi kasus nama file gambar yang akan dijadikan sample : maskoki.jpg). Gambar yang akan dijadikan sample di simpan pada folder yang sama dengan file source coding kemudian nama yang di panggil harus sama dengan nama yang berada pada folder.

Misalkan nama gambar yang dijadikan sample pada folder adalah maskoki.jpg, maka pemanggilannya harus sesuai dengan nama tersebut.

```
ikan = cv2.imread('maskoki.jpg')
```

Kemudian cek type gambar apakah sudah benar dengan mengetikan source coding berikut :

```
1 type(ikan)
```

Bisa juga mengecek ukuran dari gambar

```
1 ikan.shape
```

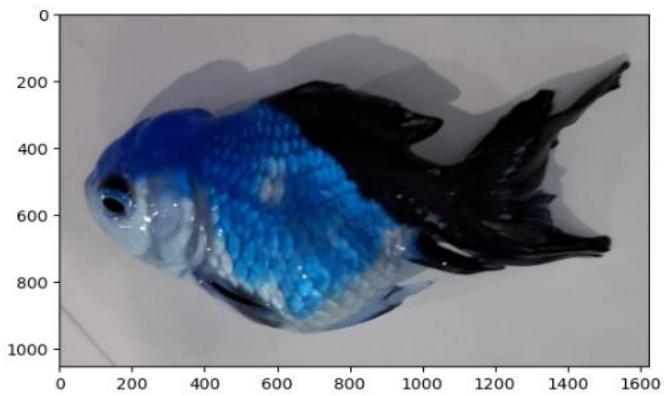
```
Out[9]: (1053, 1625, 3)
```

Untuk melihat gambar yang telah di baca, dapat di tampilkan dengan grafik matplotlib.

```
1 import matplotlib.pyplot as plt
2 plt.imshow(ikan)
```

Maka, akan tampil seperti berikut :

```
Out[10]: <matplotlib.image.AxesImage at 0x1fa22ceddd0>
```

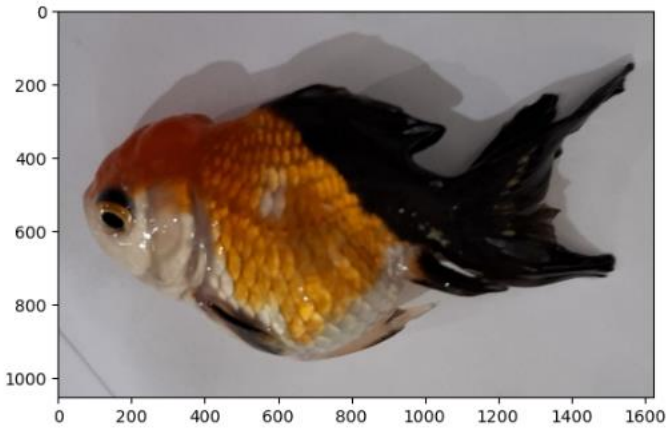


Untuk melihat gambar nyata dari gambar, rubah gambar matplotlib dari bgr opencv ke rgb matplotlib.

```
1 gbrRGB=cv2.cvtColor(ikan,cv2.COLOR_BGR2RGB)
2 plt.imshow(gbrRGB)
```

Hasil Output yang di peroleh sebagai berikut :

```
Out[11]: <matplotlib.image.AxesImage at 0x1fa22d4d150>
```



a. OS

Library OS merupakan bagian dari modul Python standar yang digunakan untuk berinteraksi dengan sistem operasi. Library ini juga menyediakan berbagai fungsi untuk bekerja dengan file dan direktori, seperti navigasi direktori, pembacaan dan penulisan file, dan manajemen lingkungan kerja. Library OS tidak perlu diunduh karena library ini bagian dari standar Python.

```
[1]: import os
```

Pada gambar di atas, kita mengimpor modul OS yang menyediakan fungsi untuk berinteraksi dengan sistem operasi.

```
[2]: current_directory = os.getcwd()
     print(f"Direktori kerja saat ini: {current_directory}")

     Direktori kerja saat ini: C:\Users\rahman\Documents\Notebook
```

Selanjutnya, ***current_directory*** disini menampung nilai dari ***os.getcwd()***, dimana berfungsi untuk mendapatkan path direktori kerja yang sedang aktif. Terakhir, ***print*** mencetak string "***Direktori kerja saat ini…***".

E. IO

Library ini menyediakan alat untuk bekerja dengan input dan output berbasis stream. Sebagai contoh, io.StringIO dan io.BytesIO digunakan untuk menangani data teks atau byte dalam memori seperti halnya file. Ini berguna ketika Anda ingin mengelola input/output secara langsung dalam memori tanpa menyimpan ke dalam file fisik. Sama seperti library OS, tidak perlu diunduh, karena library ini bagian dari standar Python.

```
[1]: import io
```

Pada gambar di atas, kita mengimpor modul IO, yang berisi kelas StringIO untuk operasi file-like berbasis string.

```
[2]: string_io = io.StringIO()
```

Pada baris ini, ***string_io*** berfungsi untuk menampung nilai dari objek ***io.StringIO()***, sehingga nantinya teks dapat ditulis dan dibaca dari objek tersebut.

```
[3]: string_io.write("Halo! Ini adalah contoh StringIO.")

[3]: 33
```

Kemudian, baris ini menulis teks ke dalam objek StringIO. Angka ***33*** disini merupakan output ketika kita menjalankan baris tersebut, karena

metode ***write()*** dari objek ***StringIO*** mengembalikan jumlah karakter yang berhasil ditulis ke dalam buffer, bukan isi dari teks yang ditulis.

```
[4]: string_io.seek(0)

[4]: 0
```

Setelah itu, baris ini mengatur posisi baca/tulis kembali ke awal (posisi ke-0) agar teks yang ditulis dapat dibaca dari awal.

```
[5]: print("Isi dari StringIO:")
     print(string_io.read())

     Isi dari StringIO:
     Halo! Ini adalah contoh StringIO.
```

Terakhir, baris ini berfungsi untuk membaca dan mencetak seluruh isi ***StringIO*** yang telah disimpan sebelumnya.

F. Pillow

Library ini merupakan modul untuk manipulasi gambar yang memungkinkan berbagai operasi, seperti membuka, memodifikasi, dan menyimpan gambar dalam berbagai format (JPEG, PNG, BMP, dll.). Image digunakan untuk memuat dan memanipulasi gambar, ImageFont untuk menambahkan font khusus ke gambar, dan ImageDraw untuk menggambar bentuk atau teks pada gambar.

- Cara installasi Pillow

```
# Cara install Pillow
pip install pillow
```

```
[1]: from PIL import Image, ImageFont, ImageDraw
```

Pada gambar di atas, kita mengimpor kelas ***Image***, ***ImageFont***, dan ***ImageDraw*** dari modul ***Pillow***, yang digunakan untuk manipulasi gambar.

```
[2]: img = Image.new('RGB', (200, 100), color='white')
```

Pada baris ini, gambar baru berukuran ***200x100*** piksel dibuat dengan mode warna ***RGB*** dan latar belakang putih, yang disimpan dalam variabel ***img***.

```
[3]: draw = ImageDraw.Draw(img)
```

Selanjutnya, baris ini membuat objek ***ImageDraw*** yang memungkinkan kita menggambar pada gambar ***img***.

```
[4]: font = ImageFont.load_default()
```

Kemudian, baris ini memuat font bawaan, yang akan digunakan untuk teks pada gambar.

```
[5]: text = "Hello, Jupyter Notebook!"
     draw.text((10, 40), text, font=font, fill="black")
```

Variabel text disini akan menyimpan tulisan "***Hello, Jupyter Notebook!***", kemudian akan ditambahkan ke gambar, diletakkan pada koordinat (10, 40), dengan warna tulisan hitam.

```
[6]: img.show()
```

Terakhir, baris diatas berfungsi untuk membuka dan menampilkan gambar yang dihasilkan.

G. Ultralytics

Library ini digunakan untuk mengimplementasikan model-model object detection yang sudah dilatih sebelumnya. Library yang kita gunakan ini, saat ini mendukung dua jenis model, yakni YOLO dan RTDETR.

- Cara Installasi Ultralutics dengan mengetikan kode program berikut :

```
# Cara install Ultralytics
pip install ultralytics
```

- Kode Program untuk mengimpor kelas ***YOLO***, dan ***RTDETR*** dari modul ***Ultralytics***, yang digunakan untuk mendeteksi objek nantinya.

```
[1]: from ultralytics import YOLO, RTDETR
```

BAB 2
EDGE DETECTION & IMAGE SEGMENTATION

A. Deteksi Tepi

Deteksi tepi (*Edge Detection*) pada suatu citra adalah suatu proses yang menghasilkan tepi-tepi dari obyek citra, tujuannya adalah untuk menandai bagian yang menjadi detail citra dan memperbaiki detail dari citra yang kabur, yang terjadi karena error atau adanya efek dari proses akuisisi citra.

1. Deteksi Tepi Laplacian of Gaussian

- Langkah pertama import terlebih dahulu Open CV, Numpy dan Matplotlib

```
1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
```

- Membaca gambar yang sebelumnya telah di upload pada folder yang sama dengan source coding.

```
1 ikan1 = cv2.imread('ikankecil.png',0)
2 ikan2 = cv2.imread('ikankecilcrop.png',0)
```

- Lakukan oprasi deteksi tepi dengan metode Laplacian of Gaussian

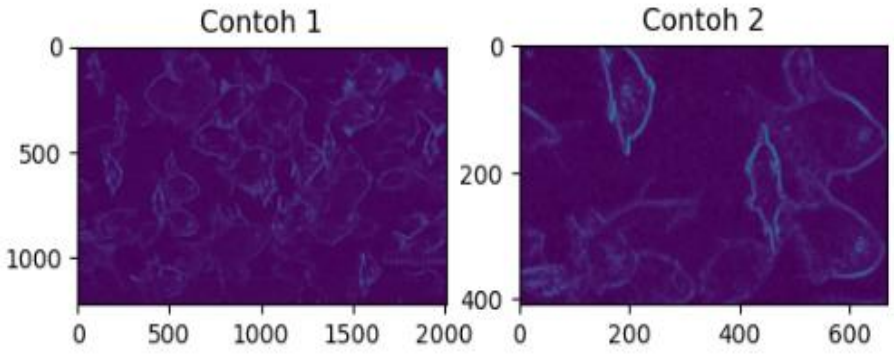
```
def laplacian_of_gaussian(img):
    img_laplace_blur = cv2.GaussianBlur(img.copy(), (3,3), 0)
    img_laplace = cv2.Laplacian(img_laplace_blur, cv2.CV_16S, ksize=3)
    img_laplace_abs = cv2.convertScaleAbs(img_laplace)
    return img_laplace_abs
```

```
ikan1_log = laplacian_of_gaussian(ikan1)
ikan2_log = laplacian_of_gaussian(ikan2)
```

- Menampilkan data :

```
plt.subplot(121);plt.imshow(ikan1_log[...,::-1]);plt.title("Contoh 1");
plt.subplot(122);plt.imshow(ikan2_log[...,::-1]);plt.title("Contoh 2");
```

Hasil Output yang di peroleh sebagai berikut :



2. Deteksi Tepi Canny

- Langkah pertama importkan library yang akan di gunakan :

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from scipy.signal import convolve2d
from scipy.ndimage import median_filter
import matplotlib.gridspec as gridspec
```

- Membaca gambar yang sebelumnya telah di upload pada folder yang sama dengan source coding:

```
img_ikan = cv2.imread('ikankecilcrop.png')
```

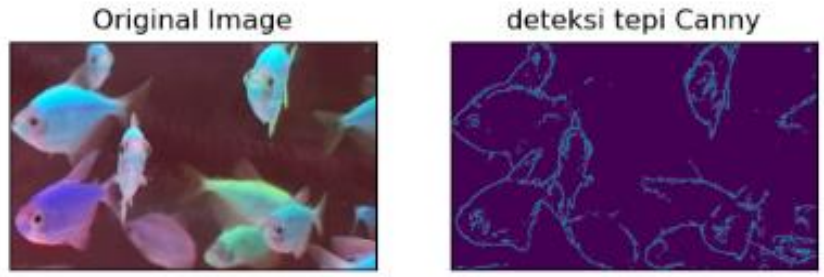
- Menerapkan algoritma deteksi tepi Canny pada gambar, menggunakan kode program berikut :

```
edges = cv2.Canny(image=img_ikan, threshold1=127, threshold2=127)
```

- Menampilkan gambar asli dan hasil deteksi tepi canny

```
plt.subplot(121),plt.imshow(img_ikan)
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(edges)
plt.title('deteksi tepi Canny'), plt.xticks([]), plt.yticks([])
plt.show()
```

Output Hasil Deteksi Tepi Canny :



3. Deteksi Tepi Canny Dengan Nilai Median

- Hitung nilai pixel median

```
med_val = np.median(img_ikan)
```

- Batas bawah adalah 0 atau 70% dari nilai median, mana saja yang lebih tinggi

```
lower = int(max(0, 0.7* med_val))
```

- Batas atas adalah 255 atau 30% di atas nilai median, mana saja yang lebih rendah

```
upper = int(min(255,1.3 * med_val))
```

- Menerapkan algoritma deteksi tepi Canny dengan nilai median

```
edges = cv2.Canny(image=img_ikan, threshold1=lower , threshold2=upper)
```

- Menampilkan gambar asli dan hasil deteksi tepi canny dengan nilai batas median

```
plt.subplot(121),plt.imshow(img_ikan)
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(edges)
plt.title('deteksi tepi Canny dengan batas median'), plt.xticks([]), plt.yticks([])
plt.show()
```

Output Yang Di Hasilkan untuk Deteksi Tepi Canny dengan Nilai Batas Median :

4. Deteksi Tepi dengan Threshold

Threshold atau ambang batas dalam konteks deteksi tepi adalah nilai numerik yang digunakan untuk membedakan antara tepi yang "kuat" dan "lemah".

Pixel dengan nilai gradien (perubahan intensitas) di atas ambang batas dianggap sebagai tepi, sedangkan yang di bawahnya dianggap bukan tepi.

```
edges = cv2.Canny(image=img_ikan, threshold1=0, threshold2=255)
```

```
# menampilkan gambar asli dan hasil deteksi tepi canny dengan nilai treshold
plt.subplot(121),plt.imshow(img_ikan)
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(edges)
plt.title('deteksi tepi Canny dengan treshold'), plt.xticks([]), plt.yticks([])
plt.show()
```

5. Deteksi Tepi Sobel

Tinjau pengaturan pixel disekitar pixel (x,y)

$$\begin{bmatrix} A0 & a1 & a2 \\ a7 & (x,y) & a3 \\ a6 & a5 & a4 \end{bmatrix}$$

Operator sobel adalah magnitude dari gradien yang dihitung dengan :

$$M = \sqrt{sx^2 + sy^2} \quad \text{atau } M = |sx| + |sy|$$

Turunan parsial dihitung dengan :

sx = (a2+ca3+a4) – (a0+ca7+a6)

sy = (a0+ca1+a2) – (a6+ca5+a4)

Dengan konstanta c=2 dalam bentuk mask, sx dan sy dinyatakan sebagai

$$Sx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad Sy = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

- Contoh citra yang akan dilakukan pendeteksian tepi dengan operator sobel :

3	4	2	5	7
2	1	6	4	2
3	5	7	2	4
4	2	5	7	1
2	5	1	6	9

citra awal

*	*	*	*	*
*	18			*

citra hasil konvolusi

$$Sx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad Sy = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

sx = 3x(-1)+2x(-2)+ 3x(-1)+2x(1)+6x(2)+7x(1) = 11

sy = 3x(1)+4x(2)+2x(1)+3x(-1)+5x(-2)+7x(-1) = 7

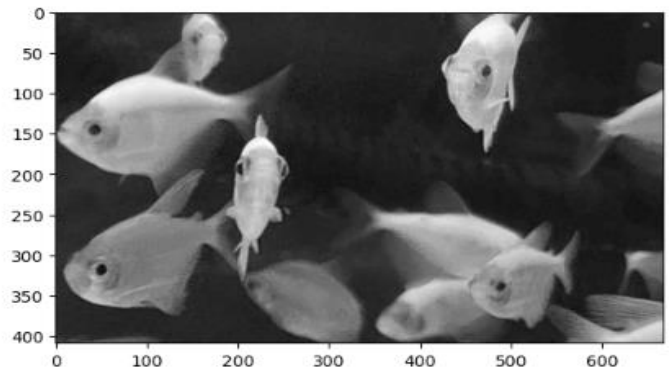
maka M = 18

- Membaca gambar yang sebelumnya telah di upload pada folder yang sama dengan source coding.

```
image = cv2.imread('ikankecilcrop.png',0)
plt.imshow(image,cmap='gray')
```

Output yang di hasilkan :

```
<matplotlib.image.AxesImage at 0x22a05e471d0>
```



- Matrik array sobel_x, sobel_y, dan laplacian merupakan representasi dari kernel (filter) yang digunakan dalam operasi

konvolusi pada citra untuk deteksi tepi. Masing-masing kernel ini memiliki karakteristik dan tujuan yang berbeda dalam mendeteksi tepi.

```
sobel_x = np.array([[-1,0,1],
                    [-2,0,1],
                    [-1,0,1]])

sobel_y = np.array([[-1,-2,-1],
                    [0,0,0],
                    [1,2,1]])

laplacian = np.array([[1,1,1],
                      [1,-8,1],
                      [1,1,1]])
```

Penjelasan kernel di atas :

Sobel x, didesain untuk mendeteksi tepi vertikal. Nilai-nilai negatif di sebelah kiri dan positif di sebelah kanan pada baris tengah mengindikasikan bahwa perubahan intensitas yang signifikan secara horizontal akan menghasilkan nilai gradien yang tinggi.

Sobel y, didesain untuk mendeteksi tepi horizontal. Nilai-nilai negatif di atas dan positif di bawah pada kolom tengah mengindikasikan bahwa perubahan intensitas yang signifikan secara vertikal akan menghasilkan nilai gradien yang tinggi.

Pasangan Sobel, dengan menggabungkan hasil dari Sobel x dan Sobel y, kita dapat menghitung magnitudo gradien dan arah gradien, yang memberikan informasi lebih lengkap tentang tepi.

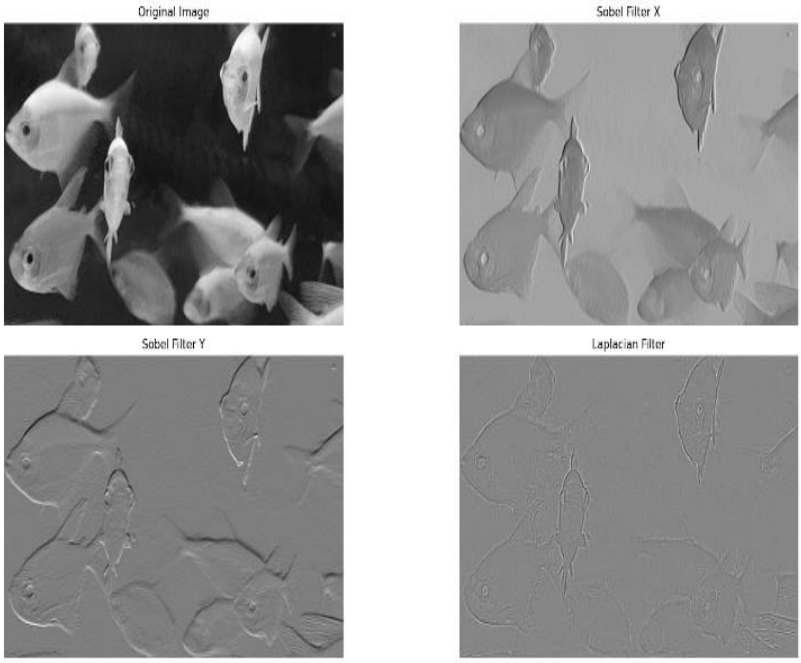
- Menerapkan operasi konvolusi pada gambar untuk semua setiap filter

```
sobel_x_filtered_img = convolve2d(image, sobel_x, mode='same', boundary='symm')
sobel_y_filtered_img = convolve2d(image, sobel_y, mode='same', boundary='symm')
laplacian_filtered_img = convolve2d(image, laplacian, mode='same', boundary='symm')
```

- Terapkan untuk semua gambar

```
plt.figure(figsize=(20,20))
gs = gridspec.GridSpec(4, 2)
gs.update(wspace=0.025, hspace=0.1)

fig = plt.subplot(gs[0])
plt.imshow(image, cmap='gray')
plt.title("Original Image")
plt.axis('off')

fig = plt.subplot(gs[1])
plt.imshow(sobel_x_filtered_img, cmap='gray')
plt.title("Sobel Filter X")
plt.axis('off')

fig = plt.subplot(gs[2])
plt.imshow(sobel_y_filtered_img, cmap='gray')
plt.title("Sobel Filter Y")
plt.axis('off')

fig = plt.subplot(gs[3])
plt.imshow(laplacian_filtered_img, cmap='gray')
plt.title("Laplacian Filter")
plt.axis('off')

plt.show()
```

Output hasil dari deteksi tepi sobel sebagai berikut :



6. Deteksi Tepi Lapasian

- Import terlebih dahulu library yang akan digunakan

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

- Baca gambar yang sebelumnya telah di upload pada folder yang sama dengan file coding.

