

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/358220979>

Pengolahan Citra Digital Menggunakan Python

Book · January 2022

CITATION
1

READS
26,398

1 author:



Finki Dona Marleny

University of Muhammadiyah Banjarmasin

40 PUBLICATIONS 22 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Analisis perbandingan algoritma pelatihan Backpropagation Neural Network untuk prediksi harga saham [View project](#)



UML : unified modeling language : panduan belajar pemodelan visual [View project](#)

PENGOLAHAN CITRA DIGITAL Menggunakan Python



PENGOLAHAN CITRA DIGITAL Menggunakan Python

Finki Dona Marleny



Pengolahan Citra Digital Menggunakan Python

Finki Dona Marleny



**Pengolahan Citra Digital
Menggunakan Python**

Penulis:
Finki Dona Marleny

ISBN : 978-623-315-853-4

Design Cover :
Retnani Nur Brilian

Layout :
Eka Safitry

Penerbit CV. Pena Persada
Redaksi :
Jl. Gerilya No. 292 Purwokerto Selatan, Kab. Banyumas
Jawa Tengah
Email : penerbit.penapersada@gmail.com
Website : penapersada.com Phone : (0281) 7771388
Anggota IKAPI

All right reserved
Cetakan pertama : 2021

Hak Cipta dilindungi oleh undang-undang. Dilarang
memperbanyak karya tulis ini dalam bentuk apapun tanpa
izin penerbit

KATA PENGANTAR

Salah satu bidang kecerdasan buatan yang banyak dikembangkan saat ini adalah pengolahan citra digital dan visi komputer (*Computer vision*). Yang mana saat ini banyak data menggunakan gambar dan video sebagai interaksi untuk saling terhubung, berbagi informasi dan juga sebagai hiburan. Buku ini mengulas tentang teori dan praktik pengolahan citra digital menggunakan Bahasa python.

Alhamdulillahi rabbil 'alamin Segala puji dan syukur penulis panjatkan ke hadirat Allah SWT atas segala rahmat, karunia dan kesehatan yang telah diberikan sehingga penulis dapat menyelesaikan dan menyusun buku ini hingga rampung. Penulis memiliki motivasi untuk terus dapat menulis dan membagi walaupun tulisan yang tersusun didalam buku ini jauh dari sempurna.

Terimakasih penulis ucapan yang sebesar-besarnya kepada semua pihak yang telah mendukung dan selalu menginspirasi dalam penyusunan buku ini. Kritik dan saran penulis butuhkan sebagai bahan masukan dan motivasi penulis untuk terus semangat menulis dan membagi informasi serta pengetahuan. Semoga buku ini dapat bermanfaat bagi para pembaca.

Banjarmasin, 2021
Penulis

DAFTAR ISI

KATA PENGANTAR	iii
DAFTAR ISI.....	iv
BAB 1 PENDAHULUAN COMPUTER VISION.....	1
A. Sejarah dan Perkembangan Teknologi <i>Computer Vision</i>	1
B. Cara kerja Computer Vision	3
1. Computer Vision: Apa dan Mengapa itu Penting?	4
2. Penerapan Teknologi Computer Vision	5
C. Pengolahan Citra Digital.....	6
1. Representasi Citra Digital	11
2. Kualitas citra	11
3. Penerapan Pengolahan Citra Digital	12
D. Tools untuk Belajar Pengolahan Citra Digital.....	14
1. Matlab	15
2. Scilab	15
3. GNU Octave	16
4. Anaconda Navigator.....	16
5. Jupyter Notebook	23
Latihan:.....	25
Soal:.....	33
BAB 2 MENGENAL PYTHON LIBRARIES UNTUK PEMROSESAN GAMBAR.....	35
A. OpenCV	35
B. Scikit-Image	36
C. SciPy.....	37

D. Pillow/PIL.....	38
E. NumPy.....	38
F. Mahotas	39
G. SimpleITK.....	40
H. Pgmagick	40
Latihan:.....	42
Soal:.....	46
BAB 3 KONSEP DASAR PENGOLAHAN CITRA DIGITAL.....	48
A. Jenis-jenis Citra	49
1. Citra Berwarna	49
2. Citra Berskala keabuan (Grayscale)	50
3. Citra Biner	50
B. Kualitas Citra	51
Latihan:.....	52
Soal:.....	55
BAB 4 OPERASI PIXEL DAN HISTOGRAM.....	57
A. Operasi Piksel	57
B. Histogram Citra	57
C. Tingkat kecerahan, kontras dan peregangan	58
1. Meregangkan kontras.....	59
2. Kombinasi kecerahan dan kontras	60
3. Menampilkan Histogram Citra	60
Latihan:.....	62
Soal:.....	68
BAB 5 OPERASI KETETANGGAAN PIKSEL	70
A. Pengertian Operasi Ketetanggaan.....	70
B. Filter Batas (Filter Max-Min).....	71

C. Filter Rerata (Mean)	71
D. Filter Median.....	72
E. Filter lolos-bawah (low-pass filter)	72
F. Filter Lolos Rendah.....	72
G. Filter lolos-tinggi	73
H. Sifat Filter Lolos-tinggi	73
I. Filter Lolos Tinggi.....	74
J. Filter “high boost”	74
K. Efek emboss	75
L. Filter Linier vs Filter Non-Linier	76
Latihan:.....	78
Soal:.....	86
BAB 6 OPERASI GEOMETRIK PADA CITRA	88
A. Operasi Geometrik pada Citra	88
B. Penggeseran citra	88
C. Pemutaran citra	89
D. Interpolasi piksel	90
E. Pemutaran citra berdasarkan sebarang koordinat	90
F. Transformasi affine	92
G. Efek Ripple (Bergelombang).....	92
H. Transformasi twirl.....	92
I. Pengolahan Citra di Kawasan Frekuensi.....	93
J. Filter Lolos-Rendah (Low-pass Filter).....	94
Latihan:.....	95
Soal:.....	99

BAB 7 OPERASI MORFOLOGI	101
A. Operasi Morfologi Untuk Pengolahan Citra	101
1. Operasi dilasi.....	103
2. Operasi Erosi	103
B. Elemen Penstruktur (Structure Element)	104
C. Operasi Opening dan Closing	104
Latihan:.....	106
Soal:.....	111
BAB 8 PENGOLAHAN CITRA BINER	113
Representasi Bentuk	113
A. Deteksi Tepi.....	113
B. Rantai Kode (Code Chain)	115
Latihan:.....	116
Soal:.....	121
BAB 9 PENGOLAHAN CITRA BERWARNA	123
A. Teori Dasar Warna	123
B. Ruang Warna	123
C. Ruang Warna RGB	124
D. Ruang warna CMYK.....	125
E. Ruang Warna YIQ	126
F. Ruang Warna YC _b C _r	126
Latihan:.....	129
Soal:.....	139
BAB 10 SEGMENTASI CITRA BERWARNA	140
A. Pengantar Segmentasi citra	140
B. Thresholding Metode Otsu	140
C. Multilevel Thresholding	141

1. Adaptive Thresholding.....	142
D. Segmentasi Chan-veze	142
Latihan:.....	144
Soal:.....	148
BAB 11 EKSTRAKSI FITUR BENTUK DAN KONTUR	150
A. Ekstraksi fitur	150
B. Sifat bundar.....	151
C. Convex hull dan soliditas	152
D. Momen invariant.....	153
E. Momen Zernike	154
Latihan:.....	156
BAB 12 EKSTRAKSI FITUR KONTUR DAN BENTUK	159
A. Fitur tekstur	159
B. Tekstur berbasis histogram.....	161
C. Gray Level Co-occurrence Matrices (GLCM).....	163
Latihan:.....	165
Daftar Pustaka	176
INDEKS.....	181
Kunci Jawaban Soal:	182
PROFIL PENULIS	183

BAB 1

PENDAHULUAN COMPUTER VISION

A. Sejarah dan Perkembangan Teknologi *Computer Vision*

Pada akhir 1960-an, Computer Vision (Visi komputer) atau bagaimana komputer melihat dimulai dari universitas yang merintis kecerdasan buatan. Dimana teknologi ini bertujuan untuk meniru sistem penglihatan manusia. Perbedaan visi komputer dari bidang pengolahan gambar digital pada masa itu adalah keinginan untuk mengekstraksi struktur tiga dimensi dari gambar dengan tujuan mencapai pemahaman penuh. Studi pada tahun 1970-an membentuk fondasi awal untuk pengembangan algoritma komputer visi yang ada saat ini, termasuk ekstraksi tepi dari gambar, pelabelan garis, pemodelan non-polihedral dan polihedral, representasi objek sebagai interkoneksi dari struktur yang lebih kecil, aliran optik, dan estimasi gerak. Beberapa dekade berikutnya ditandai dengan perkembangan analisis matematika yang lebih maju secara aspek kuantitatif dari komputer visi. Pada 1990-an, beberapa topik penelitian sebelumnya menjadi lebih aktif dalam penelitian visi komputer yaitu penelitian dalam rekonstruksi 3D proyektif menyebabkan pemahaman yang lebih baik mengenai kalibrasi kamera. Perubahan signifikan terjadi pada bidang ini yaitu dengan meningkatnya interaksi antara bidang grafis komputer dengan pengembangan menggunakan teknologi komputer visi.

Computer vision/ visi komputer adalah teknologi yang menggabungkan kamera, sistem komputasi, perangkat lunak, dan kecerdasan buatan (*Artificial Intelligent/AI*) sehingga sistem dapat melihat dan mengidentifikasi objek. Teknologi yang diterapkan pada komputer visi dapat bermanfaat di

berbagai lingkungan yaitu sistem dapat mengenali objek dan orang dengan cepat, menganalisis demografi khalayak, memeriksa hasil produksi, melacak pergerakan, dan masih banyak yang dapat dikembangkan dengan teknologi ini.

Visi Komputer merupakan bidang ilmiah antardisiplin ilmu yang membahas bagaimana komputer dapat memperoleh pemahaman tingkat tinggi dari gambar atau video digital. Dari perspektif teknik, bidang ini berupaya mengotomatiskan tugas-tugas yang dapat dilakukan oleh sistem penglihatan manusia. Tugas penglihatan komputer yang meliputi metode untuk memperoleh, memproses, menganalisis dan memahami gambar digital, serta ekstraksi data dimensi tinggi dari dunia nyata untuk menghasilkan informasi numerik atau simbolis, misalnya, dalam bentuk keputusan. Dalam konteks ini berarti transformasi gambar secara visual membutuhkan suatu input atau masukan agar menjadi deskripsi mengenai gambaran sekitar yang dapat berinteraksi dengan proses pemikiran sistem dan memperoleh tindakan yang sesuai. Pemahaman mentransformasi gambar ini dapat dilihat sebagai penguraian informasi simbolik dari data gambar menggunakan model yang dibangun dengan bantuan disiplin ilmu lainnya seperti geometri, fisika, statistik, dan teori pembelajaran serta algoritma komputasi.

Sub-domain dari penglihatan komputer meliputi rekonstruksi adegan, deteksi peristiwa, pelacakan video, pengenalan objek, estimasi gambar 3D, pembelajaran, pengindeksan, estimasi gerakan, dan pemulihan gambar. Berikut ini adalah bagaimana cara komputer melihat citra atau gambar.



Yang dilihat Manusia

0	3	2	5	4	7	6	9	8
3	0	1	2	3	4	5	6	7
2	1	0	3	2	5	4	7	6
5	2	3	0	1	2	3	4	5
4	3	2	1	0	3	2	5	4
7	4	5	2	3	0	1	2	3
6	5	4	3	2	1	0	3	2
9	6	7	4	5	2	3	0	1
8	7	6	5	4	3	2	1	0

Yang dilihat computer

Gambar 1.1: Ilustrasi perbedaan citra yang dilihat manusia dan computer

Salah satu bidang teknologi yang memungkinkan sebuah komputer bisa melihat objek di sekitarnya adalah *computer vision*. Computer vision memungkinkan komputer untuk mengidentifikasi dan memproses objek dengan cara yang sama seperti manusia. Contoh yang paling sering digunakan saat ini adalah penggunaan **scanQR** pada smartphone sebagai salah satu cara untuk melakukan pembayaran. Ketika kode scanQR dideteksi oleh perangkat cerdas seperti handphone, maka kode yang telah dipindai ini akan mengirimkan sebuah perintah khusus, bisa berupa perintah pembayaran, membuka kunci, maupun aktivitas lainnya.

B. Cara kerja Computer Vision

Saat bermain puzzle terdapat potongan-potongan gambar dan seluruh potongan-potongan gambar itu tersebar dalam berbagai bentuk pinggiran potongan gambar yang berbeda-beda bentuk membentuk suatu pola potongan, kemudian untuk mendapatkan satu kesatuan gambar yang

utuh diperlukan susunan potongan gambar yang sesuai. Dari ilustrasi permainan puzzle itulah cara kerja yang dimiliki computer vision saat menerjemahkan sebuah objek dan memberi label pada objek tersebut.

Saat menyusun puzzle, otak manusia bekerja memasangkan potongan gambar dengan pasangan yang cocok untuk membentuk gambar yang sesuai agar tidak salah dalam menempatkan gambar. Tugas ini dilakukan oleh jaringan saraf atau instruksi algoritma dalam computer vision yang akan memisahkan bagian gambar yang berbeda-beda, mengidentifikasi bagian tepi gambar hingga kemudian menggabungkannya menjadi satu persatu sampai membentuk gambar yang utuh. Komputer tidak diberi gambaran besar dari data yang dimasukkan, karena komputer akan secara otomatis akan memberi label pada gambar tersebut (misalnya: kucing, wajah, atau kertas).

Jadi, ketika manusia diberi ‘inputan’ dengan jutaan gambar tentang suatu objek misalnya objek kucing, maka otomatis komputer akan mengarahkannya pada algoritma yang memungkinkan untuk menganalisis warna dalam foto, bentuk, jarak antar satu bentuk dengan yang lain, di mana objek berbatasan satu dengan yang lain, dan seterusnya sampai ia mampu mengidentifikasi profil apa saja yang diperlukan untuk melabeli sebuah objek sebagai “kucing”.

1. Computer Vision: Apa dan Mengapa itu Penting?

Perumpamaan di atas merupakan penjelasan sederhana tentang cara kerja computer vision. Jika diterjemahkan dengan penjelasan yang lebih kompleks, computer vision membagi gambar-gambar yang telah dimasukkan menjadi bentuk piksel. Masing-masing gambar akan diterjemahkan dalam bentuk angka. Meskipun objek yang dimasukkan berbentuk tiga dimensi, komputer tetap akan menerjemahkannya menjadi dua dimensi, karena keterbatasan memori yang membuatnya

hanya mampu menyimpan garis-garis linear dalam satu bidang.

Gambar dalam bentuk dua dimensi yang terdiri dari angka-angka akan dijumlahkan oleh komputer menjadi nilai-nilai tertentu hingga membentuk sebuah kode warna, seperti #FF0000 atau 255 untuk angka merah. Komputer biasanya membaca warna sebagai serangkaian angka-angka yang membentuk tiga warna dasar, yakni merah, hijau, dan biru (red, green, blue atau biasa disingkat dengan RGB). Setiap piksel memiliki tiga nilai tersebut untuk disimpan di dalam sistem pada posisi tertentu. Hal ini bertujuan untuk proses identifikasi warna agar komputer bisa memberi penjelasan profil yang lebih detail dari sebuah objek.

2. Penerapan Teknologi Computer Vision

Sistem deteksi wajah atau facial recognition (FR) termasuk ke dalam pengembangan teknologi computer vision pada level yang lebih tinggi. Sistem FR memungkinkan algoritma dalam computer vision untuk mendeteksi fitur-fitur pada wajah manusia dan membandingkannya dengan profil diri dari wajah yang terdeteksi tersebut, mulai dari nama, jenis kelamin dan usia. FR juga dapat digunakan untuk mengautentikasi identitas pemilik wajah demi menjaga keamanan data. Media sosial kini juga telah menerapkan teknologi FR untuk mendeteksi dan menandai pengguna sehingga dapat menyesuaikan preferensi pemilik akun berdasarkan ketertarikan dan history pencarinya. Lembaga hukum dan keamanan juga mulai menggunakan teknologi FR untuk mengantisipasi ancaman-ancaman dari penjahat melalui deteksi emosi.

C. Pengolahan Citra Digital

Pengolahan Citra Digital (*Digital Image Processing*) merupakan suatu bidang ilmu yang mempelajari tentang bagaimana suatu citra dibentuk, diolah, dan dianalisis sehingga menghasilkan informasi yang dapat dipahami oleh manusia. Pengolahan Citra Digital adalah serangkaian usaha untuk melakukan transformasi suatu citra atau gambar menjadi citra lain dengan menggunakan teknik tertentu.

