

Pemrograman Python untuk Pengolahan Citra Digital

Diktat kuliah

DR. ARYA ADHYAKSA WASKITA



Daftar Isi

Daftar Isi	iii
Daftar Gambar	v
Daftar Program	vii
KATA PENGANTAR	ix
1 Instalasi Python	1
1.1 Sejarah singkat	1
1.2 Interpreter Python	1
1.3 Anaconda	7
2 Dasar Pemrograman Python	13
2.1 Pendahuluan	13
2.2 Struktur Data	15
2.2.1 List	16
2.2.2 Tuple	17
2.2.3 Dictionary	18
2.3 Operasi Berkas	19
3 Pustaka Scikit-Image	23
3.1 Pendahuluan	23
3.2 Sub modul I/O	24
4 Histogram dan statistik citra	27
4.1 Pendahuluan	27
4.2 Sub modul exposure	28
4.2.1 <code>equalize_hist</code>	29
4.2.2 <code>equalize_adapthist</code>	33
4.2.3 <code>rescale_intensity</code>	33
4.2.4 <code>adjust_gamma</code>	36
4.2.5 <code>adjust_log</code>	37

5 Perbaikan Citra	39
5.1 Pendahuluan	39
5.2 <i>random_noise</i>	39
5.3 <i>Spatial denoizing</i>	41
5.4 <i>Frequency domain denoizing</i>	42
6 Warna	47
7 Fitur citra	49
7.1 <i>Histogram of Oriented Gradients</i>	49
7.2 <i>Gray Level Co-occurrence Matrix</i>	50
8 Deteksi Tepi	51
8.1 Deteksi tepi <i>canny</i>	51
9 Morfologi Citra	53
10 Ekstraksi Fitur Bentuk	55
10.1 Pendahuluan	55
10.2 Jumlah obyek pada citra	59
10.2.1 <i>Difference of Gaussian</i>	60
10.3 Fitur bentuk	60
Bibliografi	63

Daftar Gambar

1.1	Guido van Rossum	1
1.2	Dialog instalasi <i>interpreter</i> Python	2
1.3	Pilihan paket pendukung sebelum instalasi dilakukan	2
1.4	Dialos selama proses instalasi berlangsung	3
1.5	Dialog tanda selesai instalasi	3
1.6	Lokasi instalasi <i>interpreter</i> Python	4
1.7	<i>Interpreter</i> Python siap digunakan	4
1.9	Hasil upgrade pip	4
1.8	Daftar paket yang terpasang	5
1.10	Instalasi pustaka scikit-image menggunakan pip	5
1.11	Instalasi pustaka <i>dependent</i>	5
1.12	Daftar terakhir paket terpasang	6
1.13	Daftar menu aplikasi pendukung Python	6
1.14	Aplikasi IDLE	6
1.15	Pilihan <i>platform</i> instalasi Anaconda	7
1.16	Dialog pembuka instalasi	8
1.17	Menyetujui kesepakatan	8
1.18	Pilihan pengguna Anaconda	8
1.19	Target instalasi	9
1.20	Menjadikan Anaconda sebagai sistem utama Python	9
1.21	Proses instalasi	9
1.22	Instalasi selesai	10
1.23	10
1.24	Aplikasi Jupyter	11
1.25	Terminal pada aplikasi Jupyter	11
1.26	Python Shell pada aplikasi Jupyter	12
1.27	Aplikasi Spyder	12
2.1	Python shell sedang menerima perintah	14
2.2	Variabel a sebagai obyek	14
2.3	Menampilkan dokumentasi obyek integer a	15
2.4	Menampilkan dokumentasi obyek integer a menggunakan fungsi help	15

2.5	Proses penambahan elemen <code>list</code>	16
2.6	Perbandingan penambahan elemen <code>list</code> menggunakan fungsi (a). <code>append</code> dan (b). <code>extend</code>	17
2.7	Penambahan karakter 'x' ke variabel <code>a</code> di posisi pertama	17
2.8	Mengeluarkan elemen tertentu dari <code>list</code>	17
2.9	Mengeluarkan elemen terakhir dari variabel <code>list</code>	17
2.10	Beberapa operasi yang dilakukan pada variabel <code>tuple</code>	18
2.11	Menambahkan elemen ke variabel <code>dictionary</code>	18
2.12	Mengeluarkan pasangan <i>key-value</i> dari variabel <code>dictionary</code>	19
3.1	Pengeolahan citra untuk pengenalan obyek [Gonzalez and Woods, 2008]	24
3.2	Citra uji <i>baboon</i>	24
3.3	Berkas yang berada di dalam <i>directory skimage</i>	25
3.4	Citra skala keabuan	26
4.1	Histogram ekstraksi RGB citra baboon	28
4.2	Histogram ekstraksi RGB citra baboon dengan sub modul <code>exposure</code>	29
4.3	Citra (a). gelap, (b). terang, (c). kontras rendah, (d) kontras tinggi, masing-masing dengan representasinya histogramnya [Gonzalez and Woods, 2008]	31
4.4	Perbandingan citra baboon dalam (a). skala keabuan dan (b). mengalami proses histogram <i>equalization</i>	32
4.5	Perbandingan histogram citra baboon (a). sebelum dan (b). sesudah mengalami proses histogram <i>equalization</i>	32
4.6	Perbandingan citra baboon (a). sebelum dan (b). sesudah mengalami proses <i>equal_adapthist</i>	33
4.7	Perbandingan histogram citra baboon (a). sebelum dan (b). sesudah mengalami proses <i>equal_adapthist</i>	34
4.8	Perbandingan (a). citra baboon dan (b). histogramnya pada rentang intensitas 0-0.25	35
4.9	Perbandingan (a). citra baboon dan (b). histogramnya pada rentang intensitas 0.35-0.6	35
4.10	Perbandingan (a). citra baboon dan (b). histogramnya pada rentang intensitas 0.35-0.6	36
4.11	Respon koreksi γ [Gonzalez and Woods, 2008]	36
4.12	Perbandingan citra baboon dengan (a) nilai $\gamma < 1$, (b). awal, serta (c). dengan nilai $\gamma > 1$	37
4.13	Faktor koreksi log [Gonzalez and Woods, 2008]	38
4.14	Perbandingan citra baboon (a) hasil transformasi logaritmik, (b). awal, serta (c). <i>inverse</i> logaritmik	38
5.1	Perbandingan (a). citra asli, (b). citra dengan penambahan <i>noise Gaussian</i> dengan nilai <code>mean=0</code> dan <code>var=0.01</code>	41

5.2 Hasil pengurangan <i>noise</i> dengan <i>structuring element</i> (a). <code>square(3)</code> , (b). <code>square(5)</code> , (c). <code>disk(1)</code> dan (d). <code>disk(2)</code>	42
5.3 Fungsi periodik yang merupakan kombinasi dari beberapa fungsi periodik lain	43
5.4 Ilustrasi kurva dengan fungsi gaussian	44
5.5 Ilustrasi perbandingan nilai σ terhadap frekuensi	44
5.6 Hasil penyaringan frekuensi rendah, sedang dan tinggi pada citra RGB	45
7.1 Fitur HOG dari citra baboon dengan nilai orientasi (a). 1 dan (b). 10	50
8.1 Citra (a). awal bunga, yang selanjutnya dideteksi tepi menggunakan fungsi <code>canny</code> dengan variasi nilai σ (b). 1, (c). 3 dan (d). 5	52
10.1 Citra bunga tulip	56
10.2 Citra biner yang dihasilkan dari citra tulips dalam skala keabuan	56
10.3 Citra biner yang dihasilkan dari komponen warna merah dari citra tulips	57
10.4 Citra biner yang dihasilkan dari komponen warna hijau dari citra tulips	58
10.5 Citra biner yang dihasilkan dari komponen warna biru dari citra tulips	59

Daftar Program

2.1	135788.xml	19
2.2	Menysun ulang struktur berkas	20
2.3	Contoh citra tanpa informasi jenis	22
3.1	Membaca/membuka citra	24
4.1	Histogram ekstraksi RGB	27
4.2	Histogram ekstraksi RGB dengan sub modul <code>exposure</code>	28
4.3	Penyamaan histogram	32
4.4	Penyamaan histogram adaptif	33
4.5	Penskalaan intensitas	34
4.6	Koreksi γ	37
4.7	Koreksi logaritmik	38
5.1	Pengurangan <i>noise</i> dengan filter spasial	41
7.1	<i>HOG</i>	49
8.1	Deteksi tepi <code>canny</code>	51
10.1	Melihat kinerja <i>thresholding</i> secara visual	55
10.2	Pembentukan berkas fitur bentuk dari obyek daun Flavia	60

Kata Pengantar

Diktat kuliah ini hanya merupakan pelengkap agar mahasiswa dapat lebih mudah memahami materi pengolahan citra digital. Penggunaan ilustrasi lain dari perangkat lunak berbayar dapat saja diberikan. Tetapi, karena pertimbangan kemandirian dan lisensi, maka saya memutuskan untuk menyusun diktat ini berbasis pada pustaka berlisensi publik dan berbasis bahasa pemrograman Python, **scikit-image**. Python dipertimbangkan karena banyak pustaka ilmiah yang sudah umum digunakan dan terus dikembangkan yang berbasis pada Python. Dalam pengolahan citra, selain **scikit-image**, ada juga **OpenCV** untuk *Computer Vision*. Dalam pembelajaran mesin, **scikit-learn** adalah pustaka yang juga banyak digunakan. Bahkan **tensorflow**, pustaka yang banyak digunakan dalam penelitian *deep learning* juga berbasis pada Python. Saya yakin, dengan mempelajari diktat ini, mahasiswa mampu mandiri dalam penguasaan bahasa pemrograman Python yang pada akhirnya mampu membuat mahasiswa lebih adaptif terhadap pustaka berbasis python, baik untuk tujuan ilmiah maupun bisnis. Mahasiswa pun diharapkan menjadi lebih kreatif dalam melakukan penelitian hingga mengembangkan produk perangkat lunak, maupun prototipe perangkat keras cerdas berbasis Python tanpa harus terbebani masalah lisensi.

Secara umum, diktat ini dibagi ke dalam bagian pendahuluan yang membahas tentang sejarah singkat Python yang dilanjutkan ke bagian instalasi. Instalasi ini, meskipun sangat sederhana, terutama pada sistem operasi Linux, dapat menjadi sangat merepotkan bagi beberapa mahasiswa, terutama ketika mereka menggunakan sistem operasi Windows. Karena itu, instalasi akan dilakukan di sistem operasi Windows. Bagian selanjutnya adalah dasar-dasar pemrograman Python, terutama struktur data (**list**, **tuple** dan **dictionary**), interaksi dengan *file*, hingga mempelajari penggunaan fungsi yang terdapat dalam pustaka tertentu. Sedangkan bagian terakhir dari diktat ini akan sepenuhnya diisi dengan fitur pustaka **scikit-image**, yang saat diktat ini disusun berada pada rilis 0.17.2.

Diktat ini tidak ditujukan untuk menjadi rujukan dalam teknik pengolahan citra. Sehingga penjelasan teoritis terkait pengolahan citra akan diberikan dalam porsi yang sangat minim dan hanya ditujukan sebagai pelengkap saja. Selain itu, dalam diktat ini banyak menggunakan sumber dari situs web dan akan disampaikan secara detil alamat sumber tersebut dalam diktat. Diharapkan, mahasiswa tidak takut mencoba karena ada begitu banyak sumber yang dapat digunakan untuk belajar. Hanya kesungguhan kitalah yang akan menjadi pembeda. Akhirnya, selamat mencoba pengalaman baru.

Serpong, 6 Juni 2021

Dr. Arya Adhyaksa Waskita

Bab 1

Instalasi Python

1.1 Sejarah singkat

Python dibangun oleh Guido van Rossum (Gambar 1.1¹) pada sekitar tahun 1980 di *Centrum Wiskunde & Informatica* (CWI) di Belanda [Hunt, 2019]. Nama Python diambil dari program TV favorit Guido yang berjudul ”Monty Pythons Flying Circus” yang tayang pada kisaran tahun 1969-1974.



Gambar 1.1: Guido van Rossum

1.2 Interpreter Python

Seperti telah dijelaskan di bagian Pengantar, instalasi *interpreter* Python dilakukan di sistem operasi Windows 7. Tahapan instalasi ini mengasumsikan bahwa tidak ada kendala apapun terkait sistem operasi. Selanjutnya mahasiswa diminta untuk mengunduh *interpreter* Python melalui laman <https://www.python.org/downloads/> sesuai kebutuhannya.

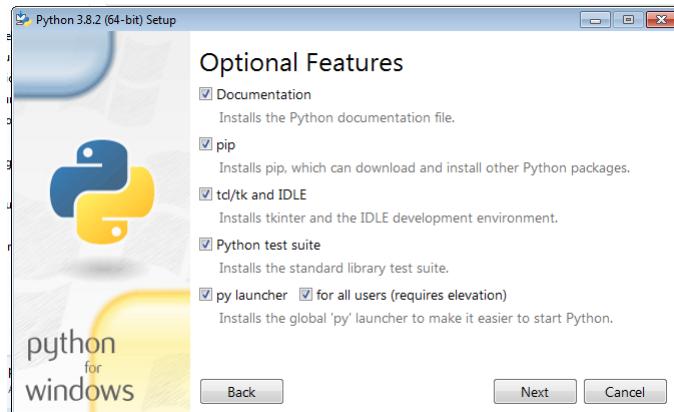
Mengeksekusi unduhan tersebut akan memunculkan dialog seperti pada Gambar 1.2. Pastikan untuk memilih konfigurasi PATH secara otomatis agar ketika proses instalasi selesai, *interpreter* Python dapat dijalankan dari mana saja di sistem komputer masing-masing. Untuk kondisi di mana terjadi kesalahan, akan muncul dialog yang memberi kita kesempatan untuk melihat *log*. Buka log tersebut dan lihat sumber dari kesalahan instalasi yang sedang terjadi.

¹<https://gvanrossum.github.io/images/guido-headshot-2019.jpg>



Gambar 1.2: Dialog instalasi *interpreter* Python

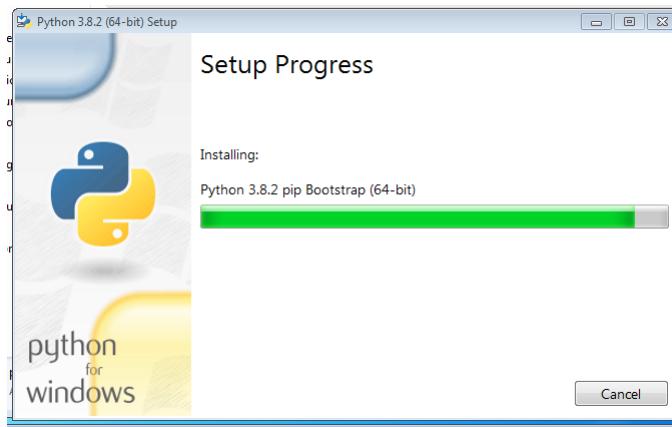
Pilihan opsi *Customize installation* akan menampilkan dialog seperti Gambar 1.3. Pastikan semua pilihan dipilih. Kemudian, selama proses instalasi berlangsung, pengguna akan disuguhkan dialog seperti Gambar 1.4. Tunggu sampai dialog tanda selesai dikeluarkan seperti pada Gambar 1.5.



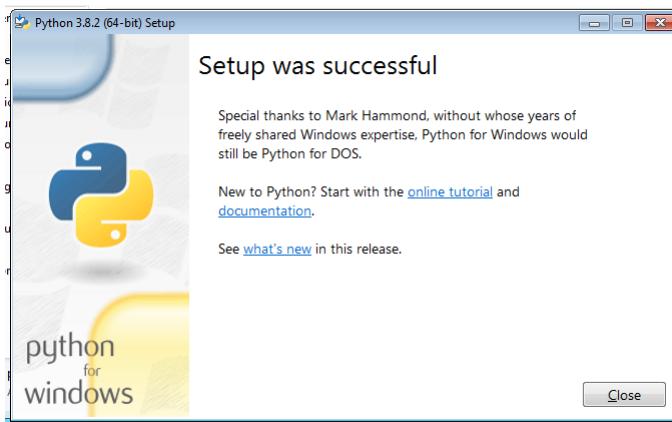
Gambar 1.3: Pilihan paket pendukung sebelum instalasi dilakukan

Seperti telah ditunjukkan pada Gambar 1.2 tentang informasi lokasi *interpreter* Python dilakukan, dapat juga dibuktikan melalui aplikasi CMD seperti Gambar 1.6. Sedangkan *interpreter* Python dapat diujicobakan dengan menuliskan perintah `python` di aplikasi CMD. Akan muncul dialog seperti Gambar 1.7. *Interpreter* Python siap digunakan, ditandai dengan munculnya karakter `>>>`.

Tahapan selanjutnya adalah instalasi pustaka `scikit-image`. Proses instalasinya dilakukan dengan aplikasi pengelola paket Python yang bernama `pip`. Silakan lihat Gambar 1.3. `pip` ada di urutan kedua dari fitur tambahan. `pip` dapat digunakan untuk melihat paket apa saja yang telah terpasang di sistem kita. Caranya dengan menjalankan perintah `python -m pip list` seperti ditunjukkan Gambar 1.8.

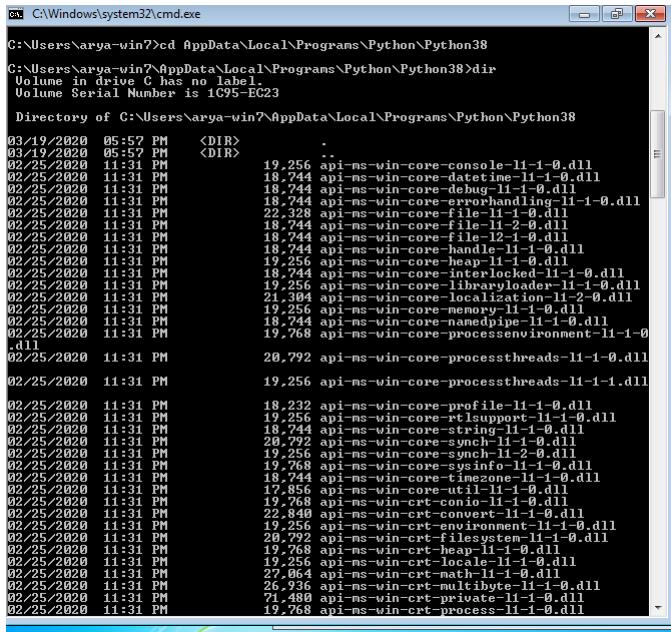


Gambar 1.4: Dialos selama proses instalasi berlangsung



Gambar 1.5: Dialog tanda selesai instalasi

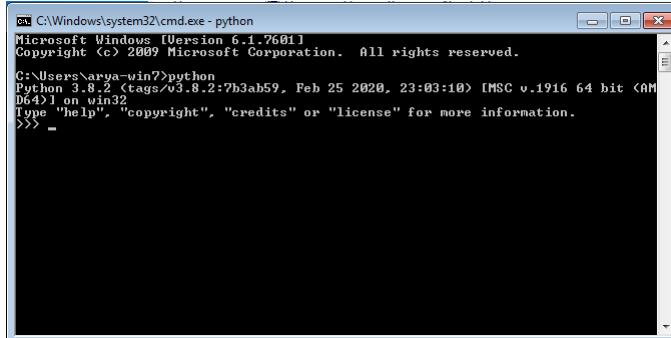
pip dapat juga digunakan untuk meng-upgrade paket yang telah terpasang, bahkan dirinya sendiri. Untuk meng-upgrade paket pip itu sendiri, dapat dilakukan dengan menjalankan perintah `python -m pip install --upgrade pip` seperti Gambar 1.9. Perhatikan versi pip yang ada di Gambar 1.8 dan Gambar 1.9.



```
C:\Windows\system32\cmd.exe
C:\Users\arya-win7\AppData\Local\Programs\Python\Python38>dir
Volume in drive C has no label.
Volume Serial Number is 1C95-EC23

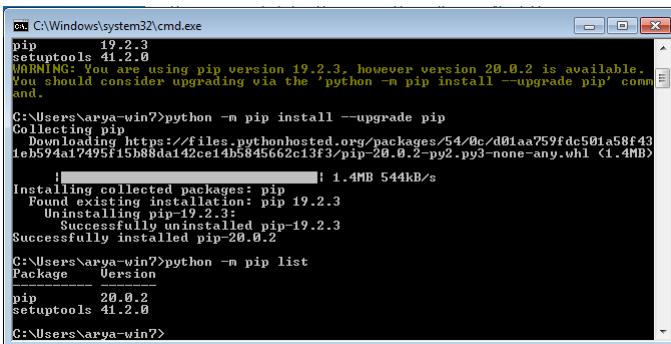
Directory of C:\Users\arya-win7\AppData\Local\Programs\Python\Python38

03/19/2020  05:57 PM    <DIR>          .
03/19/2020  05:57 PM    <DIR>          ..
02/25/2020  11:31 PM           19.256 api-ms-win-core-console-l1-1-0.dll
02/25/2020  11:31 PM           18.744 api-ms-win-core-datetime-l1-1-0.dll
02/25/2020  11:31 PM           19.256 api-ms-win-core-debug-l1-1-0.dll
02/25/2020  11:31 PM           18.744 api-ms-win-core-errorhandling-l1-1-0.dll
02/25/2020  11:31 PM           22.328 api-ms-win-core-file-l1-1-0.dll
02/25/2020  11:31 PM           18.744 api-ms-win-core-file-l1-2-0.dll
02/25/2020  11:31 PM           18.744 api-ms-win-core-file-l2-1-0.dll
02/25/2020  11:31 PM           18.744 api-ms-win-core-handle-l1-1-0.dll
02/25/2020  11:31 PM           19.256 api-ms-win-core-heapsnapshot-l1-1-0.dll
02/25/2020  11:31 PM           19.256 api-ms-win-core-interlocked-l1-1-0.dll
02/25/2020  11:31 PM           21.304 api-ms-win-core-localization-l1-2-0.dll
02/25/2020  11:31 PM           19.256 api-ms-win-core-memory-l1-1-0.dll
02/25/2020  11:31 PM           18.744 api-ms-win-core-namedpipe-l1-1-0.dll
02/25/2020  11:31 PM           19.768 api-ms-win-core-processenvironment-l1-1-0.dll
02/25/2020  11:31 PM           20.792 api-ms-win-core-processsthandles-l1-1-0.dll
02/25/2020  11:31 PM           19.256 api-ms-win-core-processsthandles-l1-1-1.dll
02/25/2020  11:31 PM           18.232 api-ms-win-core-profile-l1-1-0.dll
02/25/2020  11:31 PM           19.256 api-ms-win-core-rtlsupport-l1-1-0.dll
02/25/2020  11:31 PM           19.244 api-ms-win-core-string-l1-1-0.dll
02/25/2020  11:31 PM           20.792 api-ms-win-core-synch-l1-1-0.dll
02/25/2020  11:31 PM           19.256 api-ms-win-core-synch-l1-2-0.dll
02/25/2020  11:31 PM           19.768 api-ms-win-core-sysinfo-l1-1-0.dll
02/25/2020  11:31 PM           18.744 api-ms-win-core-timezone-l1-1-0.dll
02/25/2020  11:31 PM           17.856 api-ms-win-core-util-l1-1-0.dll
02/25/2020  11:31 PM           19.768 api-ms-win-crt-conio-l1-1-0.dll
02/25/2020  11:31 PM           22.840 api-ms-win-crt-convert-l1-1-0.dll
02/25/2020  11:31 PM           19.256 api-ms-win-crt-environment-l1-1-0.dll
02/25/2020  11:31 PM           20.792 api-ms-win-crt-fs-l1-1-0.dll
02/25/2020  11:31 PM           19.768 api-ms-win-crt-heap-l1-1-0.dll
02/25/2020  11:31 PM           19.256 api-ms-win-crt-locale-l1-1-0.dll
02/25/2020  11:31 PM           27.064 api-ms-win-crt-math-l1-1-0.dll
02/25/2020  11:31 PM           26.936 api-ms-win-crt-multibyte-l1-1-0.dll
02/25/2020  11:31 PM           71.488 api-ms-win-crt-private-l1-1-0.dll
02/25/2020  11:31 PM           19.768 api-ms-win-crt-process-l1-1-0.dll
```