```
GbrIkan = cv2.imread("ikankecil.png")
```

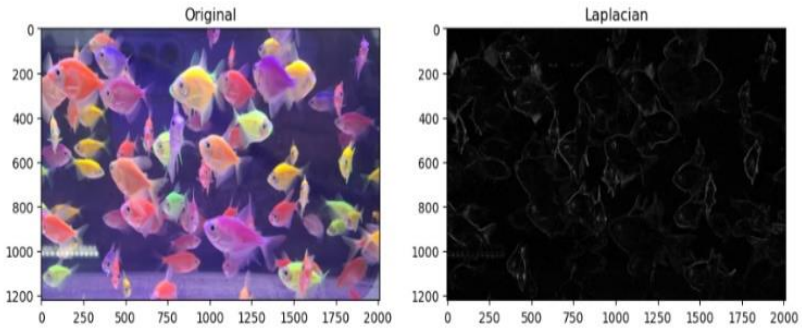
- Menghilangkan noise dengan memburamkan dengan filter Gaussian

```
GbrIkan = cv2.GaussianBlur(GbrIkan, (3, 3), 0)
```

- Lakukan deteksi tepi dengan metode lapasian :

```
GbrIkan_gray = cv2.cvtColor(GbrIkan, cv2.COLOR_BGR2GRAY)
Gbr_Lp = cv2.Laplacian(GbrIkan_gray, cv2.CV_16S, ksize=3)
Hasil = cv2.convertScaleAbs(Gbr_Lp)


plt.figure(figsize=[12,7])
plt.subplot(121);plt.imshow(GbrIkan[...,::-1]);plt.title("Original");
plt.subplot(122);plt.imshow(Hasil, cmap='gray');plt.title("Laplacian");
```

Matplotlib menampilkan gambar berdasarkan baris dan column seperti pada coding di atas subplot (121) artinya adalah satu baris dua kolom. Keluaran yang di peroleh sebagai berikut:



B. Segmentasi Citra

Segmentasi membagi citra ke dalam sejumlah region atau obyek, level untuk pembagian tergantung pada masalah yang diselesaikan.Proses segmentasi berhenti ketika obyek yang diinginkan dalam aplikasi telah terisolasi.

- Mempersiapkan library yang akan digunakan

```
import sys
import numpy as np
import skimage.color
import skimage.filters
import skimage.io
import cv2
import numpy as np
import matplotlib.pyplot as plt
from skimage.io import imread, imshow
from skimage.color import rgb2gray, rgb2hsv
from skimage.morphology import area_opening
from skimage.exposure import histogram
from skimage.filters import threshold_otsu
```

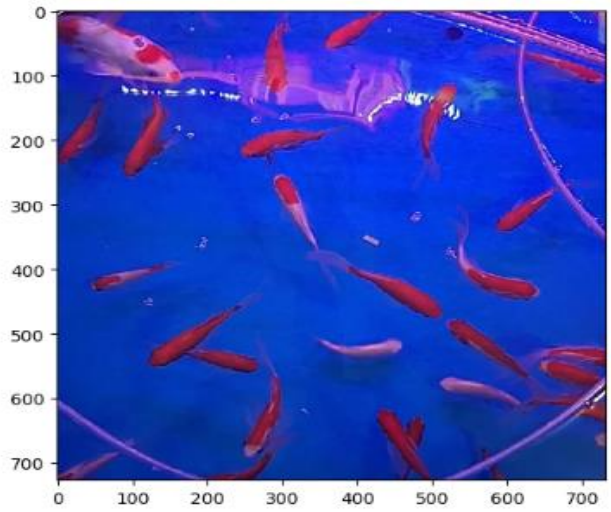
- Membaca gambar yang sebelumnya telah di upload pada folder yang sama dengan file coding

```
ikan = cv2.imread("ikanbanyak.jpg",0)
```

- Menampilkan gambar asli

```
ikan1 = plt.imread('ikanbanyak.jpg')
imshow(ikan1);
```

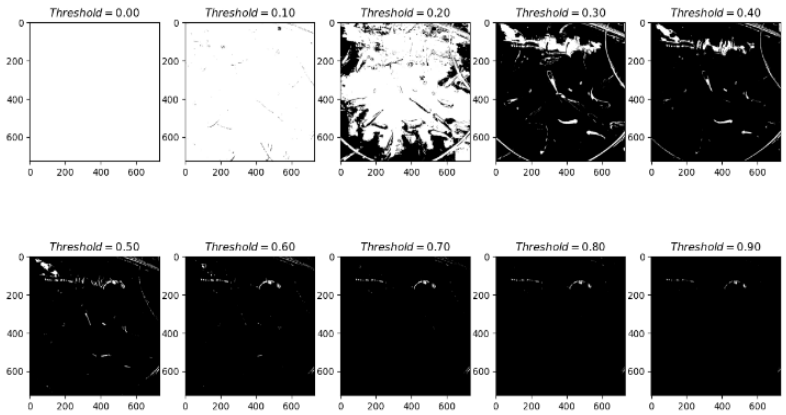
Output yang di hasilkan :



- Memvisualisasikan efek nilai ambang batas yang berbeda pada gambar skala abu-abu. Untuk beberapa gambar biner dengan ambang batas yang berbeda-beda, dapat diperoleh bagaimana thresholding dapat memengaruhi gambar yang dihasilkan dan memilih ambang batas yang optimal untuk aplikasi yang lebih spesifik.

```
th_values = np.linspace(0, 1, 11)
fig, axis = plt.subplots(2, 5, figsize=(15,8))
ikan_gray = rgb2gray(ikan1)
for th, ax in zip(th_values, axis.flatten()):

    ikan_binarized = ikan_gray < th
    ax.imshow(ikan_binarized,cmap='binary')
    ax.set_title('$Threshold = %.2f$' % th)
```

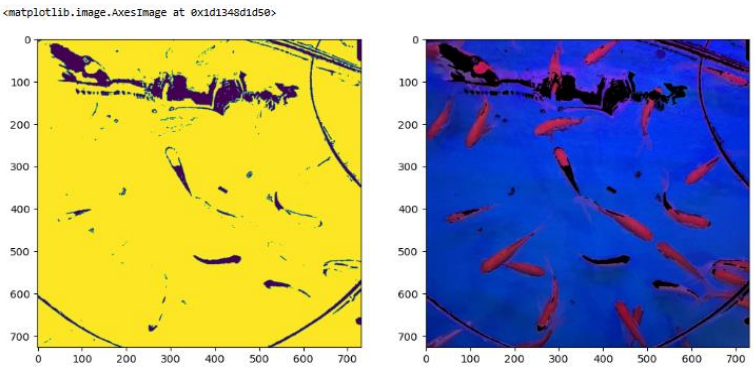
Menghasilkan output sebagai berikut :



- Menerapkan thresholding Otsu untuk membuat mask biner yang menyoroti objek latar depan dalam gambar. Kemudian, fungsi masked_image digunakan untuk mengisolasi objek latar depan dengan mengalikan gambar asli dengan mask biner.

```
def masked_image(image, mask):
    r = image[:,:,0] * mask
    g = image[:,:,1] * mask
    b = image[:,:,2] * mask
    return np.dstack([r,g,b])
fig, ax = plt.subplots(1, 2, figsize=(12,6))
thresh = threshold_otsu(ikan_gray)
ikan_otsu  = ikan_gray < thresh
ax[0].imshow(ikan_otsu)
filtered = masked_image(ikan1, ikan_otsu)
ax[1].imshow(filtered)
```

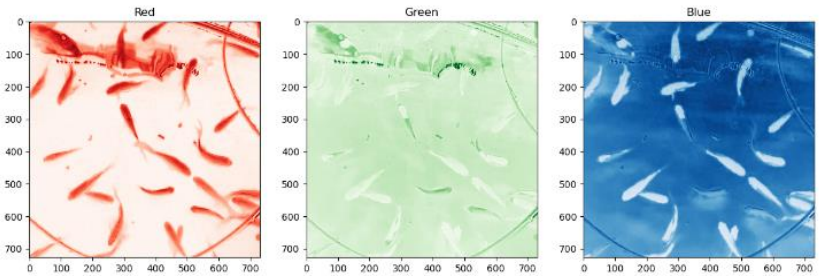
Output yang di hasilkan dari perhitungan otsu di atas adalah sebagai berikut :



- Memvisualisasikan masing-masing saluran warna, untuk memperoleh komposisi warna gambar dan area potensial untuk pemrosesan atau analisis gambar, yang berfungsi untuk koreksi warna, penyaringan warna, atau pendeteksian objek.

```
fig, ax = plt.subplots(1, 3, figsize=(15,6))
ax[0].imshow(ikan1[:,:,0], cmap='Reds')
ax[0].set_title('Red')
ax[1].imshow(ikan1[:,:,1], cmap='Greens')
ax[1].set_title('Green')
ax[2].imshow(ikan1[:,:,2], cmap='Blues')
ax[2].set_title('Blue');
```

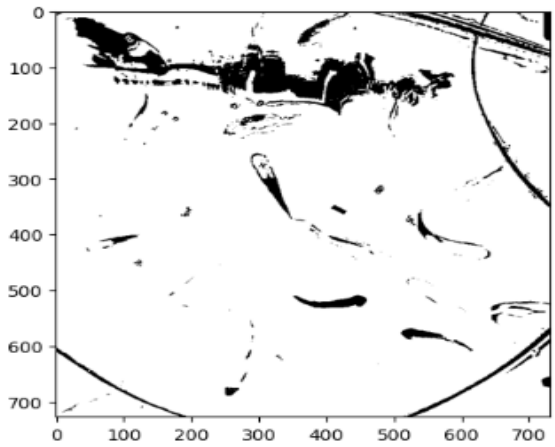
Output yang di hasilkan dari visualisasi di atas adalah sebagai berikut :



- Menerapkan thresholding Otsu untuk secara otomatis menentukan ambang batas optimal untuk binarisasi, dan kemudian menampilkan gambar biner yang dihasilkan.

```
img = cv2.imread('ikanbanyak.jpg',0)
ret,thr = cv2.threshold(img, 0, 255, cv2.THRESH_OTSU)
plt.imshow(thr,cmap='binary')
plt.show()
```

Output yang di hasilkan adalah sebagai berikut :



Segmentasi Citra dengan Morphological Operation

Oprasi morfologi merupakan teknik pengolahan citra berdasarkan bentuk segmen dan region dalam image. Langkah yang dilakukan dalam oprasi ini adalah :

- Import library yang di gunakan

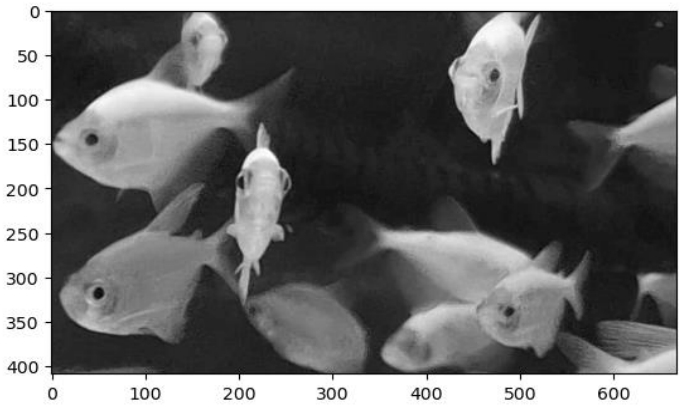
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

- Membaca gambar yang sebelumnya telah di upload pada folder yang sama denga file coding

```
img_ikan = cv2.imread('ikankecilcrop.png',0)
plt.imshow(img_ikan,cmap='gray')
```

Output yang di hasilkan :

<matplotlib.image.AxesImage at 0x141079a2ed0>



- Lakukan oprasi morfologi, dengan terlebih dahulu mengubah gambar menjadi bluring dan buat histogramnya

```
1 img_blur = cv2.medianBlur(img_ikan,5)
2 hist = cv2.calcHist([img_blur],[0],None,[256],[0,256])
3 plt.axis(False)
```

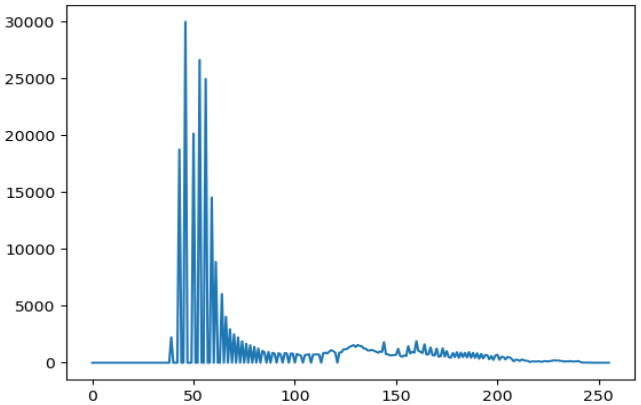
Output yang di hasilkan :

```
Out[4]: (0.0, 1.0, 0.0, 1.0)
```

- Menampilkan histogram dari hasil bluring :

```
1 plt.plot(hist)
2 plt.show()
```

Output yang dihasilkan :



- Lakukan perhitungan threshold , dimana cv2. THRESH_BINARY ini adalah tipe thresholding. Tipe ini menetapkan piksel ke nilai maksimum (255) atau nilai minimum (0) berdasarkan nilai ambang batas.

```
ret, img_tresh = cv2.threshold(img_blur,110,255,cv2.THRESH_BINARY)
plt.axis(False)
plt.imshow(img_tresh,cmap='gray')
```

Output yang di hasilkan adalah sebagai berikut :

```
Out[7]: <matplotlib.image.AxesImage at 0x141079f4e50>
```

- Operasi morfologi mengombinasikan erosi dan pelebaran. Ini digunakan untuk mengisi lubang atau celah kecil pada gambar dan menghilangkan noise kecil yang terisolasi.

```
kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(5,5))
img_close = cv2.morphologyEx(img_tresh,cv2.MORPH_CLOSE, kernel)
plt.axis(False)
plt.imshow(img_close,cmap='gray')
```

Output yang di hasilkan sebagai berikut :

```
Out[8]: <matplotlib.image.AxesImage at 0x141079da8d0>
```

- Erosi adalah operasi morfologi yang mengecilkan batas-batas wilayah putih dalam gambar. Ini digunakan untuk menghilangkan detail kecil atau noise dari gambar.

```
1 kernel = cv2.getStructuringElement(cv2.MORPH_CROSS,(3,8))
2 img_erode = cv2.erode(img_close,kernel,iterations=2)
3 plt.axis(False)
4 plt.imshow(img_erode,cmap='gray')
```

Output yang di hasilkan adalah :

```
Out[12]: <matplotlib.image.AxesImage at 0x141042de2d0>
```

- Filter median adalah teknik pemfilteran non-linear yang menggantikan setiap nilai piksel dengan nilai median piksel tetangganya di dalam kernel. Hal ini membantu mengurangi

noise dan memperhalus gambar sekaligus mempertahankan tepiannya.

```
median_img = cv2.medianBlur(img_erode,7)
plt.axis(False)
plt.imshow(median_img,cmap='gray')
```

Output yang di hasilkan adalah :

```
<matplotlib.image.AxesImage at 0x14107a650d0>
```

- Dilasi adalah operasi morfologi yang memperluas batas-batas wilayah putih dalam gambar. Ini digunakan untuk mengisi celah atau lubang pada gambar, dan juga dapat digunakan untuk menebalkan objek.

```
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(5,5))
img_dilation = cv2.dilate(median_img,kernel,iterations=1)
plt.axis(False)
plt.imshow(img_dilation,cmap='gray')
```

Output yang dihasilkan

```
<matplotlib.image.AxesImage at 0x14107af4e50>
```

- Berikutnya mengubah piksel putih menjadi hitam dan piksel hitam menjadi putih. Ini operasi yang berfungsi untuk meningkatkan kontras, dan menyoroti fitur.

```
1 hasil = ~img_dilation
2 plt.axis(False)
3 plt.imshow(hasil,cmap='gray')
```

Output yang di hasilkan :

```
<matplotlib.image.AxesImage at 0x14107acf950>
```

Segmentasi Citra berdasarkan Region Grow

- Mempersiapkan library yang akan di pergunakan

```
%matplotlib inline
import pywt
import matplotlib.pyplot as plt
import numpy as np
```

- Membaca gambar yang telah di upload pada folder yang sama dengan file coding

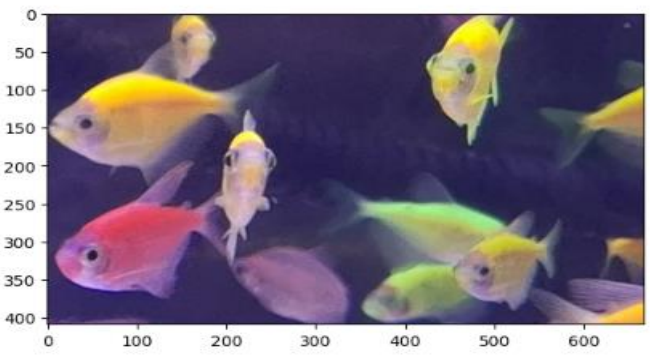
```
ikan = plt.imread("ikankecilcrop.png")
```

- Menampilkan gambar yang akan di lakukan segmentasi

```
plt.imshow(ikan)
```

Hasil Output yang di tampilkan :

```
<matplotlib.image.AxesImage at 0x149badfc650>
```



- Gunakan fungsi “shape” untuk mendapatkan dimensi atau ukuran dari sebuah objek gambar.

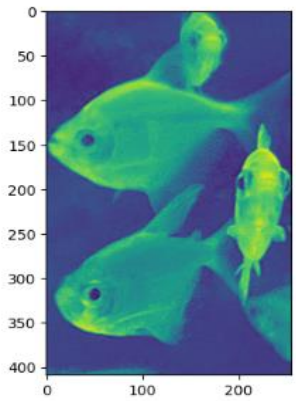
```
ikan.shape
```

- Mengubah gambar berwarna menjadi grayscale dengan menghitung rata-rata intensitas dari ketiga channel warna. Dan menampilkan sebagian dari gambar grayscale: Hanya menampilkan kolom ke-0 hingga ke-254.