Citra adalah gambar dua dimensi yang bisa ditampilkan pada layar komputer sebagai himpunan atau diskrit nilai digital yang disebut pixel atau elemen gambar. Secara matematis, citra merupakan fungsi kontinu dari intensitas cahaya pada bidang dua dimensi. Berdasarkan bentuk sinyal penyusunnya, citra dapat digolongkan menjadi dua jenis yaitu citra analog dan citra digital. Citra analog adalah citra yang dibentuk dari sinyal analog yang bersifat kontinyu, sedangkan citra digital adalah citra yang dibentuk dari sinyal digital yang bersifat diskrit.

Citra analog dihasilkan dari alat akuisisi citra analog, contohnya adalah mata manusia dan kamera analog. Gambaran yang tertangkap oleh mata manusia dan foto atau film yang tertangkap oleh kamera analog merupakan contoh dari citra analog. Citra tersebut memiliki kualitas dengan tingkat kerincian atau resolusi yang sangat baik tetapi memiliki kelemahan di antaranya adalah tidak dapat disimpan, diolah, dan diduplikasi di dalam komputer.

Citra digital merupakan representasi dari fungsi intensitas cahaya dalam bentuk diskrit pada bidang dua dimensi. Citra tersusun oleh sekumpulan piksel (picture element) yang memiliki koordinat (x,y) dan amplitudo $f(x,y)$. Koordinat (x,y) menunjukkan letak/posisi piksel dalam suatu citra, sedangkan amplitudo $f(x,y)$ menunjukkan nilai intensitas warna citra.

Pada umumnya, berdasarkan kombinasi warna pada piksel, citra dibagi menjadi tiga jenis yaitu citra RGB, citra grayscale, dan citra biner. Citra pada Gambar 1 termasuk

dalam jenis citra RGB truecolor 24-bit. Citra tersebut tersusun oleh tiga kanal warna yaitu kanal merah, kanal hijau, dan kanal biru. Masing-masing kanal warna memiliki nilai intensitas piksel dengan kedalaman bit sebesar 8-bit yang artinya memiliki variasi warna sebanyak 2^8 derajat warna (0 s.d 255).

Pada kanal merah, warna merah sempurna direpresentasikan dengan nilai 255 dan hitam sempurna dengan nilai 0. Pada kanal hijau, warna hijau sempurna direpresentasikan dengan nilai 255 dan hitam sempurna dengan nilai 0. Begitu juga pada kanal biru, warna biru sempurna direpresentasikan dengan nilai 255 dan hitam sempurna dengan nilai 0. Representasi citra RGB dan masing-masing kanal warna penyusunnya ditunjukkan pada Gambar berikut.



Citra RGB



Kanal Merah



Kanal Hijau



Kanal Biru

Gambar 1.2: Representasi citra RGB dan kanal warna penyusunnya

Setiap piksel pada citra RGB, memiliki intensitas warna yang merupakan kombinasi dari tiga nilai intensitas pada kanal R, G, dan B. Sebagai contoh, suatu piksel yang memiliki nilai intensitas warna sebesar 255 pada kanal merah, 255 pada

kanal hijau, dan 0 pada kanal biru akan menghasilkan warna kuning. Pada contoh lain, suatu piksel yang memiliki nilai intensitas warna sebesar 255 pada kanal merah, 102 pada kanal hijau, dan 0 pada kanal biru akan menghasilkan warna orange. Banyaknya kombinasi warna piksel yang mungkin pada citra RGB truecolor 24-bit adalah sebanyak $256 \times 256 \times 256 = 16.777.216$. Representasi nilai intensitas piksel dengan kombinasi warna R, G, dan B ditunjukkan pada Gambar berikut.

Yellow R = 255 G = 255 B = 0	Orange R = 255 G = 102 B = 0	Green R = 0 G = 255 B = 0
Cyan R = 0 G = 255 B = 255	Violet R = 204 G = 102 B = 204	White R = 255 G = 255 B = 255
Black R = 0 G = 0 B = 0	Turquoise R = 102 G = 255 B = 204	Brown R = 153 G = 102 B = 51

Gambar 1.3: Representasi piksel dengan kombinasi warna R, G, dan B

Jenis citra yang kedua adalah citra grayscale. Citra grayscale merupakan citra yang nilai intensitas pikselnya didasarkan pada derajat keabuan. Pada citra grayscale 8-bit, derajat warna hitam sampai dengan putih dibagi ke dalam 256 derajat keabuan di mana warna hitam sempurna direpresentasikan dengan nilai 0 dan putih sempurna dengan nilai 255. Citra RGB dapat dikonversi menjadi citra grayscale sehingga dihasilkan hanya satu kanal warna. Persamaan yang umumnya digunakan untuk mengkonversi citra RGB truecolor 24-bit menjadi citra grayscale 8-bit adalah

$$Grayscale = 0.2989 * R + 0.5870 * G + 0.1140 * B$$

di mana:

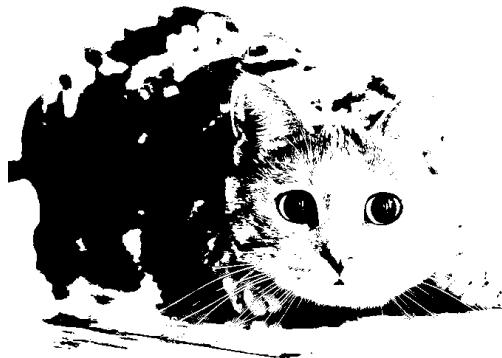
- Grayscale adalah nilai intensitas citra grayscale,
- R adalah nilai intensitas piksel pada kanal merah,
- G adalah nilai intensitas piksel pada kanal hijau, dan
- B adalah nilai intensitas piksel pada kanal biru.

Citra hasil konversi RGB menjadi grayscale ditunjukkan pada Gambar 1.4.



Gambar 1.4: Citra hasil konversi RGB menjadi grayscale

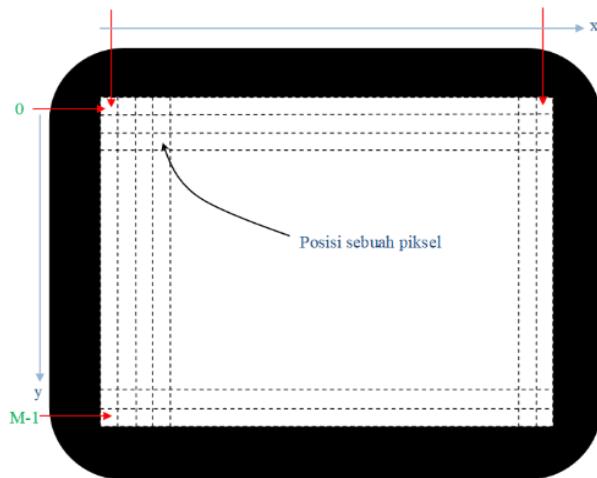
Jenis citra yang ketiga adalah citra biner. Citra biner adalah citra yang pikselnya memiliki kedalaman bit sebesar 1 bit sehingga hanya memiliki dua nilai intensitas warna yaitu 0 (hitam) dan 1 (putih). Citra grayscale dapat dikonversi menjadi citra biner melalui proses thresholding. Dalam proses thresholding, dibutuhkan suatu nilai threshold sebagai nilai pembatas konversi. Nilai intensitas piksel yang lebih besar atau sama dengan nilai threshold akan dikonversi menjadi 1. Sedangkan nilai intensitas piksel yang kurang dari nilai threshold akan dikonversi menjadi 0. Misalnya nilai threshold yang digunakan adalah 128, maka piksel yang mempunyai intensitas kurang dari 128 akan diubah menjadi 0 (hitam) dan yang lebih dari atau sama dengan 128 akan diubah menjadi 1 (putih).



Gambar 1.5: Citra hasil konversi RGB menjadi biner

Thresholding pada umumnya digunakan dalam proses segmentasi citra. Proses tersebut dilakukan untuk memisahkan antara foreground (objek yang dikehendaki) dengan background (objek lain yang tidak dikehendaki). Pada hasil segmentasi, foreground direpresentasikan oleh warna putih (1) dan background direpresentasikan oleh warna hitam (0). Pada kasus segmentasi pada satu citra saja, kita dapat menentukan nilai threshold dengan metode trial and error. Namun pada kasus segmentasi pada citra dengan jumlah yang banyak, dibutuhkan suatu metode untuk menentukan nilai threshold secara otomatis. Nilai threshold dapat diperoleh secara otomatis dengan menggunakan metode Otsu (1979).

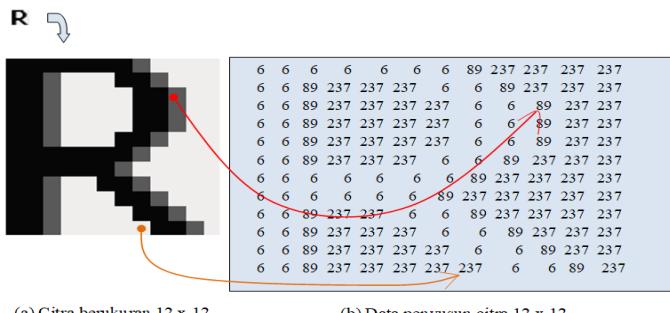
Citra digital dibentuk oleh kumpulan titik yang dinamakan piksel (pixel atau “picture element”). Setiap piksel digambarkan sebagai satu kotak kecil. Setiap piksel mempunyai koordinat posisi.



Gambar 1.6: Ilustrasi Posisi Pixel

1. Representasi Citra Digital

Citra digital sesungguhnya dibentuk melalui pendekatan yang dinamakan kuantisasi. Kuantisasi adalah prosedur yang dipakai untuk membuat suatu isyarat yang bersifat kontinu ke dalam bentuk diskrit



(a) Citra berukuran 12 x 12

(b) Data penyusun citra 12 x 12

Gambar 1.7: Representasi citra digital

2. Kualitas citra

Di samping cacah intensitas kecerahan, jumlah piksel yang digunakan untuk menyusun suatu citra mempengaruhi kualitas citra. Istilah resolusi citra biasa dinyatakan jumlah piksel pada arah lebar dan tinggi.

Resolusi piksel biasa dinyatakan dengan notasi $m \times n$, dengan m menyatakan tinggi dan n menyatakan lebar dalam jumlah piksel.

3. Penerapan Pengolahan Citra Digital

Pemanfaatan teknologi Pengolahan citra digital banyak diterapakan pada sistem cerdas, saat ini tidak asing bagi kita melihat bahkan menggunakan pengolahan citra dan pemrosesan citra dikehidupan sehari-hari. Mulai dari fitur pada kamera dan video di smartphone yang menggunakan kecerdasan buatan seperti kamera yang dapat menghasilkan gambar dengan filter-filter unik dan menarik, CCTV yang dapat memantau pergerakan dan lain sebagainya. Beberapa aplikasi dari pengolahan citra diantaranya diterapkan pada bidang: Militer, kedokteran, biologi, pendidikan, geografi dan geologi, kepolisian atau hukum, perdagangan, hiburan dan komunikasi data.

a. Aplikasi Deteksi, klasifikasi dan identifikasi objek

Ditahun 2020 setidaknya ada puluhan aplikasi gratis maupun berbayar yang menerapkan Deteksi, klasifikasi dan identifikasi objek, beberapa aplikasi ini sempat menghebohkan dunia karena keunikannya dalam menerapkan identifikasi objek citra. Berikut aplikasi-aplikasi yang banyak digunakan yaitu Luxad, Blippar, Face2gene, FaceApp, Applock, FacePhi, Log Me, FaceFirst, Pictriev, BetaFace, Kuznech Face Detection dan lain-lain.



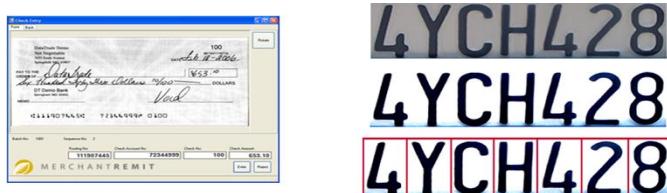
Gambar 1.8 aplikasi FaceApp

b. Aplikasi Pengenalan pola

Pengenalan pola adalah disiplin ilmu yang mengklasifikasikan object berdasarkan citra, berat atau parameter-parameter yang telah ditentukan kedalam sejumlah kategori atau kelas. Contoh aplikasi yang menerapkan pengenalan pola adalah sebagai berikut:

c. Machine Vision

Pengenalan pola menjadi dasar dari sistem mesin ini. Mesin ini menangkap sebuah atau sekelompok object dengan kamera dan selanjutnya dianalisa untuk di deskripsikan object atau benda tersebut



Gambar 1.9 Pengenalan Pola Citra

d. Character recognition (OCR)

Salah satu area pengenalan pola yang secara umum menangani permasalahan otomatisasi dan informasi. Sistem OCR mempunyai front end device yang terdiri dari pembangkit cahaya, lensa scan, document transport dan sebuah detektor.

e. Computer aided diagnosis

Sistem ini membantu dokter dalam mengambil keputusan suatu diagnosa

f. Speech recognition

Pengenalan pola suara salah satu aplikasi yang berkembang saat ini. Sistem ini mengijinkan kita untuk berkomunikasi antara manusia dengan memasukkan data ke computer. Meningkatkan efisiensi industri manufaktur, mengontrol mesin dengan berbicara pada mesin itu.

g. Face recognition

Pengenalan wajah adalah sebuah system yang mengenali image wajah manusia yang digunakan dalam otomatisasi dan security sebuah industry.

h. Biometrics

Biometric beguna untuk mengenali suatu pola mahluk hidup yang dihubungkan dengan parameter ? parameter psikologi maupun tingkah laku

i. Image Data Base Retrieval

Image Data Base Retrieval Adalah sebuah sistem untuk pengembalian citra data base. Sebuah sistem pengambilan gambar yang digunakan untuk browsing, mencari dan mengambil gambar dari database besar dari gambar digital. Metode pengambilan gambar yang paling tradisional dan umum menggunakan beberapa metode penambahan metadata seperti teks, kata kunci, judul, atau deskripsi ke gambar sehingga pengambilan dapat dilakukan pada kata-kata anotasi.

j. Data mining

Adalah pengelompokan pola objek sejumlah data yang terurut dengan harapan dapat memberikan informasi yang berguna dan diinginkan.

k. Bioinformatics

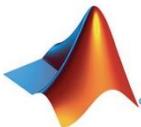
Bioinformatik berhubungan erat dengan disiplin kedokteran, pengenalan pola atau image dari suatu image penyakit atau pola dalam sebuah analisa diagnosa penyakit atau pengenalan pola pola yang berhubungan dengan dunia biologi secara umum

D. Tools untuk Belajar Pengolahan Citra Digital

Pengolahan Citra Digital dapat menggunakan *tools* seperti *Matlab*, *Octave*, *Scilab*, *Anaconda Navigator*, *Jupyter Notebook* dan sebagainya. Berikut ini beberapa aplikasi dan tools yang cukup bermanfaat untuk dicoba.

1. Matlab

MATLAB (*Matrix Laboratory*) adalah sebuah lingkungan komputasi numerikal dan bahasa pemrograman komputer generasi keempat. Dikembangkan oleh The MathWorks, MATLAB memungkinkan pengguna untuk memanipulasi matriks, penggunaan plot, fungsi dan data serta implementasi algoritma. Matlab juga dapat membantu dalam pembuatan antarmuka pengguna, dan dengan program dalam bahasa lainnya. Meskipun hanya bernuansa numerik, sebuah *toolbox* yang menggunakan mesin simbolik MuPAD, memungkinkan akses terhadap kemampuan aljabar komputer. Sebuah paket tambahan, Simulink, menambahkan simulasi grafis multiranah dan Desain Berdasar-Model dan dinamik.



Gamba 1.10r: Matlab

2. Scilab

Scilab adalah paket komputasi numerik yang dikembangkan sejak 1990 oleh para peneliti dari INRIA dan ENPC, tepatnya sejak pendirian konsorsium Scilab pada Mei 2003. Scilab adalah bahasa pemrograman tingkat tinggi, sebagian besar kegunaannya didasarkan pada seputar kemampuan menspesifikasi banyak komputasi dengan sedikit baris kode. Scilab melakukan hal ini dengan mengabstraksi tipe data primitif kepada matriks ekuivalen menurut fungsinya.

Scilab memiliki kesamaan fungsionalitas dengan MATLAB, tetapi tersedia untuk diunduh tanpa biaya lisensi. Program ini memungkinkan pengguna untuk melakukan komputasi pada cakupan luas operasi-operasi matematika dari operasi yang relatif sederhana seperti

perkalian hingga kepada operasi tingkat tinggi seperti korelasi dan aritmetika kompleks. Perangkat ini sering dipakai untuk pemrosesan sinyal, analisis statistika, perbaikan gambar, simulasi dinamika fluida, dan lain-lain.