Gambar 1.6: Lokasi instalasi *interpreter* Python


```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\arya-win7>python
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 23:03:10) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> -
```

Gambar 1.7: *Interpreter* Python siap digunakan


```
C:\Windows\system32\cmd.exe
pip      19.2.3
setuptools 41.2.0
WARNING: You are using pip version 19.2.3, however version 20.0.2 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Users\arya-win7>python -m pip install --upgrade pip
Collecting pip
  Downloading https://files.pythonhosted.org/packages/54/0c/d01aa759fdc501a58f431eb594a17495f15b88da142ce14b5845662c13f3/pip-20.0.2-py2.py3-none-any.whl (1.4MB)
     |██████████| 1.4MB 544kB/s
Installing collected packages: pip
  Found existing installation: pip 19.2.3
    Uninstalling pip-19.2.3:
      Successfully uninstalled pip-19.2.3
      Successfully installed pip-20.0.2
C:\Users\arya-win7>python -m pip list
Package          Version
pip            20.0.2
setuptools      41.2.0
C:\Users\arya-win7>
```

Gambar 1.9: Hasil upgrade pip

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\arya-win>python
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 23:03:10) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()

C:\Users\arya-win>python -m pip list
Package          Version
----
pip              19.2.3
setuptools       41.2.0
WARNING: You are using pip version 19.2.3, however version 20.0.2 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Users\arya-win>_
```

Gambar 1.8: Daftar paket yang terpasang

Sedangkan untuk memasang pustaka **scikit-image**, jalankan perintah `python -m pip install scikit-image` pada aplikasi CMD seperti Gambar 1.10.

```
C:\Windows\system32\cmd.exe - python -m pip install scikit-image
C:\Users\arya-win>python -m pip install scikit-image
Collecting scikit-image
  Downloading scikit_image-0.16.2-cp38-cp38-win_amd64.whl (25.8 MB)
    ! [  0.0%] 3.4 MB 1.3 MB/s eta 0:00:18
```

Gambar 1.10: Instalasi pustaka **scikit-image** menggunakan pip

Jika ada pustaka lain yang menjadi ketergantungan dari pustaka yang akan diinstal, pip akan melakukan instalasi secara otomatis. Gambar 1.11 menunjukkan proses tersebut. Hal ini akan sangat memudahkan pengguna mengelola pustaka Python yang digunakan.

```
C:\Windows\system32\cmd.exe - python -m pip install scikit-image
C:\Users\arya-win>python -m pip install scikit-image
Collecting scikit-image
  Downloading scikit_image-0.16.2-cp38-cp38-win_amd64.whl (25.8 MB)
    ! [  0.0%] 3.4 MB 1.3 MB/s
Collecting networkx<2.0
  Downloading networkx-2.4-py3-none-any.whl (1.6 MB)
    ! [  0.0%] 1.6 MB 1.3 MB/s
Collecting PyWavelets<0.4.0
  Downloading PyWavelets-1.1.1-cp38-cp38-win_amd64.whl (4.3 MB)
    ! [  0.0%] 4.3 MB 1.1 MB/s
Collecting pillow<4.3.0
  Downloading Pillow-7.0.0-cp38-cp38-win_amd64.whl (2.0 MB)
    ! [  0.0%] 2.0 MB 1.3 MB/s eta 0:00:01
```

Gambar 1.11: Instalasi pustaka *dependent*

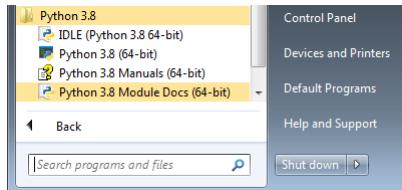
Setelah selesai, kita dapat kembali melihat daftar paket yang terpasang melalui pengelolaan

pip yang ditunjukkan Gambar 1.12.

```
C:\Users\arya-win7>python -m pip list
Package          Version
cycler           0.10.0
decorator        4.4.2
imageio          2.8.0
kiwisolver       1.1.0
matplotlib       3.2.1
networkx         2.4
numpy            1.18.2
pillow           2.0.0
pip              20.0.2
opyparsing        2.4.6
python-dateutil  2.8.1
PyWavelets        1.1.1
scikit-image     0.16.2
scipy            1.4.1
setuptools       41.2.0
six              1.14.0
C:\Users\arya-win7>
```

Gambar 1.12: Daftar terakhir paket terpasang

Menu aplikasi pendukung Python akan muncul seperti Gambar 1.13. Menu kedua pada Gambar 1.13 akan memunculkan aplikasi CMD yang sama dengan yang ditunjukkan Gambar 1.7, tetapi tanpa perlu memanggil perintah `python` terlebih dahulu. CMD secara otomatis akan memunculkan Python `shell` seperti Gambar 1.7.



Gambar 1.13: Daftar menu aplikasi pendukung Python

IDLE adalah antarmuka *interpreter* Python seperti ditunjukkan Gambar 1.14. Dalam Gambar 1.14 juga terlihat bahwa kita berhasil meng-*import* pustaka `scikit-image`, yang dalam IDLE di Windows 7 disebut sebagai `skimage`. Jika Anda sedang menggunakan Ubuntu, kemudian menggunakan pustaka `scikit-image` yang diperoleh dari *repository* Ubuntu (bukan dari `pip`), pustaka `scikit-image` juga di-*import* dengan nama `skimage`. Berhasilnya sebuah pustaka Python di-*import* adalah ketika tidak ada komentar yang muncul setelah perintah `import` tersebut.

```
>>> import skimage
```

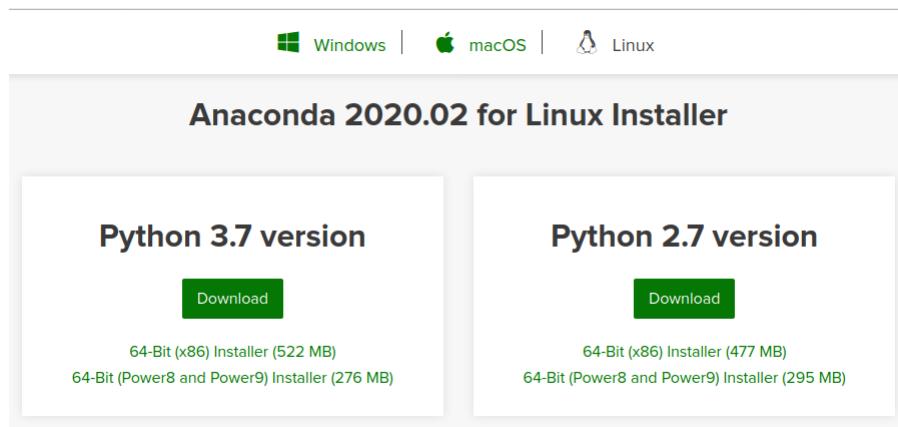
Gambar 1.14: Aplikasi IDLE

Selanjutnya, jika ditemukan petunjuk untuk masuk ke Python `Shell`, Anda dapat menggu-

nakan aplikasi IDLE, atau menggunakan terminal (di Linux)/CMD (di Windows) dengan terlebih dahulu menjalankan perintah `python`.

1.3 Anaconda

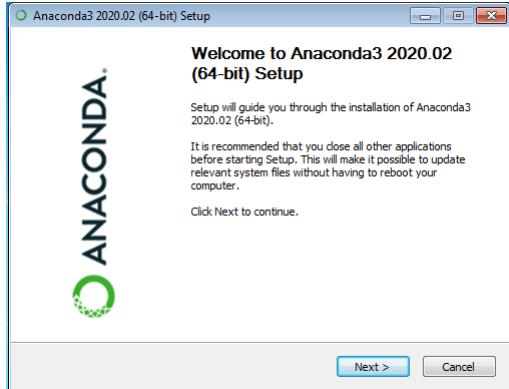
Selain pilihan manual seperti yang telah dijelaskan di Sub bab 1.2, Anaconda bisa menjadi opsi lain yang lebih bersifat otomatis. Saya menyebutnya otomatis karena Anaconda sejumlah pustaka Python, terutama yang banyak digunakan di *Data Mining*, *Machine Learning* atau *Data Science* telah dikemas di dalam Anaconda. Bahkan beberapa editor yang populer untuk Python juga dikemasnya. Anaconda bahkan mengemasnya khusus untuk *platform* yang berbeda. Anda dapat menghubungi alamat <https://www.anaconda.com/> untuk mengunduh aplikasinya. Sesuaikan kebutuhan Anda dengan pilihan yang ada seperti ditunjukkan Gambar 1.15.



Gambar 1.15: Pilihan *platform* instalasi Anaconda

Instalasi Anaconda akan menghadirkan dialog seperti ditunjukkan Gambar 1.16 - Gambar 1.22. Anaconda akan meletakkan pustaka di lokasi `C:\\\\ProgramData\\\\Anaconda3` yang berbeda dengan `pip` seperti terlihat di Gambar 1.19. Sedangkan di Gambar 1.21 terlihat sejumlah pustaka penting seperti `scikit-image` dan `scikit-learn` tengah diinstal.

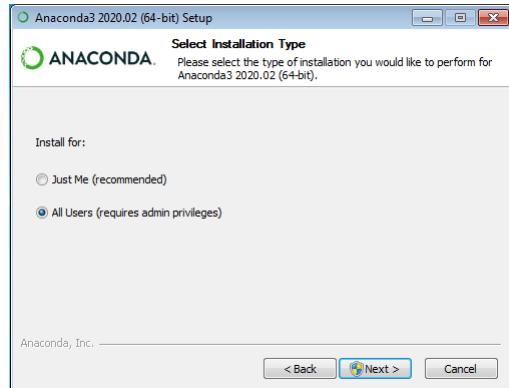
aplikasi ini akan menghadirkan antarmuka seperti tampak



Gambar 1.16: Dialog pembuka instalasi

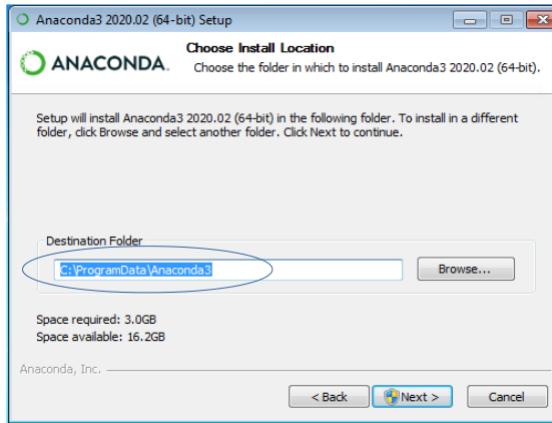


Gambar 1.17: Menyetujui kesepakatan

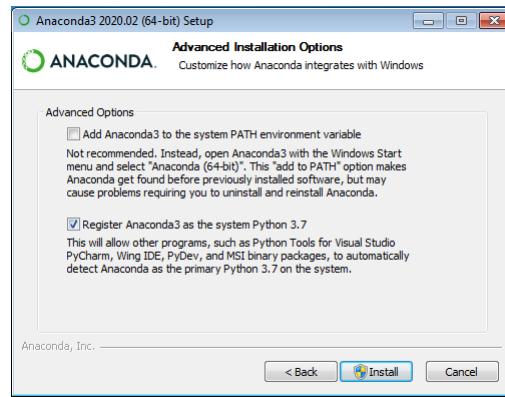


Gambar 1.18: Pilihan pengguna Anaconda

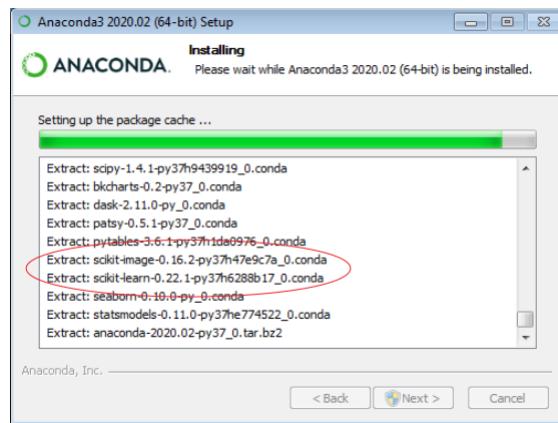
Instalasi Anaconda akan membuat menu seperti pada Gambar 1.23. Di situ terlihat sejumlah



Gambar 1.19: Target instalasi

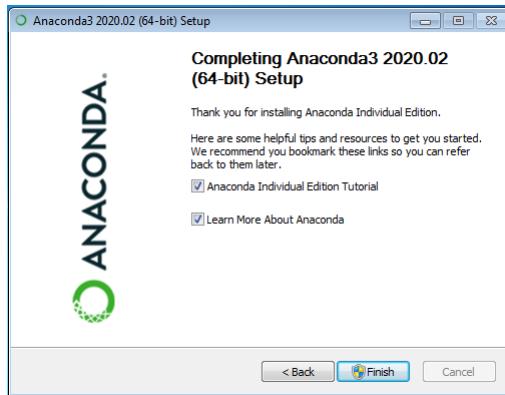


Gambar 1.20: Menjadikan Anaconda sebagai sistem utama Python



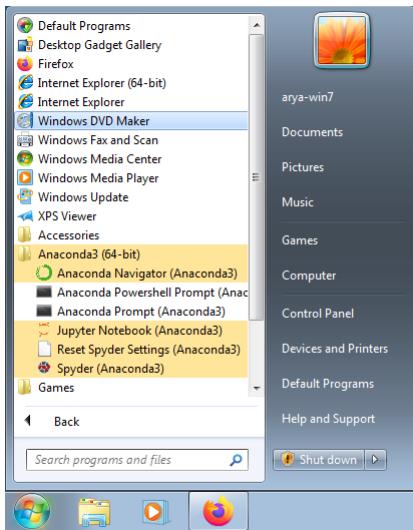
Gambar 1.21: Proses instalasi

aplikasi yang dapat digunakan untuk mengembangkan kode komputer berbasis Python seperti Jupyter dan Spyder. Untuk Jupyter, aplikasi ini akan menghadirkan antarmuka seperti tampak

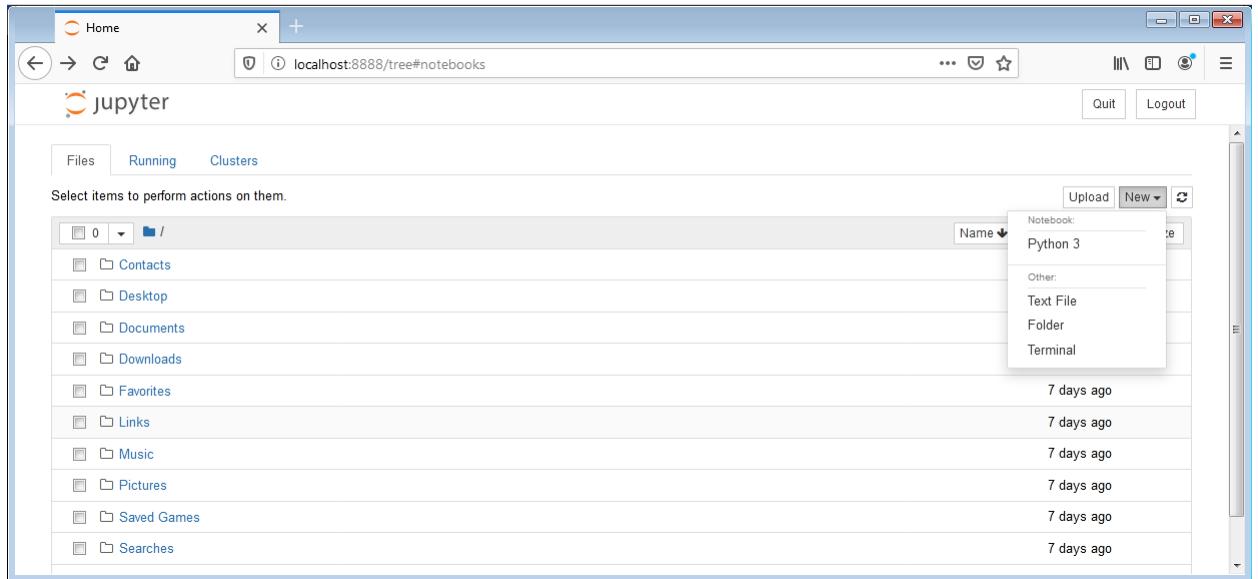


Gambar 1.22: Instalasi selesai

pada Gambar 1.24. Di sisi kanan atas terlihat beberapa opsi antarmuka untuk mengelola proyek Python dengan Jupyter, seperti Terminal Gambar 1.25 atau Python Shell di bawah Jupyter seperti Gambar 1.26 yang perannya seperti IDLE di Gambar 1.14. Sedangkan untuk Spyder, akan tampak antarmuka seperti Gambar 1.27.



Gambar 1.23



Gambar 1.24: Aplikasi Jupyter

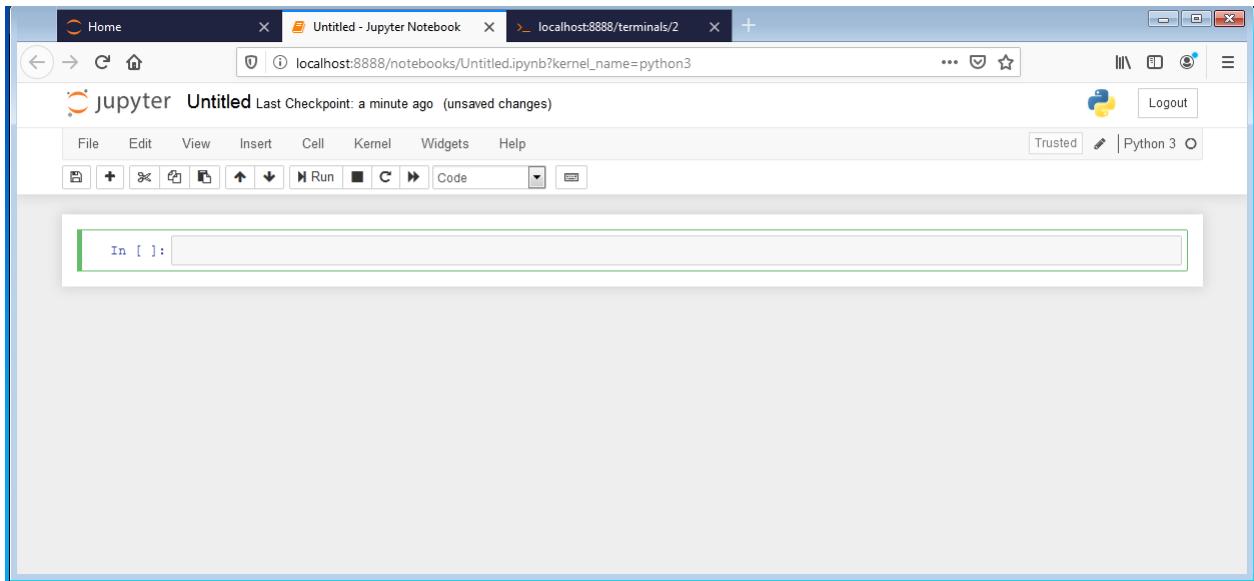
The screenshot shows the Jupyter Terminal application window. The title bar indicates it's connected to 'localhost:8888/terminals/2'. The terminal window has a title 'jupyter'. It displays a Windows PowerShell session. The output shows the following Python code execution:

```

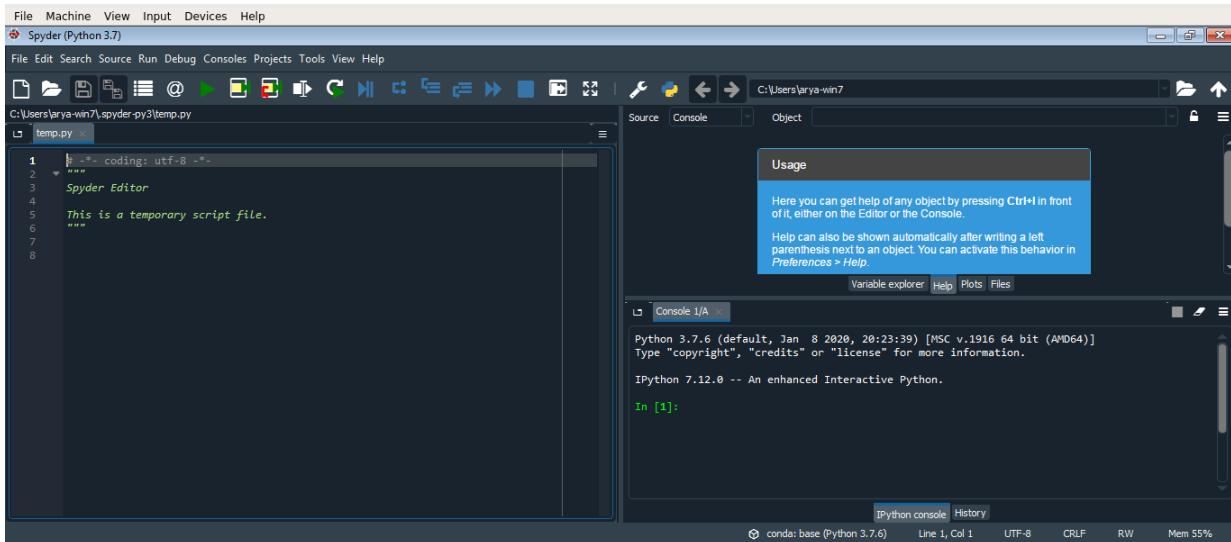
PS C:\Users\arya-win7\Documents> python
Python 3.7.6 (default, Jan  8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> dir(np)
['ALLOW_THREADS', 'AxisError', 'BUFSIZE', 'CLIP', 'ComplexWarning', 'DataSource', 'ERR_CALL', 'ERR_DEFAULT', 'ERR_IGNORE',
 'ERR_LOG', 'ERR_PRINT', 'ERR_RAISE', 'ERR_WARN', 'FLOATING_POINT_SUPPORT', 'FPE_DIVIDEBYZERO', 'FPE_INVALID', 'FPE_OVERFLOW',
 ',', 'FPE_UNDERFLOW', 'False', 'Inf', 'Infinity', 'MAXDIMS', 'MAY_SHARE_BOUNDS', 'MAY_SHARE_EXACT', 'MachAr', 'ModuleDeprecationWarning',
 'NaN', 'NINF', 'NZERO', 'NaN', 'PINF', 'PZERO', 'RAISE', 'RankWarning', 'SHIFT_DIVIDEBYZERO', 'SHIFT_INVALID',
 'SHIFT_OVERFLOW', 'SHIFT_UNDERFLOW', 'ScalarType', 'Tester', 'TooHardError', 'True', 'UFUNC_BUFSIZE_DEFAULT', 'UFUNC_PYV
ALS_NAME', 'VisibleDeprecationWarning', 'WRAP', 'NoValue', '_UFUNC_API', '_NUMPY_SETUP__', '__all__', '__builtins__', '__
cached__', '__config__', '__dir__', '__doc__', '__file__', '__getattr__', '__git_revision__', '__loader__', '__mkl_version__',
 '__name__', '__package__', '__path__', '__spec__', '__version__', '__add_newdoc_ufunc__', '__distributor_init__', '__globa
ls__', '__mat__', '__pytesttester__', '__abs__', '__absolute__', '__absolute_import__', '__add__', '__add_docstring__', '__add_newdoc_ufunc__',
 '__alen__', '__all__', '__allclose__', '__alltrue__', '__amax__', '__amin__', '__angle__', '__any__', '__append__', '__apply_along_axis__',
 '__apply_over_axes__', '__arange__', '__arccos__', '__arccosh__', '__arcsin__', '__arcsinh__', '__arctan__', '__arctan2__', '__arctanh__',
 '__argmax__', '__argmin__', '__argsort__', '__argwhere__', '__around__', '__array__', '__array2string__', '__array_equal__',
 '__array_equiv__', '__array_repr__', '__array_split__', '__array_str__', '__asanyarray__', '__asarray__', '__asarray_chkfinite__',
 '__ascontiguousarray__', '__asfarray__', '__asfortranarray__', '__asmatrix__', '__asscalar__', '__atleast_1d__', '__atleast_2d__',
 '__atleast_3d__', '__average__', '__bartlett__', '__base_repr__', '__binary_repr__', '__bincount__', '__bitwise_and__', '__bitwi
se_not__', '__bitwise_or__', '__bitwise_xor__', '__blackman__', '__block__', '__bmat__', '__bool__', '__bool__', '__broadcast__',
 '__broadcast_arrays__', '__broadcast_to__', '__busday_count__', '__busday_offset__', '__busdaycalendar__', '__byte__', '__byte_bounds__',
 '__bytes0__', '__bytes__', '__c__', '__can__']

```

Gambar 1.25: Terminal pada aplikasi Jupyter



Gambar 1.26: Python Shell pada aplikasi Jupyter



Gambar 1.27: Aplikasi Spyder

Bab 2

Dasar Pemrograman Python

2.1 Pendahuluan

Bahasa pemrograman Python memiliki 4 sifat dasar berikut¹.

1. *Interpreter.* Python diproses oleh *interpreter*, sehingga tidak perlu dikompilasi untuk menjalankannya. Hal ini seperti dijumpai pada bahasa pemrograman PHP yang sangat populer itu.
2. Interaktif. Anda dapat berinteraksi dengan Python dengan memberikannya perintah satu per satu melalui Python **shell**. Setiap perintah yang diberikan langsung akan direspon. Selain itu, Python bersifat *self explained*. Jika ada fungsi dari suatu obyek yang tidak kita ketahui, kita bisa mempelajarinya langsung dari dokumentasi di Python **shell**.
3. Berorientasi obyek. Ada semacam slogan bahwa “*Everything is object in Python*”. Seperti telah dipahami melalui kuliah Rekayasa Perangkat Lunak, orientasi obyek menyebabkan variabel dan fungsi (sering disebut sebagai *state* dan *behavior*) terkemas dalam sebuah obyek, sehingga memudahkan pengelolaan variabel. Fungsi yang melekat pada sebuah obyek juga dapat diturunkan dari satu obyek ke obyek lain sehingga tidak perlu dideklarasi ulang. Namun, fitur orientasi obyek ini pemberlakunya bagi pemrogram tidak seketal seperti yang dilakukan di Java. Jika Java mengharuskan pemrogram mendeklarasikan kelas untuk membuat program yang bahkan sangat sederhana, maka Python tidak mengharuskannya.
4. Bahasa pemrograman untuk pemula. Hal ini disebabkan karena Python sangat sederhana, tidak memerlukan banyak deklarasi yang seringkali menyulitkan, bahkan menakutkan bagi pemula. Selain itu, Python juga mendukung pengembangan aplikasi untuk banyak *platform*, dari aplikasi *embedded* hingga *web* dan *mobile*.

Untuk sifat dasar pertama dan kedua, dapat dilihat ilustrasinya di Gambar 2.1. Dalam Gambar 2.1, Python **shell** dipanggil dengan perintah `python3`. Hal tersebut disebabkan karena

¹<https://www.tutorialspoint.com/python/index.htm>

Ubuntu (yang sedang digunakan adalah Ubuntu 18.04) secara *default* menyertakan Python versi 2.x. Sedangkan untuk Python versi 3.x harus dijalankan dengan perintah `python3`. Di Gambar 2.1 terlihat bahwa ada dua perintah yang diberikan secara berurutan. Tetapi, Python akan meresponnya satu per satu. Sedangkan untuk keluar dari Python `shell`, berikan perintah `exit()`.

```
arya@arya-pc:~$ python3
Python 3.6.9 (default, Nov  7 2019, 10:44:02)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Hello world!')
Hello world!
>>> 3+7
10
>>> exit() -
```

Gambar 2.1: Python `shell` sedang menerima perintah

Untuk sifat dasar ketiga dapat diilustrasikan melalui Gambar 2.2. Kita dapat mengetahui jenis obyek dari variabel `a` dengan fungsi `type(a)`. Sedangkan untuk melihat fungsi dan variabel apa saja yang terkandung pada variabel `a`, kita dapat menggunakan fungsi `dir(a)`. Tetapi, meskipun semuanya di dalam Python adalah obyek, penggunaan Python tidak mengharuskan kita mendeklarasi kelas secara eksplisit. Dengan menuliskan perintah `a=3`, Python tahu bahwa obyek `a` adalah obyek dari kelas `integer`. Bahkan, di Gambar 2.1, operasi aritmatika dapat dilakukan tanpa mendeklarasi variabel.

```
arya@arya-pc:~$ python3
Python 3.6.9 (default, Nov  7 2019, 10:44:02)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> a=3
>>> dir(a)
['__abs__', '__add__', '__and__', '__bool__', '__cell__', '__class__', '__delattr__',
 '__dir__', '__divmod__', '__doc__', '__eq__', '__float__', '__floor__', '__floordiv__',
 '__format__', '__ge__', '__getattr__', '__getnewargs__', '__gt__',
 '__hash__', '__index__', '__init__', '__init_subclass__', '__int__', '__invert__',
 '__le__', '__lshift__', '__lt__', '__mod__', '__mul__', '__ne__', '__neg__',
 '__new__', '__or__', '__pos__', '__pow__', '__radd__', '__rand__', '__rdiv__',
 '__reduce__', '__reduce_ex__', '__repr__', '__rfloordiv__', '__rlshift__',
 '__rmod__', '__rmul__', '__ror__', '__round__', '__rpow__', '__rrshift__',
 '__rshift__', '__rsub__', '__rtruediv__', '__rxor__', '__setattr__', '__str__',
 '__subclasshook__', '__truediv__', '__trunc__', '__xor__',
 '__bit_length__', 'conjugate', 'denominator', 'from_bytes', 'imag', 'numerator',
 'real', 'to_bytes']
>>> type(a)
<class 'int'>
>>> █
```

Gambar 2.2: Variabel `a` sebagai obyek

Di Gambar 2.2 terlihat ada entitas yang diawali dan/atau diakhiri dengan karakter dua *underscore* ('`_`') atau sering disebut sebagai *dunder*² (*double underscore*) oleh komunitas pemrogram Python. Hal tersebut merupakan bagian dari PEP (*Python Enhancement Proposals*) ke-8 tentang *Style Guide for Python Code*³.

Di Gambar 2.2 juga terlihat bahwa obyek `a` memiliki fungsi `__doc__`. Fungsi inilah yang akan memberikan penjelasan singkat kepada kita tentang obyek yang sedang menjadi perhatian.

²<https://dbader.org/blog/meaning-of-underscores-in-python>

³<https://www.python.org/dev/peps/pep-0008/>

Untuk menggunakannya, jalankan perintah `a.__doc__` seperti ditunjukkan Gambar 2.3. Dengan `a` adalah nama variabel untuk obyek yang sedang menjadi perhatian.

```
aryag@aryag-pc:~$ python3
Python 3.6.9 (default, Nov  7 2019, 10:44:02)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> a=3
>>> a.__doc__
"int(x=0) -> integer\nint(x, base=10) -> integer\n\nConvert a number or string to an integer, or return 0 if no arguments\nare given. If x is a number, return x.__int__(). For floating point\nnumbers, this truncates towards zero.\n\nIf x is not a number or if base is given, then x must be a string,\nbytes, or bytearray instance representing an integer literal in the\ngiven base. The literal can be preceded by '+' or '-' and be surrounded\nby whitespace. The base defaults to 10. Valid bases are 0 and 2-36.\nBase 0 means to interpret the base from the string as an integer literal.\n>>> int('0b100', base=0)\n4"
>>> 
```

Gambar 2.3: Menampilkan dokumentasi obyek `integer a`

Format dokumentasi seperti yang ditunjukkan pada Gambar 2.3 sulit untuk dipahami. Pendekatan lain untuk mempelajari dokumentasi sebuah pustaka adalah dengan menggunakan fungsi `help`. Untuk kasus seperti Gambar 2.3, perintah yang dijalankan adalah `help(a)` (**BUKAN** `a.__doc__`). Hasilnya ditunjukkan pada Gambar 2.4. Untuk keluar dari modus dokumentasi tersebut, pengguna tinggal memberi perintah `q` setelah tanda titik dua (Gambar 2.4). Sedangkan untuk melihat isi dokumentasi selanjutnya pengguna dapat menggunkana tombol spasi di papan ketik.

```
Help on int object:

class int(object)
|   int(x=0) -> integer
|   int(x, base=10) -> integer
|
|   Convert a number or string to an integer, or return 0 if no arguments
|   are given. If x is a number, return x.__int__(). For floating point
|   numbers, this truncates towards zero.
|
|   If x is not a number or if base is given, then x must be a string,
|   bytes, or bytearray instance representing an integer literal in the
|   given base. The literal can be preceded by '+' or '-' and be surrounded
|   by whitespace. The base defaults to 10. Valid bases are 0 and 2-36.
|   Base 0 means to interpret the base from the string as an integer literal.
|   >>> int('0b100', base=0)
|   4
|
|   Methods defined here:
|
|   __abs__(self, /)
|       abs(self)
|
|   __add__(self, value, /)
|       Return self+value.
|
|   __and__(self, value, /)
|       Return self&value.
|
|   __bool__(self, /)
|       self != 0
|
|   :| 
```

Gambar 2.4: Menampilkan dokumentasi obyek `integer a` menggunakan fungsi `help`

2.2 Struktur Data

Strukur data yang dimaksud di sini adalah data *array*/larik dan sejenisnya, serta cara penggunaannya. Tidak jarang, fungsi dalam pustaka *scikit-image* menerima argumen atau mengembalikan

likan nilai dalam bentuk data *array* atau sejenisnya.

2.2.1 List

List adalah *array* yang paling banyak digunakan. Kita dapat menyimpan sejumlah nilai, dari tipe apapun ke dalam *list*, bahkan menambah atau mengurangi isinya. Untuk yang pernah mempelajari bahasa pemrograman C, tentu paham betapa sulitnya melakukan hal tersebut di C. Untuk C++ *list* dapat terapkan lebih mudah dengan bantuan *standard template library*⁴

Sebuah variabel *list*, misalnya *a*, diinisiasi dengan perintah *a=[]*. Maka, variabel *a* memiliki sejumlah fungsi yang bisa dilihat dengan perintah *dir(a)*. Diktat ini hanya akan membahas fungsi-fungsi yang sering digunakan saja. Fungsi lain bisa dipelajari sendiri dengan bantuan perintah *help(a.nama_fungsi)*, dengan *a* adalah obyek *list*.

1. *append*. Fungsi ini menambahkan elemen baru ke variabel *list*. Perhatikan Gambar 2.5.

Variabel *a* yang awalnya kosong, kemudian diisi satu per satu menggunakan perintah *append*. Variabel *a* terakhir memiliki dua elemen, masing-masing bertipe *integer* dan *character*.

```
>>> a=[]
>>> a.append(3)
>>> a
[3]
>>> a.append('3')
>>> a
[3, '3']
>>> █
```

Gambar 2.5: Proses penambahan elemen *list*

2. *extend*. Fungsi ini memiliki tugas yang sama dengan *append* dengan sedikit perbedaan.

Perhatikan Gambar 2.6. Di Gambar 2.6(a), variabel *a* ditambahkan sebuah elemen berupa variabel *list* *b* menggunakan fungsi *append*. Variabel *b* yang telah memiliki dua elemen ditambahkan ke variabel *a* sebagai satu elemen. Hal tersebut terlihat dari dijalankannya perintah *len(a)*.

Sementara di Gambar 2.6(b), proses yang sama dilakukan menggunakan fungsi *extend*. Fungsi *extend* akan menambahkan variabel *b* ke dalam variabel *a* tidak sebagai *list* secara keseluruhan, tetapi menambahkan masing-masing elemen variabel *b* ke dalam *a*. Itu sebabnya, hasil penambahan *b* ke dalam *a* membuat *a* saat ini memiliki dua elemen.

Di Program 2.2, ditunjukkan variabel *list* yang salah satunya digunakan sebagai kriteria *looping*, tepatnya dilakukan oleh variabel *hilang* yang dideklarasikan di baris ke-9.

3. *insert*. Selain menambahkan elemen ke variabel *list* di posisi akhir, penambahan elemen juga dapat dilakukan di posisi tertentu. Perhatikan Gambar 2.7. Penambahan karakter 'x' pada posisi pertama dari *list* dilakukan dengan perintah *a.insert(0, 'x')*. Hal ini disebabkan karena indeks dari elemen *list* dimulai dari 0.

⁴https://en.wikipedia.org/wiki/Standard_Template_Library

```
>>> a=[]
>>> b=[1,2]
>>> a.append(b)
>>> a
[[1, 2]]
>>> len(a)
1
(a)                                         (b)
```

Gambar 2.6: Perbandingan penambahan elemen `list` menggunakan fungsi (a). `append` dan (b). `extend`

```
>>> a
[1, 2]
>>> a.insert(0,'x')
>>> a
['x', 1, 2]
>>>
```

Gambar 2.7: Penambahan karakter 'x' ke variabel `a` di posisi pertama

4. `remove`. Selain menambahkan elemen ke variabel `list`, kita dapat juga membuang salah satu elemen yang ada di posisi tertentu di dalam `list`. Perhatikan figurename 2.8. Perintah `remove` digunakan untuk mengeluarkan elemen tertentu dari `list`. Jika ada lebih dari satu elemen yang sama yang akan dikeluarkan, maka elemen terpilih untuk dikeluarkan adalah elemen yang muncul pertama kali pada `list`.

```
>>> a=['b','c','b','a']
>>> a.remove('b')
>>> a
['c', 'b', 'a']
>>> a.remove('b')
>>> a
['c', 'a']
>>>
```

Gambar 2.8: Mengeluarkan elemen tertentu dari `list`

5. `pop`. Fungsi ini akan mengeluarkan elemen terakhir dari `list`. Perhatikan Gambar 2.9.

```
>>> a=[1,2,3,4]
>>> a.pop()
4
>>> a
[1, 2, 3]
>>>
```

Gambar 2.9: Mengeluarkan elemen terakhir dari variabel `list`

2.2.2 Tuple

`Tuple` adalah jenis *array* selain `list` yang di Python dideklarasikan dengan perintah `a=()`. Operasi pada `tuple` lebih cepat dilakukan jika dibandingkan dengan `list`. Hal ini disebabkan

karena `tuple` bersifat statis karena elemen yang ada di dalamnya tidak dapat diubah, kecuali yang bersifat *mutable*. Karena bersifat statis, deklarasi variabel `a=()` tidak akan bermanfaat. Perhatikan Gambar 2.10.

Di Gambar 2.10, sebuah variabel `a` memiliki empat elemen, di mana elemen ke-4 merupakan sebuah `list`. Elemen ke-4 diakses dengan indeks 3 (karena indeks `tuple` dimulai dari 0). Ketika diakses, isi dari elemen ke-4 tersebut dapat diubah karena bersifat *mutable*. Sebaliknya, ketika elemen lain (dalam hal ini elemen ke-2) akan diubah nilainya, Python menolaknya.

```
>>> a
(1, 2, 3, ['x', 'b'])
>>> a[3]
['x', 'b']
>>> a[3][1]='y'
>>> a
(1, 2, 3, ['x', 'y'])
>>> a[1]
2
>>> a[2]=7
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>>
```

Gambar 2.10: Beberapa operasi yang dilakukan pada variabel `tuple`

Untuk Program 2.2, variabel `hilang` yang dideklarasikan di baris ke-9 sebagai variabel `list`, dapat juga dideklarasikan sebagai `tuple`. Hal ini disebabkan karena sepanjang Program 2.2 dijalankan, elemen yang tersimpan di dalam variabel `hilang` tidak berubah.

2.2.3 Dictionary

`Dictionary` merupakan *array* yang elemen penyusunnya merupakan pasangan *key-value*. Setiap elemen akan diindeks berdasarkan *key*. `Dictionary` dideklarasikan menggunakan perintah `a={}`. Untuk menambah elemen ke variabel `dictionary`, gunakan perintah seperti Gambar 2.11.

```
>>> a={}
>>> a['nama']='Arya Adhyaksa Waskita'
>>> a
{'nama': 'Arya Adhyaksa Waskita'}
>>>
```

Gambar 2.11: Menambahkan elemen ke variabel `dictionary`

Kita juga dapat mengeluarkan sebuah elemen dari variabel `dictionary`. Karena elemennya merupakan pasangan *key-value*, maka ketika dikeluarkan, pasangan *key-value* tersebut tidak ada lagi di variabel `dictionary`. Perhatikan Gambar 2.12, elemen yang memiliki *key* berupa karakter `'nama'` akan dikeluarkan menggunakan fungsi `pop`. Karena memerlukan argumen berupa *key*, maka fungsi `pop` dapat mengeluarkan elemen yang posisinya di mana saja di dalam variabel `dictionary`, tidak harus di posisi terakhir.