```
ikanGray = (ikan[:,:,0]  + ikan[:,:,1] + ikan[:,:,2]) / 3
plt.imshow(ikanGray[:,0:255])
```

Output yang di hasilkan dari oprasi di atas :

```
<matplotlib.image.AxesImage at 0x149bb80e2d0>
```

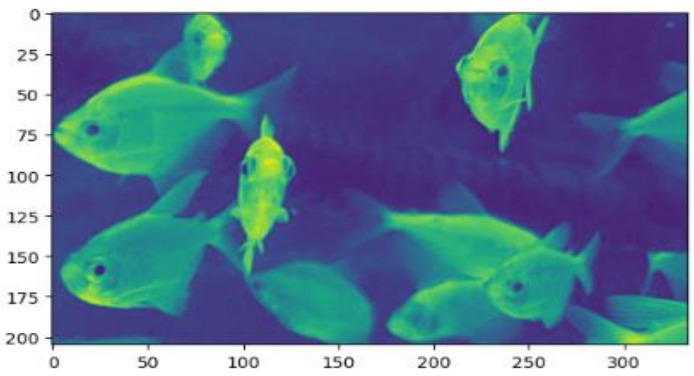


- Lakukan transformasi wavelet Haar pada gambar grayscale kemudian menampilkan koefisien aproksimasi (approximation coefficients) dari hasil transformasi tersebut.

```
coeffs = pywt.dwt2(ikanGray,'haar')
cA, (cH, cV, cD) = coeffs
plt.imshow(cA)
```

Output yang di hasilkan sebagai berikut :

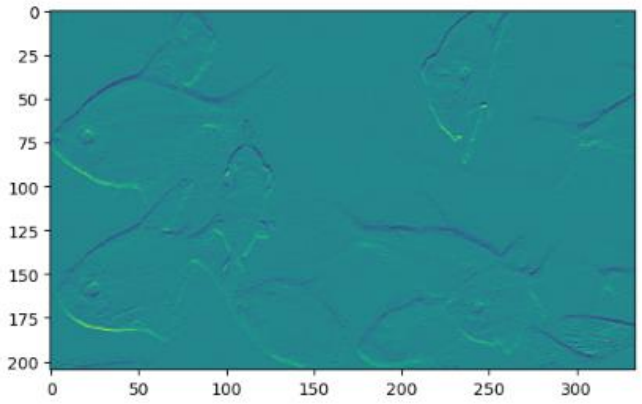
```
<matplotlib.image.AxesImage at 0x149bb87fbd0>
```



- Output dari komponen cH :

```
1 plt.imshow(cH)
```

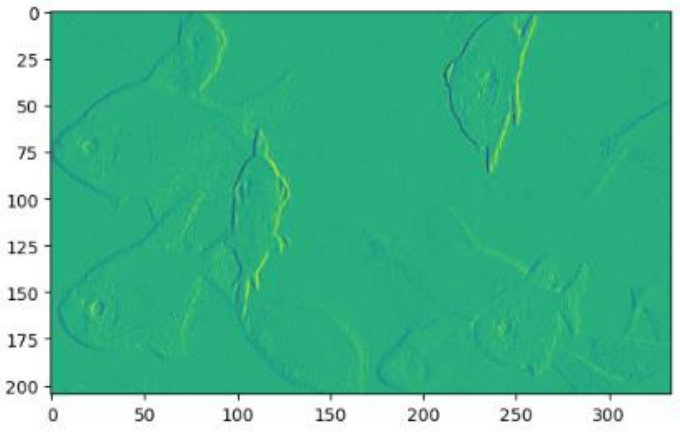
```
<matplotlib.image.AxesImage at 0x149bd1c4cd0>
```



- Output dari komponen cV :

```
1 plt.imshow(cV)
```

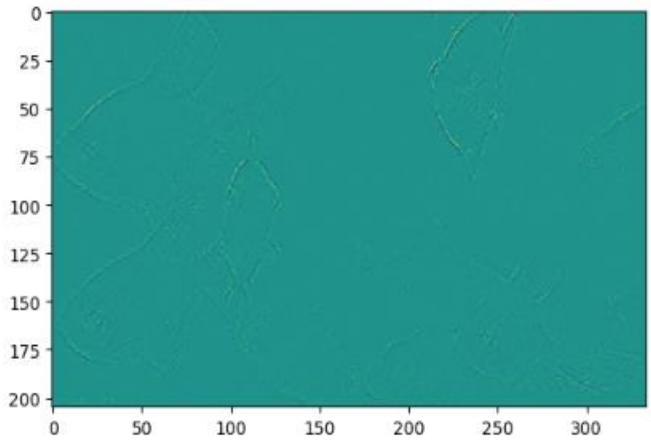
```
<matplotlib.image.AxesImage at 0x149bb871cd0>
```



- Output dari komponen cD :

```
1 plt.imshow(cD)
```

```
<matplotlib.image.AxesImage at 0x149bd282c50>
```



- Fungsi pixelDiff menghitung perbedaan kuadrat antara dua piksel dalam gambar dan membandingkannya dengan ambang batas yang diberikan.

```
1 def pixelDiff(image, ay, ax, by, bx, threshold):
2     (height, width) = image.shape
3 
4     a = image[ay, ax]
5     b = image[by, bx]
6     return np.sum((a-b)**2) <= threshold**2
```

- Mengelompokkan piksel yang memiliki intensitas atau warna yang serupa, membentuk wilayah yang terhubung, yang merupakan teknik mendasar dalam segmentasi gambar.

```
1  def discoverRegion(image, raster, yseed, xseed, threshold):
2      assert(raster[yseed, xseed] == -1)
3      region = np.max(raster) + 1
4      queue = [(yseed, xseed)]
5      raster[yseed,xseed] = region
6  
7      while len(queue) > 0:
8          (y,x) = queue.pop(0)
9  
10         for (p,q) in [(y-1,x),(y+1,x),(y,x-1),(y,x+1)]:
11             if raster[p,q] == -1 and pixelDiff(image, y, x, p, q, threshold):
12                 raster[p,q] = region
13                 queue.append((p,q))
```

- Fungsi fullRegionGrow mengimplementasikan algoritma yang lengkap untuk menyegmentasikan gambar ke dalam beberapa wilayah berdasarkan kemiripan piksel.

```
def fullRegionGrow(image, threshold):
    (height, width) = image.shape
    raster = np.zeros((height, width))-1

    # Write a board of -2 around the image.
    for y in range(height):
        raster[y,width-1] = -2
        raster[y,0] = -2
    for x in range(width):
        raster[height-1,x] = -2
        raster[0,x] = -2

    # Begin
    for y in range(height):
        for x in range(width):
            if raster[y,x] == -1:
                discoverRegion(image, raster, y, x, threshold)
    return raster
```

- Melakukan segmentasi pada gambar grayscale dengan menggunakan algoritma region growing dengan ambang batas 4/256.

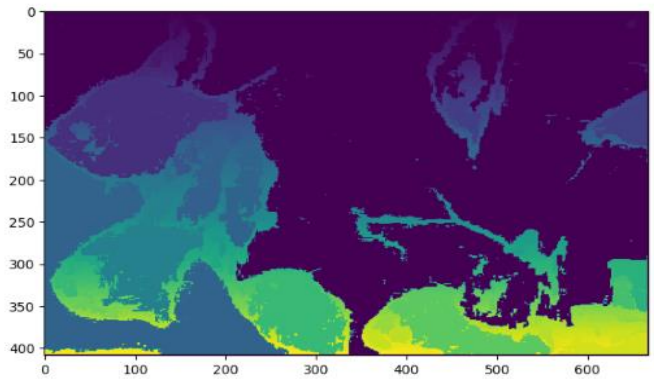
```
raster = fullRegionGrow(ikanGray, 4/256)
```

- Menampilkan hasil dari segmentasi :

```
plt.figure(figsize=(8,8))
plt.imshow(raster)
```

Output Hasil Segmentasi sebagai berikut :

```
<matplotlib.image.AxesImage at 0x149bd71b890>
```



Segmentasi Citra Berdasarkan Area

Segmentasi citra berdasarkan area adalah teknik dalam pengolahan citra dengan tujuan untuk membagi sebuah citra menjadi beberapa wilayah (region). Langkah yang dilakukan adalah :

- Memasukan library yang akan di gunakan

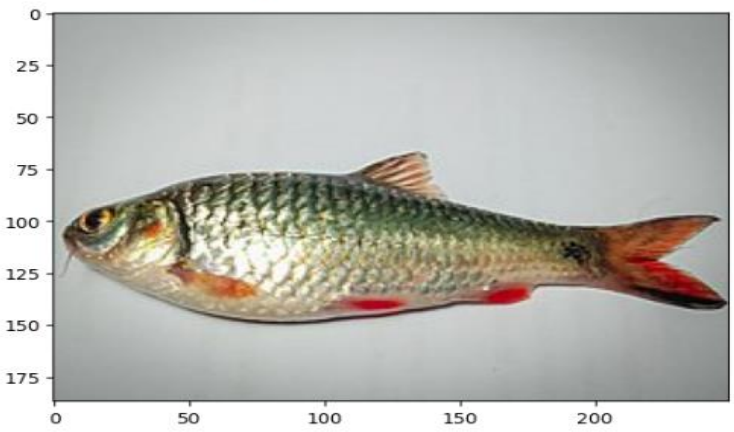
```
%matplotlib inline
import cv2
import numpy as np
from matplotlib import pyplot as plt
```

- Membaca gambar yang sebelumnya telah di upload pada folder yang sama denga file coding

```
img = cv2.imread('ikanbesar.jpg')
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

Output yang dihasilkan adalah sebagai berikut :

```
Out[7]: <matplotlib.image.AxesImage at 0x253c794df10>
```



- Melakukan oprasi segmentasi citra dengan menentukan mendefinisikan area milik objek

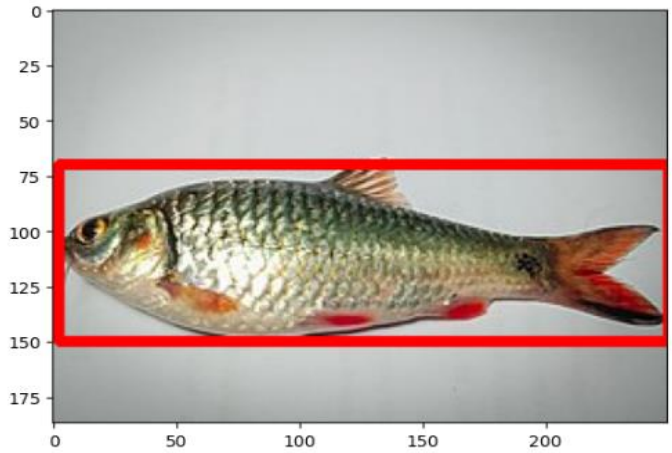
```
RECT = (2,70,250,150)
```

- Memvisualisasikan area yang dipilih. Koordinat persegi panjang ditentukan oleh x1, y1, x2, dan y2, dan warna serta ketebalannya ditentukan.

```
x1, y1, x2, y2 = RECT
tmp = cv2.rectangle(np.copy(img), (x1,y1), (x2,y2), (0,0,255), 3)
plt.imshow(cv2.cvtColor(tmp, cv2.COLOR_BGR2RGB))
```

Output yang di hasilkan sebagai berikut :

```
Out[9]: <matplotlib.image.AxesImage at 0x253c7739690>
```



- Mengalokasikan memori untuk hasil algoritma. Pertama membuat sebuah mask dengan ukuran yang sama dengan gambar asli dan Mask ini akan digunakan untuk menyimpan informasi setiap pixel.

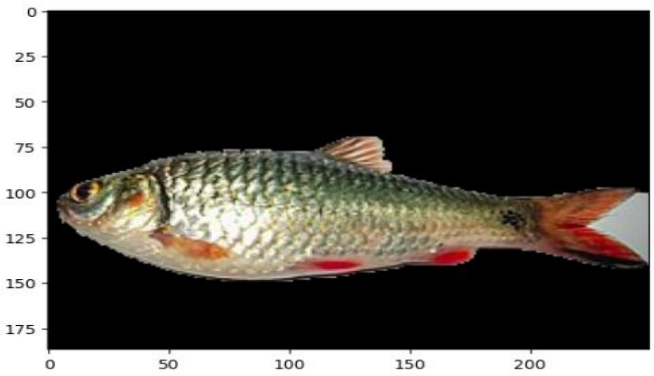
```
mask = np.zeros(img.shape[:2], np.uint8)
bgdModel = np.zeros((1,65), np.float64)
fgdModel = np.zeros((1,65), np.float64)

cv2.grabCut(img, mask, RECT, bgdModel,
            fgdModel, 5, cv2.GC_INIT_WITH_RECT);
```

- Berikutnya mengekstrak objek latar depan dari gambar dengan menetapkan piksel latar belakang menjadi 0 (hitam).

```
mask2 = np.where((mask==0)|(mask==2), 0, 1).astype('uint8')
result = img * mask2[:,:,np.newaxis]
plt.imshow(cv2.cvtColor(result, cv2.COLOR_BGR2RGB))
```

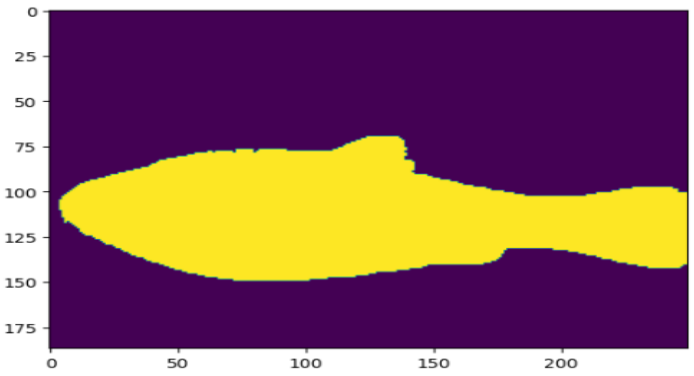
Hasil output seperti berikut :

```
<matplotlib.image.AxesImage at 0x253c7935810>
```



- **Menampilkan hasil akhir**

```
<matplotlib.image.AxesImage at 0x253c787fcd0>
```



BAB 3
IMAGE CONTOUR

Metode pengolahan citra atau gambar yang digunakan untuk mengambil ciri dari suatu gambar agar dapat dibedakan sebelum diklasifikasikan. Salah satu caranya adalah dengan contour gambar.

CONTOH : Contour Gambar

Contoh 1:

- Mempersiapkan library yang akan di gunakan

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
```

- Memanggil gambar dan merubahnya menjadi gambar grey

```
Gambar_ikan = cv2.imread("ikanbesar.jpg")
gray = cv2.cvtColor(Gambar_ikan, cv2.COLOR_BGR2GRAY)
```

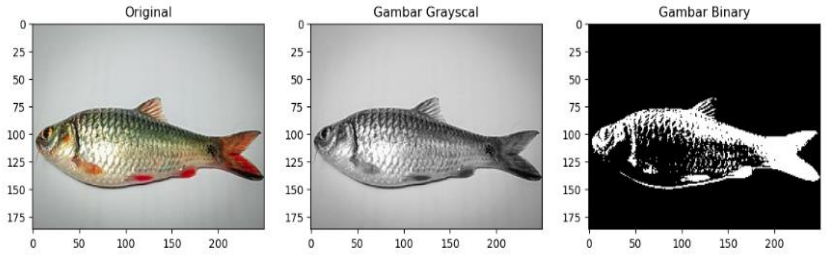
- Merubah gambar menjadi biner

```
_, binary_image = cv2.threshold(gray,127,255,cv2.THRESH_BINARY_INV)
```

- Menampilkan hasil perubahan gambar

```
plt.figure(figsize=[14,8])
plt.subplot(131);plt.imshow(Gambar_ikan[...,::-1]);plt.title("Original");
plt.subplot(132);plt.imshow(gray, cmap='gray');plt.title("Gambar Grayscal");
plt.subplot(133);plt.imshow(binary_image, cmap='gray');plt.title("Gambar Binary");
```

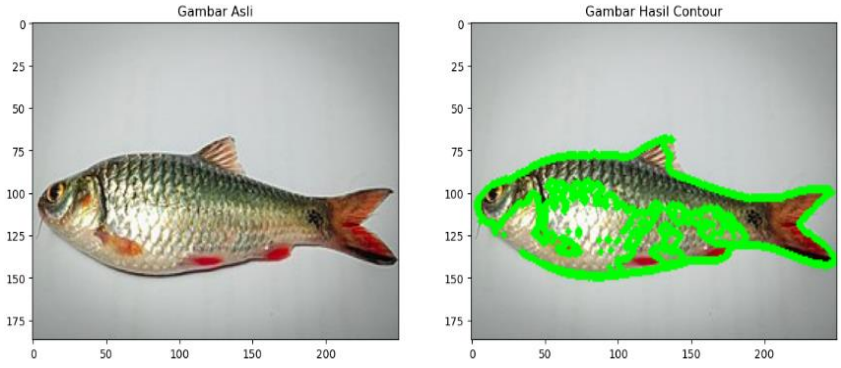
Hasil Output yang di peroleh adalah sebagai berikut :



- Menentukan ukuran kernel dengan penutupan yang merupakan operasi morfologi yang menggabungkan erosi dan pelebaran, yang dapat membantu mengisi lubang kecil atau celah pada objek. Kemudian menampilkan hasil contour gambar.

```
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5), (2, 2))
closing = cv2.morphologyEx(binary_image, cv2.MORPH_CLOSE, kernel)

contours, _ = cv2.findContours(closing, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
all_contours = cv2.drawContours(input_image.copy(), contours, -1, (0,255,0), 3)
plt.figure(figsize=[14,8])

plt.subplot(121);plt.imshow(Gambar_ikan[...,::-1]);plt.title("Gambar Asli");
plt.subplot(122);plt.imshow(all_contours[...,::-1]);plt.title("Gambar Hasil Contour");
```

Hasil Output sebagai berikut :



Contoh 2:

- Mempersiapkan library yang akan di gunakan

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
```

- Membaca gambar yang sebelumnya telah di simpan pada folder yang sama dengan file coding.