Gambar 1.11: Scilab

3. GNU Octave

GNU Octave merupakan perangkat lunak yang dapat didistribusikan kembali secara bebas. Pengguna dapat mendistribusikan ulang atau memodifikasinya di bawah ketentuan *GNU General Public License* (GPL) yang diterbitkan oleh Free Software Foundation. GNU Octave digunakan untuk analisis numerik, dan setara dengan kemampuan perangkat lunak MATLAB. Dikembangkan oleh John W. Eaton yang rilis pada tahun 1988. GNU Octave adalah bahasa tingkat tinggi, terutama ditujukan untuk komputasi numerik. Ini menyediakan antarmuka baris perintah yang nyaman untuk memecahkan masalah linier dan nonlinier secara numerik, dan untuk melakukan eksperimen numerik lainnya menggunakan bahasa yang sebagian besar kompatibel dengan Matlab.



Gambar 1.12: GNU Octave

4. Anaconda Navigator

Didalam buku ini penulis akan menggunakan tools untuk mendukung proses belajar dan Latihan pengolahan data menggunakan Jupyter Notebooks yang ada di

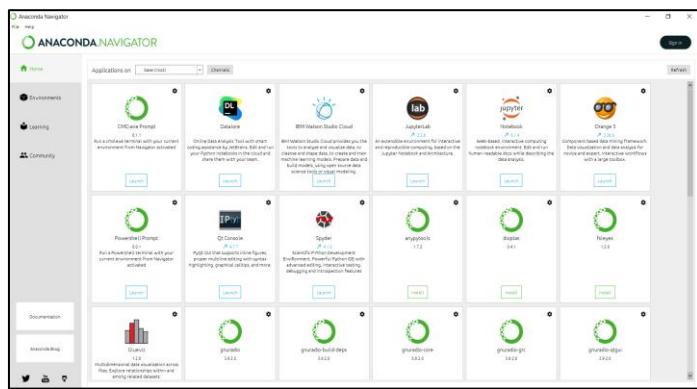
Anaconda Navigator. Oleh sesab itu maka penulis akan menjabarkan lebih Panjang tentang penggunaan Anaconda navigator serta Jupyter Notebooks. Berikut ini ulasannya.

Anaconda Navigator adalah antarmuka pengguna grafis desktop (GUI) yang disertakan dalam distribusi Anaconda® yang memungkinkan para pengguna meluncurkan aplikasi dan mengelola paket, lingkungan, dan saluran conda dengan mudah tanpa menggunakan perintah baris perintah. Navigator dapat mencari paket di Anaconda.org atau di Repositori Anaconda local yang tersedia untuk Windows, macOS, dan Linux.



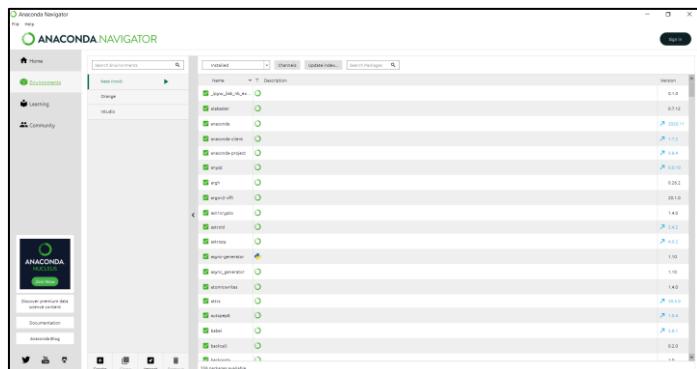
Gambar 1.13: Anaconda

Anaconda Navigator merupakan cara mudah untuk mengakses beberapa tools lain yang berada pada navigasi anaconda, pengguna dapat dengan mudah menunjuk dan klik untuk bekerja dengan paket dan lingkungan navigator tanpa perlu mengetikkan perintah conda di jendela terminal. Pengguna dapat menggunakannya untuk menemukan paket yang sesuai dengan menginstal, menjalankan paket, dan memperbarui, semua dapat dilakukan di dalam Navigator.



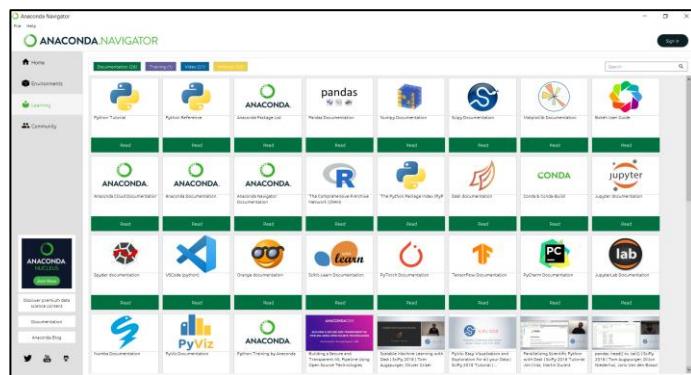
Gambar 1.14: Jendela Anaconda Navigator

Pada *Environments* tab memungkinkan pengguna untuk mengelola install-an lingkungan , paket , dan saluran .



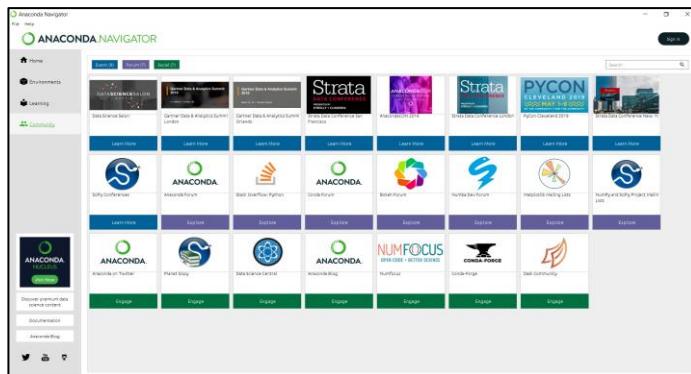
Gambar 1.15: Environments Tab

Pada tab *Learning* , Pengguna dapat mempelajari lebih lanjut tentang Navigator, platform Anaconda, dan tentang ilmu data secara terbuka. Pengguna dapat meng-Klik tombol Dokumentasi, Pelatihan, Webinar, atau Video, sehingga pengguna dapat belajar lebih dalam dengan meng-klik item apa saja untuk membuka di jendela browser.



Gambar 1.16: Learning Tab

Pada tab *Community*, Pengguna dapat mempelajari lebih lanjut tentang acara, forum dukungan gratis, dan jejaring sosial yang berkaitan dengan Navigator. Caranya dengan meng-Klik tombol Acara, Forum, atau Sosial, lalu klik item apa saja untuk membukanya di jendela browser.



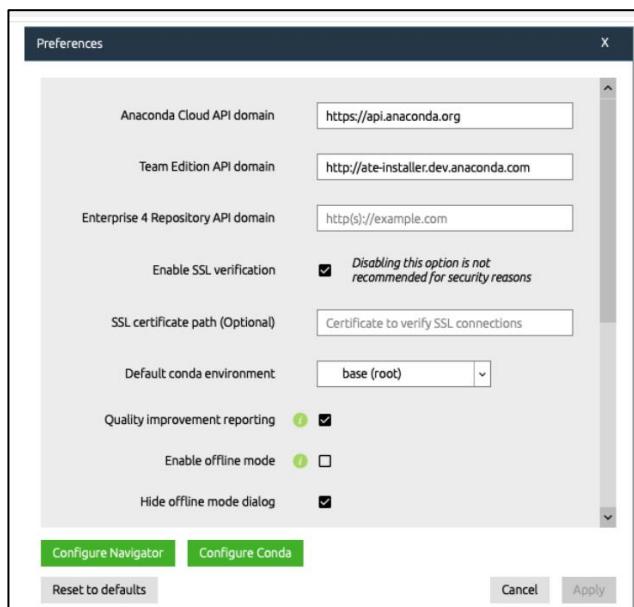
Gambar 1.17: Community Tab

Preferensi - memungkinkan pengguna untuk mengatur preferensi Navigator.

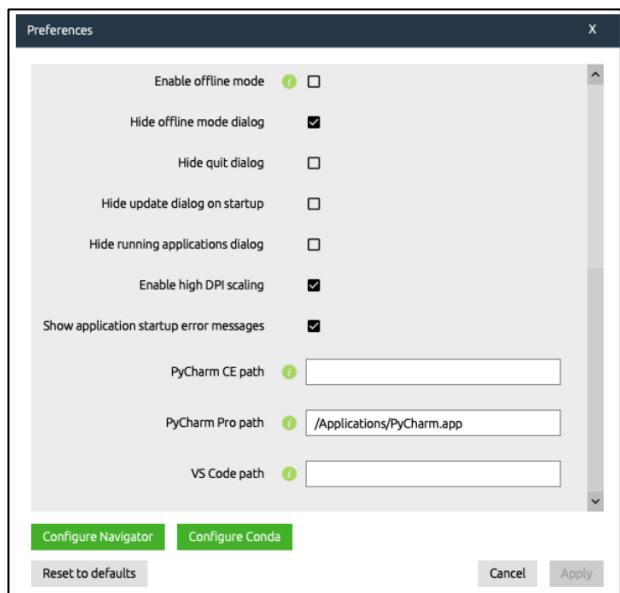
Di jendela Preferensi pengguna dapat melakukan:

- Setel domain API Anaconda.org jika pengguna akan menggunakan saluran dan paket dari anaconda.org.

- Setel domain API Edisi Tim jika pengguna akan menggunakan saluran dan paket dari server TE.
- Setel domain Enterprise 4 Repository API jika pengguna akan menggunakan saluran dan paket dari server Repo 4.
- Mengaktifkan atau menonaktifkan verifikasi SSL.
- Secara opsional, tetapkan sertifikat untuk memverifikasi koneksi SSL.
- Mengalihkan opsi untuk memberikan informasi yang tidak dapat diidentifikasi secara pribadi untuk membantu meningkatkan produk.
- Mengaktifkan atau menonaktifkan mode offline.
- Menyembunyikan kotak dialog mode offline.
- menyembunyikan kotak dialog Keluar saat keluar dari program.
- menyembunyikan kotak dialog Perbarui saat memulai program.
- menyembunyikan dialog Tutup aplikasi yang berjalan, yang biasanya ditampilkan saat keluar dari program jika masih ada aplikasi yang berjalan yang diluncurkan dari Navigator.
- mengubah tampilan Navigator dengan opsi Aktifkan penskalaan DPI Tinggi. Opsi ini dapat berguna jika Navigator tidak ditampilkan dengan benar pada beberapa layar DPI tinggi.
- Menampilkan pesan kesalahan startup aplikasi.
- Menyetel jalur Edisi Komunitas PyCharm jika tidak diinstal di lokasi default.
- Menyetel jalur PyCharm Pro jika tidak diinstal di lokasi default.
- Menetapkan jalur Kode Visual Studio jika tidak diinstal di lokasi default.



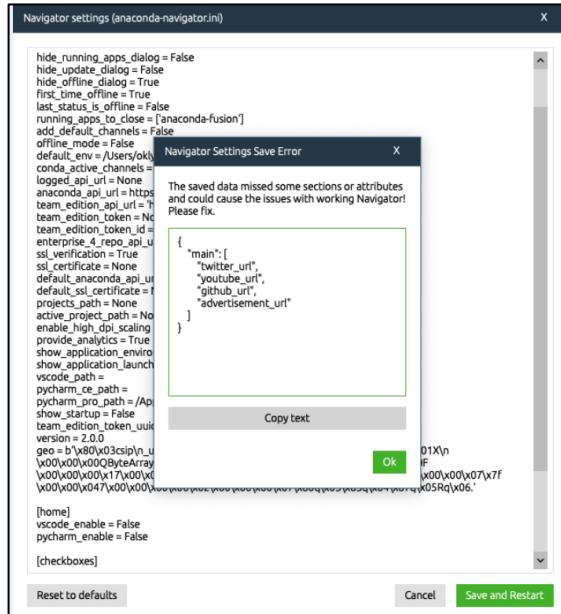
Gambar 1.18: Preference Anaconda cloud API domain



Gambar 1.19: Preference mode online/offline



Gambar 1.20: Navigator Settings



Gambar 1.21: Navigator Settings



Gambar 1.22: Conda Settings

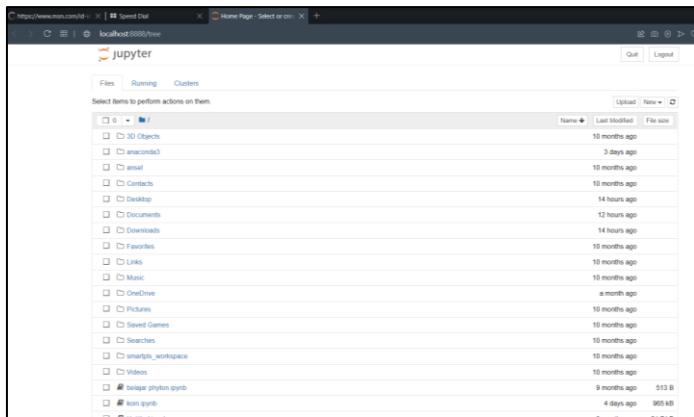
5. Jupyter Notebook

Notebook Jupyter adalah *open-source web application* yang memungkinkan para pengguna membuat dan berbagi dokumen yang berisi kode langsung, persamaan, visualisasi, dan teks naratif. Penggunaannya meliputi: pembersihan dan transformasi data, simulasi numerik, pemodelan statistik, visualisasi data, pembelajaran mesin, dan banyak lagi.



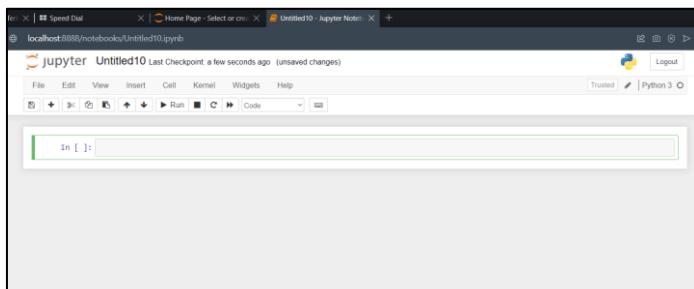
Gambar 1.23: Jupyter Notebook

Terdapat 3 tab utama pada *Home Page*, Jupyter Notebook yaitu *File*, *Running*, *Cluster*, pada sisi kanan di ujung terdapat tab *Quit* dan *Logout*. Dan tab *Upload* dan *New*. Pada sisi dibawah terdapat folder-folder untuk dipilih, yaitu Pilih item untuk melakukan tindakan pada item tersebut, pada folder yang dipilih akan menempatkan simpanan file yang akan dibuat.



Gambar 1.24: Home page

Jika memilih pilihan *New -> Python3*, maka akan tampil jendela seperti di bawah ini, yaitu jendela editor *Python3*.



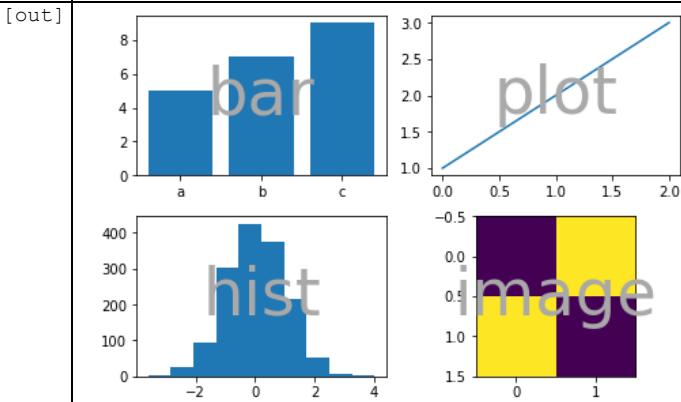
Gambar 1.25: Jendela file baru

Latihan:

Latihan ini merupakan pemahaman dalam Komposisi gambar yang kompleks dan sistematik, untuk mengenal perintah-perintah dasar di dalam memahami komposisi gambar anda dapat mencoba untuk Latihan dengan code berikut.