```

>>> a['npm']='40081'
>>> a
{'nama': 'Arya Adhyaksa Waskita', 'npm': '40081'}
>>> a.pop('nama')
'Arya Adhyaksa Waskita'
>>> a
{'npm': '40081'}
>>

```

Gambar 2.12: Mengeluarkan pasangan *key-value* dari variabel dictionary

2.3 Operasi Berkas

Yang dimaksud sebagai operasi berkas di sub bab ini ditujukan untuk memberikan solusi otomatis melakukan operasi pengolahan citra pada sejumlah besar citra (khususnya), atau berkas digital secara umum. Sebagai ilustrasi, dataset citra terkait tumbuhan salah satunya dapat diperoleh di PlantCLEF2017⁵.

Ketika berkas tersebut diekstrak, kita akan memperoleh directory data sebagai directory teratas dari dataset. Di dalamnya ada cukup banyak directory yang diberi nama berupa deretan angka. Di dalam directory tersebut, akan ada pasangan berkas dengan nama sama dari jenis xml dan jpg. Isi dari berkas berekstensi xml ditunjukkan oleh Program 2.1. Berkas ini berada dalam directory 9982.

Program 2.1: 135788.xml

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <Image>
3      <FileName>135788.jpg</FileName>
4      <Species>Allium cernuum Roth</Species>
5      <Origin>eol</Origin>
6      <Author>2004 robert sivinski, robert sivinski, calphotos</Author>
7      <Content>Flower</Content>
8      <Genus>Allium</Genus>
9      <Family>Liliaceae</Family>
10     <ObservationId>61431</ObservationId>
11     <MediaId>135788</MediaId>
12     <YearInCLEF>PlantCLEF2017</YearInCLEF>
13     <LearnTag>Train</LearnTag>
14     <ClassId>9982</ClassId>
15 </Image>

```

Yang perlu diperhatikan dari element berkas xml di Program 2.1 adalah sebagai berikut.

- **FileName:** elemen ini berisi informasi nama berkas
- **Content:** elemen ini berisi informasi jenis citra. Program 2.1 menunjukkan bahwa citra yang sedang diamati adalah bunga.
- **Family, Genus, Species:** masing-masing menunjukkan tingkatan taksonomi dari tumbuhan. Dalam konteks klasifikasi tanaman, umumnya informasi spesies yang diperlukan. Tetapi, karena berkas pada dataset tidak tersusun dalam taksonomi, maka informasi yang

⁵<http://otmedia.lirmm.fr/LifeCLEF/PlantCLEF2017/TrainPackages/PlantCLEF2017Train1EOL.tar.gz>

disimpan pada elemen tersebut bermanfaat ketika kita ingin menyusun ulang struktur berkas citra berdasarkan taksonominya.

Operasi berkas yang dicontohkan dalam diktat ini adalah menyusun ulang citra berdasarkan jenis citra, apakah itu bunga atau daun. Kemudian di dalam *directory* jenis citra tersebut, citra akan disusun mengikuti hirarki taksonominya. Sehingga akan ada struktur *directory* seperti *Flower/Compositae/Carthamus/Carthamus caeruleus L..* Pada kasus ini, berkas-berkas di dalamnya merupakan bunga dari Family Compositae, Genus Carthamus dan Species *Carthamus caeruleus L..* Programnya disajikan pada Program 2.2. Di dalamnya ada sejumlah operasi berkas seperti pencarian dalam *directory* tertentu, pencarian berkas dengan ekstensi tertentu sampai membuat *directory* dan menduplikasi berkas dari satu *directory* ke *directory* lain.

Program 2.2: Menysun ulang struktur berkas

```

1 import os,shutil
2 import string
3 a=[]
4 berkas='FileName'
5 jenis='Content'
6 family='Family'
7 genus='Genus'
8 spesies='Species'
9 hilang=['<FileName>','</FileName>','</Family>','<Family>','<Content>','</Content>','<Genus>','</Genus>','<
    ↪ Species>','</Species>','\t','\n','']
10 """Jumlah file xml"""
11 k=0
12
13 """Jumlah file xml yang gagal dibaca"""
14 l=0
15 sukses=file('succeed','w')
16 gagal=file('failed','w')
17 for i in os.listdir('.'):
18     if os.path.isdir(i):
19         for j in os.listdir(i):
20             if j.endswith('.xml'):
21                 k=k+1
22                 namafile=str(i)+'/'+str(j)
23                 with open(namafile) as f:
24                     dest1=''
25                     dest2=''
26                     dest3=''
27                     dest4=''
28                     dest=''
29                     src=''
30                     for baris in f:
31                         if berkas in baris:
32                             src=baris
33                             for b in hilang:
34                                 if b in src:
35                                     src=src.replace(b,"")
36                         if jenis in baris:
37                             dest1=baris
38                             for b in hilang:
39                                 if b in dest1:
40                                     dest1=dest1.replace(b,"")
41                         if dest1 == '':
42                             dest1='Undefined'
```

```

43         if family in baris:
44             dest2=baris
45             for b in hilang:
46                 if b in dest2:
47                     dest2=dest2.replace(b,"")
48                 if dest2 == '':
49                     dest2='Undefined'
50
51         if genus in baris:
52             dest3=baris
53             for b in hilang:
54                 if b in dest3:
55                     dest3=dest3.replace(b,"")
56                 if dest3 == '':
57                     dest3='Undefined'
58
59         if spesies in baris:
60             dest4=baris
61             for b in hilang:
62                 if b in dest4:
63                     dest4=dest4.replace(b,"")
64                 if dest4 == '':
65                     dest4='Undefined'
66
67         dest=dest1+'/'+dest2+'/'+dest3+'/'+dest4
68         if not dest in a:
69             a.append(dest)
70             try:
71                 os.makedirs(dest)
72                 print('Directory '+dest+' created')
73             except:
74                 print('Directory '+dest+' not created')
75
76         src=str(i)+'/src'
77         print(dest,src)
78         try:
79             shutil.copy(src,dest)
80             print('File '+src+' copied')
81             sukses.write(namafile+'\n')
82         except:
83             print('File '+src+' not copied')
84             l=l+1
85             gagal.write(namafile+'\n')
86
87         f.close()
88     print(str(l)+' dari '+str(k)+' citra gagal disalin')
89     gagal.close()
90     sukses.close()
91     for i in a:
92         print i

```

Di Program 2.2 disediakan juga pendekripsi kesalahan `try-except` ketika operasi berkas dilakukan. Hal ini dimaksudkan agar jalannya program tidak terhenti ketika kesalahan terjadi. Pengguna cukup mengetahui dari pesan kesalahan yang didefinisikan.

Jika fokus penelitian kita hanya pada bunga, maka kita hanya akan melakukan pengolahan pada citra yang berisi obyek bunga. Demikian juga untuk `Family`, `Genus` maupun `Species` tertentu.

Pada kondisi tertentu, misalnya seperti ditunjukkan Program 2.3, tidak ada informasi jenis

citra. Hal ini terlihat dari tidak adanya isi elemen **Content**. Untuk citra yang salah satu atau seluruh elemennya tidak bernilai, Program 2.2 akan mengelompokkannya sebagai **Undefined**. Untuk citra yang dideskripsikan oleh berkas **xml** seperti Program 2.3 akan disimpan dalam *directory Undefined/Pinaceae/Pinus/Pinus pinea L..*

Program 2.3: Contoh citra tanpa informasi jenis

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <Image>
3      <FileName>301518.jpg</FileName>
4      <Species>Pinus pinea L.</Species>
5      <Origin>eol</Origin>
6      <Author>2016 zoya akulova, zoya akulova, calphotos</Author>
7      <Content></Content>
8      <Genus>Pinus</Genus>
9      <Family>Pinaceae</Family>
10     <ObservationId>227161</ObservationId>
11     <MediaId>301518</MediaId>
12     <YearInCLEF>PlantCLEF2017</YearInCLEF>
13     <LearnTag>Train</LearnTag>
14     <ClassId>252027</ClassId>
15 </Image>
```

Operasi pencarian semua berkas pada *directory* tertentu juga bermanfaat ketika diperlukan operasi pengolahan citra seperti **resize** atau **rescale**. Fitur *Histogram of Oriented Gradients* seperti akan dijelaskan pada sub bab 7.1 sangat dipengaruhi oleh ukuran citra. Sehingga citra yang akan dianalisis harus dalam ukuran yang sama untuk mengetahui aspek pembedanya, yaitu obyek yang terdapat di dalam citra. Tentu tidak efisien jika operasi **resize** atau **rescale** harus dilakukan secara manual.

Bab 3

Pustaka Scikit-Image

3.1 Pendahuluan

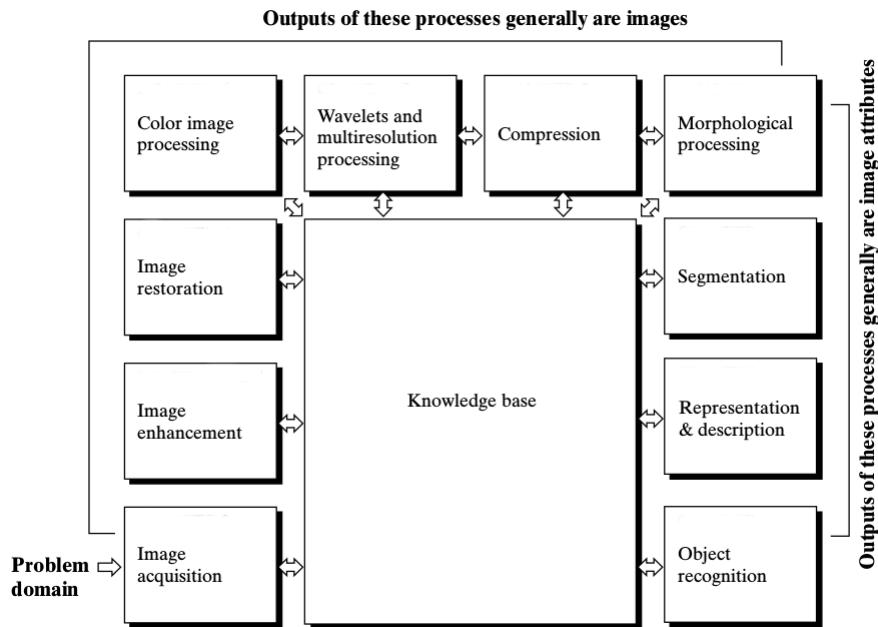
Saat diktat ini disusun, versi stabil terbaru dari pustaka `scikit-image` adalah 0.16.2. Diktat ini disusun berdasarkan penjelasan yang disajikan di <https://scikit-image.org/>. Sedangkan alur penyajiannya didasarkan pada kebutuhan untuk mendapatkan fitur citra.

Seperti dijelaskan [Gonzalez and Woods, 2008] pada Gambar 3.1, pengolahan citra mentar-getkan kemampuan pengenalan obyek. *Image enhancement* dan *Image restoration* digunakan untuk mendapatkan fitur citra yang optimal. Hal ini disebabkan karena pada kondisi tertentu, citra mengandung banyak sekali *noise* yang menyebabkan fiturnya sulit diekstraksi. Hal ini dapat membuat pengenalan obyek di dalam citra tidak maksimal.

Enhancement dan *Restoration* pada citra dapat dilakukan pada domain spasial maupun frekuensi. Pada domain spasial, citra diperlakukan seperti apa adanya, yaitu matriks dengan ukuran sebanyak piksel penyusun, yang berisi intensitas warna pada setiap element matriks. Sedangkan untuk domain frekuensi, citra dianggap sebagai representasi sejumlah gelombang elektromagnetik dengan beragam frekuensi yang menjadi satu. Komponen berfrekuensi tinggi direpresentasi oleh gradasi intensitas warna yang cepat pada domain spasial. Sebaliknya, komponen berfrekuensi rendah direpresentasikan oleh gradasi intensitas warna yang lambat pada domain spasial. *Enhancement* dan *Restoration* citra dapat dilakukan menggunakan transformasi Fourier maupun wavelet (Gambar 3.1).

Tahapan ekstraksi fitur yang tidak menjadi fokus pada diktat ini berdasarkan Gambar 3.1 adalah kompresi. Yang mungkin masih dapat dikategorikan sebagai kompresi feature selection yang merupakan pemilihan fitur hasil ekstraksi yang paling dominan dalam mencirikan suatu obyek di dalam citra. Tetapi, jika yang dimaksud adalah kompresi citra dari sudut pandang ukuran, maka hal tersebut tidak dibahas dalam diktat ini. Kompresi citra untuk mengurangi ukuran, baik untuk mengefisienkan media penyimpanan maupun jalur komunikasi sudah tidak menjadi fokus para peneliti saat ini. Selain karena kapasitas media penyimpanan dan *bandwidth* komunikasi yang semakin besar dan semakin murah, kompresi ukuran citra yang tidak tepat dapat mengurangi informasi penting yang dapat menjadi fitur citra tersebut. Akibatnya, kemampuan pengenalan obyek dalam citra menurun.

Terakhir, fitur yang berhasil diekstraksi dari berbagai metode pengolahan citra akan menjadi masukan bagi pustaka Python lain seperti **scikit-learn** dan **tensorflow**.



Gambar 3.1: Pengeolahan citra untuk pengenalan obyek [Gonzalez and Woods, 2008]

3.2 Sub modul I/O

Penjelasan tentang pengolahan citra berbasis **scikit-image** akan dimulai dengan sub module I/O (*Input/Output*). Pengguna harus memahami cara **scikit-image** membaca sebuah citra dan representasi dari pembacaan tersebut dalam komputer. Sebagai ilustrasi, citra uji berupa hewan *baboon*¹ ditunjukkan pada Gambar 3.2.



Gambar 3.2: Citra uji *baboon*

Gambar 3.2 berukuran 512x512 piksel yang berarti akan ada 3 matriks berukuran 512x512, masing-masing untuk warna merah, hijau dan biru. Setiap elemen matriks akan bernilai integer di antara 0 dan 255. Untuk membaca citra digital, digunakan fungsi **imread**, sebuah fungsi yang terdefinisi di bawah sub modul **scikit-image/io**. Masukkan perintah Program 3.1 berikut di Python **shell** seperti Gambar 1.7.

Program 3.1: Membaca/membuka citra

¹<https://homepages.cae.wisc.edu/~ece533/images/baboon.png>

```

1 >>> from skimage import io
2 >>> img=io.imread('baboon.png')
3 >>> type(img)
4 <class 'numpy.ndarray'>
5 >>> img.shape
6 (512, 512, 3)
7 >>> img2=io.imread('baboon.png', True)
8 >>> img2.shape
9 (512, 512)
10 >>> io.imsave('baboonGS.png', img2)

```

Perintah di baris ke-1 menunjukkan cara untuk meng-*import* pustaka *io*. Di sistem operasi Windows®, lokasi pustakanya ditunjukkan di Gambar 1.2. Sedangkan di sistem operasi GNU-Linux, lokasi pustakanya berada di */home/arya/.local/lib/python3.6/site-packages/skimage*. Di bawahnya, terdapat struktur directory seperti ditunjukkan Gambar 3.3. Terlihat bahwa *io* adalah *sub directory* yang membuat cara pemanggilan pustaka adalah seperti baris ke-1 pada Program 3.1. Cara lainnya adalah dengan mengganti perintah di baris ke-1 dengan *import skimage.io*. *Directory* seperti yang ditunjukkan Gambar 3.3 sama dengan daftar sub modul dari pustaka *scikit-image*². Karenanya, pola pemanggilan pustaka juga memiliki pola yang sama dengan *io*.

```

arya@arya-pc:~/._local/lib/python3.6/site-packages/skimage$ ls
_build.py    exposure    graph    morphology    segmentation    viewer
color        external    _init_.py    __pycache__    setup.py
conftest.py   feature    io        registration    _shared
data         filters    measure    restoration    transform
draw         future    metrics    scripts      util
arya@arya-pc:~/._local/lib/python3.6/site-packages/skimage$

```

Gambar 3.3: Berkas yang berada di dalam *directory* *skimage*

Untuk baris ke-2 Program 3.1, ditunjukkan cara untuk menggunakan fungsi *imread*. Karena pustaka *io* di-*import* menggunakan perintah *from skimage import io*, maka fungsi *imread* digunakan seperti pada baris ke-2. Jika pustaka *io* di-*import* dengan perintah *import skimage.io*, maka fungsi *imread* digunakan dengan perintah *img=skimage.io.imread('baboon.png')*. Perlu diperhatikan, cara pembacaan citra seperti baris ke-2 hanya untuk kondisi di mana citra *baboon.png* berada pada *directory* yang sama dengan lokasi Python shell dipanggil. Variabel *img* pada baris ke-2 menunjukkan pointer ke citra yang dibaca.

Jenis data dari variabel *img* diketahui dengan cara seperti ditunjukkan pada baris ke-3. Terlihat bahwa *img* merupakan variabel *numpy array*. Sedangkan untuk mengetahui ukuran dari *numpy array* digunakan perintah pada baris ke-5. Terlihat bahwa variabel *img* adalah 3 buah matriks berdimensi dua berukuran 512x512. Hal ini menunjukkan bahwa citra yang sedang dibaca terdiri dari 3 komponen warna, masing-masing adalah R (*Red*), G (*Green*), dan B (*Blue*).

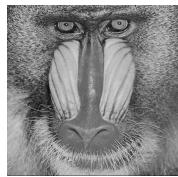
Untuk mengakses komponen warna tertentu (merah, hijau atau biru), gunakan perintah

²<https://scikit-image.org/docs/stable/api/api.html>

`img[:, :, 0]` untuk komponen warna merah serta `img[:, :, 1]` dan `img[:, :, 2]` masing untuk komponen warna hijau dan biru. Pola akses matriksnya sama dengan apa yang dilakukan pada Matlab®.

Untuk membaca citra dalam bentuk skala keabuan, berikan perintah seperti baris ke-7. Baris ke-9 menunjukkan bahwa citra yang dibaca telah dikonversi ke dalam skala keabuan sehingga hanya terdiri dari 1 matriks berukuran 512x512.

Untuk menyimpan citra yang tadi dibaca dalam bentuk skala keabuan, dapat digunakan perintah di baris ke-10. Argumen pertama ('baboonGS.png') adalah nama berkas citra yang akan disimpan, sedangkan argumen kedua (`img2`) adalah matriks citra dalam skala keabuan. Hasilnya ditunjukkan pada Gambar 3.4.



Gambar 3.4: Citra skala keabuan

Sampai di sini, pustaka `numpy` tidak dibahas secara detil. Bagi yang tertarik dapat mempelajarinya secara daring di alamat <https://numpy.org/>. Untuk melihat fungsi apa saja yang dapat dilakukan oleh obyek `numpy` dapat diketahui dengan memberikan perintah `dir(img)` di Python `shell`, dengan `img` adalah obyek dari kelas `numpy`.

Bab 4

Histogram dan statistik citra

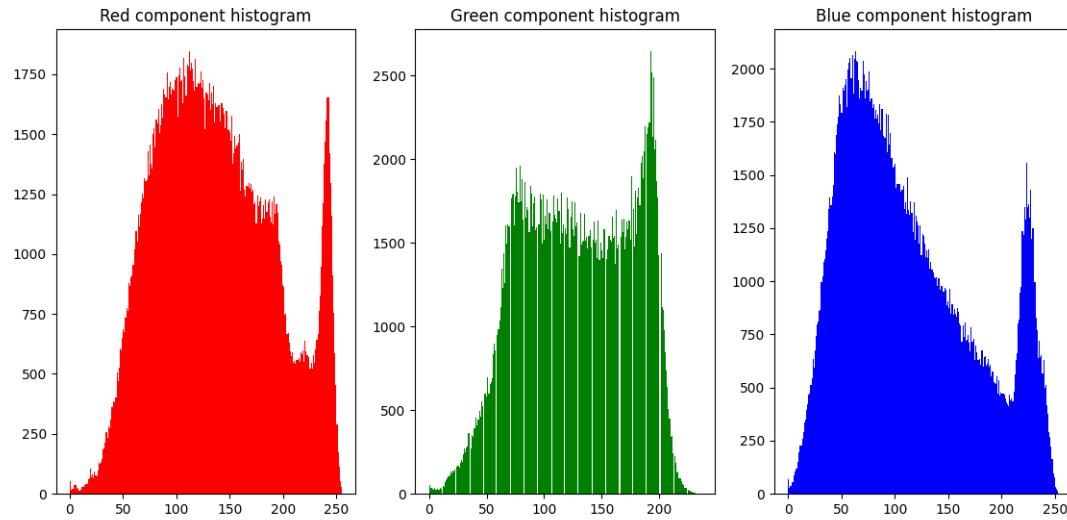
4.1 Pendahuluan

Histogram digunakan untuk menggambarkan statistik citra dalam format visual yang mudah diinterpretasi [Burger and Burge, 2016]. Histogram menunjukkan distribusi frekuensi piksel dengan intensitas tertentu, dari 0 sampai 255. Sebagai ilustrasi, citra yang ditunjukkan pada Gambar 3.2 akan dibaca komponen warnanya kemudian diplot grafik histogramnya. Programnya dapat dilihat di Program 4.1. Hasilnya ditunjukkan pada Gambar 4.1 setelah menjalankan perintah `python3 nama_file.py` di terminal.

Program 4.1: Histogram ekstraksi RGB