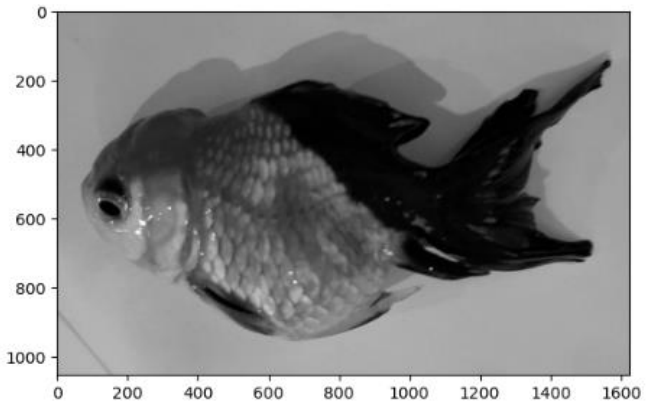
```
I_org = plt.imread('maskoki.jpg')
I = cv2.cvtColor(I_org,cv2.COLOR_RGB2GRAY)
```

- Menampilkan gambar

```
plt.imshow(I,cmap='gray')
```

Output Yang Di Hasilkan :

```
<matplotlib.image.AxesImage at 0x1756ff6ea90>
```



- NumRows, NumCols fungsinya adalah untuk mendapatkan dimensi (ukuran) dari sebuah gambar dan menyimpannya dalam dua variabel

```
numRows , numCols = I.shape[0] , I.shape[1]
```

- Gerakan arah ke pergeseran koordinat yang sesuai. khususnya untuk operasi yang melibatkan lingkungan piksel.

```
directions = {'left':(0,-1),
              'right':(0,1),
              'up':(-1,0),
              'down':(1,0),
              'up_left':(-1,-1),
              'up_right':(-1,1),
              'down_left':(1,-1),
              'down_right':(1,1)}
```

- Larik C yang dihasilkan dapat digunakan untuk berbagai tugas pemrosesan gambar

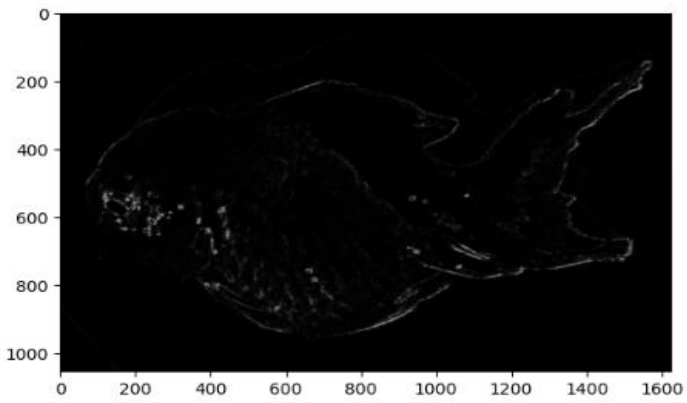
```
C = np.zeros(I.shape)
for i in range(2,numRows-2):
    for j in range(2,numCols-2):
        minSSD = -1
        for d in directions.values():
            u , v = d[0] , d[1]
            P1 = I[i-1:i+2,j-1:j+2]
            P2 = I[i+u-1:i+u+2,j+v-1:j+v+2]
            ssd = np.sum((P1-P2)**2)
            if minSSD == -1:
                minSSD = ssd
            elif ssd < minSSD:
                minSSD = ssd
        C[i,j] = minSSD
```

- Menampilkan gambar

```
plt.imshow(C,cmap='gray')
```

Menghasilkan Output sebagai berikut :

```
<matplotlib.image.AxesImage at 0x17570b38d10>
```

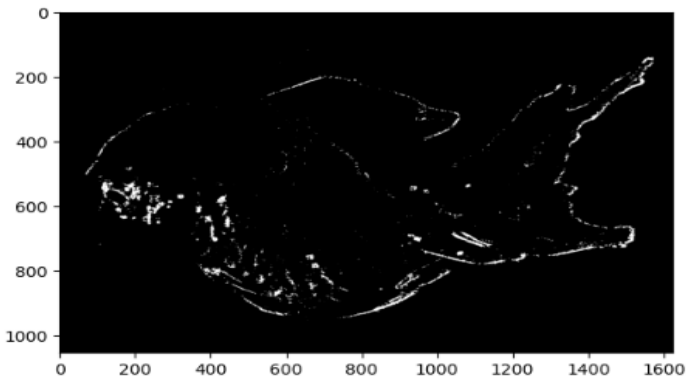


- Ambang batas yang lebih tinggi akan menghasilkan segmentasi yang lebih konservatif, sedangkan ambang batas yang lebih rendah akan menghasilkan segmentasi yang lebih baik.

```
th = C.mean()+3*C.std()
C_image = C>th
plt.imshow(C_image,cmap='gray')
```

Output yang dihasilkan sebagai berikut :

```
<matplotlib.image.AxesImage at 0x17570bdcbd0>
```

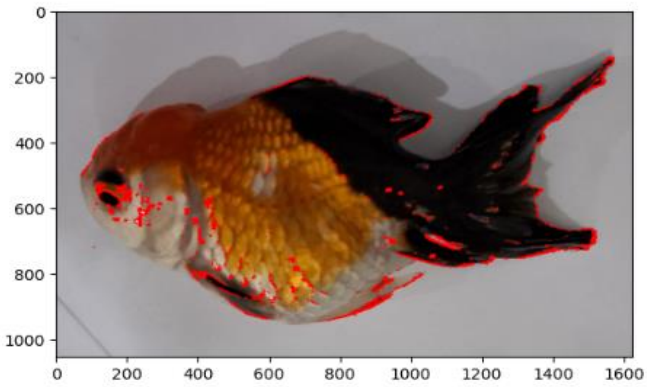


- Fungsi cv2.circle untuk mengontrol ukuran lingkaran yang digambar di sekeliling tepi

```
I2 = I_org[:,:,::-1]
idx = np.argwhere(C_image)
r,c = idx[:,0],idx[:,1]
for i in range(len(r)):
    I2 = cv2.circle(np.float32(I2),(c[i],r[i]),radius=1,color=(0,0,255),thickness=-1)
plt.imshow(np.array(I2[:,:,::-1],np.uint8))
```

Output yang di hasilkan adalah :

```
<matplotlib.image.AxesImage at 0x17573f5cbd0>
```



CONTOH : **Contour Tulisan**

- Mempersiapkan library yang akan di gunakan

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
```

- Memanggil gambar dan merubahnya menjadi gambar grey

```
input_gambar = cv2.imread("tulisanikan.png")
gray_gambar = cv2.cvtColor(input_image, cv2.COLOR_BGR2GRAY)
```

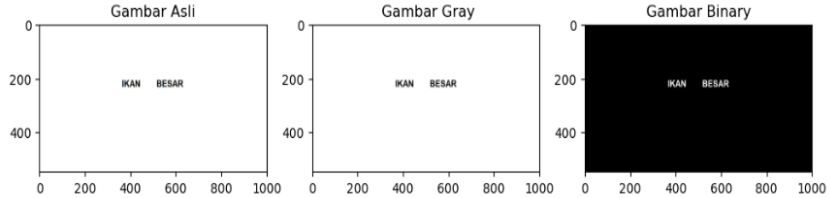
- Merubah gambar menjadi biner

```
_, binary_gambar = cv2.threshold(gray_gambar,127,255,cv2.THRESH_BINARY_INV)
```

- Menampilkan hasil perubahan gambar

```
plt.figure(figsize=[12,8])
plt.subplot(131);plt.imshow(input_gambar, cmap='gray');plt.title("Gambar Asli");

plt.subplot(132);plt.imshow(gray_gambar, cmap='gray');plt.title(" Gambar Gray");
plt.subplot(133);plt.imshow(binary_gambar, cmap='gray');plt.title("Gambar Binary");
```

Hasil Output yang di hasilkan :



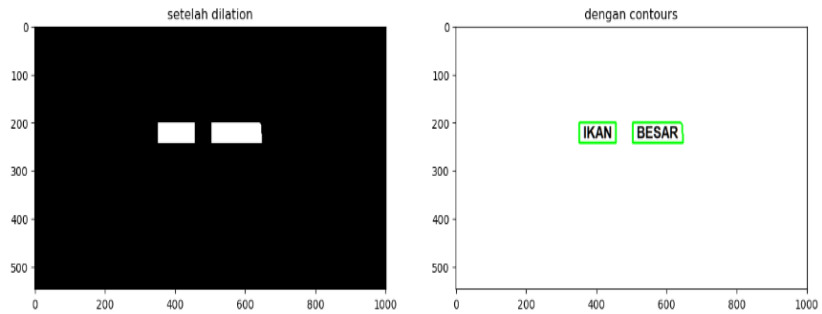
- Kontur adalah batas objek dalam gambar. Mode pengambilan cv2.RETR_EXTERNAL hanya mengambil kontur terluar, sedangkan metode pendekatan cv2.CHAIN_APPROX_SIMPLE memampatkan titik-titik kontur untuk merepresentasikan bentuk kontur secara lebih efisien.

```
kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(15,10))

dilation = cv2.dilate(binary_gambar, kernel, iterations = 2)

outputGambar = input_gambar.copy()

contours, hierarchy = cv2.findContours(dilation,
                                       cv2.RETR_EXTERNAL,
                                       cv2.CHAIN_APPROX_SIMPLE)

all_contours = cv2.drawContours(input_gambar, contours, -1, (0,255,0), 3)
```

- Menampilkan hasil Contour

```
plt.figure(figsize=[15,8])
plt.subplot(121);plt.imshow(dilation, cmap='gray');plt.title("setelah dilation");
plt.subplot(122);plt.imshow(all_contours, cmap='gray');plt.title("dengan contours");
```

Hasil Output sebagai berikut :

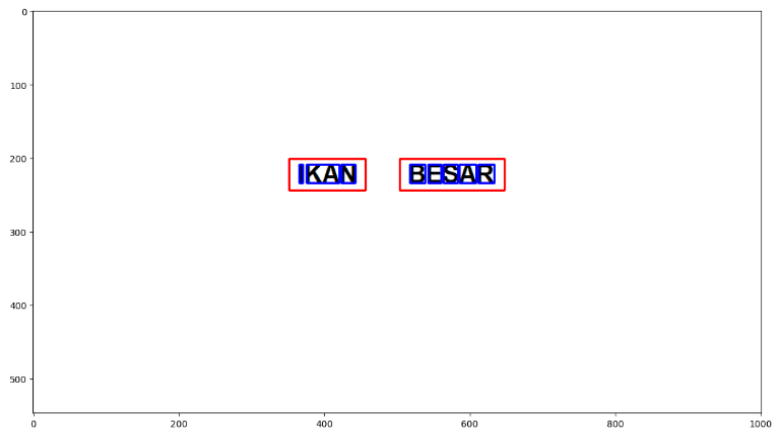


- Khusus untuk mendeteksi dan mengekstrak karakter masing-masing dari gambar. Berikut ini adalah rincian mengenai apa yang dilakukan.

```
MIN_CONTOUR_AREA = 5


for contour in contours:
    if cv2.contourArea(contour) > MIN_CONTOUR_AREA:

        [intX, intY, intW, intH] = cv2.boundingRect(contour)

        cv2.rectangle(outputGambar,(intX, intY), (intX+intW,intY+intH), (0, 0, 255), 2)

        # menentukan subgambar kata dari setiap kontur kata
        imgROI = binary_gambar[intY:intY+intH, intX:intX+intW]


        subContours, subHierarchy = cv2.findContours(imgROI.copy(),
                                            cv2.RETR_EXTERNAL,
                                            cv2.CHAIN_APPROX_SIMPLE)

        for subContour in subContours:

            [pointX, pointY, width, height] = cv2.boundingRect(subContour)


            cv2.rectangle(outputGambar,(intX+pointX, intY+pointY),(intX+pointX+width, intY+pointY+height), (255, 0, 0),2

plt.figure(figsize=[15,8])
plt.imshow(outputGambar[...,::-1])
```

Hasil Output seperti berikut :

```
<matplotlib.image.AxesImage at 0x17764866b90>
```



BAB 4
IMAGE MATCHING

Pencocokan citra (image matching) merupakan salah satu bagian dari pengolahan citra yang dilakukan untuk mencari citra lain yang sejenis atau memiliki kemiripan. Salah satu parameter yang merepresentasikan tingkat kemiripan antara dua buah citra adalah jarak euclidean.

A. Pencocokan Citra Sederhana

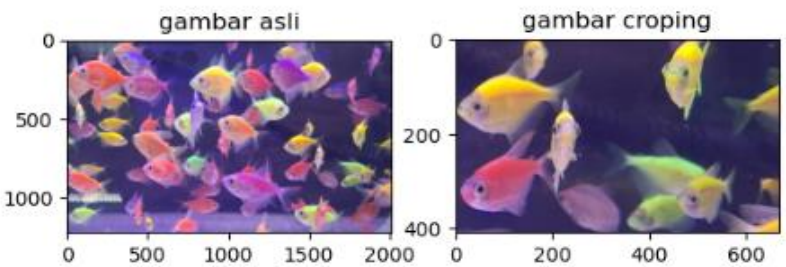
- Mempersiapkan library yang akan di gunakan

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

- Membaca gambar yang sebelumnya telah di simpan pada folder yang sama dengan file coding.

```
Gambar = cv2.imread('ikankecil.png')
GambarCroping = cv2.imread('ikankecilcrop.png')
plt.subplot(121);plt.imshow(Gambar[...,::-1]);plt.title("gambar asli");
plt.subplot(122);plt.imshow(GambarCroping[...,::-1]);plt.title("gambar croping");
```

Hasil Output yang di peroleh :



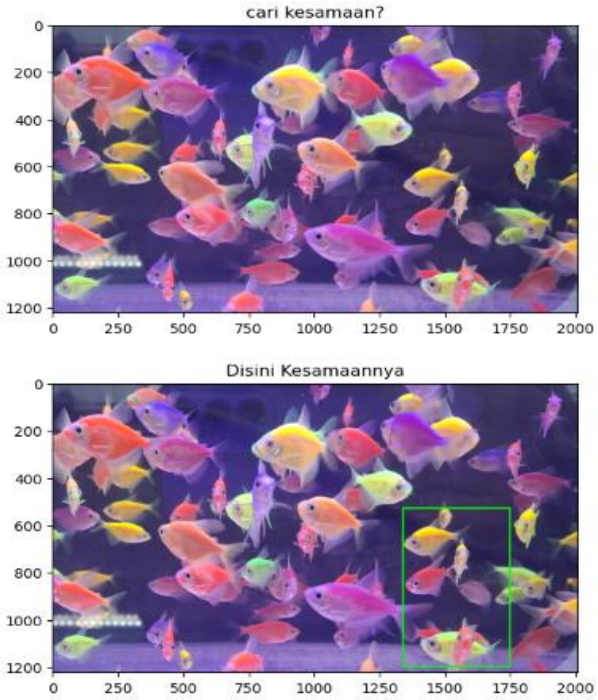
- Melakukan pencocokan gambar

```
Gambar = cv2.imread('ikankecil.png')
cv2.imshow(Gambar)
cv2.waitKey(0)
plt.imshow(cv2.cvtColor(Gambar, cv2.COLOR_BGR2RGB))
plt.title('cari kesamaan?')
plt.show()

gray = cv2.cvtColor(Gambar, cv2.COLOR_BGR2GRAY)

GambarCroping = cv2.imread('ikankecilcrop.png',0)
h, w = GambarCroping.shape

result = cv2.matchTemplate(gray, GambarCroping, cv2.TM_CCOEFF)
_, _, _, max_loc = cv2.minMaxLoc(result)

top_left = max_loc
bottom_right = (top_left[0] + h, top_left[1] + w)
cv2.rectangle(Gambar, top_left, bottom_right, (0,255,0), 5)
```

- Menampilkan hasil kecocokan gambar

```
cv2.imshow(Gambar)
cv2.waitKey(0)

plt.imshow(cv2.cvtColor(Gambar, cv2.COLOR_BGR2RGB))
plt.title('Disini Kesamaannya')
plt.show()


cv2.destroyAllWindows()
```

Output yang di hasilkan dari operasi di atas :



B. Pencocokan Citra dengan Deteksi Brute Force dengan ORB

Pencocokan citra dengan menggunakan deteksi fitur merupakan teknik yang umum digunakan untuk menemukan kesamaan antara dua atau lebih gambar. Salah satu metode deteksi fitur yang populer adalah ORB (Oriented FAST and Rotated BRIEF).

- Mempersiapkan library yang akan di gunakan

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
```

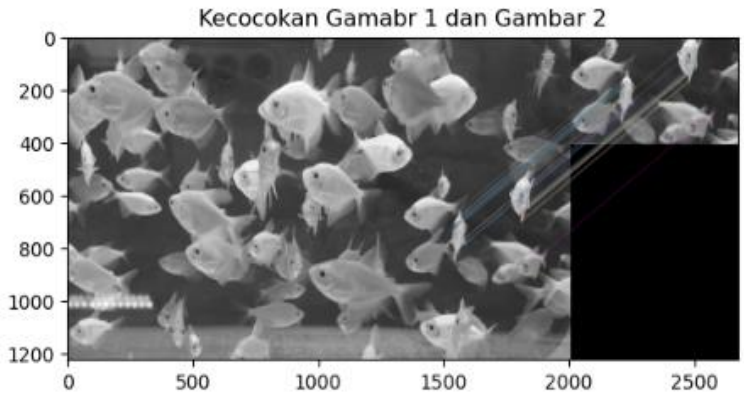
- Membaca gambar yang sebelumnya telah di simpan pada folder yang sama dengan file coding. Gambar terdiri dari dua gambar yang pertama gambar asli, yang kedua hasil potongan / cropping dari gambar asli.

```
Gambar1 = cv2.imread('ikankecil.png',0)  # queryImage (data gambar asli )
Gambar2 = cv2.imread('ikankecilcrop.png',0) # trainImage (data gambar hasil croping)
```

- Melakukan pencocokan citra, pertama-tama mendeteksi keypoints dan menghitung deskriptornya menggunakan detektor fitur yang sesuai (ORB). Kemudian, mencocokkan deskriptor menggunakan pencocokan Brute-Force. Pencocokan terbaik dipilih dan divisualisasikan dengan menggambar garis antara keypoint yang sesuai dalam dua gambar.

```
orb = cv2.ORB_create()

kp1, des1 = orb.detectAndCompute(Gambar1,None)
kp2, des2 = orb.detectAndCompute(Gambar2,None)

bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)

matches = bf.match(des1,des2)
matches = sorted(matches, key = lambda x:x.distance)
Gambar3 = cv2.drawMatches(Gambar1,kp1,Gambar2,kp2,matches[:30], flags=2, outImg=Gambar1)

cv2.imshow('Kecocokan Gamabr 1 dan Gambar 2', Gambar3)
cv2.waitKey()

plt.imshow(Gambar3)
plt.title('Kecocokan Gamabr 1 dan Gambar 2')
plt.show()

cv2.destroyAllWindows()
```

- **Hasil yang di peroleh untuk pencocokan citra sebagai berikut :**



- Pencocokan citra dengan Deteksi Brute Force Line

Deteksi garis brute force adalah metode yang sederhana namun efektif untuk pencocokan citra. Meskipun memiliki beberapa keterbatasan

- Mempersiapkan library yang akan di gunakan

```
%matplotlib inline
import numpy as np
import cv2
from matplotlib import pyplot as plt
```

- Membaca gambar yang sebelumnya telah di simpan pada folder yang sama dengan file coding. Gambar terdiri dari dua gambar

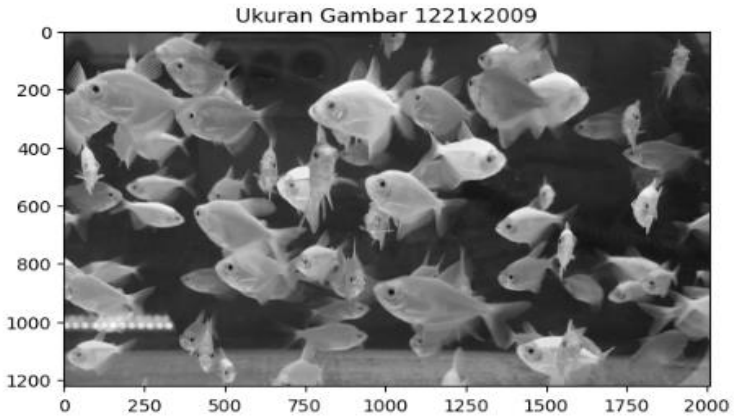
yang pertama gambar asli, yang kedua hasil potongan / cropping dari gambar asli.

```
Gambar1 = cv2.imread('ikankecil.png', cv2.IMREAD_GRAYSCALE) # gambar asli
Gambar2 = cv2.imread('ikankecilcrop.png', cv2.IMREAD_GRAYSCALE) # gambar hasil croping
```

- Menampilkan gambar Asli yaitu gambar pertama yang telah di rubah menjadi gray

```
plt.imshow(Gambar1, cmap='gray')
plt.title('Ukuran Gambar %dx%d' % Gambar1.shape);
```

Output yang dihasilkan sebagai berikut :

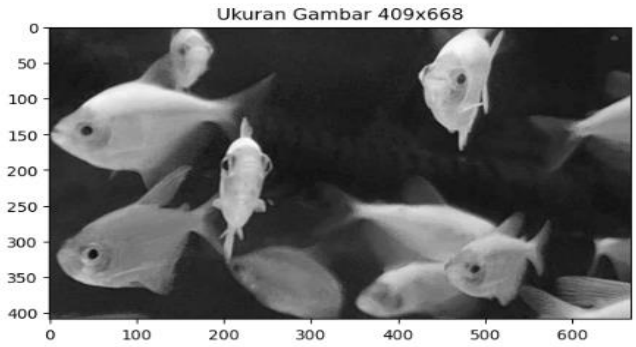


- Menampilkan gambar ke dua, yaitu hasi cropping dari gambar pertama

```
plt.imshow(Gambar2, cmap='gray')
plt.title('Ukuran Gambar %dx%d' % Gambar2.shape)
```

Output yang dihasilkan adalah sebagai berikut :

```
Text(0.5, 1.0, 'Ukuran Gambar 409x668')
```



- Menginisialisasi detektor titik dan menjalankan pencarian titik dan perhitungan descriptor.