[in]	<pre>import matplotlib.pyplot as plt import numpy as np def identify_axes(ax_dict, fontsize=48): kw = dict(ha="center", va="center", fontsize=fontsize, color="darkgrey") for k, ax in ax_dict.items(): ax.text(0.5, 0.5, k, transform=ax.transAxes, **kw)</pre>
[in]	<pre>np.random.seed(19680801) hist_data = np.random.randn(1_500) fig = plt.figure(constrained_layout=True) ax_array = fig.subplots(2, 2, squeeze=False) ax_array[0, 0].bar(["a", "b", "c"], [5, 7, 9]) ax_array[0, 1].plot([1, 2, 3]) ax_array[1, 0].hist(hist_data, bins="auto") ax_array[1, 1].imshow([[1, 2], [2, 1]]) identify_axes({(j, k): a for j, r in enumerate(ax_array) for k, a in enumerate(r)},)</pre>
[out]	<p>The output consists of four subplots arranged in a 2x2 grid. Top-left: A bar chart with three bars labeled 'a', 'b', and 'c' on the x-axis. The values are approximately 5, 7, and 9 respectively. A label '(0, 0)' is overlaid on the first bar. Top-right: A line plot with three points at (1, 1), (2, 2), and (3, 3). A label '(0, 1)' is overlaid on the first point. Bottom-left: A histogram of random data with a peak around zero. A label '(1, 0)' is overlaid on the histogram. Bottom-right: A 2x2 heatmap with alternating yellow and dark purple squares. A label '(1, 1)' is overlaid on the top-right square.</p>

```
[in] fig = plt.figure(constrained_layout=True)
ax_dict = fig.subplot_mosaic(
    [
        ["bar", "plot"],
        ["hist", "image"]
    ],
)
ax_dict["bar"].bar(["a", "b", "c"], [5, 7, 9])
ax_dict["plot"].plot([1, 2, 3])
ax_dict["hist"].hist(hist_data)
ax_dict["image"].imshow([[1, 2], [2, 1]])
identify axes(ax dict)
```

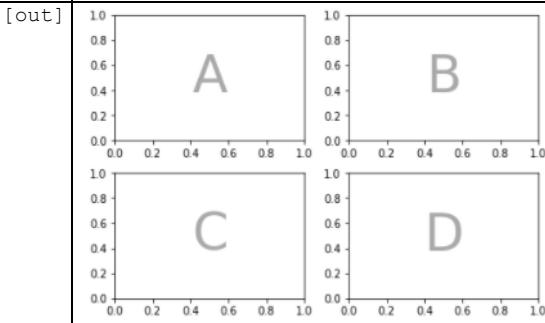


```
[in] print(ax_dict)
```

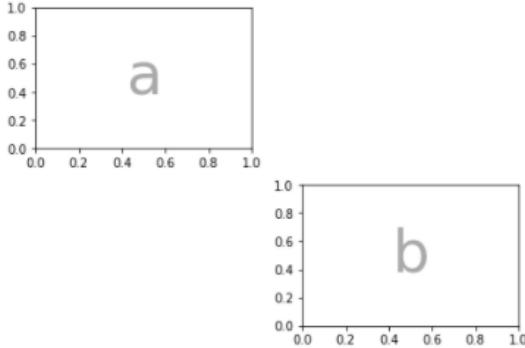
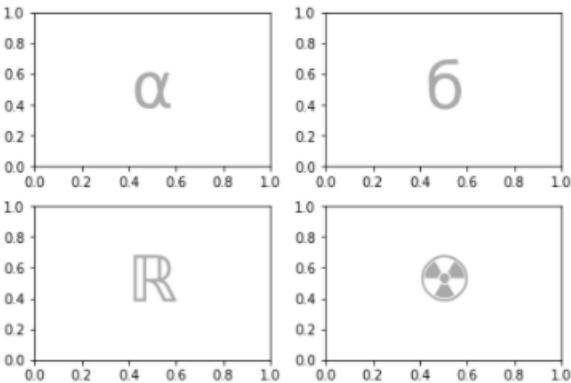
```
[out] {'hist': <AxesSubplot:label='hist'>, 'image': <AxesSubplot:label='image'>, 'bar': <AxesSubplot:label='bar'>, 'plot': <AxesSubplot:label='plot'>}
```

```
[in] mosaic = """
    AB
    CD
    """
```

```
[in] fig = plt.figure(constrained_layout=True)
ax_dict = fig.subplot_mosaic(mosaic)
identify axes(ax dict)
```



[in]	<pre>axd = plt.figure(constrained_layout=True).subplot_mosaic(""" ABD CCD """) identify_axes(axd)</pre>
[out]	
[in]	<pre>axd = plt.figure(constrained_layout=True).subplot_mosaic(""" .C. BBB .D. """) identify_axes(axd)</pre>
[out]	
[in]	<pre>axd = plt.figure(constrained_layout=True).subplot_mosaic(""" aX Xb """ , empty_sentinel="X",)</pre>

	<code>identify axes (axd)</code>
[out]	 <p>The output shows two separate subplots. The top subplot contains the letter 'a' and the bottom subplot contains the letter 'b', both centered within their respective square axes.</p>
[in]	<pre>axd = plt.figure(constrained_layout=True).subplot_mosaic(""" αβ R:: """) identify axes (axd)</pre>
[out]	 <p>The output shows four subplots arranged in a 2x2 grid. The top-left subplot contains the Greek letter alpha (α), the top-right subplot contains the Greek letter beta (β), the bottom-left subplot contains the letter R, and the bottom-right subplot contains a radiation symbol (a circle with three segments).</p>
[in]	<pre>axd = plt.figure(constrained_layout=True).subplot_mosaic(""" .a. bAc .d. """ , gridspec_kw={ # set the height ratios between the rows "height_ratios": [1, 3.5, 1], # set the width ratios between the columns "width ratios": [1, 3.5, 1], },)</pre>

	<code>identify_axes(axd)</code>
[out]	
[in]	<pre> mosaic = """AA BC""" fig = plt.figure() axd = fig.subplot_mosaic(mosaic, gridspec_kw={ "bottom": 0.25, "top": 0.95, "left": 0.1, "right": 0.5, "wspace": 0.5, "hspace": 0.5, },) identify_axes(axd) axd = fig.subplot_mosaic(mosaic, gridspec_kw={ "bottom": 0.05, "top": 0.75, "left": 0.6, "right": 0.95, "wspace": 0.5, "hspace": 0.5, },) identify_axes(axd) </pre>

[out]	
[in]	<pre>axd = plt.figure(constrained_layout=True).subplot_mosaic("AB", subplot_kw={"projection": "polar"}) identify_axes(axd)</pre>
[out]	
[in]	<pre>axd = plt.figure(constrained_layout=True).subplot_mosaic([["main", "zoom"], ["main", "BLANK"]], empty_sentinel="BLANK", gridspec_kw={"width_ratios": [2, 1]},) identify_axes(axd)</pre>
[out]	

[in]	<pre> inner = [["inner A"], ["inner B"],] outer nested mosaic = [["main", inner], ["bottom", "bottom"],] axd = plt.figure(constrained layout=True).subplot mosaic(outer nested mosaic, empty sentinel=None) identify axes(axd, fontsize=36) </pre>
[out]	
[in]	<pre> mosaic = np.zeros((4, 4), dtype=int) for j in range(4): mosaic[j, j] = j + 1 axd = plt.figure(constrained layout=True).subplot mosaic(mosaic, empty sentinel=0,) identify axes(axd) </pre>
[out]	

Soal:

Pilih salah satu jawaban yang benar dengan memberi tanda pada lembar jawaban.

1. Bagaimana cara kerja Computer Vision?
 - a. Mengidentifikasi dan memproses objek
 - b. Menghitung dan memfilter teks
 - c. Menghitung Gambar
 - d. Menyimpan file Gambar
 - e. Membuat gambar 3D
2. Computer vision membagi gambar-gambar yang telah dimasukkan menjadi sebuah bentuk, bentuk apakah yang diterjemahkan komputer untuk memproses gambar?
 - a. Bentuk Teks
 - b. Bentuk Audio
 - c. Bentuk Gambar
 - d. Bentuk Piksel dan bentuk angka
 - e. Bentuk
3. Pada umumnya, berdasarkan kombinasi warna pada piksel, citra dibagi menjadi beberapa jenis yaitu citra...
 - a. CMYK, RGB, Citra grayscale
 - b. Citra Biner dan Citra analog
 - c. RGB, citra grayscale, dan citra biner
 - d. Citra grayscale dan Citra analog
 - e. Citra biner, citra grayscale dan citra analog
4. Jika representasi warna kuning(*yellow*) memiliki nilai intensitas piksel R:255 G:255 B:0, maka berapakah nilai intensitas piksel warna Putih(*white*)...
 - a. R: 0 G:255 B:0
 - b. R: 0 G:255 B:255
 - c. R: 102 G:255 B:204
 - d. R: 255 G:255 B:255
 - e. R: 204 G:102 B:204

5. Yang tidak termasuk dalam tipe warna adalah ...
- a. Biner
 - b. RGB
 - c. Grayscale
 - d. CMYK
 - e. Piksel

BAB 2

MENGENAL PYTHON LIBRARIES UNTUK PEMROSESAN GAMBAR

Dalam bab ini penulis akan membuat daftar pustaka pemrosesan gambar paling berguna dan paling sering digunakan dengan Python, berikut 8 *Python Libraries* untuk pemrosesan gambar teratas yang digunakan dalam Pembelajaran Mesin/*Machine Learning*.

A. OpenCV

OpenCV adalah perpustakaan(*Library*) *open-source* yang dikembangkan oleh Intel pada tahun 2000. Hal ini sebagian besar digunakan dalam tugas-tugas *computer visi* seperti deteksi objek, deteksi wajah, pengenalan wajah, segmentasi gambar, dll tetapi juga berisi banyak fungsi berguna yang mungkin perlu dalam Pengolahan citra digital.



Gambar 2.1: OpenCV

OpenCV (opencv.org) merupakan sebuah framework yang banyak digunakan untuk keperluan pemrosesan image dan video. OpenCV bersifat gratis dan tersedia untuk berbagai bahasa pemrograman, seperti C++, Python, Java dan lain-lain pada platform Windows, Linux, iOS, Android dan seterusnya. OpenCV dapat digunakan untuk melakukan

berbagai operasi terhadap citra dan video dengan berbagai algoritma terbaru serta banyak digunakan pada berbagai sistem pengolahan citra yang populer.

Contoh kode berikut adalah untuk memanggil file gambar dengan OpenCV :

```
import cv2 as cv  
img = cv.imread('kucing.jpg')
```

B. Scikit-Image

Scikit-Image adalah library pemrosesan gambar berbasis python yang memiliki beberapa bagian yang ditulis dalam *Cython* (*Cython* adalah bahasa pemrograman yang merupakan superset dari bahasa pemrograman Python yang dirancang untuk memiliki kinerja seperti bahasa pemrograman C.) untuk mencapai kinerja yang baik.



Gambar 2.2: Sicikit-Image

Berikut Ini Scikit_Image termasuk algoritma untuk:

- Segmentasi,
- Transformasi geometris,
- Manipulasi ruang warna,
- Analisis,
- Filtering ,
- Morfologi,
- Morphology, dan banyak lagi

Scikit-image menggunakan *array NumPy* sebagai objek gambar.

Contoh kode berikut merupakan contoh untuk memfilter gambar dengan scikit-image.

```
from skimage import data, io, filters

image = data.coins()
# ... or any other NumPy array!
edges = filters.sobel(image)
io.imshow(edges)
io.show()
```

C. SciPy

Scipy digunakan untuk perhitungan matematis dan ilmiah tetapi juga dapat melakukan pemrosesan gambar multi-dimensi menggunakan submodul *scipy.ndimage*. Fungsi ini menyediakan fungsi untuk beroperasi pada array Numpy n-dimensi.



Gambar 2.3: Scipy

Scipy menawarkan operasi pemrosesan gambar yang paling umum digunakan seperti:

- Reading Images
- Image Segmentation
- Convolution
- Face Detection
- Feature Extraction

Contoh berikut adalah contoh untuk melakukan pemrosesan gambar multi-dimensi menggunakan submodul *scipy.ndimage*.

```
from scipy import misc, ndimage
from matplotlib import pyplot as plt

face = misc.face()
blurred_face = ndimage.gaussian_filter(face, sigma=3)
```

D. Pillow/PIL

PIL (*Python Imaging Library*) adalah *open-source library* untuk tugas pemrosesan gambar yang membutuhkan bahasa pemrograman python. PIL dapat melakukan tugas pada gambar seperti membaca, mengubah skala, menyimpan dalam format gambar yang berbeda.

Contoh kode untuk membuka, memutar, dan menampilkan gambar (*menggunakan penampil default*).

```
from PIL import Image
with Image.open("kucing.jpg") as im:
    im.rotate(45).show()
```

Fungsi

```
PIL.Image.open(fp, mode='r', formats=None)
```

Fungsi untuk membuka dan mengidentifikasi file gambar yang diberikan.

Fungsi ini mengidentifikasi file, tetapi file tetap terbuka dan data gambar sebenarnya tidak dibaca dari file sampai pengguna mencoba memproses data atau memanggil **load()**.

E. NumPy

Gambar pada dasarnya adalah array nilai piksel di mana setiap piksel diwakili oleh nilai 1 (*skala abu-abu*) atau 3 (RGB). Oleh karena itu, NumPy dapat dengan mudah melakukan tugas seperti pemotongan gambar, penyembunyian, atau manipulasi nilai piksel.



Gambar 2.4: NumPy

Contoh kode untuk Pembuatan himpunan (array).

```
import numpy as np
x = np.array([1, 2, 3])
x
```

F. Mahotas

Mahotas adalah perpustakaan pemrosesan gambar dan visi komputer lain yang dirancang untuk informatika bioimage. Ia membaca dan menulis gambar dalam array NumPy, dan diimplementasikan dalam C++ dengan antarmuka python yang lebih halus.

Fungsi Mahotas yang populer adalah sebagai berikut:

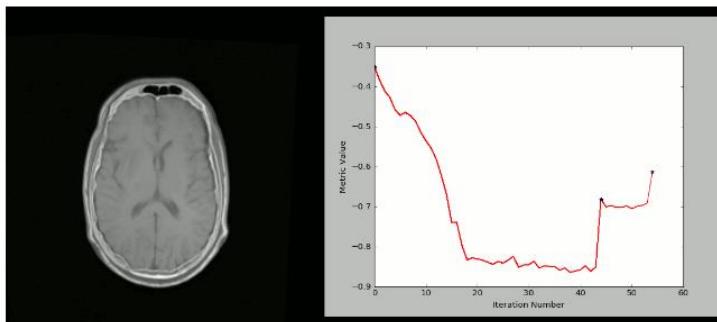
- Watershed
- Convex points calculations.
- Hit & miss. thinning
- Zernike & Haralick, local binary patterns, and TAS features.
- Morphological processing
- Speeded-Up Robust Features (SURF), a form of local features
- Thresholding
- Convolution.
- Sobel edge detection.

Contoh kode untuk fungsi **distance()** sederhana yang menghitung peta jarak:

```
import mahotas
dmap = mahotas.distance(f)
```

G. SimpleITK

ITK atau *Insight Segmentation and Registration Toolkit* adalah platform *open-source* yang banyak digunakan untuk Segmentasi Gambar dan Registrasi Gambar (proses yang melapisi dua atau lebih gambar).



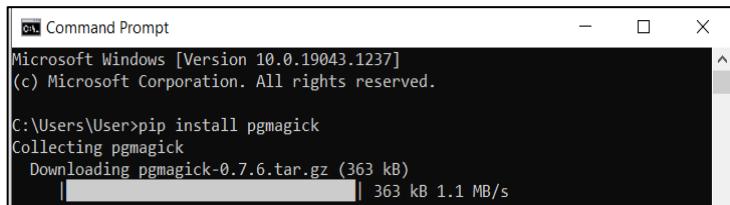
Gambar 2.5: hasil gambar menggunakan simpleITK

H. Pgmagick

Pgmagick adalah pengikatan GraphicsMagick untuk Python yang menyediakan utilitas untuk tampil pada gambar seperti mengubah ukuran, rotasi, penajaman, gambar gradien, menggambar teks, dll.