```
1 from matplotlib import pyplot as plt
2 from skimage import io
3
4 img=io.imread('../pics/baboon.png')
5 red=img[:, :, 0]
6 row, column=red.shape
7 r=red.reshape(row*column)
8 green=img[:, :, 1]
9 row, column=green.shape
10 g=green.reshape(row*column)
11 blue=img[:, :, 2]
12 row, column=blue.shape
13 b=blue.reshape(row*column)
14 f,(ax1, ax2, ax3)=plt.subplots(1,3)
15 n1, bins1, patches1 = ax1.hist(r, 256, facecolor='red')
16 ax1.set_title('Red component histogram')
17 n2, bins2, patches2 = ax2.hist(g, 256, facecolor='green')
18 ax2.set_title('Green component histogram')
19 n3, bins3, patches3 = ax3.hist(b, 256, facecolor='blue')
20 ax3.set_title('Blue component histogram')
21 plt.show()
```

Baris ke-1 dari Program 4.1 adalah perintah meng-*import* pustaka `matplotlib` yang bertugas membuat plot histogram seperti Gambar 4.1. Kata kunci `as` di baris ke-1 tersebut digunakan untuk membuat alias dari nama pustaka yang di-*import*, dalam hal ini adalah `pyplot`. Kemudian, di baris ke-3, mahasiswa harus berhati-hati dalam meletakkan citra baboon tersebut. Dalam Program 4.1.



Gambar 4.1: Histogram ekstraksi RGB citra baboon

Seperti yang telah disebutkan, citra yang dibaca menggunakan fungsi `io.imread` menghasilkan matriks yang berisi intensitas komponen warna untuk setiap piksel. Matriks tersebut disajikan dalam bentuk `numpy array` (baris ke-4 Program 3.1). Parameter statistik dapat dengan mudah diketahui dari bentuk `numpy array` tersebut. Masukkan perintah `dir(img)` di Python `shell`, dengan `img` adalah representasi `numpy array` dari citra yang dibaca. Kita akan melihat parameter statistik seperti `min` (intensitas terendah), `max` (intensitas tertinggi), `mean` (intensitas rata-rata) atau `var` (*variance* dari semua nilai intensitas). Parameter statistik dari intensitas komponen warna citra dapat dijadikan fitur untuk mengenali obyek tertentu [Rosyani et al., 2018].

4.2 Sub modul exposure

Pustaka `scikit-image` memiliki sub modul khusus yang diberi nama `exposure`. Program 4.2 mengilustrasikan fungsi yang sama dengan dengan Program 4.1. Sedangkan histogram komponen warna merah, hijau dan biru ditunjukkan oleh Gambar 4.2.

Program 4.2: Histogram ekstraksi RGB dengan sub modul `exposure`

```

1 from matplotlib import pyplot as plt
2 from skimage import io
3 from skimage import exposure as ex
4
5 img=io.imread('../pics/baboon.png')
6 red=img[:, :, 0]
7 green=img[:, :, 1]
8 blue=img[:, :, 2]
9 f,(ax1, ax2, ax3)=plt.subplots(1,3)
10 hist1,bin_centers1=ex.histogram(red)

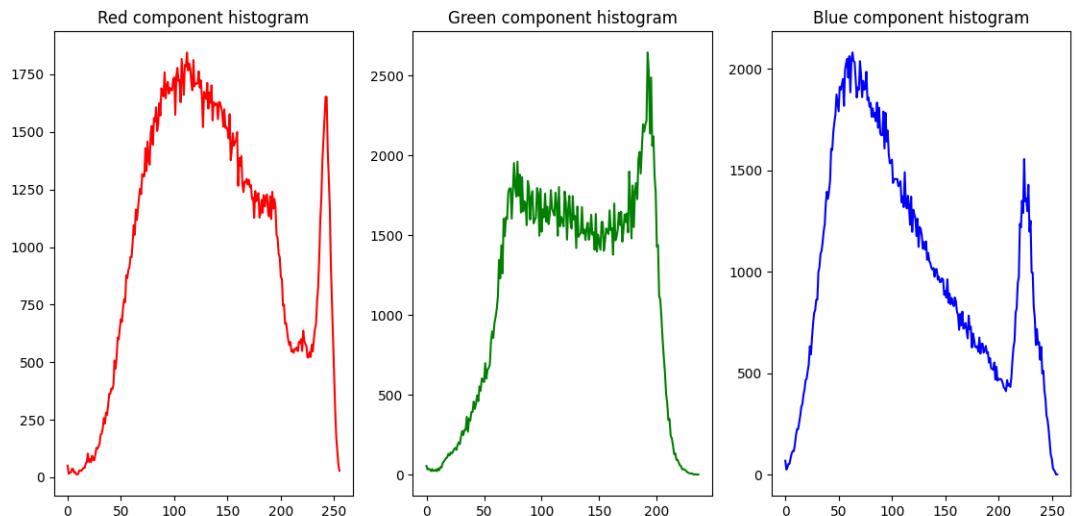
```

```

11 ax1.plot(bin_centers1,hist1,color='red')
12 ax1.set_title('Red component histogram')
13 hist2,bin_centers2=ex.histogram(green)
14 ax2.plot(bin_centers2,hist2,color='green')
15 ax2.set_title('Green component histogram')
16 hist3,bin_centers3=ex.histogram(blue)
17 ax3.plot(bin_centers3,hist3,color='blue')
18 ax3.set_title('Blue component histogram')
19 plt.show()

```

Perbedaan antara Program 4.1 dan Program 4.2 adalah bahwa Program 4.1 mengolah frekuensi intensitas komponen warna menggunakan pustaka `pyplot` yang merupakan sub modul dari `matplotlib`. Sedangkan Program 4.2 mengolah frekuensi intensitas komponen warna menggunakan pustaka `exposure` yang merupakan sub modul dari `scikit-image`. Hasilnya merupakan jumlah piksel yang memiliki intensitas warna pada setiap kanal di antara 0 sampai 255. Jumlah piksel pada setiap kanal intensitas warna tersebut yang selanjutnya diplot oleh pustaka `pyplot`.



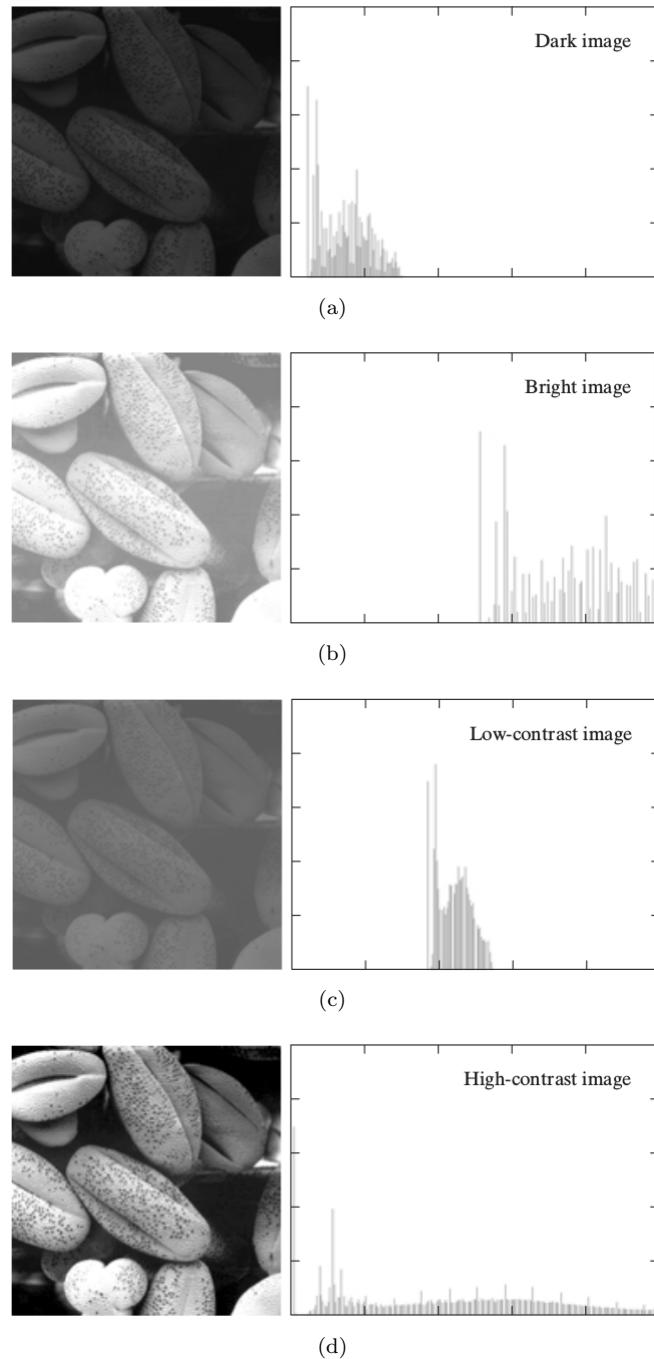
Gambar 4.2: Histogram ekstraksi RGB citra baboon dengan sub modul `exposure`

Selanjutnya, setiap fungsi dalam sub modul exposure akan dicontohkan satu per satu terhadap citra uji baboon (Gambar 3.2).

4.2.1 `equalize_hist`

Citra dengan intensitas warna yang tidak merata menyebabkan kualitas citra menurun. Sebagai contoh, Gambar 4.3(a) menunjukkan citra dengan mayoritas semua pikselnya memiliki intensitas rendah. Secara visual citra sulit diinterpretasi. Kemudian, Gambar 4.3(b) menunjukkan citra yang mayoritas pikselnya memiliki intensitas tinggi. Citra seperti itu pun juga memiliki interpretasi visual yang rendah. Sebaliknya, Gambar 4.3(c) menunjukkan citra yang intensitas piksel-pikselnya yang tidak sama dengan 0 berkumpul pada daerah tertentu. Sama dengan

kedua citra sebelumnya, citra dengan kontras rendah tersebut sulit diinterpretasi secara visual. Terakhir, Gambar 4.3(d) merupakan citra yang piksel-pikselnya memiliki intensitas merata, tidak hanya berkumpul di rentang nilai intensitas tertentu saja. Ternyata citra seperti inilah yang lebih mudah diinterpretasi secara visual. Dalam hal ini, batas antar obyek jelas. Kondisi ini memudahkan citra disegmentasi untuk memilih obyek tertentu untuk selanjutnya dikenali.



Gambar 4.3: Citra (a). gelap, (b). terang, (c). kontras rendah, (d) kontras tinggi, masing-masing dengan representasinya histogramnya [Gonzalez and Woods, 2008]

Selanjutnya, akan ditunjukkan proses penyamaan histogram pada citra baboon (Gambar 3.4). Perhatikan Program 4.3. Hasilnya dapat dilihat di Gambar 4.4(a) dan Gambar 4.4(b). Terlihat bahwa Gambar 4.4(b) yang telah mengalami proses histogram *equalization* memiliki

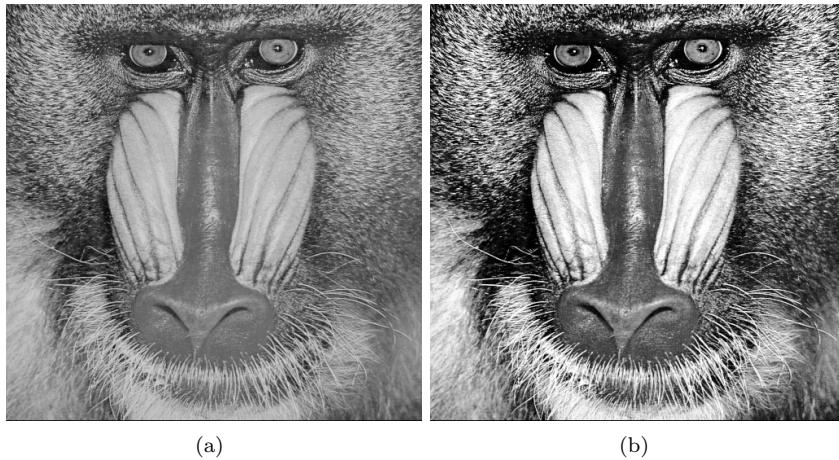
kontras yang lebih baik. Hal ini didukung dengan perbandingan histogram antara sebelum (Gambar 4.5(a)) dan sesudah (Gambar 4.5(b)) proses *equalization* dilakukan.

Program 4.3: Penyamaan histogram

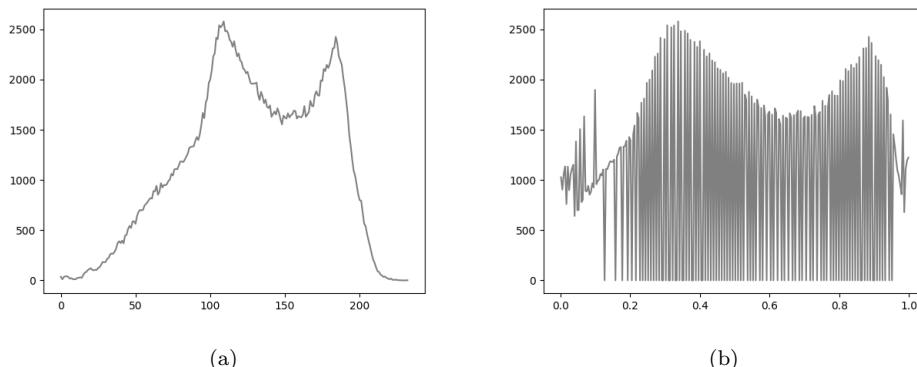
```

1 from matplotlib import pyplot as plt
2 from skimage import io, exposure as ex
3
4 img=io.imread('../pics/baboonGS.png')
5 imgEq=ex.equalize_hist(img,nbins=256)
6 hist,bins=ex.histogram(imgEq)
7 c=plt.plot(bins,hist,color='gray')
8 io.imsave('../pics/baboonGSEq.png',imgEq)
9 plt.show()

```



Gambar 4.4: Perbandingan citra baboon dalam (a). skala keabuan dan (b). mengalami proses histogram *equalization*



Gambar 4.5: Perbandingan histogram citra baboon (a). sebelum dan (b). sesudah mengalami proses histogram *equalization*

4.2.2 equalize_adapthist

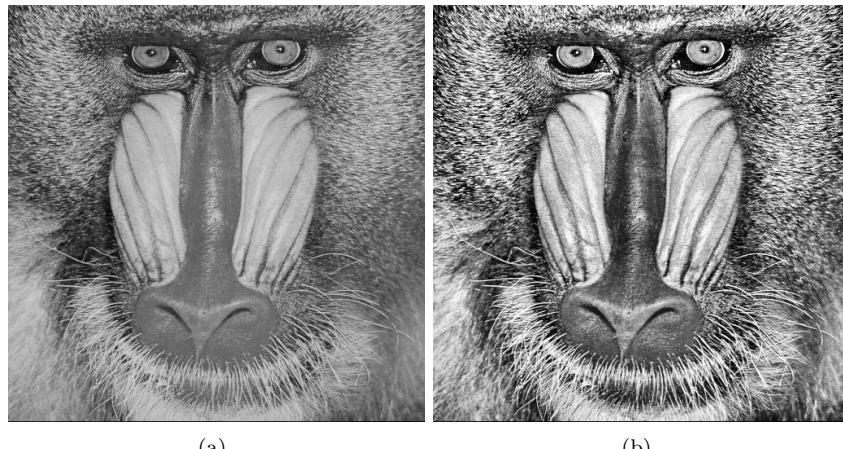
Fungsi ini menjalankan algoritma untuk perbaikan kontras lokal¹ menggunakan histogram yang dihitung pada daerah yang berbeda dari citra. Selanjutnya, detil lokal dapat ditingkatkan baik pada daerah yang lebih gelap maupun yang lebih terang dari kebanyakan daerah di dalam citra. Dengan menjalankan Program 4.4, kita membandingkan citra asli (Gambar 4.6(a)) terhadap citra yang telah mengalami proses `equalize_adapthist` (Gambar 4.6(b)). Sedangkan Gambar 4.7(b) dan 4.7(a) masing-masing merupakan histogram dari citra sebelum dan sesudah mengalami proses `equalize_adapthist`

Program 4.4: Penyamaan histogram adaptif

```

1 from skimage import io
2 from skimage import exposure as ex
3 from matplotlib import pyplot as plt
4
5 img=io.imread('../pics/baboon.png', True)
6 imgEq=ex.equalize_adapthist(img)
7 io.imsave('../pics/baboonAdapthist.png', imgEq)
8 y,x=ex.histogram(imgEq)
9 c=plt.plot(x,y,color='gray')
10 plt.show()

```

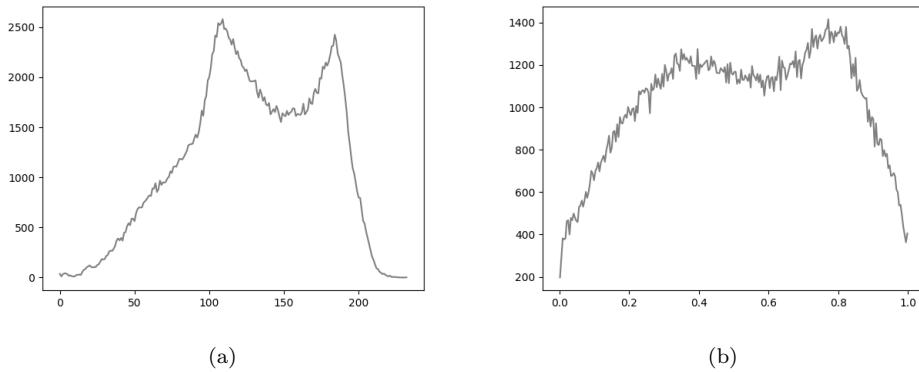


Gambar 4.6: Perbandingan citra baboon (a). sebelum dan (b). sesudah mengalami proses `equal_adapthist`

4.2.3 rescale_intensity

Fitur yang ditawarkan oleh fungsi `rescale_intensity` adalah mengubah rentang intensitas warna, apakah seluruhnya atau sebagian menjadi rentang warna yang baru. Konsepnya mirip dengan normalisasi dalam arti memberi rentang terendah dan tertinggi yang baru terhadap citra. Pengujian berikut akan menggeser rentang citra baboon yang semula berada di kisaran 0 s/d 1 menjadi tiga jenis,masing-masing 0 s/d 0.25, 0.35 s/d 0.6 dan 0.75 s/d 1. Penskalaan seperti

¹https://scikit-image.org/docs/stable/api/skimage.exposure.html#skimage.exposure.equalize_adapthist



Gambar 4.7: Perbandingan histogram citra baboon (a). sebelum dan (b). sesudah mengalami proses *equal_adapthist*

ini disebabkan karena citra yang dibaca adalah citra RGB dan dikonversi dalam skala keabuan seperti ditunjukkan pada baris ke-5 Program 4.5 pada argumen `True`. Jika citra yang dibaca adalah citra dalam skala keabuan, maka rentang nilai intensitasnya adalah nilai riil, bukan hasil normalisasi. Karenanya, ketika citra yang dibaca adalah citra dengan skala keabuan, bukan citra RGB, maka skala intensitas awal maupun target seperti ditunjukkan pada baris ke-7, 12 dan 17 dari Program 4.5 harus disesuaikan. Penjelasan dan ilustrasi yang lebih detil akan disajikan pada Bab 6.

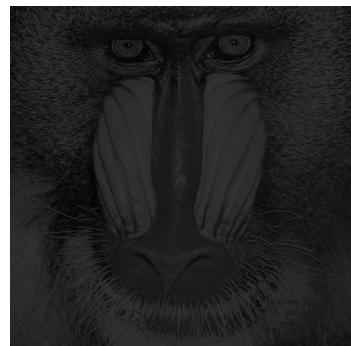
Dengan fungsi `rescale_intensity`, intensitas skala keabuan dari citra akan diskalakan ulang pada rentang yang baru. Hasil yang ditunjukkan Gambar 4.8, 4.9 dan 4.10 mirip dengan yang ditunjukkan Gambar 4.3. Bentuk histogram pada Gambar 4.8(b), 4.9(b) dan 4.10(b) juga sama. Yang berbeda adalah rentang nilai intensitas yang terdapat dalam masing-masing gambar. Pada Gambar 4.8(a), hanya terdapat piksel dengan intensitas pada rentang nilai antara 0 - 0.25. Sedangkan pada Gambar 4.9(a) dan 4.10(a) masing-masing hanya memiliki rentang nilai intensitas 0.35 - 0.6 dan 0.75 - 1.

Program 4.5: Penskalaan intensitas

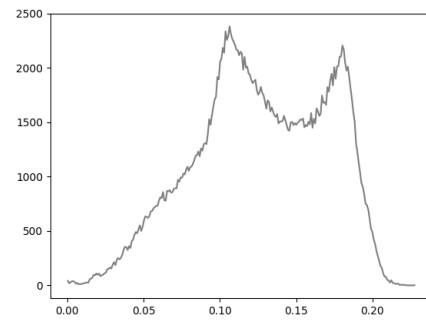
```

1  from skimage import io
2  from skimage import exposure as ex
3  from matplotlib import pyplot as plt
4
5  img=io.imread('../pics/baboon.png',True)
6  f,(ax1, ax2, ax3)=plt.subplots(1,3)
7  bb1=ex.rescale_intensity(img, in_range=(0,1), out_range=(0,0.25))
8  io.imsave('../pics/baboonScale1.png', bb1)
9  hist1,bin_centers1=ex.histogram(bb1)
10 ax1.plot(bin_centers1,hist1,color='gray')
11 ax1.set_title('0 to 0.25')
12 bb2=ex.rescale_intensity(img, in_range=(0,1), out_range=(0.35, 0.6))
13 io.imsave('../pics/baboonScale2.png', bb2)
14 hist2,bin_centers2=ex.histogram(bb2)
15 ax2.plot(bin_centers2,hist2,color='gray')
16 ax2.set_title('0.35 to 0.6')
17 bb3=ex.rescale_intensity(img, in_range=(0,1), out_range=(0.75, 1))
```