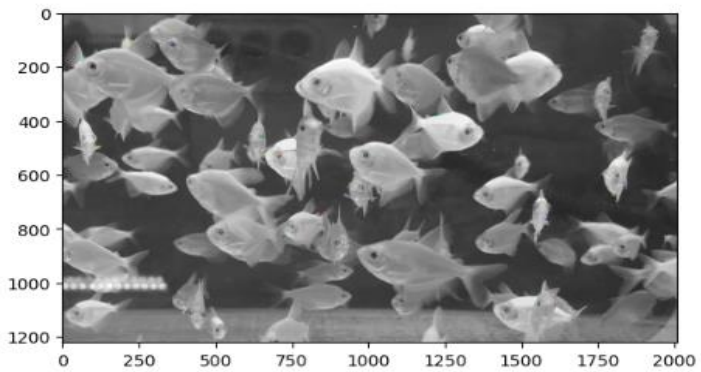
```
# menginisialisasi detektor titik
orb = cv2.ORB_create()

# menjalankan pencarian titik dan perhitungan deskriptor
kp1, des1 = orb.detectAndCompute(Gambar1, None)
kp2, des2 = orb.detectAndCompute(Gambar2, None)
```

- Menampilkan titik-titik kunci (keypoints) yang terdeteksi pada gambar pertama / gambar asli.

```
Gambar1_kp = cv2.drawKeypoints(Gambar1, kp1, np.zeros_like(Gambar1))
plt.imshow(Gambar1_kp)
```

Output yang dihasilkan sebagai berikut :

```
<matplotlib.image.AxesImage at 0x195bf49f2d0>
```

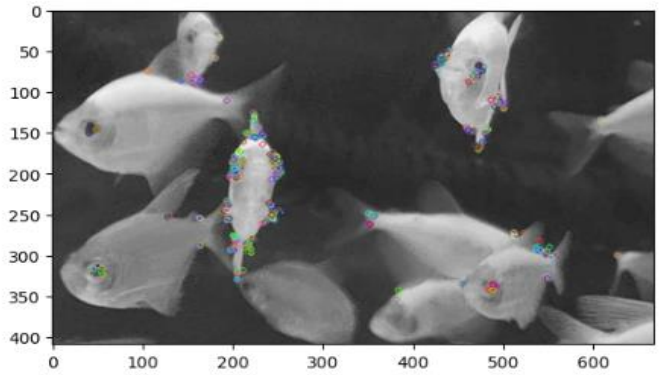


- Menampilkan titik-titik kunci (keypoints) yang terdeteksi pada gambar ke dua/ gambar hasil cropping

```
Gambar2_kp = cv2.drawKeypoints(Gambar2, kp2, np.zeros_like(Gambar1))
plt.imshow(Gambar2_kp)
```

Output yang di hasilkan sebagai berikut :

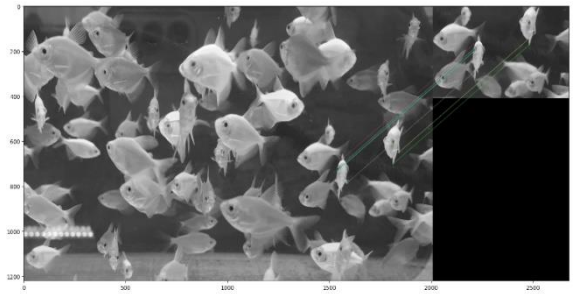
```
<matplotlib.image.AxesImage at 0x195bf55d9d0>
```



- Mencocokan dengan - brute force matching dengan mengurutkan hasil dalam urutan jarak dan menghasilkan 10 hasil perhitunga terdekat.

```
# mencocokan dengan - brute force matching
# ukuran jarak - Jarak Hamming (jumlah komponen yang tidak cocok)
bf = cv2.BFMatcher(cv2.NORM_HAMMING)

# pencocokan
matches = bf.match(des1, des2)

# mengurutkan hasil dalam urutan jarak
matches = sorted(matches, key = lambda x: x.distance)

# menghasilkan 10 hasil perhitunga terdekat
Gambar3 = np.zeros_like(Gambar2)
Gambar3 = cv2.drawMatches(Gambar1, kp1, Gambar2, kp2, matches[:7], Gambar3, flags=2)

fig = plt.gcf()
fig.set_size_inches(18.5, 10.5)
plt.imshow(Gambar3),plt.show()
```

Output yang di hasilkan adalah sebagai berikut :



```
(<matplotlib.image.AxesImage at 0x195bfe29310>, None)
```

BAB 5
IMAGE CLUSTERING

A. Clustering

Clustering adalah suatu cara menganalisa data dengan cara mengelompokan data / objek kedalam kelompok-kelompok berdasarkan suatu kesamaan tertentu.

- Mempersiapkan library yang akan digunakan

```
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt
import numpy as np
```

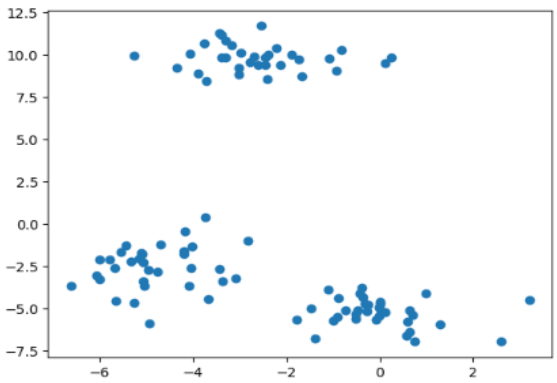
- Menentukan titik sebanyak 50 titik sebagai dataset sampel secara acak, lalu memeriksa bentuk dataset.

```
X, _ = make_blobs(random_state=50)
X.shape
```

- Buatlah scatter plot dari titik-titik data dalam array X.

```
plt.scatter(X[:, 0], X[:, 1]);
```

Output yang di hasilkan adalah :



- Membuat model K-Means yang akan mengelompokkan data ke dalam 5 cluster. Setelah model tersebut dipasang pada dataset,

model tersebut dapat digunakan untuk memprediksi label klaster untuk titik data baru.

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=5, random_state=50)
```

- Menampilkan labels sebanyak 50

```
labels = kmeans.fit_predict(X)
labels
```

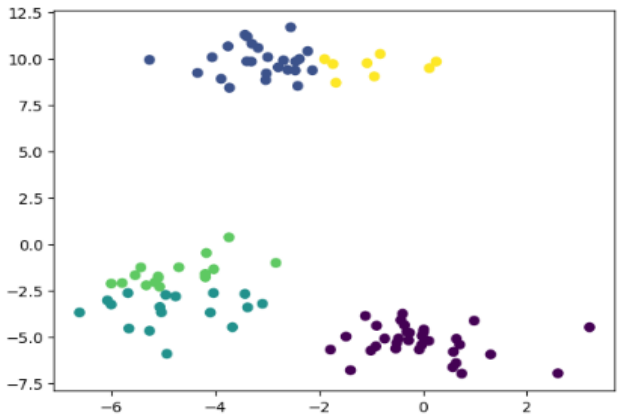
Output yang di hasilkan :

```
array([2, 3, 0, 1, 1, 3, 0, 0, 0, 3, 0, 0, 4, 0, 1, 1, 4, 3, 0, 2, 1, 0,
       3, 2, 0, 2, 3, 4, 0, 1, 4, 2, 0, 1, 1, 2, 0, 0, 0, 1, 0, 1, 0, 2,
       4, 3, 0, 1, 1, 1, 1, 0, 1, 3, 0, 3, 3, 0, 0, 2, 1, 0, 1, 1, 1, 3,
       3, 0, 1, 0, 2, 2, 2, 0, 1, 3, 0, 0, 2, 2, 1, 1, 1, 0, 2, 4, 4, 0,
       2, 0, 1, 0, 4, 0, 2, 3, 0, 3, 2, 3])
```

- Hasil akhir dari klustering 5 kelompok adalah sebagai berikut