Cara Install:

```
$ pip install pgmagick
```



Gambar 2.6: instalasi Pgmagik

Contoh kode untuk memburamkan gambar:

```
from pgmagick.api import Image
    img = Image('leena.jpeg')
        # blur image
            Img.blur(10, 5)
```

Latihan:

1. "CN" color selection

Matplotlib mengonversi warna "CN" ke RGBA saat menggambar. Bagian *Styling* berisi informasi tambahan tentang mengontrol warna dan properti gaya.

```
[in]   import numpy as np
        import matplotlib.pyplot as plt
        import matplotlib as mpl

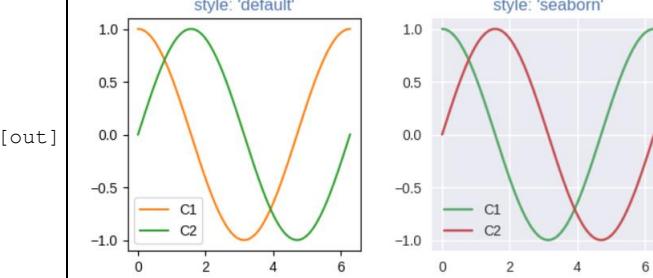
        th = np.linspace(0, 2*np.pi, 128)

        def demo(sty):
            mpl.style.use(sty)
            fig, ax = plt.subplots(figsize=(3, 3))

            ax.set_title('style: {!r}'.format(sty),
color='C0')

            ax.plot(th, np.cos(th), 'C1', label='C1')
            ax.plot(th, np.sin(th), 'C2', label='C2')
            ax.legend()

        demo('default')
        demo('seaborn')
```



2. Perbandingan antara warna X11/CSS4 dan xkcd

```
import matplotlib._color_data as mcd
import matplotlib.patches as mpatch
import matplotlib.pyplot as plt

overlap = {name for name in mcd.CSS4_COLORS
           if "xkcd:" + name in
mcd.XKCD_COLORS}

fig = plt.figure(figsize=[9, 5])
ax = fig.add_axes([0, 0, 1, 1])

n_groups = 3
n_rows = len(overlap) // n_groups + 1

for j, color_name in
enumerate(sorted(overlap)):
    css4 = mcd.CSS4_COLORS[color_name]
    xkcd = mcd.XKCD_COLORS["xkcd:" +
color_name].upper()

    col_shift = (j // n_rows) * 3
    y_pos = j % n_rows
    text_args = dict(va='center', fontsize=10,
                      weight='bold' if css4 ==
xkcd else None)
    ax.add_patch(mpatch.Rectangle((0 +
col_shift, y_pos), 1, 1, color=css4))
    ax.add_patch(mpatch.Rectangle((1 +
col_shift, y_pos), 1, 1, color=xkcd))
    ax.text(0 + col_shift, y_pos + .5, ' ' +
css4, alpha=0.5, **text_args)
    ax.text(1 + col_shift, y_pos + .5, ' ' +
xkcd, alpha=0.5, **text_args)
    ax.text(2 + col_shift, y_pos + .5, ' ' +
color_name, **text_args)

for g in range(n_groups):
    ax.hlines(range(n_rows), 3*g, 3*g + 2.8,
color='0.7', linewidth=1)
    ax.text(0.5 + 3*g, -0.5, 'X11',
ha='center', va='center')
    ax.text(1.5 + 3*g, -0.5, 'xkcd',
ha='center', va='center')

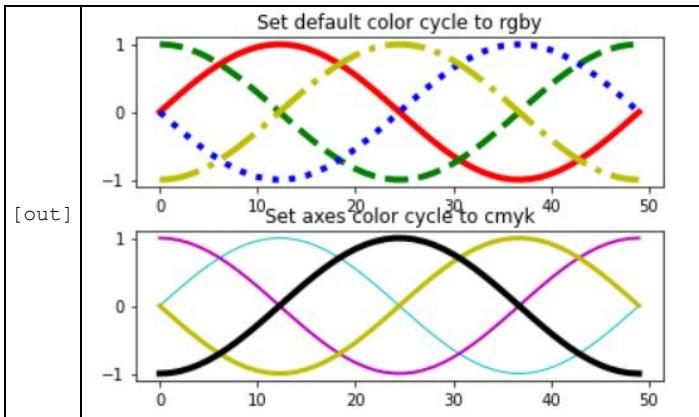
ax.set_xlim(0, 3 * n_groups)
ax.set_ylim(n_rows, -1)
ax.axis('off')

plt.show()
```

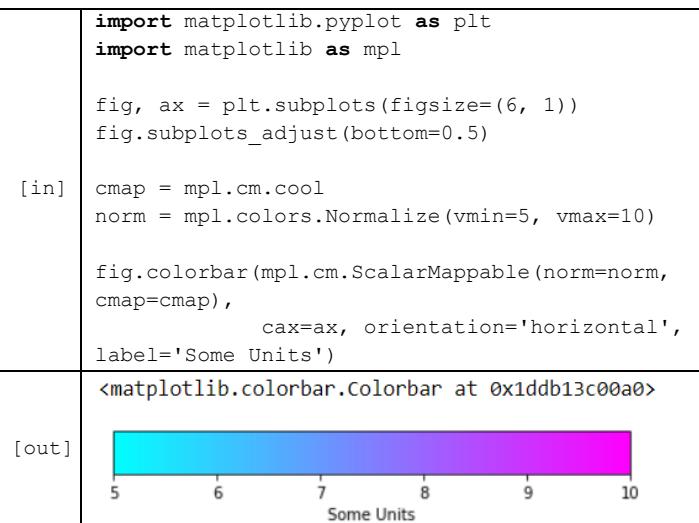
X11	xkcd	X11	xkcd	X11	xkcd
#00FFFF	#13EABC	aqua	#000000	#35B01A	green
#00FFFF	#04DBB2	aquamarine	#808080	#929591	grey
#00FFFF	#00A9F3	azure	#000000	#3CB371	indigo
#F5F5DC	#E6DA6	beige	#FFFFFF	#FFFFCB	ivory
#000000	#000000	black	#F0E68C	#AAA662	kaki
#A52A2A	#853700	brown	#E6E6FA	#C79FEF	lavender
#FFFF00	#C1FB0A	chartreuse	#ADD8E6	#7BC8F6	lightblue
#008080	#00008B	chocolate	#90EE90	#7FFF7B	lightgreen
#FF7F50	#FF0080	coral	#FF1493	#C2007B	magenta
#0000CD	#800080	crimson	#000000	#800020	maroon
#00FFFF	#00FFFF	cyan	#0000CD	#0000CD	navy
#00008B	#00008B	darkblue	#800000	#007500	olive
#008080	#008080	darkgreen	#FFA500	#FF7300	orange
#FF00FF	#FF00FF	fuchsia	#FFA500	#E420F	orangered
#FFD700	#DDB4AC	gold	#DA70D6	#C875C4	orchid
#DAA520	#FAC205	goldenrod	#FFCC00	#FF1400	pink

3. Styling dengan cycler

[in]	<pre>from cycler import cycler import numpy as np import matplotlib.pyplot as plt</pre>
[in]	<pre>x = np.linspace(0, 2 * np.pi, 50) offsets = np.linspace(0, 2 * np.pi, 4, endpoint=False) yy = np.transpose([np.sin(x + phi) for phi in offsets])</pre>
[in]	<pre>print(yy.shape)</pre>
[out]	<pre>(50, 4)</pre>
[in]	<pre>default_cycler = (cycler(color=['r', 'g', 'b', 'y']) + cycler(linestyle=['-', '--', ':', '-.']))</pre>
	<pre>plt.rc('lines', linewidth=4) plt.rc('axes', prop_cycle=default_cycler)</pre>
[in]	<pre>custom_cycler = (cycler(color=['c', 'm', 'y', 'k']) + cycler(lw=[1, 2, 3, 4]))</pre>
	<pre>fig, (ax0, ax1) = plt.subplots(nrows=2) ax0.plot(yy) ax0.set_title('Set default color cycle to rgby') ax1.set_prop_cycle(custom_cycler) ax1.plot(yy) ax1.set_title('Set axes color cycle to cmyk') # Tambahkan sedikit lebih banyak ruang di # antara dua plot. fig.subplots_adjust(hspace=0.3) plt.show()</pre>



4. Basic continuous colorbar



Soal:

Pilih salah satu jawaban yang benar dengan memberi tanda pada lembar jawaban.

1. Kode mana untuk memanggil file gambar dengan OpenCV...
 - a. cv.imread()
 - b. np.linspace()
 - c. plt.subplots()
 - d. mpl.style.use()
 - e. plt.figure()
2. Scikit_Image termasuk algoritma untuk, kecuali...
 - a. Segmentasi
 - b. Filtering
 - c. Morphology
 - d. Transformasi Geometris
 - e. Convolution
3. Yang bukan operasi pemrosesan gambar yang paling umum menggunakan Scipy diantaranya,adalah:
 - a. Segmentasi Gambar
 - b. Feature Extraction
 - c. Morphology
 - d. Face Detection
 - e. Convolution
4. Kode manakah untuk memanggil perintah filter gambar dengan scikit-image...
 - a. pip install scikit-image
 - b. from skimage import data
 - c. from skimage.util import img_as_float
 - d. from skimage import data, io, filters
 - e. from skimage import exposure

5. Algoritma *Sobel edge detection* merupakan bagian fungsi yang sering digunakan pada Python Libraries...
 - a. OpenCV
 - b. Scikit-Image
 - c. Scipy
 - d. NumPy
 - e. Mahotas

BAB 3

KONSEP DASAR PENGOLAHAN CITRA DIGITAL

Citra adalah suatu representasi atau gambaran yang menyerupai, mirip atau tiruan dari suatu objek. Pada umumnya citra terbagi 2 yaitu citra yang bersifat analog dan citra yang bersifat digital.

1. Citra Analog

Citra analog adalah citra yang bersifat berkelanjutan, seperti gambar pada monitor televisi, foto sinar X, foto yang tercetak di kertas foto, lukisan, pemandangan alam, hasil CT scan, gambar-gambar yang terekam pada pita kaset, dan lain sebagainya. Citra analog tidak dapat direpresentasikan dalam komputer, sehingga tidak bisa diproses di komputer secara langsung.



Gambar 3.1 CT Scan

Oleh sebab itu, agar ini dapat diproses di komputer, proses konversi analog ke digital harus dilakukan terlebih dahulu. Citra analog dihasilkan dari alat-alat analog, seperti video kamera analog, kamera foto analog, cam, CT scan, sensor rontgen untuk foto thorax, sensor gelombang

pendek pada sistem radar, sensor ultrasound pada sistem USG, dan lain-lain.

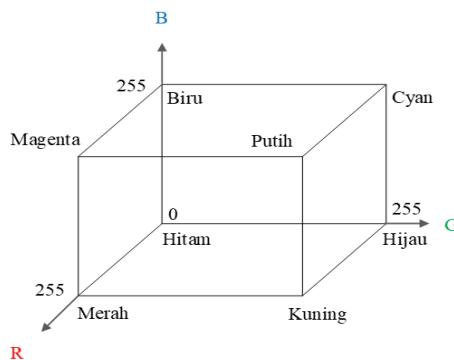
2. Citra Digital

Citra digital merupakan representatif dari citra yang diambil oleh mesin dengan bentuk pendekatan berdasarkan sampling dan kuantisasi. Sampling menyatakan besarnya kotak-kotak yang disusun dalam baris dan kolom. Dengan kata lain, sampling pada citra menyatakan besar kecilnya ukuran pixel (titik) pada citra, dan kuantisasi menyatakan besarnya nilai tingkat kecerahan yang dinyatakan dalam nilai tingkat keabuan (*grayscale*) sesuai dengan jumlah bit biner yang digunakan oleh mesin, dengan kata lain kuantisasi pada citra menyatakan jumlah warna yang ada pada citra.

A. Jenis-jenis Citra

1. Citra Berwarna

Citra berwarna, atau biasa dinamakan citra RGB, merupakan jenis citra yang menyajikan warna dalam bentuk komponen R (merah), G (hijau), dan B (biru). Setiap komponen warna menggunakan 8 bit (nilainya berkisar antara 0 sampai dengan 255). Kemungkinan warna yang bisa disajikan mencapai $255 \times 255 \times 255$ atau 16.581.375 warna.



Contoh nilai R, G dan B dari beberapa warna dasar

Warna	R	G	B
Merah	255	0	0
Hijau	0	255	0
Biru	0	0	255
Hitam	0	0	0
Putih	255	255	255
Kuning	0	255	255

2. Citra Berskala keabuan (Grayscale)

Citra grayscale menangani gradasi warna hitam dan putih, yang tentu saja menghasilkan efek warna abu-abu.

Pada jenis gambar ini, warna dinyatakan dengan intensitas yang berkisar antara 0 sampai dengan 255. Nilai 0 menyatakan hitam dan nilai 255 menyatakan putih.



Gambar 3.2 Ilustrasi citra berwarna dan citra berskala keabuan

3. Citra Biner

Citra biner adalah citra dengan setiap piksel hanya dinyatakan dengan sebuah nilai dari dua buah kemungkinan (yaitu nilai 0 dan 1).

Nilai 0 menyatakan warna hitam dan nilai 1 menyatakan warna putih.

Citra jenis ini banyak dipakai dalam pemrosesan citra, misalnya untuk kepentingan memperoleh tepi bentuk suatu objek.



Citra Daun berskala keabuan



Citra Biner

B. Kualitas Citra

Di samping cacah intensitas kecerahan, jumlah piksel yang digunakan untuk menyusun suatu citra mempengaruhi kualitas citra.

Istilah resolusi citra biasa dinyatakan jumlah piksel pada arah lebar dan tinggi. Resolusi piksel biasa dinyatakan dengan notasi $m \times n$, dengan m menyatakan tinggi dan n menyatakan lebar dalam jumlah piksel.

Kualitas citra juga dapat dinyatakan dalam bentuk Resolusi spasial yang ditentukan oleh jumlah piksel per satuan panjang.

Istilah seperti dpi (dot per inch) menyatakan jumlah piksel per inci. Misalnya, citra 300 dpi menyatakan bahwa citra akan dicetak dengan jumlah piksel sebanyak 300 sepanjang satu inci.

Jadi citra dengan resolusi ruang spasial sebesar 300 dpi dicetak di kertas dengan ukuran lebih kecil daripada yang mempunyai resolusi ruang sebesar 150 dpi, meskipun kedua gambar memiliki resolusi piksel yang sama

Latihan:

1. Skala keabu-abuan

```
In [1]: import matplotlib, cv2  
import numpy as np  
import matplotlib.pyplot as plt  
import matplotlib.image as mpimg  
%matplotlib inline
```

```
In [2]: img = cv2.imread('kucing.jpg')  
im = mpimg.imread('kucing.jpg')
```

```
In [3]: img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  
plt.imshow(img)
```

```
Out[3]: <matplotlib.image.AxesImage at 0x20b4fe51c10>
```



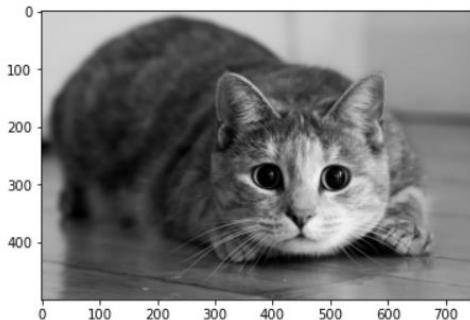
```
In [4]: gray_img = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)  
print (gray_img)
```

```
In [4]: gray_img = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
print (gray_img)
```

```
[[207 207 207 ... 166 166 166]
 [207 207 207 ... 166 166 166]
 [207 207 207 ... 166 166 166]
 ...
 [169 169 167 ... 150 151 151]
 [171 171 169 ... 151 151 151]
 [171 170 169 ... 151 151 151]]
```

```
In [5]: plt.imshow(cv2.cvtColor(gray_img, cv2.COLOR_GRAY2RGB))
```

```
Out[5]: <matplotlib.image.AxesImage at 0x20b50629c10>
```



2. Menyimpan file gambar RGB dengan library PIL dan NumPy menggunakan pil_img.save()

```
[in]   from PIL import Image
        import numpy as np

        im = np.array(Image.open('cat.png'))

        im_R = im.copy()
        im_R[:, :, (1, 2)] = 0
        im_G = im.copy()
        im_G[:, :, (0, 2)] = 0
        im_B = im.copy()
        im_B[:, :, (0, 1)] = 0

        im_RGB = np.concatenate((im_R, im_G, im_B),
                               axis=1)

        pil_img = Image.fromarray(im_RGB)
        pil_img.save('cat1.png')
```

[out]



Latihan Mandiri:

Dari Latihan Skala keabu-abuan diatas silahkan modifikasi dengan gambar/citra yang anda miliki, jelaskan dengan singkat perintah-perintah yang digunakan...

3. Memisah saluran warna keabu-abuan

Citra berwarna terdiri dari 3 kanal warna dimana citra abu-abu hanya terdiri dari 1 kanal Warna yang membawa informasi intensitas untuk setiap piksel yang menampilkan citra sebagai hitam-putih.