```
18 io.imsave('..../pics/baboonScale3.png', bb3)
19 hist3,bin_centers3=ex.histogram(bb3)
20 ax3.plot(bin_centers3,hist3,color='gray')
21 ax3.set_title('0.75 to 1')
22 plt.show()
```



(a)

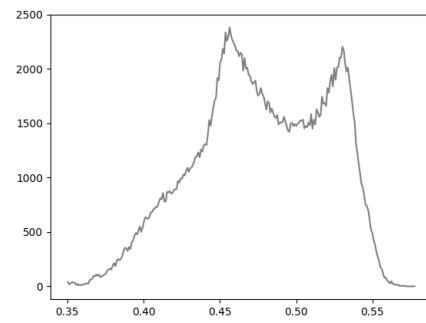


(b)

Gambar 4.8: Perbandingan (a). citra baboon dan (b). histogramnya pada rentang intensitas 0-0.25

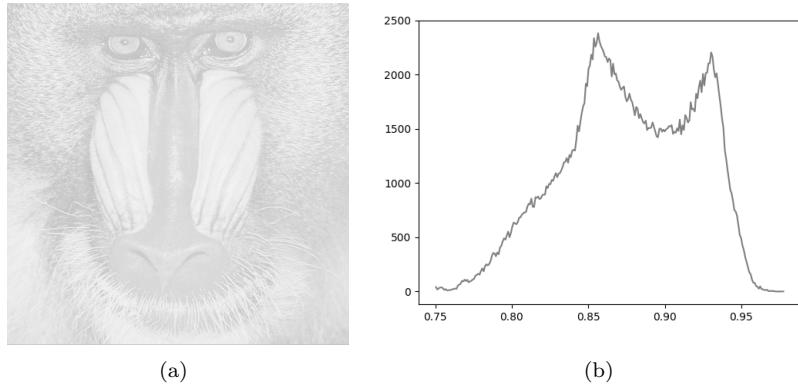


(a)



(b)

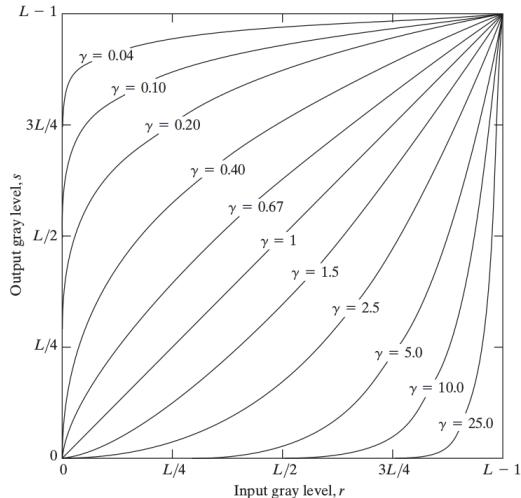
Gambar 4.9: Perbandingan (a). citra baboon dan (b). histogramnya pada rentang intensitas 0.35-0.6



Gambar 4.10: Perbandingan (a). citra baboon dan (b). histogramnya pada rentang intensitas 0.35-0.6

4.2.4 adjust_gamma

Koreksi γ perlu diperhatikan ketika citra akan disajikan pada perangkat *display* yang berbeda. Hal ini akan memperbaiki faktor *intensity to voltage response* mengikuti persamaan $s = cr^\gamma$. Response koreksi γ diilustrasikan oleh Gambar 4.11. Nilai intensitas setiap piksel akan ditransformasikan menjadi nilai intensitas baru mengikuti fungsi di Gambar 4.11.



Gambar 4.11: Respon koreksi γ [Gonzalez and Woods, 2008]

Program 4.6 menunjukkan kode program untuk koreksi γ . Ada dua variasi nilai γ , masing-masing 0.5 dan 2. Nilai γ yang lebih besar dari satu akan membuat citra semakin tampak gelap (Gambar 4.12(c)). Sedangkan nilai γ yang lebih kecil dari satu akan membuat citra semakin tampak terang (Gambar 4.12(a)). Dalam Program 4.6, faktor multiplikasi $c = 1$ sehingga tidak perlu didefinisikan secara eksplisit (hanya diperlukan 2 argumen). Untuk faktor multiplikasi

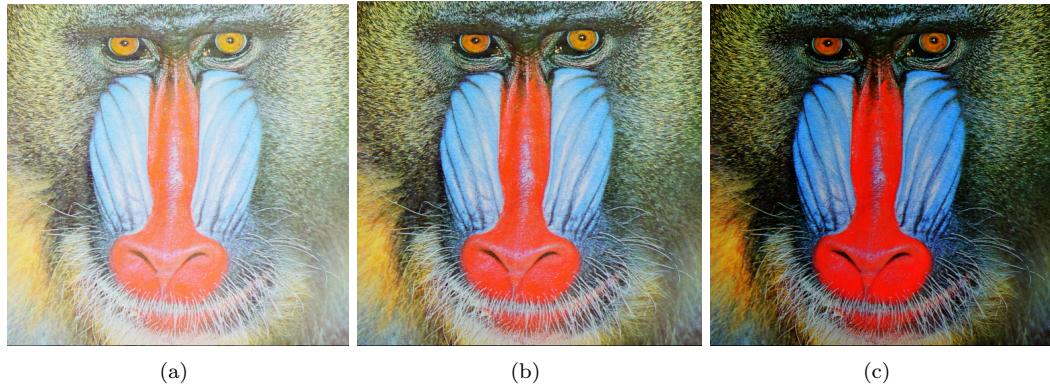
yang tidak sama dengan satu, maka diperlukan 3 argumen untuk melakukan koreksi γ .

Program 4.6: Koreksi γ

```

1 from skimage import io
2 from skimage.exposure import adjust_gamma as ag
3
4 img=io.imread('../pics/baboon.png')
5 img1=ag(img,0.5)
6 io.imsave('../pics/baboonGammaHalf.png', img1)
7 img2=ag(img,2)
8 io.imsave('../pics/baboonGammaDouble.png', img2)

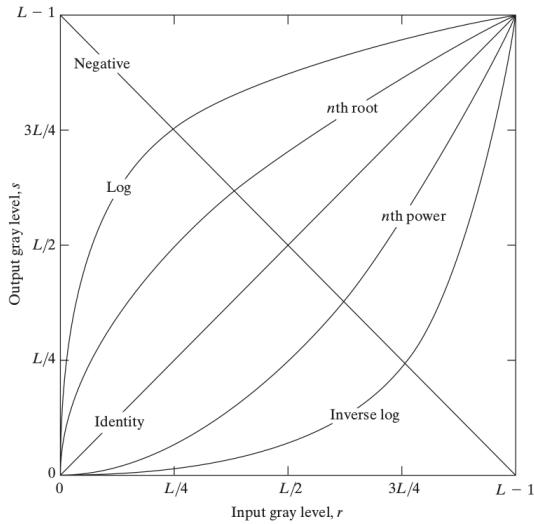
```



Gambar 4.12: Perbandingan citra baboon dengan (a) nilai $\gamma < 1$, (b). awal, serta (c). dengan nilai $\gamma > 1$

4.2.5 adjust_log

Fungsi lain untuk mentrasformasikan nilai intensitas piksel adalah fungsi koreksi logaritmik yang diilustrasikan dalam Gambar 4.13. Transformasi logaritmik akan mengikuti formula $s = c \log(1+r)$, sedangkan transformasi inverse logaritmik akan mengikuti formula $s = c(2^I - 1)$. Program 4.7 menunjukkan transformasi logaritmik pada baris ke-5 dan menghasilkan citra hasil transformasi seperti ditunjukkan Gambar 4.14(a)). Sedangkan transformasi *inverse* logaritmik ditunjukkan pada baris ke-7 dan menghasilkan citra hasil transformasi seperti ditunjukkan Gambar 4.14(c). Keduanya menggunakan faktor multiplikasi sama dengan satu. Hasil transformasi yang diperoleh tidak sesignifikan terhadap apa yang diperoleh menggunakan transformasi γ . Tetapi kecenderungannya masih sama. Untuk transformasi logaritmik pengaruhnya seperti transformasi $\gamma < 1$, yaitu membuat citra menjadi lebih terang dari sebelumnya. Sedangkan transformasi *inverse* logaritmik sama pengaruhnya seperti transformasi $\gamma > 1$, yaitu membuat citra menjadi lebih gelap dari sebelumnya. Melihat nilai rerata intensitas piksel (baris ke-8 pada Program 4.7) dapat membuktikan tingkat terang-gelap secara kuantitatif.



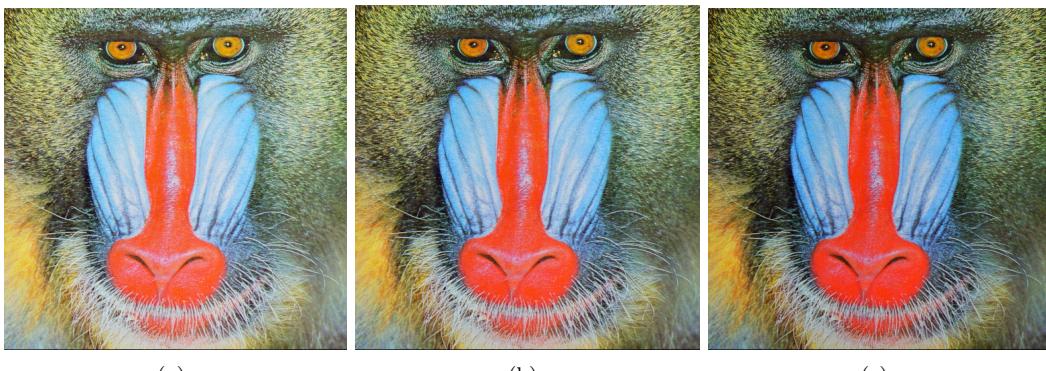
Gambar 4.13: Faktor koreksi log [Gonzalez and Woods, 2008]

Program 4.7: Koreksi logaritmik

```

1 from skimage import io
2 from skimage.exposure import adjust_log as al
3
4 img=io.imread('../pics/baboon.png')
5 img1=al(img)
6 io.imsave('../pics/baboonLog.png', img1)
7 img2=al(img, inv=True)
8 print(img1.mean(), img.mean(), img2.mean())
9 io.imsave('../pics/baboonInvLog.png', img2)

```



Gambar 4.14: Perbandingan citra baboon (a) hasil transformasi logaritmik, (b). awal, serta (c). *inverse* logaritmik

Bab 5

Perbaikan Citra

5.1 Pendahuluan

Perbaikan citra dilakukan dengan cara menghilangkan/mengurangi degradasi citra yang dapat disebabkan karena *noise* atau *blurring*. Degradasi sebuah citra digital dapat dimodelkan dengan persamaan (5.1) [Gonzalez and Woods, 2008].

$$g(x, y) = (f(x, y) * h(x, y)) + n(x, y) \quad (5.1)$$

Dari persamaan (5.1), diketahui

- $g(x, y)$ adalah citra terdegradasi
- $f(x, y)$ adalah citra asli
- $h(x, y)$ adalah filter spasial
- $n(x, y)$ adalah *noise*

Dengan demikian, secara teoritis, citra asli dapat kembali diperoleh dengan menggunakan persamaan (5.2). Tentu dengan mengetahui fungsi *noise* yang menyebabkan citra terdegradasi.

$$f(x, y) = \frac{g(x, y) - n(x, y)}{h(x, y)} \quad (5.2)$$

5.2 random_noise

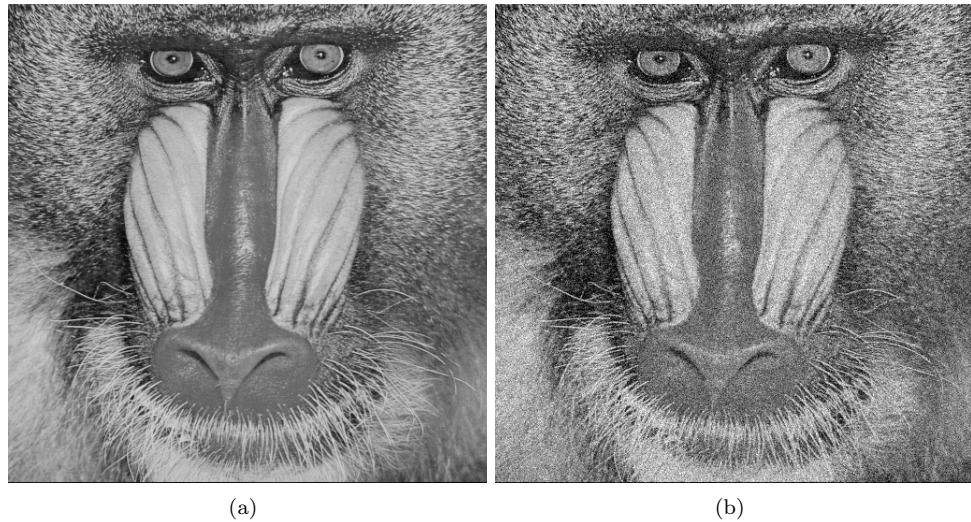
Pustaka `scikit-image` menyediakan fungsi degradasi citra dengan nama `random_noise` yang diletakkan di bawah sub modul `util`. Fungsi tersebut memfasilitasi pemberian beberapa jenis *noise* berikut karakteristiknya masing-masing. Berikut adalah beberapa contoh pemberian *noise*.

Fungsi `random_noise` menerima argumen berikut ini.

1. `image` adalah citra yang akan diberi *noise*, fungsi akan secara otomatis melakukan normalisasi nilai intensitas citra pada rentang 0 s/d 1.

2. **mode** adalah jenis *noise* yang dapat diberikan dalam tipe data **str**. Jenis *noise* tersebut adalah
 - 'gaussian' tambahan *noise* yang memenuhi distribusi Gaussian
 - 'localvar' tambahan *noise* yang memenuhi distribusi Gaussian dengan tambahan informasi *local variance* pada setiap titik citra. Jika argumen **mode** bernilai 'localvar', argumen **local_vars** harus diberikan
 - 'poisson' tambahan *noise* yang memenuhi distribusi Poisson
 - 'salt' tambahan *noise* yang mengubah titik piksel yang dipilih secara acak dengan nilai 1
 - 'pepper' tambahan *noise* yang merupakan kebalikan dari 'salt'. *Noise* 'pepper' mengubah titik piksel yang dipilih secara acak dengan nilai 1 (untuk jenis *unsigned*) atau -1 (untuk jenis *signed*)
 - 's&p' adalah tambahan *noise* yang merupakan gabungan jenis 'salt' dan 'pepper'
 - 'speckle' adalah tambahan *noise* yang mengubah nilai intensitas citra pada setiap piksel menjadi $x + (n * x)$, dengan x adalah intensitas citra asli dan n adalah fungsi *uniform noise* dengan tambahan argumen **mean** (rerata) dan **var** (varians)
3. **seed** bertipe **int** dan bersifat opsional. Tetapi, jika nilai argumen diberikan, nilai tersebut akan digunakan untuk menghasilkan nilai acak untuk membangkitkan *noise*
4. **clip** bertipe **bool** dan bersifat opsional. Jika tidak diberikan, maka fungsi **random_noise** akan memberikan nilai **True** pada argumen ini. Dampaknya, hasil pemberian *noise* dari jenis 'speckle', 'poisson' dan 'gaussian' membuat citra tetap memiliki rentang nilai intensitas [-1,1] (*unsigned*). Sebaliknya, jika secara aktif diberikan nilai **False**, citra hasil pemberian *noise* dapat memiliki nilai intensitas di luar rentang normalnya
5. **mean** adalah nilai rerata dari distribusi *noise* 'gaussian' dan 'speckle'. Argumen ini bertipe **float** dan bersifat opsional. Jika tidak diberikan, fungsi **random_noise** akan memberikan nilai 0 untuk argumen ini.
6. **var** adalah nilai varians untuk distribusi *noise* 'gaussian' dan 'speckle'. Argumen ini menerima nilai **float** dan bersifat opsional. Jika tidak diberikan, fungsi **random_noise** akan memberikan nilai 0.01
7. **local_vars** adalah argumen yang diperlukan ketika pengguna memberikan *noise* dengan **mode** 'localvar'
8. **amount** adalah argumen yang digunakan untuk menentukan proporsi jumlah piksel yang akan dikonversi nilai intensitasnya ketika **mode** 'salt', 'pepper' dan 's&p' digunakan.
9. **salt_vs_pepper** adalah argumen yang digunakan untuk menentukan proporsi jumlah piksel yang nilai intensitasnya akan dikonversi menjadi 1 ('salt') dan 0/-1 ('pepper'). Argumen ini diberikan ketika **mode** bernilai 's&p'

Gambar 5.1 menunjukkan perbandingan citra asli dan citra dengan penambahan *noise* dari fungsi Gaussian.



Gambar 5.1: Perbandingan (a). citra asli, (b). citra dengan penambahan *noise* Gaussian dengan nilai `mean=0` dan `var=0.01`

5.3 Spatial denoizing

Pustaka *scikit-image* menyediakan fungsi pengurangan *noise* secara spasial pada sub modul `skimage.filters.rank`. Sub bab ini akan menunjukkan hasil dari pengurangan *noise* secara spasial berbasis sejumlah opsi *structuring element* (dibahas di bab 9).

Program 5.1 menjelaskan proses pengurangan *noise* dengan filter `mean` dan *structuring element* berupa matriks segiempat dengan nilai elemen penyusunnya yang membentuk segiempat (`square`) dan lingkaran (`disk`). Sedangkan Gambar 5.2 menunjukkan perbedaan hasil pengurangan *noise* terhadap Gambar 5.1(b).

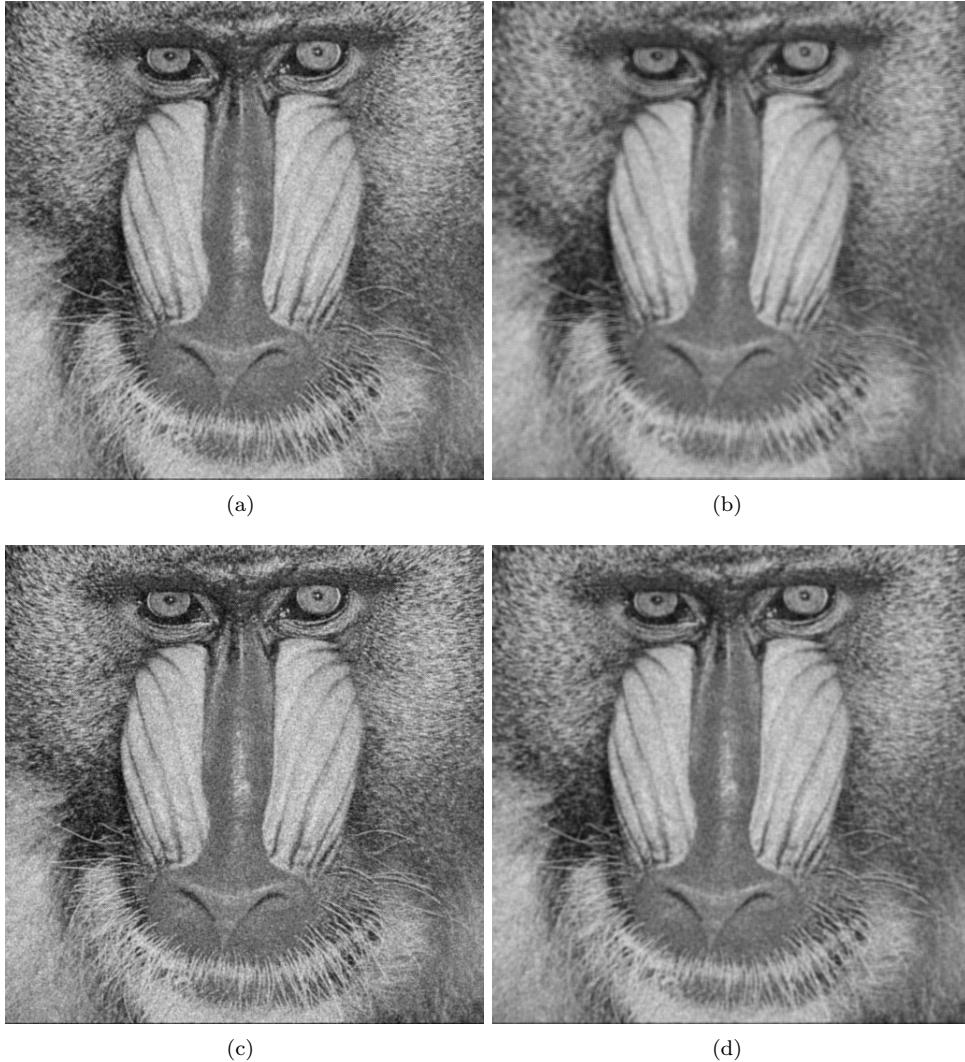
Program 5.1: Pengurangan *noise* dengan filter spasial