```
plt.scatter(X[:, 0], X[:, 1], c=labels);
```

Output yang di hasilkan sebagai berikut :

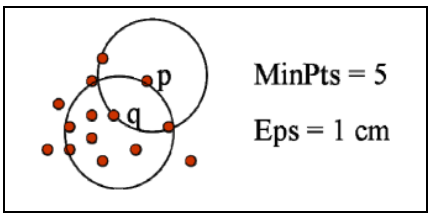


B. DBSCAN Clustering

Setiap titik cluster yang berasal dari lingkungan radius harus berisi setidaknya jumlah minimum poin yang telah ditentukan.

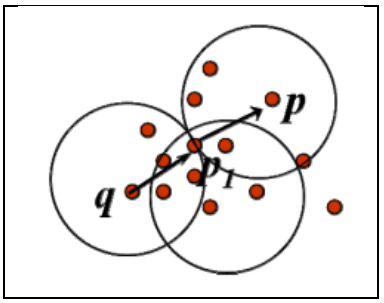
Algoritma ini membutuhkan 2 masukan parameter, yaitu Eps, jari-jari yang menentukan batas daerah sekitar titik (Epsneighbourhood) dan MinPts, jumlah minimum titik yang harus ada di lingkungan-Eps. Ide dasar dari density-based clustering berkaitan dengan beberapa definisi yaitu :

1. Neighborhood dengan radius Eps dari suatu obyek disebut Eps-neighborhood dari suatu obyek tersebut.
2. Jika Eps-neighborhood dari suatu obyek mengandung titik sekurang-kurangnya jumlah minimum, MinPts, maka suatu obyek tersebut dinamakan *core object*
3. Diberikan set obyek D, obyek p dikatakan *directly density-reachable* (kepadatan terjangkau langsung) dari obyek q jika p termasuk dalam Eps-neighborhood dari q dan q adalah *core objek.* Ilustrasi *directly density-reachable* (kepadatan terjangkau langsung)



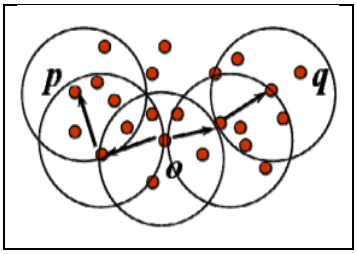
Gambar 9. Ilustrasi *directly density-reachable*

4. Sebuah obyek p adalah *density-reachable* dari obyek q dengan memperhatikan Eps dan MinPts dalam suatu set objek , D, jika terdapat serangkaian obyek p1,…,pn, p1=q dan pn=p dimana pi+1 adalah directly *density-reachable* dari pi dengan memperhatikan Eps dan MinPts, untuk 1 <= i <= n, pi elemen D. Konsep *density-reachable* di-ilustrasikan pada Gambar dibawah ini :



Gambar 10. Ilustrasi *density-reachable*

5. Sebuah obyek p adalah *density-connected* terhadap obyek q dengan memperhatikan Eps dan MinPts dalam set obyek D, jika ada sebuah obyek o elemen D sehingga p dan q keduanya density-reachable dari o dengan memperhatikan Eps dan MinPts. Gambar di bawah ini merupakan ilustrasi dari konsep *density-connected*.



Gambar 11. Ilustrasi *density- connected*

Tahapan dari algoritma DBSCAN:

1. memilih titik p sebagai titik pusat
2. Ambil semua titik yang memiliki kepadatan yang dapat dijangkau terhadap titik p
3. jika p adalah titik inti, maka cluster terbentuk
4. Jika p adalah titik batas, tidak ada titik yang density-reachable dari p dan DBSCAN mengunjungi titik berikutnya dari database.
5. Lanjutkan proses sampai semua titik telah diproses

CONTOH KASUS

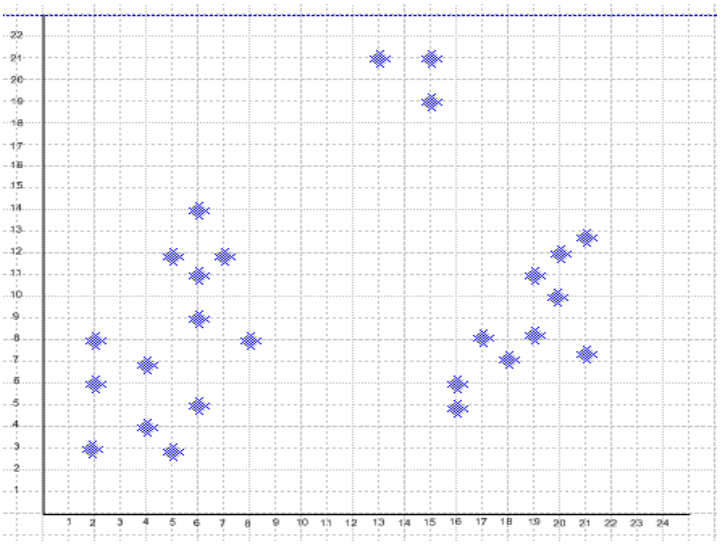
Eps = 4

MinPts = 4

Tabel node dengan titik (x,y) :

A (4,4)	H (16,7)	O (6,9)	V(21,13)
B (2,3)	I (18,7)	P (20,10)	W(6,14)
C (5,3)	J (121,7)	Q (6,11)	X(15,19)
D (6,5)	K (2,8)	R (19,11)	Y(13,21)
E (16,5)	L (8,8)	S (5,12)	Z(15,21)
F(2,6)	M (17,8)	T (7,12)	
G (4,7)	N (19,8)	U (20,12)	

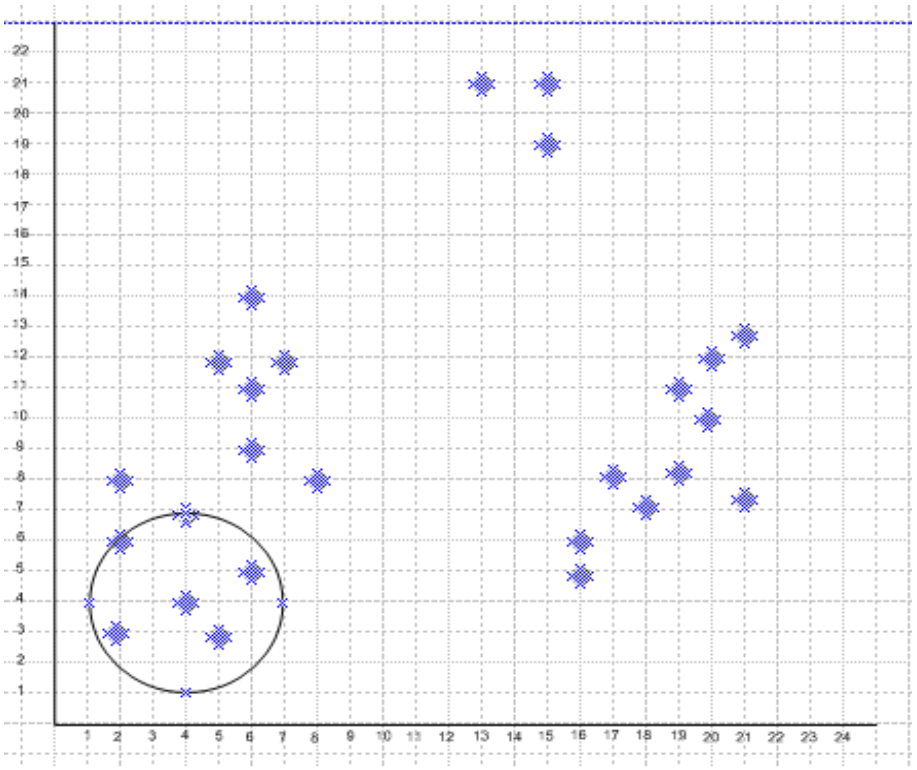
Node (x,y) dari table diatas direfresentasikan dengan gambar dibawah ini :



Iterasi 1. Dengan titik tengah (4,4) maka point yang menjadi anggota adalah 5 point yaitu B, C, D, F, G. Untuk lebih jelasnya dapat dilihat pada tabel berikut :

A (4,4)	H (16,7)	O (6,9)	V(21,13)
B (2,3)	I (18,7)	P (20,10)	W(6,14)
C (5,3)	J (121,7)	Q (6,11)	X(15,19)
D (6,5)	K (2,8)	R (19,11)	Y(13,21)
E (16,5)	L (8,8)	S (5,12)	Z(15,21)
F(2,6)	M (17,8)	T (7,12)	
G (4,7)	N (19,8)	U (20,12)	

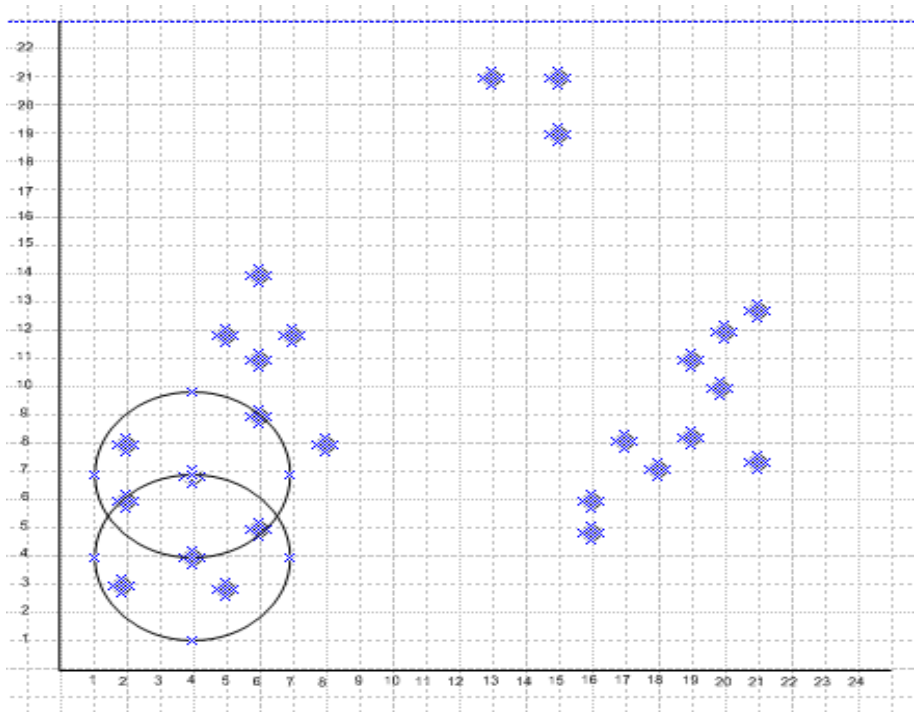
Titik Tengah = (4,4)



Iterasi 2. Dengan titik tengah (4,7) maka point yang menjadi anggota adalah 4 point yaitu A, D, F, K. Berikut penyajian dalam bentuk tabel agar lebih jelas sebagai berikut :

A (4,4)	H (16,7)	O (6,9)	V(21,13)
B (2,3)	I (18,7)	P (20,10)	W(6,14)
C (5,3)	J (121,7)	Q (6,11)	X(15,19)
D (6,5)	K (2,8)	R (19,11)	Y(13,21)
E (16,5)	L (8,8)	S (5,12)	Z(15,21)
F(2,6)	M (17,8)	T (7,12)	
G (4,7)	N (19,8)	U (20,12)	

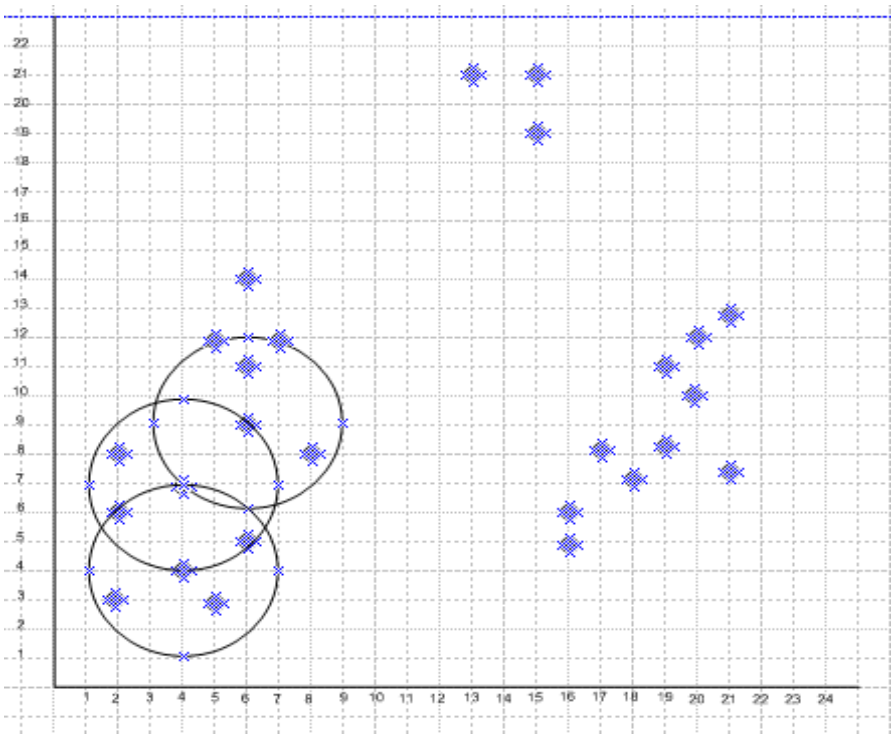
Titik Tengah = (4,7)



Iterasi 3. Dengan titik tengah (6,9) maka point yang menjadi anggota adalah 6 point yaitu D, G, L, Q, S dan T. Untuk lebih jelasnya dapat dilihat pada tabel berikut :

A (4,4)	H (16,7)	O (6,9)	V(21,13)
B (2,3)	I (18,7)	P (20,10)	W(6,14)
C (5,3)	J (121,7)	Q (6,11)	X(15,19)
D (6,5)	K (2,8)	R (19,11)	Y(13,21)
E (16,5)	L (8,8)	S (5,12)	Z(15,21)
F(2,6)	M (17,8)	T (7,12)	
G (4,7)	N (19,8)	U (20,12)	

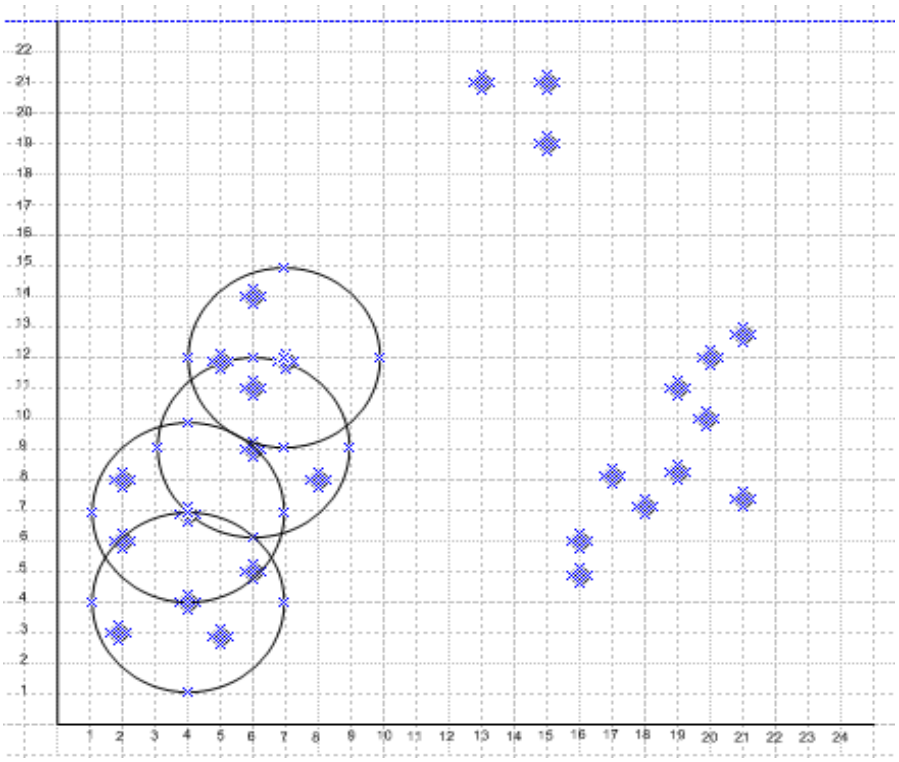
Titik Tengah (6,9)



Iterasi 4. Dengan titik tengah (7,12) maka point yang menjadi anggota adalah 7 point yaitu O, P, Q, S dan W. Untuk lebih jelasnya dapat dilihat pada tabel berikut :

A (4,4)	H (16,7)	O (6,9)	V(21,13)
B (2,3)	I (18,7)	P (20,10)	W(6,14)
C (5,3)	J (121,7)	Q (6,11)	X(15,19)
D (6,5)	K (2,8)	R (19,11)	Y(13,21)
E (16,5)	L (8,8)	S (5,12)	Z(15,21)
F(2,6)	M (17,8)	T (7,12)	
G (4,7)	N (19,8)	U (20,12)	

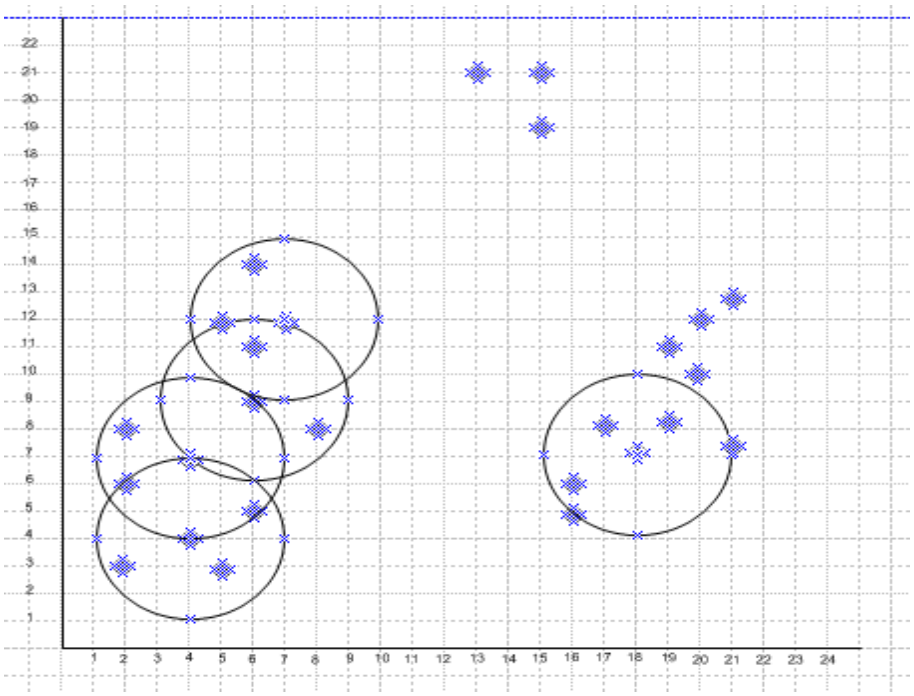
Titik Tengah (7,12)



Iterasi 5. Dengan titik tengah (16,7) maka point yang menjadi anggota adalah 5 point yaitu D, I, M dan N. Untuk lebih jelasnya dapat dilihat pada tabel berikut :

A (4,4)	H (16,7)	O (6,9)	V(21,13)
B (2,3)	I (18,7)	P (20,10)	W(6,14)
C (5,3)	J (12,7)	Q (6,11)	X(15,19)
D (6,5)	K (2,8)	R (19,11)	Y(13,21)
E (16,5)	L (8,8)	S (5,12)	Z(15,21)
F(2,6)	M (17,8)	T (7,12)	
G (4,7)	N (19,8)	U (20,12)	

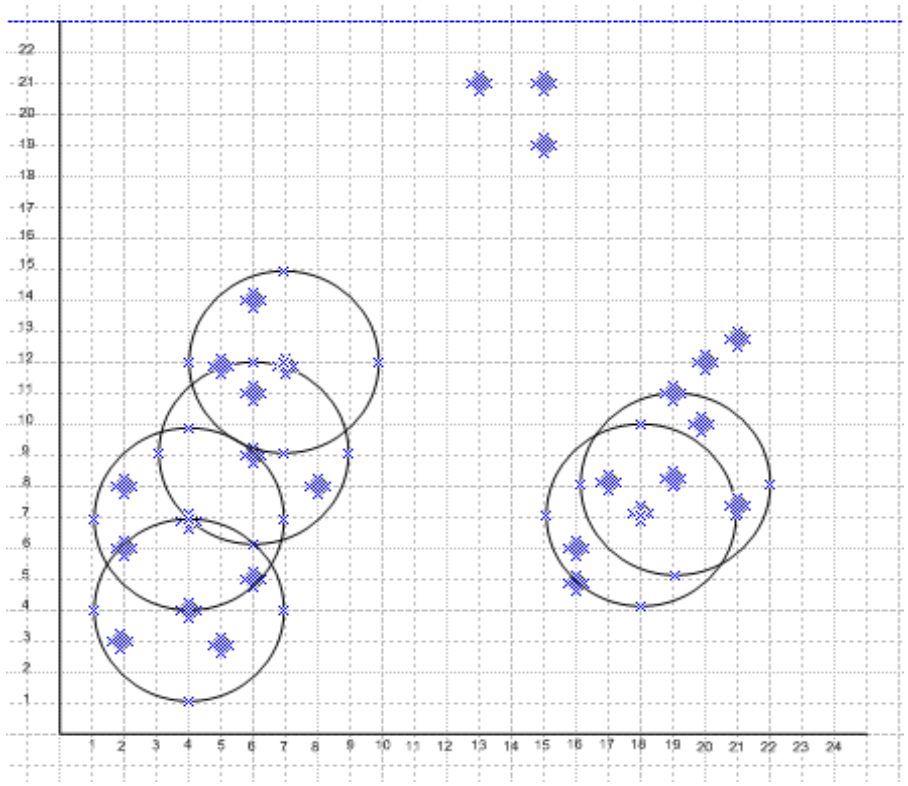
Titik Tengah (16,7)



Iterasi 6. Dengan titik tengah (19,8) maka point yang menjadi anggota adalah 5 point yaitu H,I, J, M, P dan R. Untuk lebih jelasnya dapat dilihat pada tabael berikut :

A (4,4)	H (16,7)	O (6,9)	V(21,13)
B (2,3)	I (18,7)	P (20,10)	W(6,14)
C (5,3)	J (121,7)	Q (6,11)	X(15,19)
D (6,5)	K (2,8)	R (19,11)	Y(13,21)
E (16,5)	L (8,8)	S (5,12)	Z(15,21)
F(2,6)	M (17,8)	T (7,12)	
G (4,7)	N (19,8)	U (20,12)	

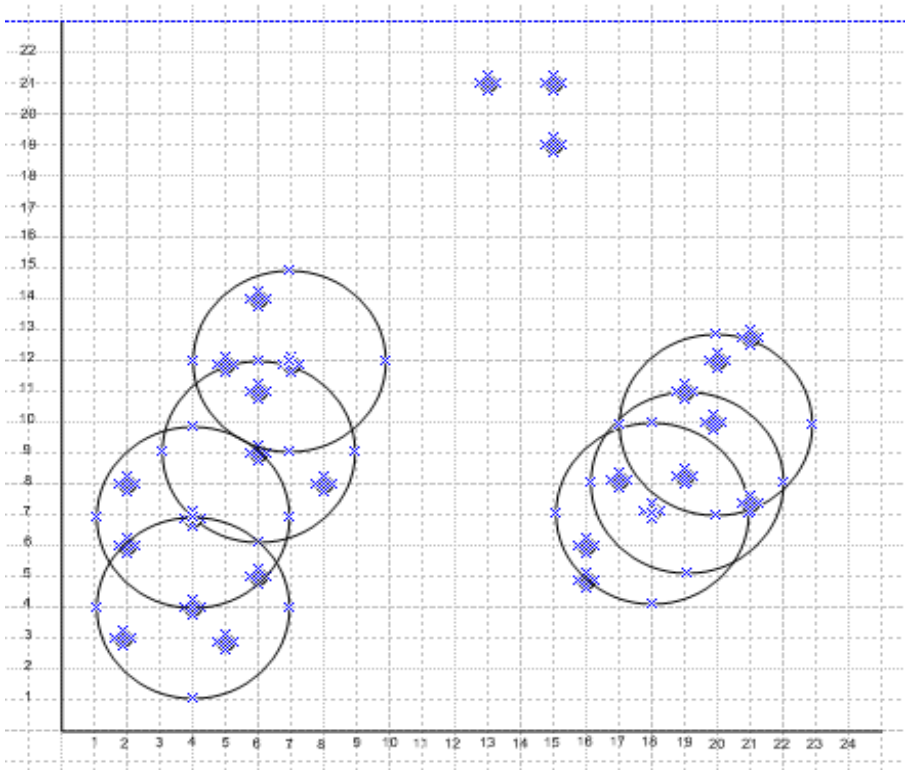
Titik Tengah (19,8)



Iterasi 7. Dengan titik tengah (20,10) maka point yang menjadi anggota adalah 4 point yaitu I, J, M, N. Untuk lebih jelasnya dapat dilihat pada tabael berikut :

A (4,4)	H (16,7)	O (6,9)	V(21,13)
B (2,3)	I (18,7)	P (20,10)	W(6,14)
C (5,3)	J (121,7)	Q (6,11)	X(15,19)
D (6,5)	K (2,8)	R (19,11)	Y(13,21)
E (16,5)	L (8,8)	S (5,12)	Z(15,21)
F(2,6)	M (17,8)	T (7,12)	
G (4,7)	N (19,8)	U (20,12)	

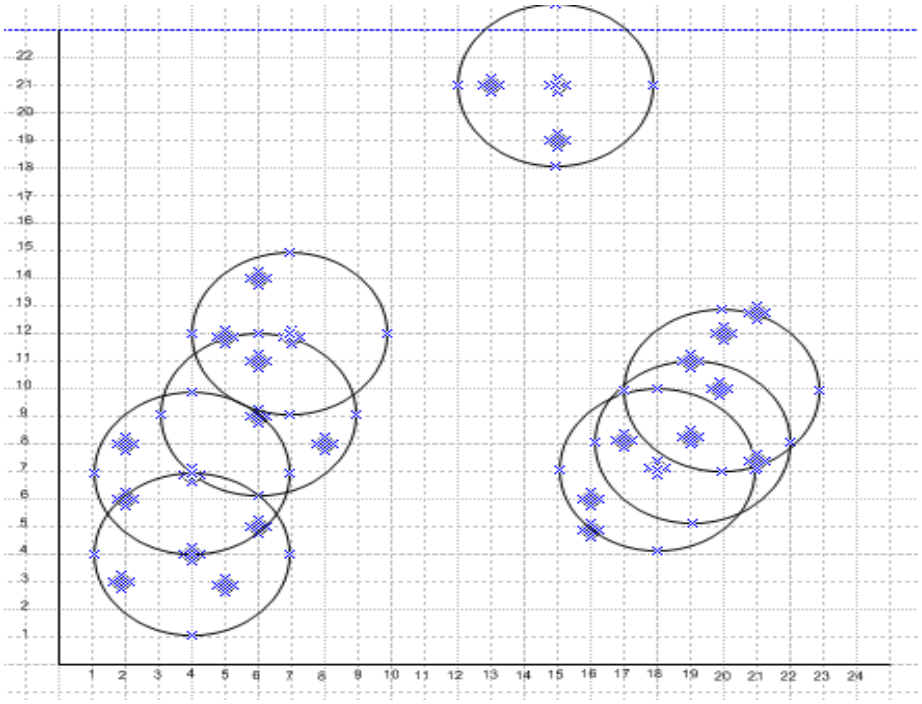
Titik Tengah (20,10)



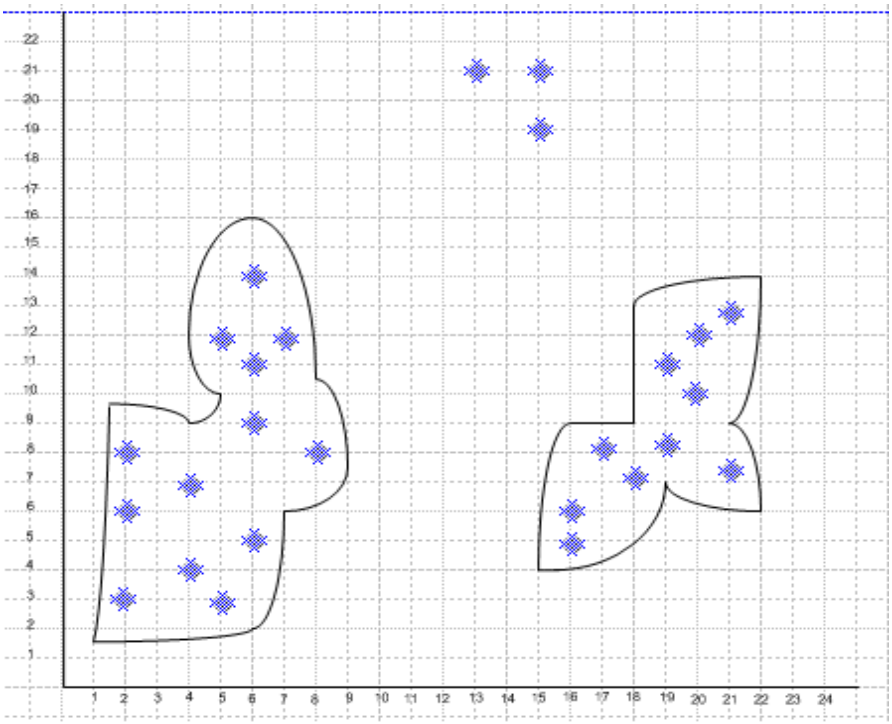
Iterasi 8. Dengan titik tengah (15,21)maka point yang menjadi anggota adalah 2 point yaitu. Untuk lebih jelasnya dapat dilihat pada tabael berikut :

A (4,4)	H (16,7)	O (6,9)	V(21,13)
B (2,3)	I (18,7)	P (20,10)	W(6,14)
C (5,3)	J (121,7)	Q (6,11)	X(15,19)
D (6,5)	K (2,8)	R (19,11)	Y(13,21)
E (16,5)	L (8,8)	S (5,12)	Z(15,21)
F(2,6)	M (17,8)	T (7,12)	
G (4,7)	N (19,8)	U (20,12)	

Titik Tengah (15,12)



Hasil Output :



Contoh selanjutnya penggunaan DB Scan sebagai berikut dengan dua puluh titik.

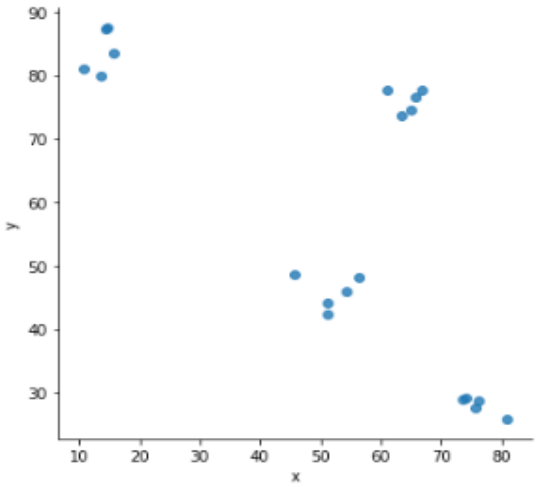
1. Menyiapkan library yang akan di gunakan:

```
import numpy as np

from dataviz import generate_clusters
from dataviz import plot_clusters
from dataviz import plot_data
from dbscan import DBSCAN
```

2. Memvisualisasikan data 20 titik yang dibuat secara acak dalam rentang [1, 100]

```
def generate_data(num_clusters: int, seed=None) -> np.ndarray:
    titik_point= 20
    spread = 7
    bounds = (1, 100)
    clusters = generate_clusters(titik_cluster,
                                 titik_point,
                                 spread,
                                 bounds,
                                 bounds,
                                 seed)
    return np.array(clusters)

num_clusters = 4
clusters = generate_data(titik_cluster, seed=1)
plot_data(clusters)
```

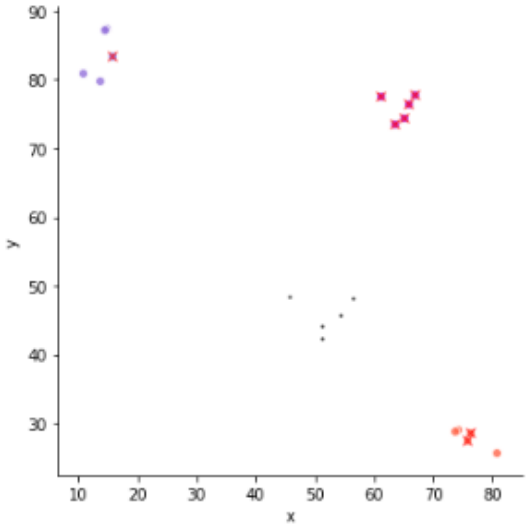
Hasil visualisasi data



3. Membuat cluster dengan metode DBCAN, Mari kita pilih `min_samples` menjadi 5 karena ada 5 titik per klaster, Jarak terbesar dari sebuah titik ke titik tengah dari setiap klaster adalah 6. Kita sebut sebagai `eps`.

```
dbscan = DBSCAN(eps=6, min_samples=5)
dbscan.fit(clusters)
plot_clusters(clusters, dbscan.labels_, dbscan.components_)
```

Output yang dihasilkan adalah sebagai berikut, Dimana titik merah adalah core points dan hitam sebagai noise points



C. K-Means Clustering

Termasuk partitioning clustering yang memisahkan data ke k daerah bagian yang terpisah. K-means algorithm sangat terkenal karena kemudahan dan kemampuannya untuk mengklaster data besar dan data outlier dengan sangat cepat.

Sesuai dengan karakteristik partitioning clustering, Setiap data harus termasuk ke cluster tertentu, dan Memungkinkan bagi setiap data yang termasuk cluster tertentu pada suatu tahapan proses, pada tahapan berikutnya berpindah ke cluster yang lain.

CONTOH 1.

Mengcluster Warna Dominan Pada Citra

- Mempersiapkan library yang akan digunakan

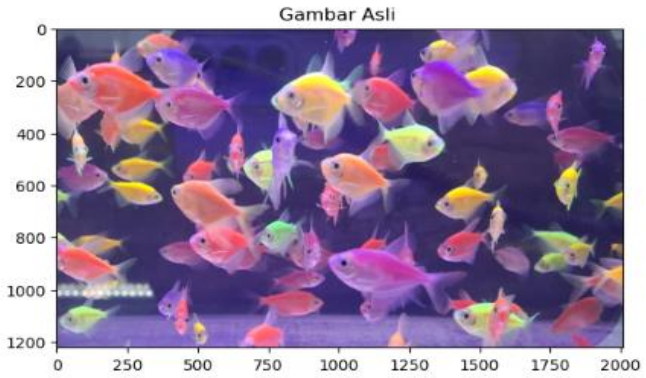
```
import cv2
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

- Membaca gambar dan menampilkannya

```
Gambar = cv2.imread("ikankecil.png")
Gambar_baru =cv2.cvtColor(Gambar,cv2.COLOR_BGR2RGB)
plt.imshow(Gambar_baru);plt.title('Gambar Asli ')
```

Output yang di hasilkan :

```
Text(0.5, 1.0, 'Gambar Asli ')
```



- Melakukan segmentasi gambar menggunakan algoritma K-Means dengan tujuan menemukan warna-warna dominan dalam gambar, dengan jumlah cluster 6.

```
n_clusters = 6

# membentuk list piksel
flat_Gambar = Gambar.reshape((-1, 3))

#menggunakan k-means untuk mengelompokkan piksel
kmeans = KMeans(n_clusters = n_clusters)
kmeans.fit(flat_Gambar)

#pusat cluster adalah warna dominan
dominant_warna = np.array(kmeans.cluster_centers_, dtype='uint8')

labels = kmeans.labels_

print(dominant_colors)
print(labels)
```

Dengan output sebagai berikut

```
[[183 114 152]
 [102  49  60]
 [125 110 207]
 [136  85 114]
 [107 199 221]
 [188 170 201]]
[4 4 4 ... 0 0 0]
```

- Menghitung persentase kemunculan setiap warna dominan yang telah ditemukan dalam sebuah gambar.

```
persentase = np.bincount(labels)/len(flat_Gambar)
persentase
```

```
array([0.18542562, 0.44958864, 0.039193  , 0.09448921, 0.12342371,
       0.10787982])
```

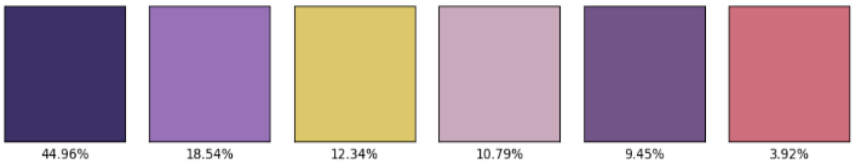
- Menggabungkan nilai persentase dan warna dominan yang sesuai ke dalam daftar tupel, kemudian mengurutkan daftar dalam urutan menurun berdasarkan nilai persentase. Hal ini memungkinkan kita untuk mengurutkan warna dominan berdasarkan penonjolan warna dalam gambar.

```
p_and_c = zip(persentase,dominant_colors)
p_and_c = sorted(p_and_c,reverse=True)
```

- Kode ini digunakan untuk memvisualisasikan warna dominan yang diidentifikasi dalam sebuah gambar menggunakan pengelompokan K-Means.

```
block = np.ones((50,50,3),dtype='uint')
plt.figure(figsize=(12,8))
for i in range(n_clusters):
    plt.subplot(1,n_clusters,i+1)
    block[:] = p_and_c[i][1][::-1]
    plt.imshow(block)
    plt.xticks([])
    plt.yticks([])
    plt.xlabel(str(round(p_and_c[i][0]*100,2))+'%')
```

Output yang di hasilkan sebagai berikut :



CONTOH 2.

Penerapan model K-Means Pada Segmentasi Gambar

Segmentasi gambar adalah proses pembagian sebuah gambar menjadi beberapa region. Salah satu metode yang populer untuk melakukan segmentasi gambar adalah algoritma K-Means.

- Mempersiapkan library yang akan di gunakan

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
```

- Membaca gambar, yang sebelumnya telah di upload pada folder yang sama dengan file coding. Dan mengcopy gambar yang telah di baca.

```
%matplotlib inline

# membaca gambar
Gambar = cv2.imread('ikankecil.png')

# mengcopy gambar
Gambar_copy = np.copy(image)
```

- Merubah gambar menjadi RGB

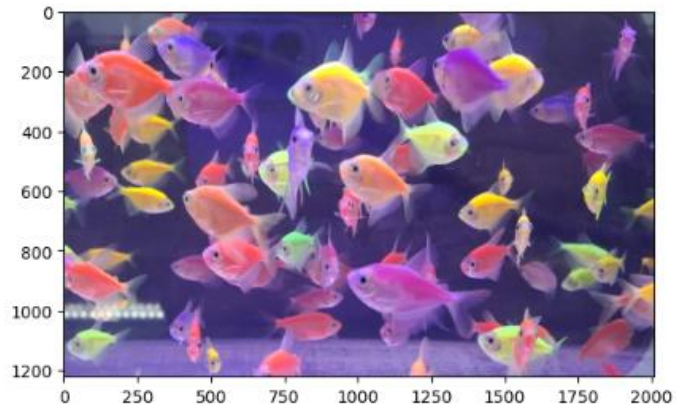
```
Gambar_copy = cv2.cvtColor(Gambar_copy, cv2.COLOR_BGR2RGB)
```

- Menampilkan gambar yang akan di lakukan proses segmentasi

```
plt.imshow(Gambar_copy)

print(Gambar_copy.shape)
```

Output yang di hasilkan adalah sebagai berikut :

```
(1221, 2009, 3)
```



- Membentuk ulang gambar menjadi susunan piksel 2D dan 3 nilai warna (RGB) dan mengonversi ke tipe float.

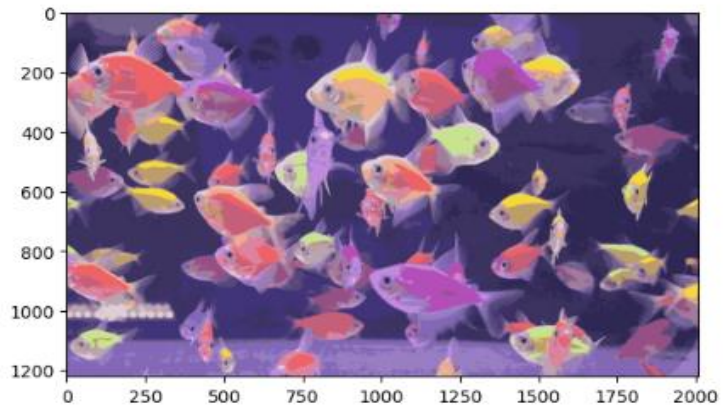
```
pixel_vals = Gambar_copy.reshape((-1,3))

pixel_vals = np.float32(pixel_vals)
```

- K-means clustering, dengan mengisi julah k =20

```
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 100, 0.2)

k = 20
retval, labels, centers = cv2.kmeans(pixel_vals, k, None, criteria, 10, cv2.KMEANS_RANDOM_CENTERS)

# Convert data into 8-bit values
centers = np.uint8(centers)
segmented_data = centers[labels.flatten()
""
# Membentuk ulang data ke dalam dimensi gambar asli
segmented_image = segmented_data.reshape((Gambar_copy.shape))
labels_reshape = labels.reshape(Gambar_copy.shape[0], Gambar_copy.shape[1])
```

- **Menampikan hasil segmentasi :**

```
<matplotlib.image.AxesImage at 0x24da7a60c90>
```

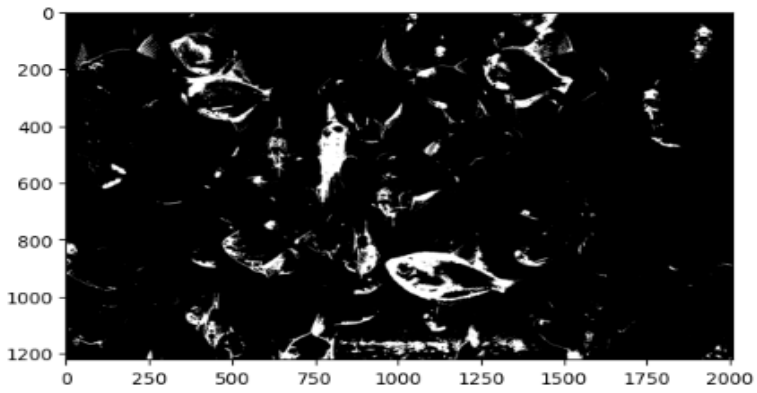


- Memvisualisasikan satu segmen dengan terelebihdahulu membuat ke dalam image grey

```
plt.imshow(labels_reshape==5, cmap='gray')
```

Output yang di hasilkan :

```
<matplotlib.image.AxesImage at 0x237143aa050>
```

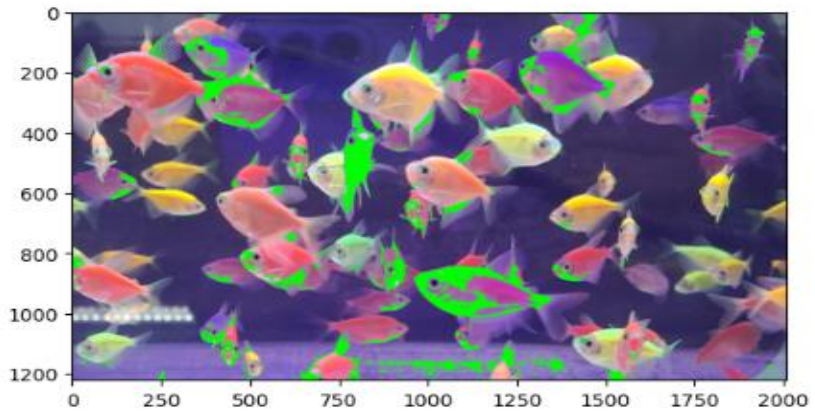


- Masking gambar dan menampilkan gambar hasil masking

```
masked_image = np.copy(image_copy)
masked_image[labels_reshape == 5] = [0, 255, 0]

plt.imshow(masked_image)
```

Output yang di hasilkan dari masking :

```
<matplotlib.image.AxesImage at 0x23715b60c10>
```



BAB 6
IMAGE CLASSIFICATION

A. Klasifikasi

Klasifikasi data adalah suatu proses yang menemukan properti-properti yang sama pada sebuah himpunan obyek di dalarn sebuah basis data, dan mengklasifikasikannya ke dalam kelas-kelas yang berbeda menurut model klasifikasi yang ditetapkan. Untuk membentuk sebuah model klasifikasi, suatu sampel basis data 'E' diperlakukan sebagai training set, dimana setiap tupel terdiri dari himpunan yang sama yang memuat atribut yang beragam seperti tupel-tupel yang terdapat dalam suatu basis data yang besar 'W'. Setiap tupel diidentifkasikan dengan sebuah label atau identitas kelas.

Tujuan dari klasifikasi ini adalah pertama-tama untuk menganalisa training data dan membentuk sebuah deskripsi yang akurat atau sebuah model untuk setiap kelas berdasarkan feature-feature yang tersedia di dalam data itu. Deskripsi dari masing-masing kelas itu nantinya akan digunakan untuk mengklasifikasikan data yang hendak di test dalam basis data 'W', atau untuk membangun suatu deskripsi yang lebih baik untuk setiap kelas dalam basis data.

B. K-Nearest Neighbor (KNN)

Algoritma K-Nearest Neighbor (KNN) merupakan algoritma supervised learning di mana hasil klasifikasi data baru berdasar kepada kategori mayoritas tetangga terdekat ke-K. Tujuan dari algoritma ini adalah mengklasifikasikan objek baru berdasarkan atribut dan data

training. Algoritma KNN menggunakan kalsifikasi ketetenggaan sebagai prediksi terhadap data baru.

CONTOH 1. Diberikan data training berikut, terdiri dari 2 atribut dengan skala kuantitatif yaitu X1penampilan ikan dan X2 prilaku ikan serta 2 kelas yaitu Sehat dan Sakit. Jika terdapat data baru dengan nilai X1=8 dan X2=4, tentukan kelasnya!

Data Training

Data Ikan	X1(penampilan)	X2(Prilaku)	Klas
	8	9	Sehat
	7	8	Sehat
	4	2	Sakit
	3	2	Sakit
	**8**	**4**	**?**

Langkah Klasifikasi :

1. Tentukan parameter K = jumlah tetangga terdekat.
   Misalkan ditetapkan Nilai parameter K = 3
2. Hitung jarak antara data baru dengan semua data training

X1	X2	*Eucledian Distance* Data baru (8,4)
8	9	$\text{Sqrt}((8-8)^2+(9-4)^2) = 5$
7	8	$\text{Sqrt}((7-8)^2+(8-4)^2) = 4{,}1$
4	2	$\text{Sqrt}((4-8)^2+(5-4)^2) = 4{,}5$
3	2	$\text{Sqrt}((3-8)^2+(2-4)^2) = 5{,}4$

3. Urutkan jarak tersebut dan tetapkan tetangga terdekat berdasarkan jarak minimum ke-K

X1	X2	*Eucledian Distance* Data baru (8,4)	Rank	Termasuk 3 tetangga terdekat
8	9	Sqrt$((8-8)^2+(9-4)^2)$ = 5	3	Ya
7	8	Sqrt$((7-8)^2+(8-4)^2$) = 4,1	1	Ya
4	2	Sqrt$((4-8)^2+(5-4)^2$) = 4,5	2	Ya
3	2	Sqrt$((3-8)^2+(2-4)^2$) = 5,4	4	Tidak

4. Periksa kelas dari tetangga terdekat

X1	X2	*Eucledian Distance* Data baru (8,4)	Termasuk 3 tetangga terdekat	Y
8	9	Sqrt$((8-8)^2+(9-4)^2)$ = 5	Ya	Sehat
7	8	Sqrt$((7-8)^2+(8-4)^2$) = 4,1	Ya	Sehat
4	2	Sqrt$((4-8)^2+(5-4)^2$) = 4,5	Ya	Sakit
3	2	Sqrt$((3-8)^2+(2-4)^2$) = 5,4	Tidak	

5. Gunakan mayoritas sederhana dari kelas tetangga terdekat sebagai nilai prediksi data baru. Hasil pada no 4 menunjukkan bahwa dari 3 tetangga terdekat, terdapat 2 kelas Sehat dan 1 kelas Sakit, maka disimpulkan bahwa data baru termasuk ke dalam kelas **Sehat.**