Kode berikut memisahkan setiap saluran warna:

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

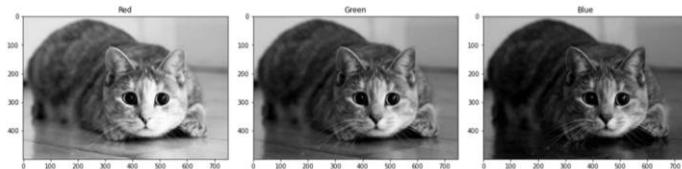
img = cv.imread('kucing.jpeg')
b, g, r = cv.split(img)

fig, ax = plt.subplots(1, 3, figsize=(16, 8))
fig.tight_layout()

ax[0].imshow(cv.cvtColor(r, cv.COLOR_BGR2RGB))
ax[0].set_title("Red")

ax[1].imshow(cv.cvtColor(g, cv.COLOR_BGR2RGB))
ax[1].set_title("Green")

ax[2].imshow(cv.cvtColor(b, cv.COLOR_BGR2RGB))
ax[2].set_title("Blue")
```



Latihan Mandiri:

Dari Latihan Memisah saluran warna keabu-abuan diatas silahkan modifikasi dengan gambar/citra yang anda miliki, jelaskan perbedaan setiap citra yang telah dimodifikasi...

Soal:

Pilih salah satu jawaban yang benar dengan memberi tanda pada lembar jawaban.

1. Berdasarkan sifatnya citra terbagi atas citra...
 - a. Citra RGB dan Citra Analog
 - b. Citra Biner dan Citra Grayscale
 - c. Citra Analog dan Citra Digital
 - d. Citra Biner dan Citra RGB
 - e. Citra Digital dan Citra biner
2. Citra grayscale menangani gradasi warna...
 - a. abu-abu, putih
 - b. abu-abu, hitam
 - c. hitam, abu-abu, putih
 - d. hitam, putih
 - e. putih,
3. `pil_img.save()` merupakan perintah untuk ...
 - a. Mendeteksi objek
 - b. Memilih warna
 - c. Membuat grafik warna
 - d. Menyimpan file gambar
 - e. Memfilter gambar berwarna
4. CT-scan dan foto hasil USG merupakan jenis dari citra ...
 - a. Citra Biner
 - b. Citra Analog
 - c. Citra Grayscale
 - d. Citra Digital
 - e. Citra Berwarna

5. Apa fungsi dari `cv2.COLOR_RGB2GRAY` dan `cv2.COLOR_BGR2GRAY` ...
 - a. Konversi warna dasar
 - b. Konversi citra berwarna
 - c. Menampilkan skala keabu-abuan
 - d. Menampilkan citra keabu-abuan
 - e. Konversi RGB ke abu-abu

BAB 4

OPERASI PIXEL DAN HISTOGRAM

A. Operasi Piksel

Operasi piksel adalah operasi pengolahan citra yang memetakan hubungan setiap piksel yang bergantung pada piksel itu sendiri.

Jika $f(y, x)$ menyatakan nilai sebuah piksel pada citra f dan $g(y, x)$ menyatakan piksel hasil pengolahan dari $f(y, x)$, hubungannya dapat dinyatakan dengan:

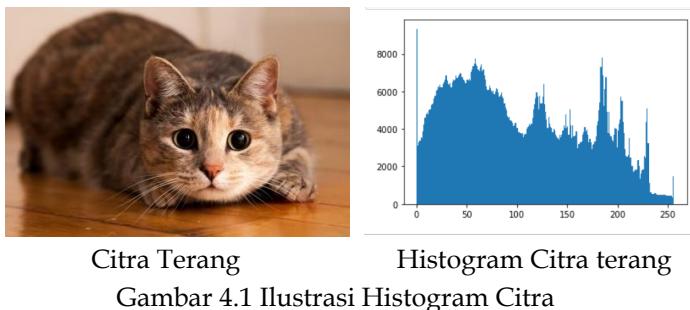
$$g(y, x) = T(f(y, x))$$

T menyatakan fungsi atau macam operasi yang dikenakan terhadap piksel $f(y, x)$

B. Histogram Citra

Histogram citra merupakan diagram yang menggambarkan frekuensi setiap nilai intensitas yang muncul di seluruh piksel citra.

Nilai yang besar menyatakan bahwa piksel-piksel yang mempunyai intensitas tersebut sangat banyak



Manfaat Histogram

- Berguna untuk mengamati penyebaran intensitas warna dan dapat dipakai untuk pengambilan keputusan misalnya dalam peningkatan kecerahan atau peregangan kontras serta sebaran warna.
- Berguna untuk penentuan batas-batas dalam pemisahan objek dari latarbelakangnya.
- Memberikan persentase komposisi warna dan tekstur intensitas untuk kepentingan identifikasi citra.
- Histogram tidak mencerminkan susunan posisi warna piksel di dalam citra. Oleh karena itu, histogram tidak dapat dipakai untuk menebak bentuk objek yang terkandung di dalam citra.

C. Tingkat kecerahan, kontras dan peregangan

Operasi ini diperlukan dengan tujuan untuk membuat gambar menjadi lebih terang. Secara matematis, peningkatan kecerahan dilakukan dengan cara menambahkan suatu konstanta terhadap nilai seluruh piksel. Misalkan, $f(y, x)$ menyatakan nilai piksel pada citra berskala keabuan pada koordinat (y, x) . Maka, citra baru adalah:

$$g(y, x) = f(y, X) + \beta$$



Sebelum dicerahkan



Sesudah dicerahkan

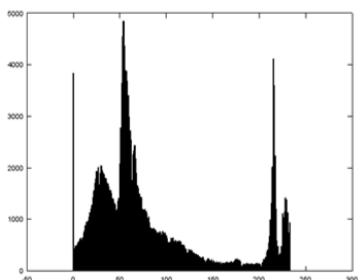
Jika dilihat melalui histogram, peningkatan kecerahan sebenarnya berefek pada penggeseran komposisi intensitas piksel ke kanan bila β berupa bilangan positif atau ke kiri jika β berupa bilangan negatif.



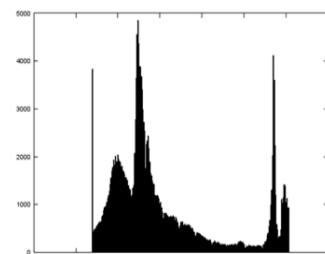
Citra "a" dengan Kecerahan Rendah



Citra "B" dengan kecerahan ditambah 20



Histogram dari gambar "a"



Histogram dari gambar "b"

1. Meregangkan kontras

Kontras dalam suatu citra menyatakan distribusi warna terang dan warna gelap. Suatu citra berskala keabuan dikatakan memiliki kontras rendah apabila distribusi warna cenderung pada jangkauan aras keabuan yang sempit. Sebaliknya, citra mempunyai kontras tinggi apabila jangkauan aras keabuan lebih terdistribusi secara melebar. Kontras dapat diukur berdasarkan perbedaan antara nilai intensitas tertinggi dan nilai intensitas terendah yang menyusun piksel-piksel dalam citra.

$$g(y, x) = \alpha f(y, X)$$

Operasi peningkatan kecerahan dan peregangan kontras dapat dilakukan sekaligus untuk kepentingan memperbaiki citra. Secara umum, gabungan kedua operasi tersebut dapat ditulis menjadi:

2. Kombinasi kecerahan dan kontras

Operasi peningkatan kecerahan dan peregangan kontras dapat dilakukan sekaligus untuk kepentingan memperbaiki citra. Secara umum, gabungan kedua operasi tersebut dapat ditulis menjadi:

$$g(y, x) = \alpha f(y, X) + \beta$$

$$g(y, x) = g_1 + ((g_2 - g_1)/(f_2 - f_1)) [f(y, X) - f_2]$$

3. Menampilkan Histogram Citra

Histogram dapat ditampilkan dengan menggunakan banyak cara. Fungsi yang dapat digunakan antara lain `cv2.calcHist()`, `np.histogram()` dan `plt.hist()`.

Adapun parameter untuk fungsi `cv2.calcHist()` adalah sebagai berikut:

```
cv.calcHist(images, channels, mask, histSize, ranges[,  
hist[, accumulate]])
```

images : gambar yang dibaca menggunakan fungsi `cv.read` dengan type `uint8` or `float32`. Gambar ini harus dinyatakan sebagai sebuah array dengan kurung siku "[img]".

channels : band yang akan dihitung histogramnya. Gunakan `[0]` untuk citra grayscale. Untuk citra berwarna, kita dapat berikan nilai `[0], [1]` atau `[2]` untuk menghitung histogram dari masing-masing band.

mask : digunakan apabila kita ingin memperoleh histogram dari sebagian citra saja.

- histSize** : Ukuran histogram, atau banyaknya kolom yang digambar. Kita dapat gunakan nilai sesuai dengan jumlah pixel [256].
- ranges** : rentang nilai. Pada citra 8 bit, nilainya adalah [0,256].

Ekstraksi Band

Sebuah citra dapat dipecah pada band penyusunnya dengan menggunakan perintah cv2.split().

```
# Band blue, green dan red masng-
masing disimpan pada variabel b,g,r
b, g, r = cv2.split(img)
```

atau bisa juga dengan menggunakan operasi index pada matriks img:

```
b = img[... ,0] # blue channel
g = img[... ,1] # green channel
r = img[... ,2] # red channel
```

Perhatikan bagaimana tiap band disusun pada variabel yang menyimpan gambar: masing-masing band disusun sebagai sebuah array di dalam array. Ingat bahwa pada Python, index sebuah array dimulai dari angka nol. Selanjutnya masing-masing band dapat ditampilkan dengan menggunakan cv2_imshow atau plt.imshow:

```
cv2_imshow(b); # menampilkan band biru
```

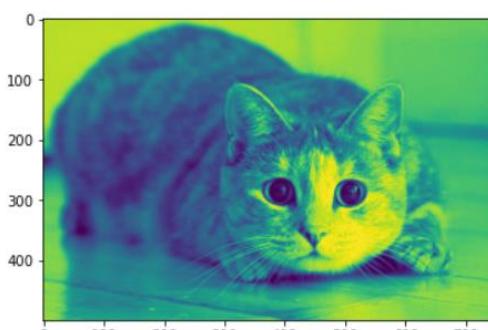
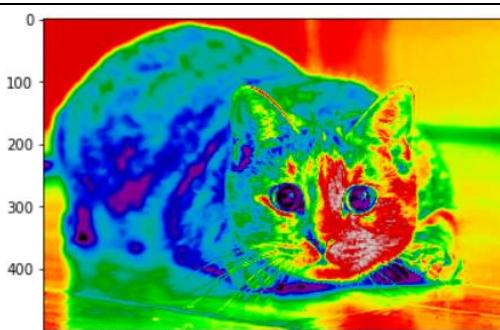
Latihan:

1. Mengubah ukuran Gambar contoh operasi piksel

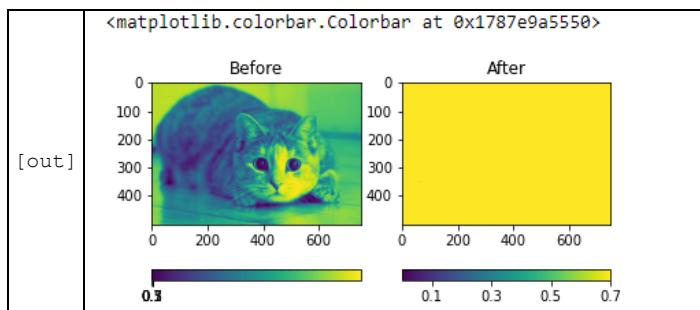
[in]	<pre>import matplotlib.pyplot as plt import matplotlib.image as mpimg %matplotlib inline from PIL import Image img = Image.open('kucing.jpg') img.thumbnail((64, 64), Image.ANTIALIAS) image in-place imgplot = plt.imshow(img)</pre>
[out]	

2. Mengimpor data gambar ke dalam array Numpy

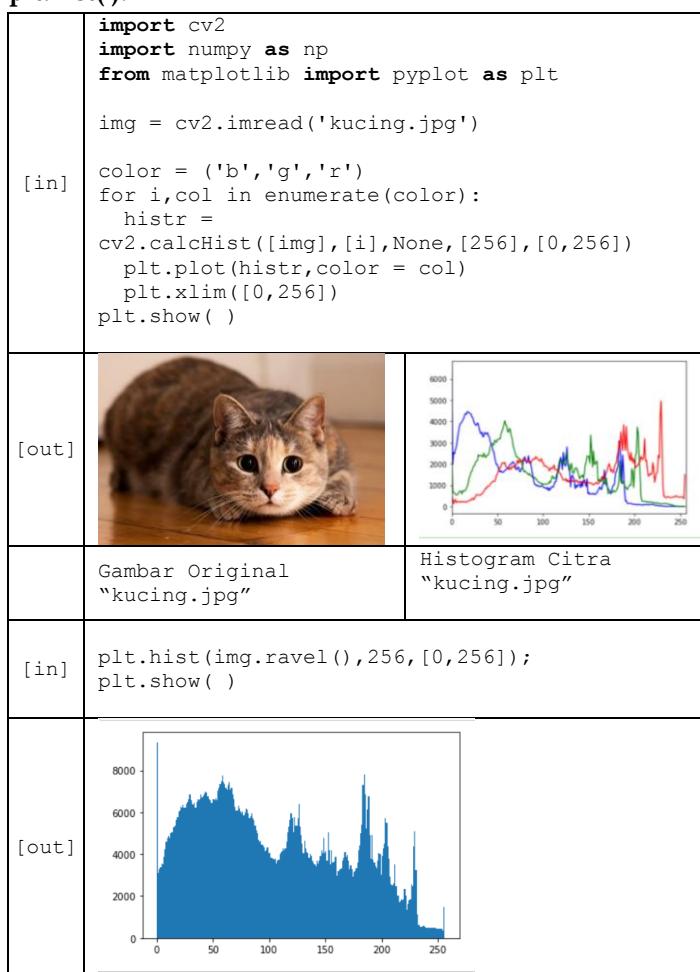
[in]	<pre>import matplotlib.pyplot as plt import matplotlib.image as mpimg %matplotlib inline from PIL import Image</pre>
[in]	<pre>img = mpimg.imread('kucing.jpg') imgplot = plt.imshow(img)</pre>
[out]	

[in]	<pre>lum_img = img[:, :, 0] plt.imshow(lum_img)</pre>
[out]	 <p><matplotlib.image.AxesImage at 0x1787c47b0d0></p>
[in]	<pre>plt.imshow(lum_img, cmap="hot")</pre>
[out]	 <p><matplotlib.image.AxesImage at 0x1787cbcac10></p>
[in]	<pre>imgplot = plt.imshow(lum_img) imgplot.set_cmap('nipy_spectral')</pre>
[out]	
[in]	<pre>plt.hist(lum_img.ravel(), bins=256, range=(0.0, 1.0), fc='k', ec='k')</pre>

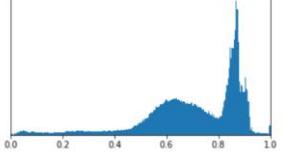
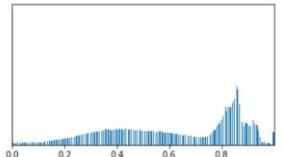
[in]	<pre>imgplot = plt.imshow(lum_img, clim=(0.0, 0.7))</pre>
[out]	
[in]	<pre>imgplot = plt.imshow(lum_img) plt.colorbar()</pre>
[out]	<pre><matplotlib.colorbar.Colorbar at 0x1787eca8820></pre>
[in]	<pre>fig = plt.figure() ax = fig.add_subplot(1, 2, 1) imgplot = plt.imshow(lum_img) ax.set_title('Before') plt.colorbar(ticks=[0.1, 0.3, 0.5, 0.7], orientation='horizontal') ax = fig.add_subplot(1, 2, 2) imgplot = plt.imshow(lum_img) imgplot.set_clim(0.0, 0.7) ax.set_title('After') plt.colorbar(ticks=[0.1, 0.3, 0.5, 0.7], orientation='horizontal')</pre>



3. Berikut adalah contoh kode untuk memanggil histogram **plt.hist()**:



4. Memanipulasi eksposur gambar

[in]	<pre>import numpy as np import matplotlib.pyplot as plt import skimage.exposure as skie %matplotlib inline</pre>
[in]	<pre>img = plt.imread('https://github.com/ipython- books/' + 'cookbook-2nd- data/blob/master/' + 'beach.png?raw=true')[..., 0]</pre>
[in]	<pre>def show(img): # Display the image. fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 3)) ax1.imshow(img, cmap=plt.cm.gray) ax1.set_axis_off() # Display the histogram. ax2.hist(img.ravel(), lw=0, bins=256) ax2.set_xlim(0, img.max()) ax2.set_yticks([]) plt.show()</pre>
[in]	<pre>show(img)</pre>
[out]	 
[in]	<pre>show(skie.rescale_intensity(img, in_range=(0.4, .95), out_range=(0, 1)))</pre>
[out]	 

Latihan Mandiri:

Modifikasi kode untuk memanggil histogram, pilihlah gambar sesuai gambar yang anda miliki dan Tampilkan histogram untuk ekstraksi *Band Merah, Hijau dan Biru...*

Soal:

Pilih salah satu jawaban yang benar dengan memberi tanda pada lembar jawaban.

1. Sebuah citra dapat dipecah pada band penyusunnya dengan menggunakan perintah ...
 - a. ax2.hist(img.ravel())
 - b. ax1.imshow()
 - c. cv2.calcHist()
 - d. cv2.split()
 - e. plt.imshow()
2. Secara matematis, peningkatan kecerahan berdasarkan dengan...
 - a. Menambahkan suatu konstanta terhadap nilai seluruh piksel
 - b. Meningkatkan kecerahan dan peregangan kontras
 - c. Penggeseran komposisi intensitas piksel
 - d. Nilai intensitas terendah
 - e. Mengubah komposisi warna dan tekstur
3. Jika dilihat melalui histogram, peningkatan kecerahan sebenarnya berefek pada...
 - a. Nilai intensitas terendah
 - b. Nilai kontras terendah
 - c. Meningkatkan kecerahan dan peregangan kontras
 - d. Menambahkan suatu konstanta terhadap nilai seluruh piksel
 - e. Penggeseran komposisi intensitas piksel
4. Operasi apa yang diperlukan untuk membuat gambar menjadi lebih terang ...
 - a. Operasi distribusi warna terang dan warna gelap
 - b. Operasi peningkatan kecerahan dan peregangan kontras
 - c. Operasi piksel warna citra
 - d. Operasi Ketetanggaan piksel
 - e. Operasi batas filter

5. Yang tidak termasuk dalam parameter fungsi **cv2.calcHist()** adalah ...
 - a. images
 - b. channels
 - c. ranges
 - d. array
 - e. histSize

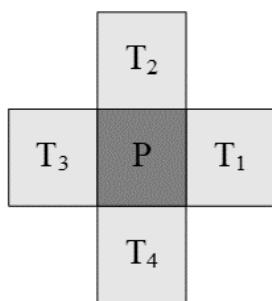
BAB 5

OPERASI KETETANGGAAN PIKSEL

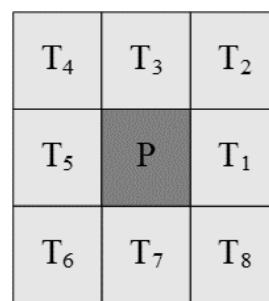
A. Pengertian Operasi Ketetanggaan

Dalam pengolahan citra digital untuk mendapatkan nilai suatu piksel diperlukan operasi ketetanggaan piksel. Setiap piksel pada citra tidak berdiri sendiri, satu dengan yang lainnya terkait dengan piksel tetangga, karena nilai suatu piksel merupakan bagian suatu objek tertentu di dalam citra. Pada pengolahan citra, ketetanggaan piksel banyak dipakai terutama pada analisis bentuk objek.

Ketetanggaan piksel yang umum dipakai adalah 4-ketetanggaan dan 8-ketetanggaan.



4 ketetanggaan



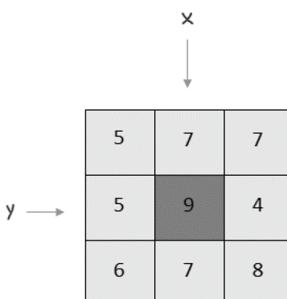
8 Ketetanggaan

Jika $P(b, k)$, maka:

- Pada 4-ketetanggaan
 $T_{-1} = (b, k + 1), T_2 = (b - 1, k),$
 $T_3 = (b, k - 1), T_4 = (b + 1, k)$
- Pada 8-ketetanggaan
 $T_{-1} = (b, k + 1), T_2 = (b - 1, k - 1)$
 $T_3 = (b, k - 1), T_4 = (b - 1, k - 1)$
 $T_5 = (b, k - 1), T_6 = (b + 1, k - 1)$
 $T_7 = (b + 1, k - 1), T_8 = (b + 1, k + 1)$

B. Filter Batas (Filter Max-Min)

Tujuan filter batas adalah untuk mencegah piksel yang intensitasnya di luar intensitas piksel-piksel tetangga. Untuk mendapatkan nilai piksel dan ketetanggannya perlu untuk menentukan nilai terendah sebagai batas bawah dan nilai tertinggi sebagai batas atas. Bila nilai piksel yang sedang dihitung (*iterasi*) lebih kecil dari batas bawah, maka diberi nilai sesuai dengan nilai batas bawah, begitu pula sebaliknya dengan batas atas.



Berdasarkan keadaan tersebut,

- $\text{minInt} = \text{minimum}(5,7,7,5,4,6,7,8) = 4;$
- $\text{maksInt} = \text{maksimum}(5,7,7,5,4,6,7,8) = 8;$
- mengingat $f(y,x)$ bernilai 9 dan lebih besar daripada 8 (maksInt) maka $g(y,x)$ bernilai 8;

C. Filter Rerata (Mean)

Filter pererataan menjumlahkan nilai piksel dengan ketetanggannya lalu dibagi untuk mendapatkan nilai rata-ratanya. dilakukan dengan menggunakan rumus:

$$g(x,y) = 1/9 \sum_{p=1}^9 f(y+p, x+q)$$

Filter pererataan dilakukan dengan menggunakan rumus:

65	50	55
78	68	60
60	60	62

$g(y, x) = \frac{1}{9} \times (65+50+55+76+68+60+60+60+62) = 61,7778$
 $\cong 62$

g(y,x)

D. Filter Median

Filter median mengurutkan nilai piksel dan ketetanggaannya dari yang terkecil kemudian didapatkan nilai tengahnya.

$$g(y, x) = \text{median}(f(y-1, x-1), f(y-1, x), f(y-1, x+1), f(y, x-1), f(y, x), f(y, x+1), f(y+1, x-1), f(y+1, x), f(y+1, x+1))$$



E. Filter lolos-bawah (low-pass filter)

Filter lolos-bawah (*low-pass filter*) adalah filter yang mempunyai sifat dapat meloloskan yang berfrekuensi rendah dan menghilangkan yang berfrekuensi tinggi. Efek filter ini membuat perubahan aras keabuan menjadi lebih lembut. Filter ini berguna untuk menghaluskan derau atau untuk kepentingan interpolasi tepi objek dalam citra.

F. Filter Lolos Rendah

Efek pengaburan citra dapat ditingkatkan dengan menaikkan ukuran kernel. Rahasia kernel yang digunakan untuk keperluan mengaburkan citra seperti berikut.

- Tinggi dan lebar kernel ganjil.
- Bobot dalam kernel bersifat simetris terhadap piksel pusat.

- Semua bobot bernilai positif.
- Jumlah keseluruhan bobot sebesar satu.

	<table border="1"> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>2</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> </table>	0	1	0	1	2	1	0	1	0		<table border="1"> <tr><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	1	1	1	1	1	1	1	1	1
0	1	0																			
1	2	1																			
0	1	0																			
1	1	1																			
1	1	1																			
1	1	1																			
1/6	#1	1/9	#2																		
	<table border="1"> <tr><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>2</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	1	1	1	1	2	1	1	1	1		<table border="1"> <tr><td>1</td><td>2</td><td>1</td></tr> <tr><td>2</td><td>4</td><td>2</td></tr> <tr><td>1</td><td>2</td><td>1</td></tr> </table>	1	2	1	2	4	2	1	2	1
1	1	1																			
1	2	1																			
1	1	1																			
1	2	1																			
2	4	2																			
1	2	1																			
1/10	#3	1/16	#4																		

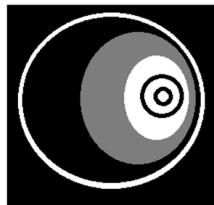
G. Filter lolos-tinggi

Filter lolos-tinggi adalah filter yang ditujukan untuk melewatkkan frekuensi tinggi dan menghalangi yang berfrekuensi rendah. Hal ini biasa dipakai untuk mendapatkan tepi objek dalam citra atau menajamkan citra.

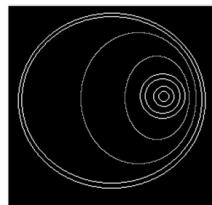
<table border="1"> <tr><td>0</td><td>-1</td><td>0</td></tr> <tr><td>-1</td><td>4</td><td>-1</td></tr> <tr><td>0</td><td>-1</td><td>0</td></tr> </table>	0	-1	0	-1	4	-1	0	-1	0	<table border="1"> <tr><td>-1</td><td>-1</td><td>-1</td></tr> <tr><td>-1</td><td>8</td><td>-1</td></tr> <tr><td>-1</td><td>-1</td><td>-1</td></tr> </table>	-1	-1	-1	-1	8	-1	-1	-1	-1	<table border="1"> <tr><td>1</td><td>-2</td><td>1</td></tr> <tr><td>-2</td><td>4</td><td>-2</td></tr> <tr><td>1</td><td>-2</td><td>1</td></tr> </table>	1	-2	1	-2	4	-2	1	-2	1
0	-1	0																											
-1	4	-1																											
0	-1	0																											
-1	-1	-1																											
-1	8	-1																											
-1	-1	-1																											
1	-2	1																											
-2	4	-2																											
1	-2	1																											
#1	#2	#3																											

H. Sifat Filter Lolos-tinggi

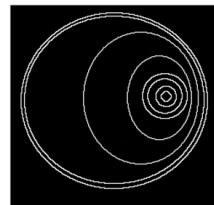
Apabila dikenakan pada area dengan perubahan aras keabuan yang lambat (frekuensi rendah), hasil berupa nol atau nilai yang sangat kecil. Apabila dikenakan pada area yang perubahan aras keabuannya cepat (frekuensi tinggi), hasil konvolusi bernali sangat besar.



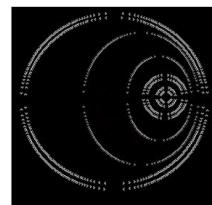
Citra bulat.png



Hasil dengan kernel #1



Hasil dengan kernel #2



Hasil dengan kernel #3

I. Filter Lolos Tinggi

Filter Lolos tinggi digunakan untuk keperluan mendeteksi tepi. Berikut ini adalah keperluan untuk filter lolos tinggi dalam mendeteksi tepi:

- Tinggi dan lebar kernel ganjil.
- Bobot dalam kernel bersifat simetris terhadap piksel pusat.
- Bobot pusat kernel bernilai positif.
- Bobot tetangga pusat kernel bernilai negatif (dapat menggunakan 4-ketetanggan atau 8 ketetanggan).
- Jumlah keseluruhan bobot sebesar satu.

J. Filter “high boost”

Filter “high boost” dapat digunakan untuk menajamkan citra melalui konvolusi. Kernel yang dapat dipakai adalah kernel filter lolos-tinggi dengan nilai di pusat diisi dengan nilai yang lebih besar daripada nilai pada posisi tersebut untuk filter lolos-tinggi.

Contoh filter high-boost:

-1	-1	-1
-1	c	-1
-1	-1	-1

$c > 8$; misalnya 9

Berikut ini kernel yang digunakan untuk keperluan menajamkan citra adalah:

- Tinggi dan lebar kernel gasal.
- Bobot dalam kernel bersifat simetris terhadap piksel pusat.
- Bobot pusat kernel bernilai positif.
- Bobot di sekeliling pusat kernel bernilai negatif (dapat menggunakan 4-ketetanggaan atau 8 ketetanggaan).
- Jumlah keseluruhan bobot lebih besar satu.
- Bobot terbesar terletak di pusat kernel

K. Effek emboss

Kernel yang digunakan:

-2	0	0
0	0	0
0	0	2



Berdasarkan citra
boneka2



Berdasarkan citra
lena256

Nilai negatif dan positif yang berpasangan menentukan perubahan kecerahan yang berefek pada penggambaran garis gelap atau terang

Nilai negatif menentukan arah emboss

-4	-4	0
-4	1	4
0	4	4

-6	0	6
-6	1	6
-6	0	6

-1	0	0
0	0	0
0	0	1



Embossing
dari arah
kiri keatas

Embossing
dari arah
kiri

Kernel #1

Hasil untuk
kernel#1

4	4	0
4	1	-4
0	-4	-4

6	0	-6
6	1	-6
6	0	-6

1	0	0
0	0	0
0	0	-1



Embossing
dari arah
kanan
bawah

Embossing
dari arah
kanan

Kernel #2

Hasil untuk
kernel#2

Pembuatan emboss terletak pada kernel konvolusi dengan sifat-sifat berikut:

- Tinggi dan lebar kernel gasal.
- Bobot dalam kernel bersifat tidak simetris terhadap piksel pusat.
- Bobot pusat kernel bernilai nol.
- Jumlah keseluruhan bobot bernilai nol.

L. Filter Linier vs Filter Non-Linier

Filter linier menerapkan penapisan (*filtering*) secara linier pada keseluruhan citra.

Contoh filter linier:

- Filter rerata
- Filter Gaussian
- Filter Laplacian (Topi Mexico)

Kelemahan filter linear, terutama ketika dipakai untuk konvolusi citra atau penghilangan derau, yaitu membuat struktur citra yang meliputi titik, tepi, dan garis ikut terkaburkan dan kualitas citra keseluruhan menurun (Burger dan Burge, 2008)

Filter non-linier menerapkan penapisan (filtering) dengan fungsi non-linier.

Contoh filter linier:

- Filter batas
- Filter rerata
- Filter median

Filter Gaussian tergolong sebagai filter lolos-rendah yang didasarkan pada fungsi Gaussian.

$$G(y,x) = e^{-\frac{x^2 + y^2}{2\sigma^2}}$$

Dalam hal ini, σ adalah deviasi standar dan piksel pada pusat (y, x) mendapatkan bobot terbesar berupa 1

Latihan:

1. Gaussian filter dari Scipy

[in]	<pre>from scipy import misc,ndimage from matplotlib import pyplot as plt face = misc.face() blurred_face = ndimage.gaussian_filter(face, sigma=3) fig, ax = plt.subplots(1, 2, figsize=(16, 8)) ax[0].imshow(face) ax[0].set_title("Original Image") ax[0].set_xticks([]) ax[0].set_yticks([]) ax[1].imshow(blurred_face) ax[1].set_title("Blurred Image") ax[1].set_xticks([]) ax[1].set_yticks([])</pre>
[out]	 

2. Linier Filtering

[in]	<pre>import cv2 as cv import numpy as np import matplotlib.pyplot as plt image = cv.imread("kucing.jpg") fig, ax = plt.subplots(1, 3, figsize=(16, 8)) fig.tight_layout() # To convolve the kernel on an image we can # use cv.filter2D ax[0].imshow(cv.cvtColor(image, cv.COLOR_BGR2RGB)) ax[0].set_title('Original Image') kernel_sharpening = np.array([[-1, -1, -1], [-1, 9, -1], [-1, -1, -1]]) kernel_sharpening_2 = np.array([[-1, -1, -1], [-1, 10, -1], [-1, -1, -1]])</pre>
------	--

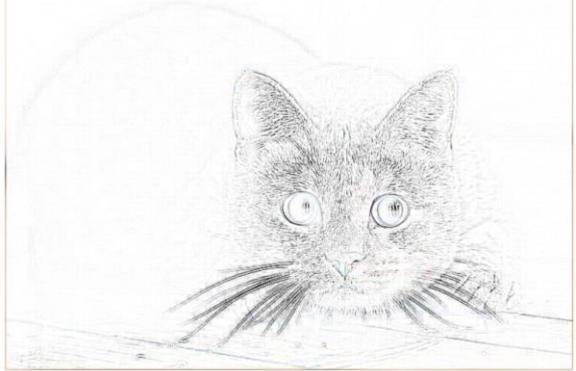
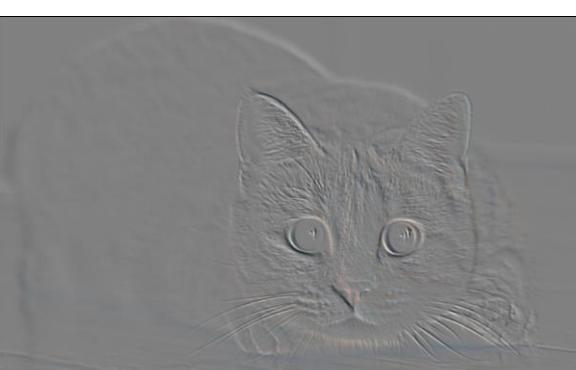
	<pre> sharpened = cv.filter2D(image, -1, kernel_sharpening) ax[1].imshow(cv.cvtColor(sharpened, cv.COLOR_BGR2RGB)) ax[1].set_title('Sharpened Kernel Image') sharpened_2 = cv.filter2D(image, -1, kernel_sharpening_2) ax[2].imshow(cv.cvtColor(sharpened_2, cv.COLOR_BGR2RGB)) ax[2].set_title('Sharpened Kernel Image 2') plt.show() </pre>
[out]	

3. Manipulasi Gambar Menggunakan PIL/Pillow

[in]	<pre> from PIL import Image im = Image.open("kucing.jpg") </pre>
[in]	<pre> im.getbands() </pre>
[out]	<pre> ('R', 'G', 'B') </pre>
[in]	<pre> im.size </pre>
[out]	<pre> (400, 600) </pre>
[in]	<pre> import base64 with open("kucing.jpg", "rb") as image: image_string = base64.b64encode(image.read()) </pre>
[in]	<pre> import io image = io.BytesIO(base64.b64decode(image_string)) Image.open(image) </pre>
[out]	

[in]	<pre>im = Image.open('kucing.jpg') box = (100, 150, 300, 300) cropped_image = im.crop(box) cropped_image</pre>
[out]	
[in]	<pre>rotated = im.rotate(180) rotated</pre>
[out]	
[in]	<pre>logo = Image.open('cat.png')</pre>
[in]	<pre>position = (38, 469) im.paste(logo, position) im.save('merged.jpg')</pre>
[in]	<pre>im = Image.open("kucing.jpg") image_copy = im.copy() position = ((image_copy.width - logo.width), (image_copy.height - logo.height)) image_copy.paste(logo, position, logo) image_copy</pre>

[out]	
[in]	<pre>from PIL import ImageEnhance enhancer = ImageEnhance.Sharpness(im) enhancer.enhance(10.0)</pre>
[out]	
[in]	<pre>enhancer = ImageEnhance.Contrast(im) enhancer.enhance(2)</pre>
[out]	

[in]	<pre>from PIL import ImageFilter im.filter(ImageFilter.CONTOUR)</pre>
[out]	
[in]	<pre>im.filter(ImageFilter.EDGE_ENHANCE)</pre>
[out]	
[in]	<pre>im.filter(ImageFilter.EMBOSS)</pre>
[out]	

[in]	<pre>im.filter(ImageFilter.FIND_EDGES)</pre>
[out]	
[in]	<pre>from PIL import ImageSequence</pre>
[in]	<pre>im = Image.open("kucing.jpg") frame_num = 1 for frame in ImageSequence.Iterator(im): frame.save("frame%d.png" % frame_num) frame_num = frame_num + 1 if frame_num == 3: break</pre>
[out]	

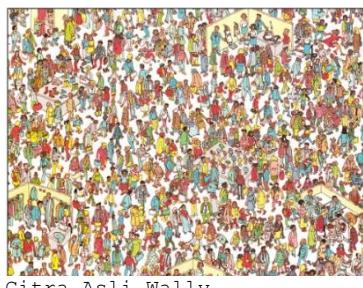
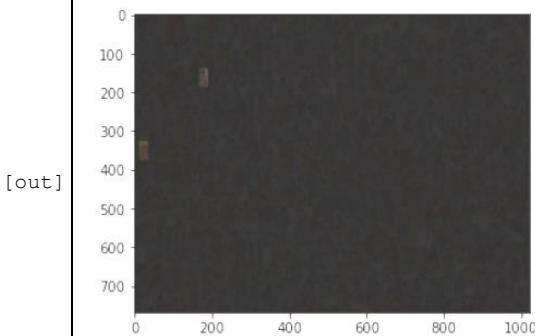
4. Cuplikan kode berikut membantu menemukan Wally di tengah keramaian.