```

1 from skimage import io
2 from skimage.filters.rank import mean
3 from skimage.morphology import square, disk
4
5 imgNoise=io.imread('../pics/baboonNoise.png')
6 a1=square(3)
7 a2=square(5)
8 imgDeNoise=mean(imgNoise,a1)
9 io.imsave('../pics/baboonSquare3.png', imgDeNoise)
10 imgDeNoise=mean(imgNoise,a2)
11 io.imsave('../pics/baboonSquare5.png', imgDeNoise)
12 a1=disk(1)
13 a2=disk(2)
14 imgDeNoise=mean(imgNoise,a1)
15 io.imsave('../pics/baboonDisk1.png', imgDeNoise)
16 imgDeNoise=mean(imgNoise,a2)

```

```
17 | io.imsave('.../pics/baboonDisk2.png', imgDeNoise)
```

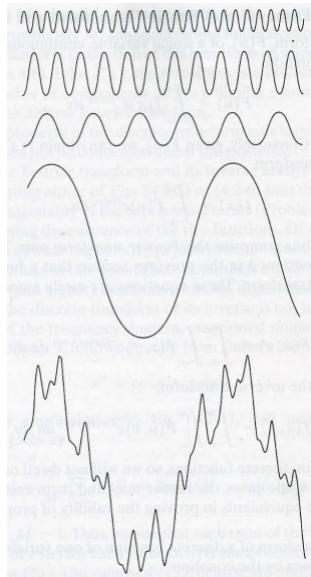


Gambar 5.2: Hasil pengurangan *noise* dengan *structuring element* (a). `square(3)`, (b). `square(5)`, (c). `disk(1)` dan (d). `disk(2)`

5.4 Frequency domain denoizing

Setiap fungsi yang berulang secara periodik dapat dinyatakan sebagai jumlah dari beberapa fungsi pada frekuensi dan amplitudo tertentu [Gonzalez and Woods, 2008]. Ilustrasinya disajikan pada Gambar 5.3. Sebuah citra digital dapat dianggap sebagai fungsi periodik yang karenanya dapat ditransformasi ke dalam domain frekuensi. Pada posisi di mana terjadi perubahan nilai intensitas piksel secara signifikan, citra dikatakan menyimpan komponen frekuensi tinggi. Sebaliknya, pada posisi di mana perubahan intensitas piksel sangat landai bahkan nyaris

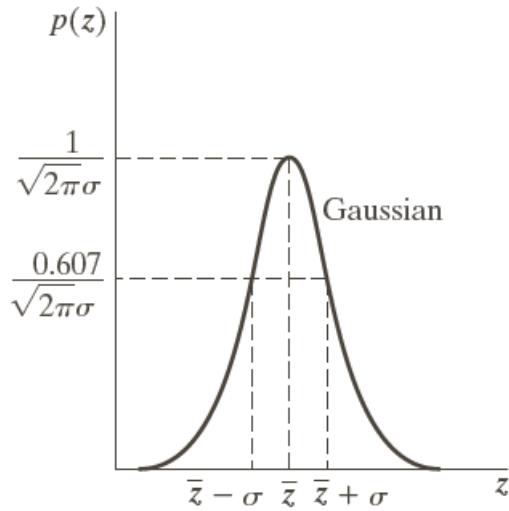
tetap, citra dikatakan menyimpan komponen frekuensi rendah.



Gambar 5.3: Fungsi periodik yang merupakan kombinasi dari beberapa fungsi periodik lain

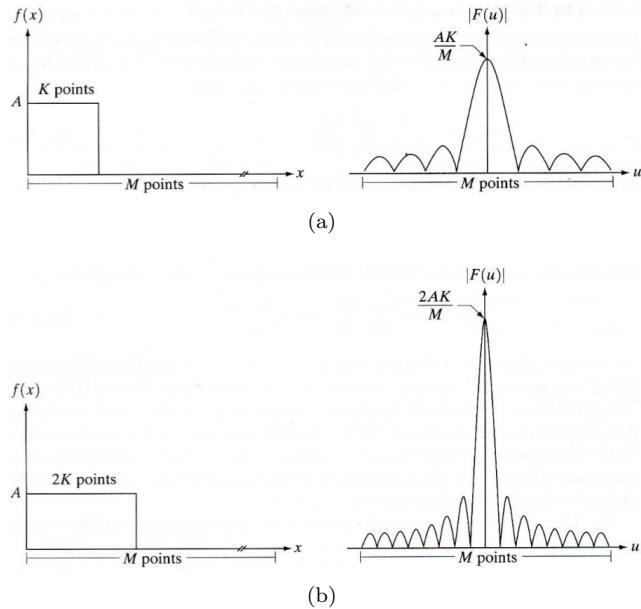
Pada frekuensi tinggi, di sutilah batas obyek umumnya berada. Mempertahankan komponen frekuensi tinggi akan menjaga kontras citra tetap baik. Sebaliknya, frekuensi rendah merepresentasikan kondisi di mana nilai intensitas piksel cenderung tetap. Mempertahankan komponen frekuensi rendah menjadi dasar dari proses penghalusan (*smoothing*) citra.

Secara teori, mempertahankan/menghilangkan komponen dengan frekuensi/rentang frekuensi tertentu dari citra dilakukan pada domain frekuensi. Citra harus ditransformasi dahulu ke domain frekuensi. Selanjutnya, komponen frekuensi yang tidak diharapkan dapat dihilangkan. Terakhir, transformasi dilakukan kembali ke domain spasial. Dengan pustaka `scikit-image`, proses transformasi dilakukan secara implisit oleh fungsi `difference_of_gaussian` di bawah sub modul `filters`. Fungsi tersebut sejatinya adalah sebuah *band pass filter*, filter untuk menyaring rentang frekuensi tertentu. Frekuensi dinyatakan sebagai nilai σ (standar deviasi). Ilustrasinya disajikan pada Gambar 5.4.



Gambar 5.4: Ilustrasi kurva dengan fungsi gaussian

Semakin besar nilai σ maka frekuensi semakin kecil. Sebaliknya, semakin kecil nilai σ , maka frekuensi semakin besar. Ilustrasinya disajikan pada Gambar 5.5. Nilai σ di Gambar 5.5(a) lebih besar dibandingkan di Gambar 5.5(b). Besarnya nilai σ ditunjukkan dengan bentuk kurva yang melebar dan dengan amplitudo kecil (Gambar 5.5(a)). Sedangkan besarnya nilai frekuensi ditunjukkan dengan bentuk kurva yang pipih dengan amplitudo besar (Gambar 5.5(b))).



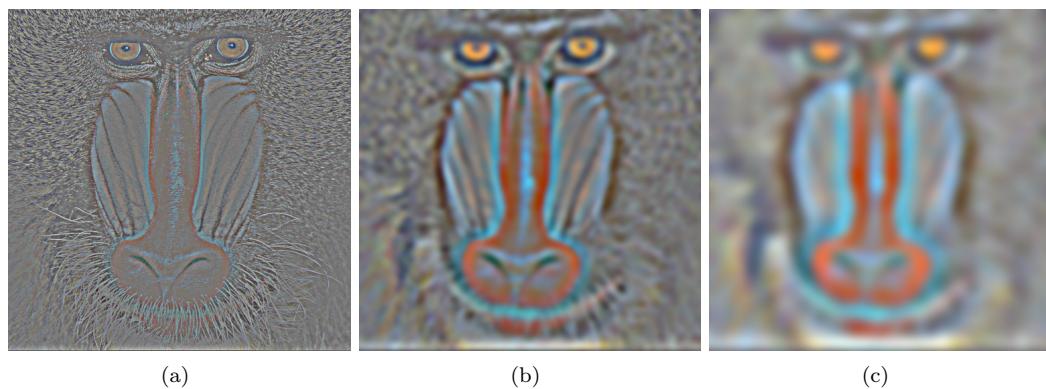
Gambar 5.5: Ilustrasi perbandingan nilai σ terhadap frekuensi

Sedangkan Gambar 5.6 menunjukkan hasil penyaringan frekuensi rendah, sedang (pada ren-

tang frekuensi tertentu), dan tinggi dari citra baboon dalam warna RGB. Penyaringan tersebut dilakukan dengan fungsi `difference_of_gaussian`. Gambar 5.6(a) diperoleh dengan memberikan argumen `low_sigma=0` dan `high_sigma=5`. Pemberian argumen dengan nilai tersebut bermakna frekuensi tertinggi yang ada di dalam citra hingga frekuensi yang setara dengan nilai $\sigma = 5$ akan dipertahankan. Terlihat bahwa Gambar 5.6(a) terlihat lebih jelas daripada kedua citra lainnya.

Selanjutnya, pembentukan citra pada Gambar 5.6(b), argumen yang diberikan untuk `low_sigma=5` dan `high_sigma=10`. Pemberian argumen tersebut akan menyaring frekuensi yang setara dengan nilai $\sigma = 10$ hingga frekuensi yang setara dengan nilai $\sigma = 5$ akan dipertahankan. Karena frekuensi yang setara dengan nilai $\sigma = 5$ atau lebih dihilangkan, maka citra di Gambar 5.6(b) terlihat semakin buram jika dibandingkan dengan citra pada Gambar 5.6(a).

Terakhir, pembentukan citra pada Gambar 5.6(c), argumen yang diberikan untuk `low_sigma=10` sedangkan argumen `high_sigma` tidak diberikan. Dengan kombinasi tersebut, frekuensi yang setara dengan nilai $\sigma = 10$ dan yang lebih rendah akan dipertahankan. Berarti yang tersisa dari citra adalah komponen frekuensi rendah. Hasilnya, Gambar 5.6(c) tampak paling buram dibanding dua citra sebelumnya.



Gambar 5.6: Hasil penyaringan frekuensi rendah, sedang dan tinggi pada citra RGB

Bab 6

Warna

Bab 7

Fitur citra

Fungsi Yang akan dijelaskan di sini merupakan sub modul `skimage.feature`.

7.1 *Histogram of Oriented Gradients*

Metode HOG diperkenalka oleh Dalal dan Triggs pada tahun 2005 [Dalal and Triggs, 2005] untuk mendeteksi obyek manusia pada citra. Perhatikan Program 7.1. Citra baboon dihitung fitur HOGnya dengan variasi 2 nilai orientasi, masing-masing 1 dan 10. Hasilnya ditunjukkan pada Gambar 7.1(a) dan 7.1(b). Citra selanjutnya dibagi menjadi sejumlah `block` yang setiap `block` berisi (3x3) `cell` sebagai dasar pembuatan histogram dari *gradient* (bobot) dan *orientation* (arah). Dan setiap `cell` berisi (8x8) piksel. Perhitungan *gradient* menggunakan L2-norm seperti dijelaskan pada tautan ini¹. Sedangkan penjelasan detil dari ekstraksi fitur metode HOG dapat dipelajari di tautan ini²³.

Program 7.1: *HOG*

```
1 from skimage import io
2 from skimage import feature as ft
3
4 img=io.imread('../pics/baboon.png')
5
6 hog1=ft.hog(img, orientations=1, pixels_per_cell=(8, 8), cells_per_block=(3, 3), block_norm='L2-Hys', visualize
7     ↪ =True, transform_sqrt=False, feature_vector=True, multichannel=None)
8 hog2=ft.hog(img, orientations=10, pixels_per_cell=(8, 8), cells_per_block=(3, 3), block_norm='L2-Hys',
    ↪ visualize=True, transform_sqrt=False, feature_vector=True, multichannel=None)
9 print(hog1[0])
10 print(hog2[0])
11 io.imsave('../pics/bhog1.png', hog1[1])
12 io.imsave('../pics/bhog10.png', hog2[1])
```

Seperti dijelaskan secara *implisit* di Program 7.1, HOG mengembalikan dua elemen `list`. Elemen pertama berisi fitur HOG berupa bobot dan arah. Fitur ini bisa dimunculkan jika argumen `feature_vector` bernilai `True`. Gambar 7.1(a) dan 7.1(b) diperoleh jika argumen

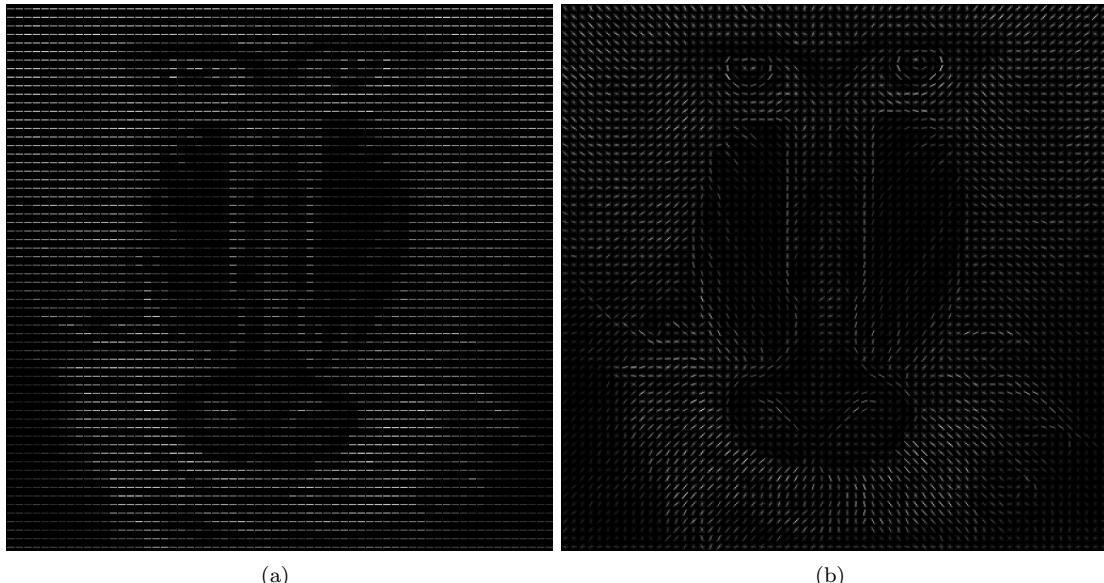
¹<https://machinelearningmastery.com/vector-norms-machine-learning/>

²<https://www.learnopencv.com/histogram-of-oriented-gradients/>

³<https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/>

`visualize` bernilai `True`. Jika salah satu argumen (apakah `feature_vector` atau `visualize`) yang bernilai `True`, fungsi `hog` akan mengembalikan `list` dengan 1 elemen saja, sesuai dengan argumen yang bernilai `True` dan nilai tersebut menempati elemen pertama dari `list` tersebut. Tetapi, jika kedua argumen tadi bernilai `False`, maka `hog` akan mengembalikan list yang menjelaskan jumlah blok pada setiap baris dan kolom yang terbentuk, jumlah sel pada setiap baris dan kolom, serta jumlah orientasinya.

Untuk citra berukuran 512x512, akan terdapat 62 blok pada setiap baris dan kolom. Untuk jumlah orientasinya, Gambar 7.1(a) yang hanya memiliki 1 orientasi seperti dideskripsikan oleh argumen `orientations` hanya direpresentasikan oleh karakter '-''. Sedangkan Gambar 7.1(b) direpresentasikan dengan lebih banyak karakter yang saling bertumpuk karena dideskripsikan oleh argumen `orientations` yang bernilai 10. Nilai bobot dan orientasi yang diperoleh dari operasi `hog` dapat digunakan untuk mengenali obyek yang berada di dalam citra. Tentu, akan lebih baik jika dalam satu citra terdapat satu obyek seperti dalam pengujian yang dilakukan alal dan Triggs [Dalal and Triggs, 2005].



Gambar 7.1: Fitur HOG dari citra baboon dengan nilai orientasi (a). 1 dan (b). 10

7.2 Gray Level Co-occurrence Matrix

Bab 8

Deteksi Tepi

Deteksi tepi adalah metode deteksi batas obyek pada citra. Dari tepi obyek tersebut segmentasi untuk memisahkan obyek dari latar dapat dilakukan. Setelah segmentasi dilakukan, obyek dapat dikenali.

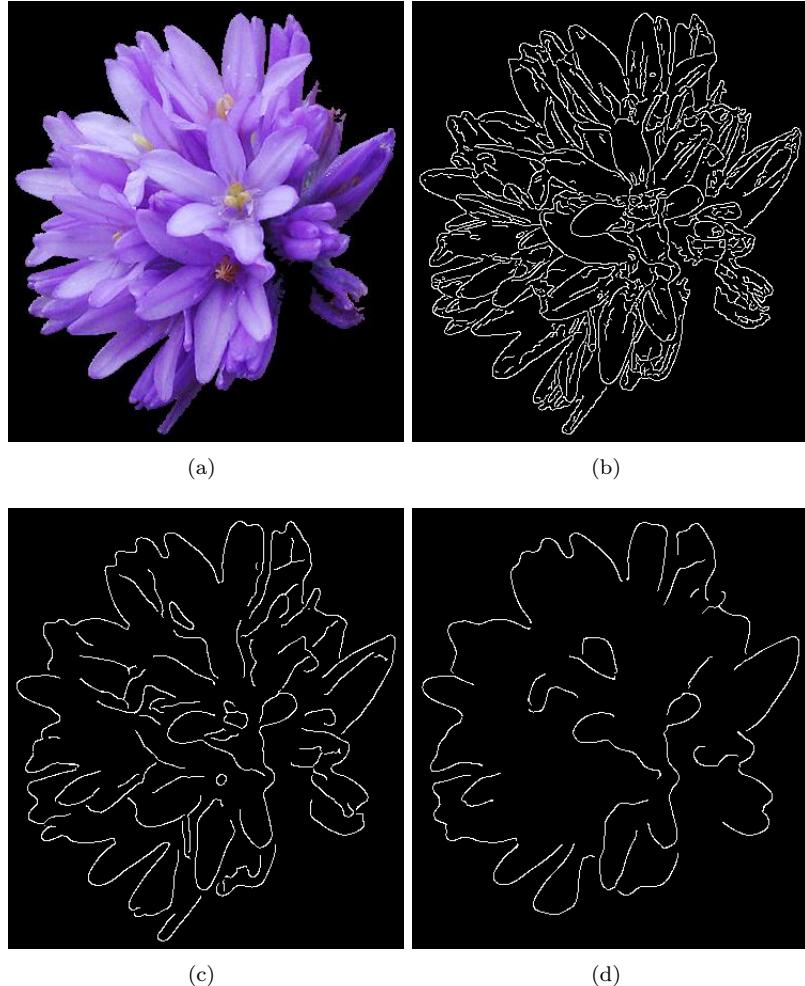
8.1 Deteksi tepi canny

Fungsi deteksi tepi `canny` dikelompokkan scikit-image sebagai bagian dari modul `feature`. Program 8.1 menunjukkan contoh deteksi tepi `canny`. Citra asli berupa sebuah bunga (Gambar 8.1(a)) dibaca sebagai citra berskala keabuan dengan perintah di baris ke-5. Deteksi tepi kemudian dilakukan dengan nilai $\sigma = 1$ seperti pada baris ke-6. Hasil deteksi tepi tersebut direpresentasikan sebagai matriks `boolean` karena elemen matriks berisi nilai `True` atau `False`. Representasi citra `boolean` tidak dapat langsung disimpan sehingga harus dikonversi. Untuk itulah dilakukan konversi citra dari bentuk matriks `boolean` ke bentuk `unsigned integer` seperti perintah pada baris ke-7. Barulah hasil konversinya disimpan sebagai berkas citra seperti baris ke-8. Hal ini berlaku untuk deteksi tepi `canny` dengan nilai $\sigma = 3$ dan $\sigma = 5$. Hasilnya masing-masing diperlihatkan di Gambar 8.1(b), Gambar 8.1(c) dan Gambar 8.1(d).

Program 8.1: Deteksi tepi `canny`