```
from sklearn.neighbors import KNeighborsClassifier
```

```
x1=[8,7,4,3]
x2=[9,8,2,2]
target=['sehat','sehat','sakit','sakit']
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
target_encoded=le.fit_transform(target)
print(target_encoded)
```

```
[1 1 0 0]
```

```
features =zip(x1,x2)
features = list(features)
features
```

```
Out[5]: [(8, 9), (7, 8), (4, 2), (3, 2)]
```

```
knn = KNeighborsClassifier(n_neighbors= 3)
knn.fit(features,target)
print(knn.predict([[8,4]]))
```

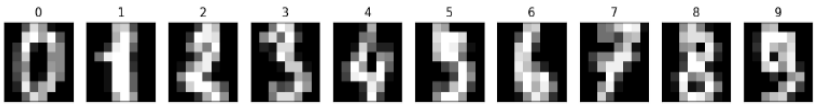
```
['sehat']
```

CONTOH 2.

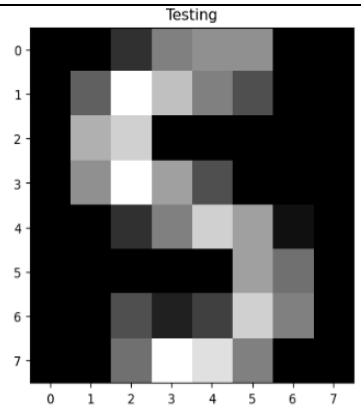
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

```
data = load_digits()

X = data.data
Y = data.target

plt.figure(figsize=(15, 3))
for label, image in enumerate(data.images[:10]):
    ax = plt.subplot(2, 10, label+1)
    plt.imshow(image, cmap='gray')
    plt.title(str(label))
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
plt.show()


x_train, x_test, y_train, y_test = train_test_split(data.data, data.target, test_size=0.3, random_state=42)
model = KNeighborsClassifier()

model.fit(x_train, y_train)

y_pred = model.predict(x_test)
```

```
accuracy = accuracy_score(y_test, y_pred) * 100
print('Model Accuracy: ' + str(accuracy) + ' %')


def predict_image_at_index(image_index):
    plt.imshow(x_test[image_index].reshape(8, 8), cmap='gray')
    plt.title(str('Testing'))
    plt.show()

    prediction = model.predict([x_test[image_index]])[0]
    real_value = y_test[image_index]

    print('Model Predicted Digit is: ' + str(prediction))
    print('Real Digit is: ' + str(real_value))



predict_image_at_index(150)
```




```
Model Accuracy: 99.25925925925925 %
```




```
Model Predicted Digit is: 5
Real Digit is: 5
```

BAB 7
DETEKSI & IDENTIFIKASI OBJEK

A. Algoritma YOLOv9

YOLO merupakan salah satu model yang digunakan dalam pendeteksian objek. YOLOv9 memiliki beberapa pembaruan dari versi sebelumnya yaitu YOLOv8, seperti penggunaan modul yang lebih luas dari YOLOv9 dibandingkan versi sebelumnya, sehingga memungkinkan arsitektur yang lebih kompleks.

Penilaian performa [10] dengan menggunakan 3 parameter yaitu precision, recall, dan mAP menemukan bahwa penerapan YOLOv9 memiliki nilai yang lebih baik dibandingkan versi sebelumnya (YOLOv8) meskipun membutuhkan memori yang lebih besar. YOLOv9 memiliki keunggulan dalam menyelesaikan tugas dengan kecepatan tinggi[11]. Berikut merupakan cara penggunaan YOLOv9.

- Impor semua modul yang diperlukan.

```
import os
import io
import sys
from PIL import Image, ImageFont, ImageDraw
from ultralytics import YOLO
```

- Mendapatkan direktori tempat program dieksekusi.

```
base_dir = os.getcwd()
```

- Membuat fungsi untuk memproses gambar menggunakan model YOLO, berikan nama fungsi process_image dengan

atribut image_path. Mendapatkan direktori model dan memuat model menggunakan YOLO.

```
model_path = os.path.join(base_dir, "models", "best_yolo.pt")
model = YOLO(model_path)
```

- Membuka gambar dan mendapatkan ukuran gambar (lebar dan tinggi).

```
img = Image.open(image_path)
img_width, img_height = img.size
```

- Melakukan prediksi deteksi objek pada gambar dengan confidence dan IoU tertentu.

```
results = model.predict(img, conf=0.75, iou=0.5)
```

- Mengambil bounding box dari hasil deteksi dan menggambarkannya pada gambar.

```
bboxes = results[0].boxes
draw = ImageDraw.Draw(img)
```

- Mengatur besaran font, membuat skala faktor untuk menyesuaikan ukuran gambar yang dimiliki, dan menyesuaikan ukuran font berdasarkan skala faktor.

```
base_font_size = 20
scale_factor = min(img_width, img_height) / 640
font_size = max(base_font_size, int(base_font_size * scale_factor))
```

- Mendapatkan direktori font, dan memuat font dengan ukuran yang telah disesuaikan.

```
font_path = os.path.join(base_dir, "fonts", "poppins_medium.ttf")
font = ImageFont.truetype(font_path, font_size)
```

- Mendefinisikan warna untuk masing-masing kelas deteksi.

```
class_colors = {
    "Bacterial Diseases": (255, 0, 0),
    "Fungal Diseases": (0, 0, 255),
    "Healthy Fish": (0, 255, 0),
    "Parasitic Diseases": (255, 0, 255),
    "White Tail Diseases": (0, 255, 255)
}
```

- Iterasikan untuk setiap bounding box yang terdeteksi.

```
for i, box in enumerate(bboxes):
    # Mengambil koordinat bounding box (x1, y1, x2, y2).
    x1, y1, x2, y2 = box.xyxy[0].tolist()

    # Mengambil confidence score untuk deteksi ini.
    confidence = box.conf[0].item()

    # Mengambil nama kelas dari hasil deteksi berdasarkan index kelas.
    label = results[0].names[int(box.cls[0].item())]

    # Mengambil warna yang sesuai dari class_colors berdasarkan label kelas.
    color = class_colors.get(label, (255, 255, 255))

    # Menggambar rectangle (bounding box) di gambar dengan warna yang ditentukan.
    draw.rectangle([x1, y1, x2, y2], outline=color, width=3)

    # Membuat teks label dengan confidence score.
    text = f"{label}: {confidence:.2f}"

    # Mendapatkan bounding box untuk teks yang akan ditampilkan.
    text_bbox = draw.textbbox((x1, y1), text, font=font)
    text_width = text_bbox[2] - text_bbox[0]
    text_height = text_bbox[3] - text_bbox[1]

    # Mendefinisikan latar belakang untuk teks label.
    text_bg = [x1, y1 - text_height, x1 + text_width, y1]
    draw.rectangle(text_bg, fill=color)  # Menggambar latar belakang teks.
    draw.text((x1, y1 - text_height), text, fill=(255, 255, 255), font=font)  # Menulis teks label di atas kotak.
```

- Menyimpan gambar yang telah diproses dalam bentuk JPEG ke dalam objek BytesIO.

```
processed_image = io.BytesIO()
img.save(processed_image, format="JPEG")
processed_image.seek(0)
```

- Mengembalikan hasil gambar yang telah diproses tadinya.

```
return processed_image
```

- Memasukkan nama gambar yang akan digunakan untuk objek deteksi nantinya, serta membuat nama keluaran dari hasil gambar yang telah diprediksi.

```
image_path = "bacterial_diseases_koki_002.jpg"
output_path = "xbacterial_diseases_koki_002.jpg"
```

- Menyimpan hasil keluaran yang telah diproses ke dalam direktori tempat program dieksekusi.

```
output_image = process_image(image_path)
with open(output_path, "wb") as f:
    f.write(output_image.read())
```

- **Berikut adalah hasil setelah mengeksekusi kode program di atas:**

Gambar 12. Sebelum Deteksi menggunakan YOLOv9

Gambar 13. Setelah Deteksi menggunakan YOLOv9

B. Algoritma RT-DETR

RTDETR merupakan salah satu model yang digunakan dalam pendeteksian objek. Dalam pendeteksian gambar buram, RT-DETR adalah salah satu solusi yang dapat digunakan. RT-DETR memiliki hasil akurasi yang lebih tinggi dibandingkan dengan YOLOv9[12]. Berikut merupakan cara penggunaan RT-DETR.

- Impor semua modul yang diperlukan.

```
import os
import io
import sys
from PIL import Image, ImageFont, ImageDraw
from ultralytics import RTDETR
```

- Mendapatkan direktori tempat program dieksekusi.

```
base_dir = os.getcwd()
```

- Membuat fungsi untuk memproses gambar menggunakan model RT-DETR, berikan nama fungsi process_image dengan atribut image_path. Mendapatkan direktori model dan memuat model menggunakan RT-DETR.

```
model_path = os.path.join(base_dir, "models", "best_rtdetr.pt")
model = RTDETR(model_path)
```

- Membuka gambar dan mendapatkan ukuran gambar (lebar dan tinggi).

```
img = Image.open(image_path)
img_width, img_height = img.size
```

- Melakukan prediksi deteksi objek pada gambar dengan confidence dan IoU tertentu.

```
results = model.predict(img, conf=0.75, iou=0.5)
```

- Mengambil bounding box dari hasil deteksi dan menggambarkannya pada gambar.

```
bboxes = results[0].boxes
draw = ImageDraw.Draw(img)
```

- Mengatur besaran font, membuat skala faktor untuk menyesuaikan ukuran gambar yang dimiliki, dan menyesuaikan ukuran font berdasarkan skala faktor.

```
base_font_size = 20
scale_factor = min(img_width, img_height) / 640
font_size = max(base_font_size, int(base_font_size * scale_factor))
```

- Mendapatkan direktori font, dan memuat font dengan ukuran yang telah disesuaikan.

```
font_path = os.path.join(base_dir, "fonts", "poppins_medium.ttf")
font = ImageFont.truetype(font_path, font_size)
```

- Mendefinisikan warna untuk masing-masing kelas deteksi.

```
class_colors = {
    "Bacterial Diseases": (255, 0, 0),
    "Fungal Diseases": (0, 0, 255),
    "Healthy Fish": (0, 255, 0),
    "Parasitic Diseases": (255, 0, 255),
    "White Tail Diseases": (0, 255, 255)
}
```

- Iterasikan untuk setiap bounding box yang terdeteksi.

```
for i, box in enumerate(bboxes):
    # Mengambil koordinat bounding box (x1, y1, x2, y2).
    x1, y1, x2, y2 = box.xyxy[0].tolist()

    # Mengambil confidence score untuk deteksi ini.
    confidence = box.conf[0].item()

    # Mengambil nama kelas dari hasil deteksi berdasarkan index kelas.
    label = results[0].names[int(box.cls[0].item())]

    # Mengambil warna yang sesuai dari class_colors berdasarkan label kelas.
    color = class_colors.get(label, (255, 255, 255))

    # Menggambar rectangle (bounding box) di gambar dengan warna yang ditentukan.
    draw.rectangle([x1, y1, x2, y2], outline=color, width=3)

    # Membuat teks label dengan confidence score.
    text = f"{label}: {confidence:.2f}"

    # Mendapatkan bounding box untuk teks yang akan ditampilkan.
    text_bbox = draw.textbbox((x1, y1), text, font=font)
    text_width = text_bbox[2] - text_bbox[0]
    text_height = text_bbox[3] - text_bbox[1]

    # Mendefinisikan latar belakang untuk teks label.
    text_bg = [x1, y1 - text_height, x1 + text_width, y1]
    draw.rectangle(text_bg, fill=color)  # Menggambar latar belakang teks.
    draw.text((x1, y1 - text_height), text, fill=(255, 255, 255), font=font)  # Menulis teks label di atas kotak.
```

- Menyimpan gambar yang telah diproses dalam bentuk JPEG ke dalam objek BytesIO.

```
processed_image = io.BytesIO()
img.save(processed_image, format="JPEG")
processed_image.seek(0)
```

- Mengembalikan hasil gambar yang telah diproses tadinya.

```
return processed_image
```

- Memasukkan nama gambar yang akan digunakan untuk objek deteksi nantinya, serta membuat nama keluaran dari hasil gambar yang telah diprediksi.

```
image_path = "bacterial_diseases_koki_002.jpg"
output_path = "xbacterial_diseases_koki_002.jpg"
```

- Menyimpan hasil keluaran yang telah diproses ke dalam direktori tempat program dieksekusi.

```
output_image = process_image(image_path)
with open(output_path, "wb") as f:
    f.write(output_image.read())
```

- **Berikut adalah hasil setelah mengeksekusi kode program di atas:**

Gambar 14. Sebelum Deteksi menggunakan RT-DETR

Gambar 15. Setelah Deteksi menggunakan RT-DETR

DAFTAR PUSTAKA

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362.

McKinney, W. (2022). *Python for Data Analysis*. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Rougier, N., Rougier, N., Visualization, S., Matplotlib, P., & Rougier, N. P. (2021). *Scientific Visualization : Python + Matplotlib*.

Sapkota, R., Meng, Z., Churuvija, M., Du, X., Ma, Z., & Karkee, M. (2024). *Comprehensive Performance Evaluation of YOLOv10, YOLOv9 and YOLOv8 on Detecting and Counting Fruitlet in Complex Orchard Environments*. *July*.

Tang, S., & Yan, W. (2024). Utilizing RT-DETR Model for Fruit Calorie Estimation from Digital Images. *Information (Switzerland)*, *15*(8).

Zayani, H. M., Ammar, I., Ghodhbani, R., Saidani, T., Sellami, R., Kallel, M., Alsuwaylimi, A. A., Alshammari, K., Alrslani, F. A. F., & Algarni, M. H. (2024). Unveiling the Potential of YOLOv9 through Comparison with YOLOv8. *International Journal of Intelligent Systems and Applications in Engineering*, *2024*(3), 2845–2854.

BIOGRAFI PENULIS

Dwi Nurul Huda, S.T., M.Kom

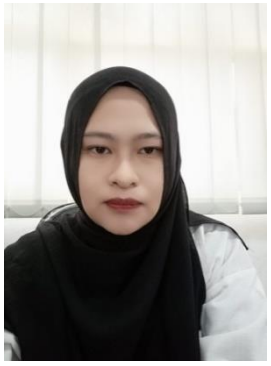
Seorang dosen di Program Studi Sistem Informasi STT Indonesia Tanjung Pinang, lahir di Bandung pada 23 September 1987. Beliau menyelesaikan pendidikan S1 di STMIK Bandung lulus tahun 2009 dan melanjutkan studi S2 di Universitas AMIKOM, lulus pada tahun 2017. Selain mengajar, Penulis juga aktif sebagai penulis buku dengan fokus pada bidang pemrograman dan tata kelola teknologi informasi.

Liza Safitri, S.T., M.Kom

Lahir di Sungai Liat pada 15 Mei 1987, Penulis saat ini menjabat sebagai Ketua Program Studi Teknik Informatika dan dosen di STT Indonesia Tanjung Pinang. Setelah menyelesaikan pendidikan sarjananya di STMIK Bandung lulus tahun 2009 dan melanjutkan studi S2 di Universitas AMIKOM, beliau aktif berkontribusi dalam pengembangan kurikulum dan penelitian di bidang teknologi informasi. Selain itu, Penulis juga telah membuktikan diri sebagai seorang penulis produktif dengan beberapa buku yang membahas topik-topik terkini dalam paket aplikasi dan tata kelola teknologi informasi.

Mochammad Rizki Romdoni, S.Kom., M.T

Seorang penulis dan Dosen Program Studi Teknik Informatika STT Indonesia Tanjung Pinang, kelahiran Garut, 10 Agustus 1983. saat ini tengah mengejar gelar doktor di bidang Teknologi Informasi dan Komunikasi (ICT) di Asia E University, Malaysia. Sebelumnya, beliau telah menyelesaikan studi S2 di Universitas Udayana. Selain itu telah menerbitkan beberapa judul buku yang berkaitan dengan pemrograman dan tata kelola teknologi informasi.



Ade Winari, M.T

Penulis adalah seorang Dosen di STT Indonesia Tanjung Pinang, kelahiran Purwakarta, 02 April 2024, Setelah lulus dari STMIK Bandung pada tahun 2009, kemudian melanjutkan studi di program pasca sarjana Universitas Udayana, tahun 2012. Selain itu telah menerbitkan beberapa judul buku yang berkaitan dengan pengolahan citra dan internet of things.

Abdur Rahman, S.Kom

Mahasiswa program studi Teknik Informatika STT Indonesia Tanjung Pinang, kelahiran Tanjungpinang, 16 Oktober 2001.

Selama menjadi mahasiswa, telah mengikuti berbagai kegiatan seperti program Bangkit dan berkolaborasi dengan penelitian dosen yang dibiayai oleh Pemerintah.

MACHINE LEARNING

untuk deteksi dan identifikasi object
menggunakan python

Buku ini membahas dasar-dasar dan teknik lanjutan dalam machine learning (pembelajaran mesin) yang digunakan untuk mendeteksi dan mengidentifikasi objek melalui pemrograman Python.

Selain membahas teori, buku ini juga berfokus pada implementasi praktis dengan memberikan contoh-contoh kode yang mudah diikuti menggunakan pustaka populer seperti TensorFlow, Keras, dan OpenCV. Pembaca akan mempelajari cara memproses gambar, melatih model deteksi objek, dan mengidentifikasi berbagai objek dalam gambar maupun video.

Buku ini dirancang untuk mahasiswa, peneliti, dan praktisi yang ingin memperdalam pemahaman tentang machine learning dalam aplikasi visi komputer, terutama dalam mendeteksi dan mengenali objek. Penjelasan disampaikan secara terstruktur, mulai dari teori dasar hingga praktik implementasi, sehingga cocok untuk pembaca dengan berbagai latar belakang, baik pemula maupun yang sudah memiliki pengalaman dalam pemrograman dan machine learning.

Dengan buku ini, pembaca diharapkan mampu mengembangkan proyek deteksi objek secara mandiri dan menerapkan teknik machine learning yang efektif untuk berbagai kebutuhan.

PT MEDIA PENERBIT INDONESIA

mediapenerbitindonesia.com
+6281362150605
Penerbit Idn
@pt.mediapenerbitidn

ISBN 978-634-7012-16-6