```
1 from skimage import io
2 from skimage import feature as ft
3 from skimage import util
4
5 img=io.imread('../pics/train1.jpg', True)
6 edge1=ft.canny(img)
7 e1=util.img_as_uint(edge1)
8 io.imsave('../pics/edgeCanny1.png',e1)
9
10 edge3=ft.canny(img, sigma=3)
11 e3=util.img_as_uint(edge3)
12 io.imsave('../pics/edgeCanny3.png',e3)
13
14 edge5=ft.canny(img, sigma=5)
15 e5=util.img_as_uint(edge5)
16 io.imsave('../pics/edgeCanny5.png',e5)
```

Baris ke-6 berbeda dari baris ke-10 dan 14 dalam hal argumen yang diberikan ke fungsi `canny`. Di baris ke-6, hanya ada 1 argumen yang diberikan, yaitu variabel *pointer* citra. Hal ini disebabkan karena fungsi `canny` memberikan nilai $\sigma = 1$ sebagai nilai *default*. Sedangkan pada baris ke-10 dan 14, nilai σ perlu diberikan karena 3 dan 5 bukan merupakan nilai *default*.



Gambar 8.1: Citra (a). awal bunga, yang selanjutnya dideteksi tepi menggunakan fungsi `canny` dengan variasi nilai σ (b). 1, (c). 3 dan (d). 5

Bab 9

Morfologi Citra

Bab 10

Ekstraksi Fitur Bentuk

10.1 Pendahuluan

Fitur bentuk citra diperoleh dengan terlebih dahulu membentuk citra biner, citra yang hanya memiliki dua nilai intensitas piksel, masing-masing 0 yang mewakili warna hitam dan 255 yang mewakili warna putih. *Object of interest* akan diberi warna putih, sementara *background* akan diberi warna hitam. Pembentukan citra biner memerlukan proses *thresholding*, proses untuk menentukan titik batas yang dari titik itulah piksel dengan intensitas tertentu diubah menjadi 0, sedangkan piksel lain nilai intensitasnya diubah menjadi 1. Proses *thresholding* ini mensyaratkan masukan berupa citra dalam skala keabuan.

Pada pustaka `scikit-image`, proses *thresholding* tersedia melalui beberapa fungsi yang didefinisikan di sub modul `skimage.filters`. Pustaka `scikit-image` bahkan menyediakan sebuah fungsi yang dapat digunakan untuk melakukan komparasi visual terhadap hasil *thresholding* menggunakan berbagai metode yang disediakan secara terpisah.

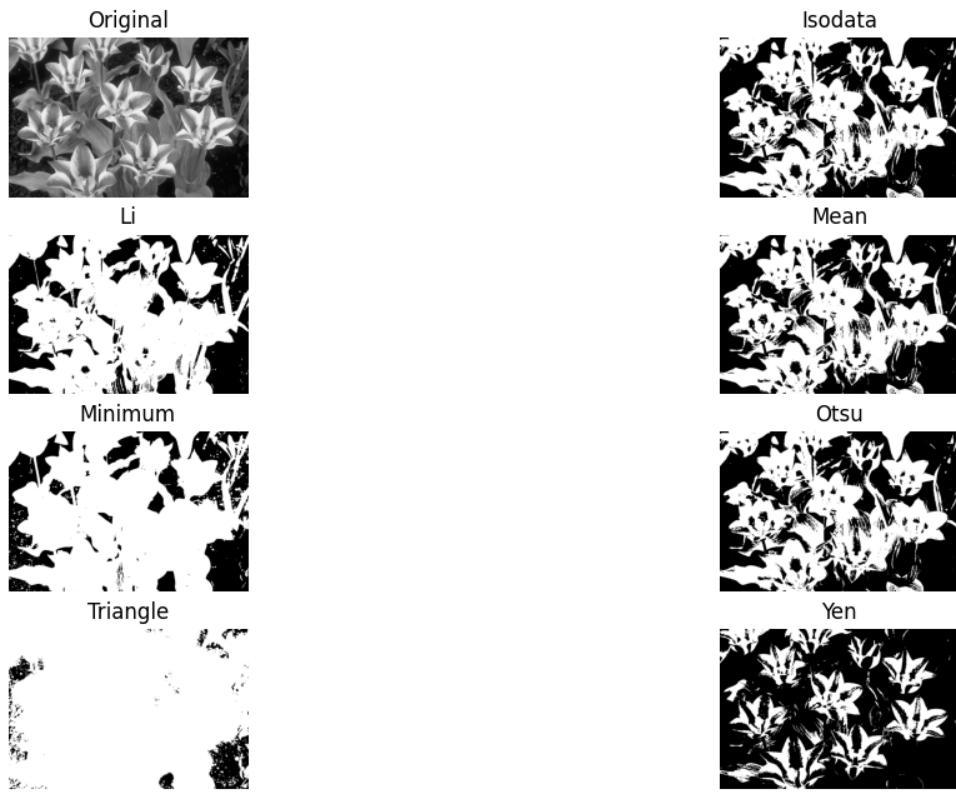
Program 10.1 menunjukkan fungsi komparasi berbagai metode *thresholding* yang disediakan oleh pustaka `scikit-image`. Pembentukan citra biner dilakukan terhadap citra yang disajikan pada Gambar 10.1. Di Program 10.1, citra dibaca dalam skala keabuan. Sedangkan Gambar 10.2 menunjukkan hasil citra binernya.

Program 10.1: Melihat kinerja *thresholding* secara visual

```
1 from skimage import io
2 from skimage.filters import try_all_threshold as allth
3 from matplotlib import pyplot as plt
4
5 a=io.imread('../pics/tulips.png', as_gray=True)
6 fig, ax = allth(a, figsize=(10, 8), verbose=False)
7 plt.show()
```



Gambar 10.1: Citra bunga tulip



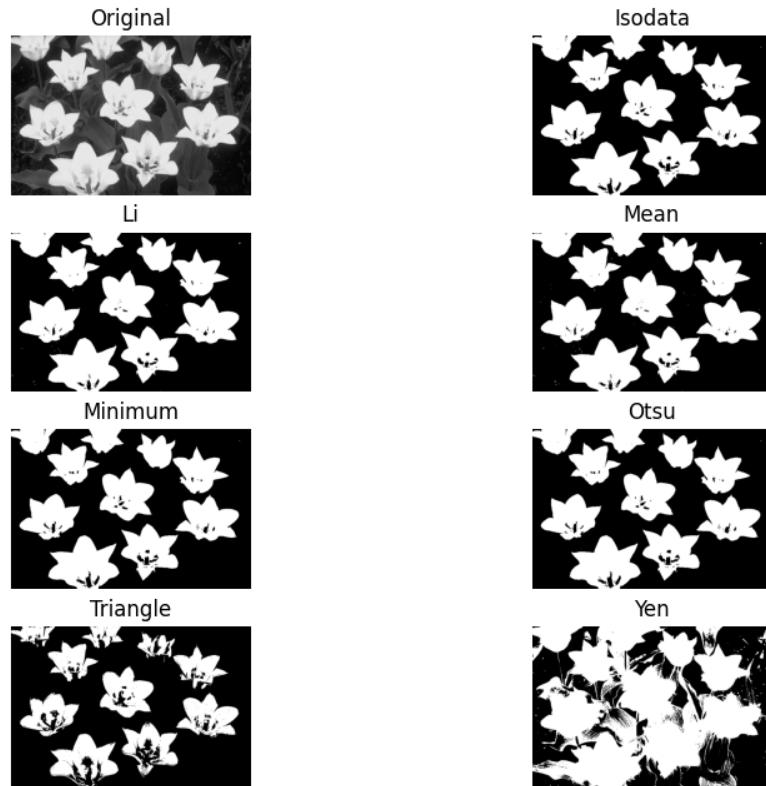
Gambar 10.2: Citra biner yang dihasilkan dari citra tulips dalam skala keabuan

Seperti telah dijelaskan di sub bab 4.1, komponen warna merah, hijau dan biru ketika disimpan secara *independent*, akan ditampilkan sebagai citra dalam skala keabuan. Tentunya, nilai *threshold* yang diperoleh dari setiap komponen warna akan berbeda, meski metode *thresholding*nya sama. Gambar 10.3 menunjukkan pembentukan citra biner dari komponen warna merah. Sedangkan Gambar 10.4 dan Gambar 10.5 masing-masing menunjukkan proses yang sama berdasarkan komponen warna hijau dan biru. Terlihat bahwa citra biner yang diperoleh

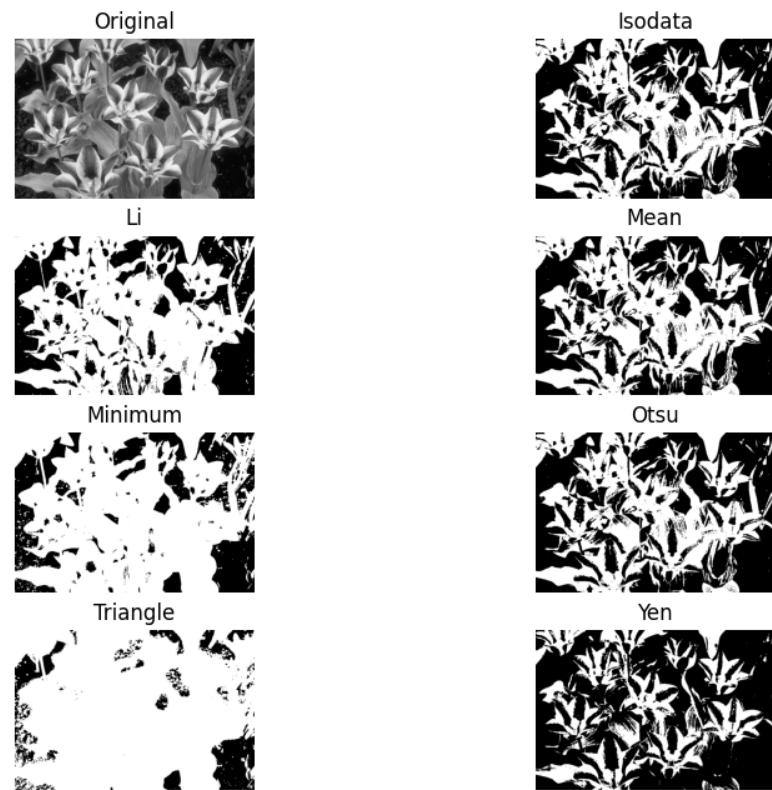
berdasarkan komponen warna merah menghasilkan segmentasi yang lebih baik daripada yang diperoleh dari komponen warna hijau dan biru.

Kekurangan yang masih ada pada citra biner dari komponen warna merah adalah adanya bagian hijau dalam lingkup kelopak bunga. Komponen warna hijau tersebut direpresentasikan sebagai nilai intensitas yang lebih kecil daripada nilai *threshold* ketika direpresentasikan dalam komponen warna merah. Inilah yang menyebabkan kelopak bunga tampak berlubang di bagian tengah. Dari Gambar 10.3, hanya metode *thresholding* Yen yang mampu membatasi daerah kelopak bunga menjadi satu kesatuan. Kekurangannya adalah, obyek lain di luar kelopak bunga masih dianggap *object of interest*.

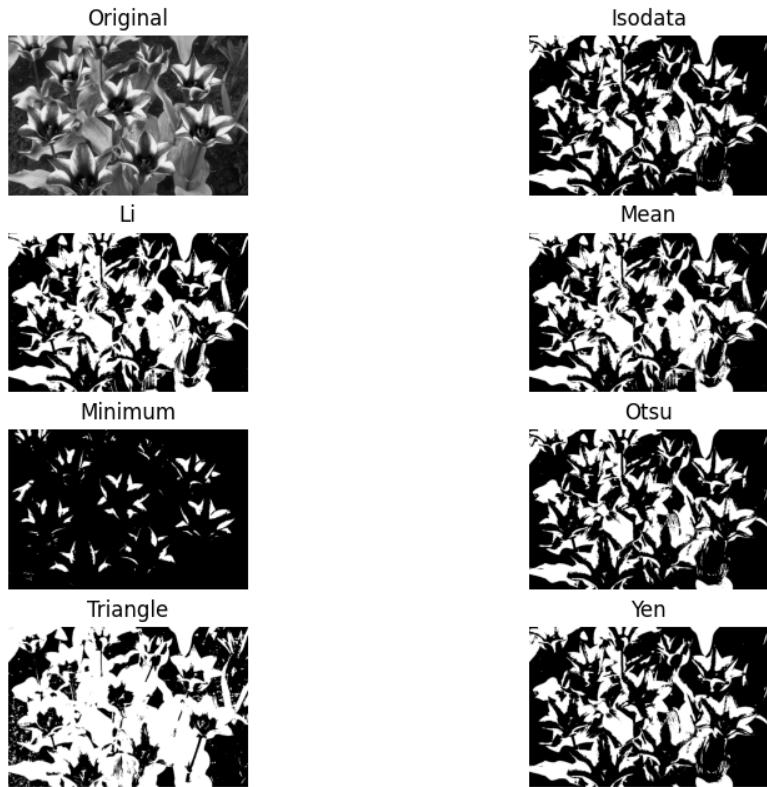
Pendekatan ini dapat dilakukan dengan menyisipkan perintah `img=a[:, :, 0]` di antara baris ke-5 dan 6 dari Program 10.1 jika ingin menggunakan komponen warna merah sebagai basis pembentukan citra biner. Sedangkan penyisipan perintah `img=a[:, :, 1]` dan `img=a[:, :, 2]` masing-masing ditujukan untuk menggunakan komponen warna hijau dan biru sebagai basis pembentukan citra biner. Sebagai tambahan, argumen ke-2 pada baris ke-5 dari Program 10.1 ditiadakan. Sedangkan argumen pertama pada baris ke-6 dari Program 10.1 diubah menjadi `img`, menyesuaikan variabel untuk setiap komponen warna.



Gambar 10.3: Citra biner yang dihasilkan dari komponen warna merah dari citra tulips



Gambar 10.4: Citra biner yang dihasilkan dari komponen warna hijau dari citra tulips



Gambar 10.5: Citra biner yang dihasilkan dari komponen warna biru dari citra tulips

10.2 Jumlah obyek pada citra

Untuk mendapatkan fitur bentuk, obyek harus dapat terpisah secara utuh dari latar (*background*). Pemisahan obyek dari latar ditunjukkan oleh citra biner yang sebelumnya dipelajari di sub bab 10.1. Citra biner sendiri dapat dengan mudah diperoleh ketika obyek dan latar sangat berbeda seperti pada kasus di sub bab 10.3. Tetapi, ketika citra yang dihadapi memiliki obyek dan latar yang rumit (Gambar 10.1), pembentukan citra biner sangat sulit dilakukan. Untuk alasan inilah, banyak penelitian yang dilakukan terkait segmentasi obyek pada citra [Wang et al., 2018].

Salah satu kriteria segmentasi obyek yang baik adalah ditemukannya jumlah obyek yang sama [Crevier, 2008]. Mendeteksi keberadaan obyek secara otomatis dilakukan dengan fungsi `blob`, yang dalam pustaka `scikit-image` diterapkan sebagai fungsi `blob_dog`, `blob_doh` dan `blob_log` di bawah sub modul `skimage.feature`. Basis identifikasi yang digunakan ketiga fungsi tersebut adalah pola hubungan intensitas piksel di mana sebuah obyek didefinisikan sebagai daerah dengan intensitas tinggi yang dikelilingi daerah dengan intensitas rendah atau sebaliknya.

Untuk mengidentifikasi kondisi tersebut, ketiga fungsi membutuhkan sejumlah argumen yang hampir sama. Dan argumen yang paling penting adalah `min_sigma` dan `max_sigma`. Seperti telah dijelaskan di sub bab 5.4, perubahan intensitas citra dapat dinyatakan dalam bentuk

frekuensi. Frekuensi tinggi merupakan representasi dari perubahan intensitas citra yang signifikan, sedangkan frekuensi rendah merupakan representasi dari intensitas citra yang cenderung stabil (tetap). Di sisi lain, frekuensi berbanding terbalik dengan σ . Semakin besar frekuensi, semakin kecil σ . Demikian sebaliknya. Argumen `min_sigma` dan `max_sigma` digunakan untuk mendeteksi daerah pada citra yang memiliki rentang frekuensi tersebut karena di situlah diduga terdapat batas obyek, daerah yang sebelumnya memiliki perubahan intensitas yang cepat kemudian melambat dan stabil.

Berikut contoh penggunaan ketiga fungsi tersebut.

10.2.1 Difference of Gaussian

10.3 Fitur bentuk

Berikut adalah fitur bentuk yang dapat diperoleh menggunakan pustaka `scikit-image`, yang dalam hal ini didefinisikan sebagai fungsi `regionprops` di dalam sub modul `measure`. Sebagai obyek kajian, Program 10.2 digunakan untuk mengambil sejumlah fitur bentuk dari fungsi `regionprops`, khususnya yang bernilai tunggal. Fitur titik pusat (*centroid*) dari *object of interest* juga akan diekstrak dengan memisahkannya menjadi titik pusat X dan Y. Citra yang akan diambil fitur bentuknya adalah citra daun flavia¹. Penggunaan citra tersebut disebabkan karena hanya ada obyek daun yang terdapat di setiap citra, sehingga tidak diperlukan lagi tahapan segmentasi. Program 10.2 juga akan membentuk berkas fitur dengan format `csv` yang siap dilatih dengan algoritma pelatihan tertentu. Fitur bentuk yang akan diekstrak adalah yang didefinisikan oleh Putzu [Putzu et al., 2014].

Program 10.2 terdiri dari dua komponen utama, masing-masing komponen data `lookup` untuk melihat kelas daun yang juga didefinisikan di laman *repository* flavia. Data `lookup` tersebut dijalankan oleh fungsi `leavesClass` di baris ke-7 yang menerima ID daun yang juga menjadi nama berkas daun yang bersangkutan. Sedangkan komponen berikutnya adalah ekstraksi fitur dan menuliskannya pada berkas fitur yang diberi nama `feature.csv`.

Program 10.2: Pembentukan berkas fitur bentuk dari obyek daun Flavia

```

1 from skimage import io
2 from skimage.color import rgb2gray
3 from skimage.filters import threshold_otsu as otsu
4 from skimage.measure import label, regionprops
5 import os, shutil, string
6
7 def leavesClass(a):
8     if a>=1001 and a<=1059:
9         return 'pubescent bamboo'
10    elif a>=1060 and a<=1122:
11        return 'Chinese horse chestnut'
12    elif a>=1123 and a<=1194:
13        return 'Chinese redbud'
14    elif a>=1195 and a<=1267:
15        return 'true indigo'
16    elif a>=1268 and a<=1323:
17        return 'Japanese maple'
```

¹<http://flavia.sourceforge.net/>

```

18     elif a>=1324 and a<=1385:
19         return 'Nanmu'
20     elif a>=1386 and a<=1437:
21         return 'castor aralia'
22     elif a>=1438 and a<=1496:
23         return 'goldenrain tree'
24     elif a>=1497 and a<=1551:
25         return 'Chinese cinnamon'
26     elif a>=1552 and a<=1616:
27         return 'Anhui Barberry'
28     elif a>=2001 and a<=2050:
29         return 'Big-fruited Holly'
30     elif a>=2051 and a<=2113:
31         return 'Japanese cheesewood'
32     elif a>=2114 and a<=2165:
33         return 'wintersweet'
34     elif a>=2166 and a<=2230:
35         return 'camphortree'
36     elif a>=2231 and a<=2290:
37         return 'Japan Arrowwood'
38     elif a>=2291 and a<=2346:
39         return 'sweet osmanthus'
40     elif a>=2347 and a<=2423:
41         return 'deodar'
42     elif a>=2424 and a<=2485:
43         return 'maidenhair tree'
44     elif a>=2486 and a<=2546:
45         return 'Crepe myrtle'
46     elif a>=2547 and a<=2612:
47         return 'oleander'
48     elif a>=2616 and a<=2675:
49         return 'yew plum pine'
50     elif a>=3001 and a<=3055:
51         return 'Japanese Flowering Cherry'
52     elif a>=3056 and a<=3110:
53         return 'Glossy Privet'
54     elif a>=3111 and a<=3175:
55         return 'Chinese Toon'
56     elif a>=3176 and a<=3229:
57         return 'peach'
58     elif a>=3230 and a<=3281:
59         return 'Ford Woodlotus'
60     elif a>=3282 and a<=3334:
61         return 'trident maple'
62     elif a>=3335 and a<=3389:
63         return 'Beale\'s barberry'
64     elif a>=3390 and a<=3446:
65         return 'southern magnolia'
66     elif a>=3447 and a<=3510:
67         return 'Canadian poplar'
68     elif a>=3511 and a<=3563:
69         return 'Chinese tulip tree'
70     elif a>=3566 and a<=3621:
71         return 'tangerine'
72     else:
73         return 'unknown'
74
75 f=open('feature.csv','w')
76 f.write('area,bbox_area,centroidX,centroidY,convex_area,eccentricity,Equivalent_diameter,euler_number,extent,
77     ↪ filled_area,major_axis_length,.minor_axis_length,orientation,perimeter,solidity,kelas\n')
78 for i in os.listdir('.'):
79     if i.endswith('.jpg'):

```

```

79     temp=i.split('..')
80     kelas=leavesClass(int(temp[0]))
81     img=io.imread(i)
82     imgGray=rgb2gray(img)
83     threshold=otsu(imgGray,256)
84     x,y=imgGray.shape
85     for k in range(x):
86         for l in range(y):
87             if imgGray[k][l]>threshold:
88                 imgGray[k][l]=0
89             else:
90                 imgGray[k][l]=1
91
92     imgLabel=label(imgGray)
93     props=regionprops(imgLabel)
94     centroid=props[0].centroid
95     f.write(str(props[0].area)+','+str(props[0].bbox_area)+','+str(centroid[0])+','+str(centroid[1])+','+str
96     ↪ (props[0].convex_area)+','+str(props[0].eccentricity)+','+str(props[0].equivalent_diameter)+','
97     ↪ +str(props[0].euler_number)+','+str(props[0].extent)+','+str(props[0].filled_area)+','+str(
98     ↪ props[0].major_axis_length)+','+str(props[0].minor_axis_length)+','+str(props[0].orientation)+'
99     ↪ ,'+str(props[0].perimeter)+','+str(props[0].solidity)+','+kelas+'\n')
100
101 f.close()

```

Bibliografi

- [Burger and Burge, 2016] Burger, W. and Burge, M. J. (2016). *Digital Image Processing: An Algorithmic Introduction Using Java*. Springer Publishing Company, Incorporated, 2nd edition.
- [Crevier, 2008] Crevier, D. (2008). Image segmentation algorithm development using ground truth image data sets. *Computer Vision and Image Understanding*, 112(2):143 – 159.
- [Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1.
- [Gonzalez and Woods, 2008] Gonzalez, R. C. and Woods, R. E. (2008). *Digital Image Processing*. Prentice Hall.
- [Hunt, 2019] Hunt, J. (2019). *A Beginners Guide to Python 3 Programming*. Springer Publishing Company, Incorporated, 1st edition.
- [Putzu et al., 2014] Putzu, L., Caocci, G., and Ruberto], C. D. (2014). Leucocyte classification for leukaemia detection using image processing techniques. *Artificial Intelligence in Medicine*, 62(3):179 – 191.
- [Rosyani et al., 2018] Rosyani, P., Taufik, M., Waskita, A. A., and Apriyanti, D. H. (2018). Comparison of color model for flower recognition. In *2018 3rd International Conference on Information Technology, Information System and Electrical Engineering (ICITISEE)*, pages 10–14.
- [Wang et al., 2018] Wang, Z., Wang, K., Yang, F., Pan, S., and Han, Y. (2018). Image segmentation of overlapping leaves based on chanvese model and sobel operator. *Information Processing in Agriculture*, 5(1):1 – 10.