```
[in] from pylab import imshow, show
import mahotas
import mahotas.demos
import numpy as np

wally = mahotas.demos.load('Wally')
wfloat = wally.astype(float)

r,g,b = wfloat.transpose((2,0,1))
w = wfloat.mean(2)
pattern = np.ones((24,16), float)

for i in range(2):
    pattern[i::4] = -1
    v = mahotas.convolve(r-w, pattern)
    mask = (v == v.max())
    mask = mahotas.dilate(mask,
        np.ones((48,24)))
    np.subtract(wally, .8*wally *
~mask[:,None], out=wally, casting='unsafe')
imshow(wally)
show()
```



Latihan Mandiri:

1. Carilah sembarang gambar (boleh dari internet atau memotret sendiri), kemudian dari gambar tersebut tampilkan dengan Python-OpenCV:
 - Gambar asli dengan menggunakan cv2_imshow()
 - Gambar asli dengan menggunakan Matplotlib
2. Pada gambar no 1, lakukan konversi BGR ke HSV kemudian tampilkan sebagai berikut (boleh menggunakan Matplotlib atau cv2.imshow())

Soal:

Pilih salah satu jawaban yang benar dengan memberi tanda pada lembar jawaban.

1. Apa tujuan dari filter batas sebuah citra ?
 - a. Untuk mendapatkan nilai piksel
 - b. Menjumlahkan nilai piksel
 - c. Mencegah piksel yang intensitasnya di luar intensitas piksel-piksel tetangga
 - d. Mengurutkan nilai piksel
 - e. Menghilangkan filter yang berfrekuensi tinggi
2. Berikut ini kode manakah yang digunakan untuk memberikan efek emboss....
 - a. ndimage.gaussian_filter
 - b. im.filter(ImageFilter.CONTOUR)
 - c. cv.filter2D
 - d. im.filter(ImageFilter.EMBOSS)
 - e. im.filter(ImageFilter.EDGE_ENHANCE)
3. Untuk menggunakan Efek pengaburan citra dapat digunakan pada perintah kode:
 - a. im.filter(ImageFilter.EDGE_ENHANCE)
 - b. ndimage.gaussian_filter
 - c. ImageSequence.Iterator()
 - d. ImageEnhance.Sharpness()
 - e. im.filter(ImageFilter.EMBOSS)
4. Efek Filter lolos tinggi dapat digunakan dengan kode
 - a. ndimage.gaussian_filter
 - b. im.filter(ImageFilter.CONTOUR)
 - c. ImageSequence.Iterator()
 - d. im.filter(ImageFilter.EMBOSS)
 - e. im.filter(ImageFilter.EDGE_ENHANCE)

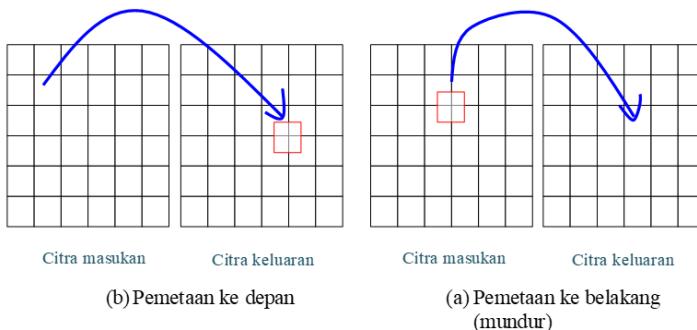
5. Bagaimana fungsi Filter median dalam memproses citra?
 - a. Untuk mendapatkan nilai piksel
 - b. Menjumlahkan nilai piksel
 - c. Mencegah piksel yang intensitasnya di luar intensitas piksel-piksel tetangga
 - d. Mengurutkan nilai piksel dari yang terkecil kemudian didaptkan nilai tengahnya
 - e. Menghilangkan filter yang berfrekuensi tinggi

BAB 6

OPERASI GEOMETRIK PADA CITRA

A. Operasi Geometrik pada Citra

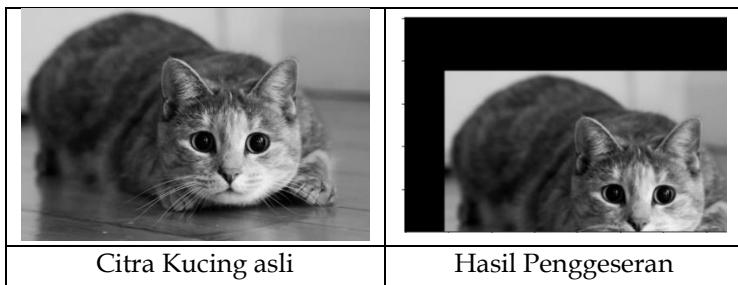
Pada pengolahan citra digital, secara umum operasi Geometrik ditujukan untuk memodifikasi koordinat pixel dalam suatu citra dengan pendekatan tertentu, namun dalam perkembangannya memodifikasi nilai skala keabuan sangat dimungkinkan dalam operasi geometrik. Operasi Geometri berhubungan dengan perubahan bentuk geometri citra, yaitu baik ukuran ataupun orientasinya. Operasi geometri diantaranya meliputi pencerminan flipping, rotasi/pemutaran Rotating, pemotongan Cropping, dan penskalaan Scaling/Zooming.



B. Penggeseran citra

Penggeseran citra ke arah mendatar atau vertikal dapat dilakukan dengan persamaan:

$$\begin{aligned}x_{baru} &= x_{lama} + S_x \\y_{baru} &= y_{lama} + S_y\end{aligned}$$



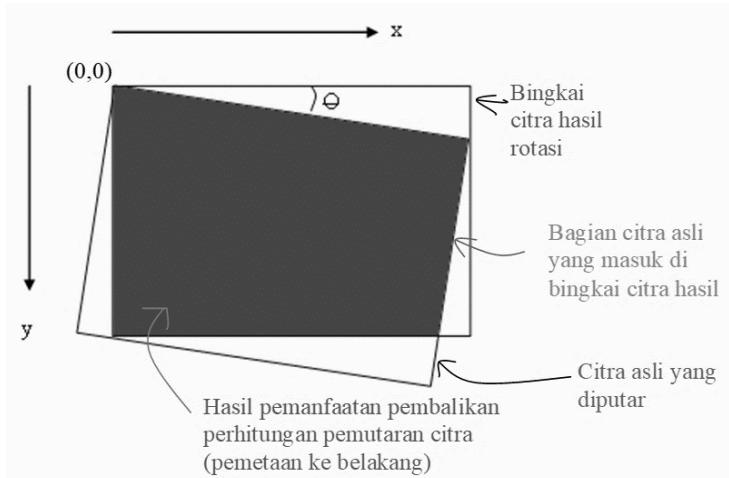
C. Pemutaran citra

Suatu citra dapat diputar dengan sudut θ seiring arah jarum jam atau berlawanan arah jarum jam dengan pusat putaran pada koordinat (0,0)

Operasi memutar citra dapat dilakukan dengan fungsi cosinus-sinus, sebagai berikut:

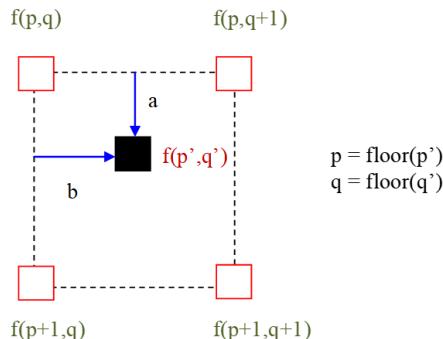
$$x_{baru} = x * \cos(\theta) + y * \sin(\theta)$$

$$y_{baru} = y * \cos(\theta) - x * \sin(\theta)$$



D. Interpolasi piksel

Hasil pemutaran citra dapat menimbulkan efek bergerigi pada objek citra yang disebabkan oleh operasi pembulatan (round atau floor).



$$f(p', q') = (1 - a)[(1 - b)f(p, q) + b f(p, q + 1)] + \\ a[(1 - b)f(p + 1, q) + b f(p + 1, q + 1)]$$

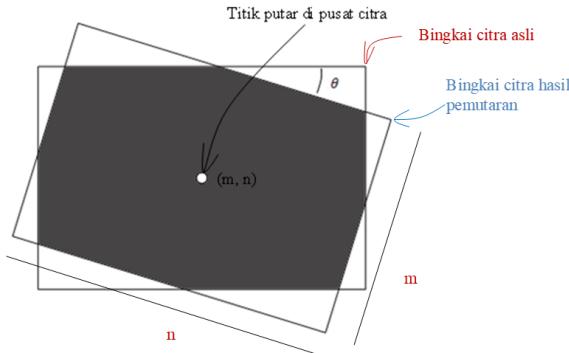
Dalam hal ini, a dan b dihitung melalui:

$$a = p' - p$$

$$b = q' - q$$

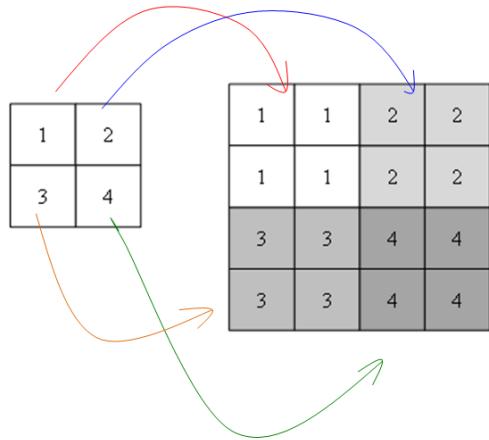
E. Pemutaran citra berdasarkan sebarang koordinat

Rotasi tidak harus berpusat pada titik (0,0), dapat dilakukan pada titik (m, n)



$$x_{baru} = (x - n) * \cos(\theta) + (y - m) * \sin(\theta) + n$$

$$y_{baru} = (y - m) * \cos(\theta) - (x - n) * \sin(\theta) + m$$



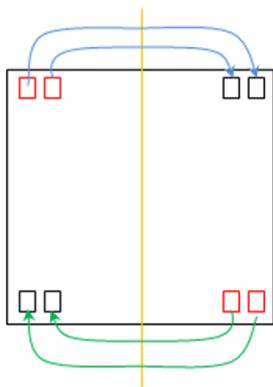
```
tinggi_baru = tinggi * sy;
```

```
lebar_baru = lebar * sx;
```

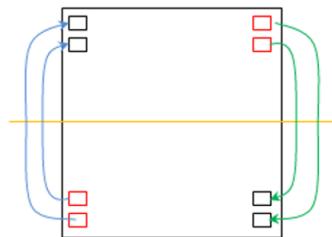
```
y2 = ((y-1) / sy) + 1;
```

```
x2 = ((x-1) / sx) + 1;
```

Pencerminan Horisontal



Pencerminan Vertikal



F. Transformasi affine

Transformasi affine adalah transformasi linear yang menyertakan penskalaan, pemutaran, penggeseran, dan shearing (pembengkokan).

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Persamaan di atas dapat ditulis pula menjadi seperti berikut:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

G. Efek Ripple (Bergelombang)

Efek ripple (riak) adalah aplikasi transformasi citra yang membuat gambar terlihat bergelombang. Efek riak dapat dibuat baik pada arah x maupun y.

$$x' = x + a_x \sin \frac{2\pi y}{T_x}$$

$$y' = y + a_y \sin \frac{2\pi x}{T_y}$$

H. Transformasi twirl

Transformasi twirl (olak atau puntiran) dilakukan dengan memutar citra berdasarkan titik pusat citra, tetapi tidak bersifat linear. Salah satu varian bentuk transformasinya, yang diadaptasi dari Burger & Burge (2008), sebagai berikut:

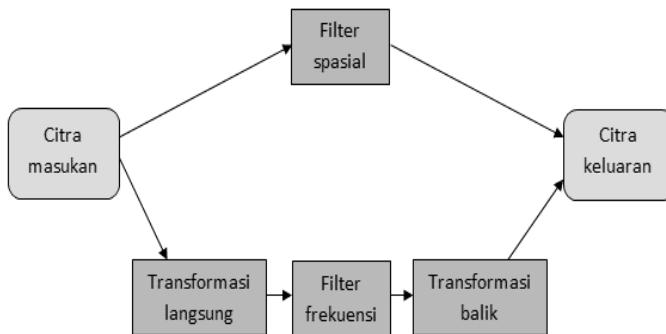
$$x' = x_c + r \cos(\beta) \quad r = \sqrt{(x - x_c)^2 + (y - y_c)^2}$$

$$y' = y_c + r \sin(\beta) \quad \beta = \text{ArcTan}(d_y, d_x) + \alpha(r_{maks} - r)/r_{maks}$$

I. Pengolahan Citra di Kawasan Frekuensi

1. Transformasi Fourier

Transformasi Citra: Spasial vs Frekuensi



Spasial = berkaitan dengan posisi pixel

Frekuensi = berkaitan dengan kuantitas pixel

Transformasi Fourier dimanfaatkan untuk memetakan citra dari kawasan spasial ke dalam kawasan frekuensi. Citra dapat diamati sebagai kumpulan gelombang sinusoid dengan frekuensi, amplitudo, dan fase yang berbeda-beda.

2. Fourier 1-D (Discrete Fourier Transform)

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) \left(\cos \left[\frac{2\pi u x}{N} \right] - j \sin \left[\frac{2\pi u x}{N} \right] \right)$$

atau

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) \exp \left[-j \frac{2\pi u x}{N} \right], \text{ dengan } u = 0, 1, 2, \dots, N-1$$

Invers Discrete Fourier Transform

$$f(x) = \sum_{u=0}^{N-1} F(u) \exp \left[j \frac{2\pi u x}{N} \right], \text{ dengan } u = 0, 1, 2, \dots, N-1$$

Contoh Perhitungan:

$f(x)$	$F(u)$	
Real	Real	Imajiner
2	3	0
4	0,25	0,25
1	-1,50	0
5	0,25	-0,25

DFT → DFT⁻¹

Fourier 2-D

$$F(u, v) = \sum_{y=0}^{M-1} \sum_{x=0}^{N-1} f(x, y) \left(\cos\left(2\pi \left(\frac{ux}{N} + \frac{vy}{M}\right)\right) - j \sin\left(2\pi \left(\frac{ux}{N} + \frac{vy}{M}\right)\right) \right)$$

Invers 2-D Fourier Transform

$$f(y, x) = \frac{1}{MN} \sum_{y=0}^{M-1} \sum_{x=0}^{N-1} F(v, u) \left(\cos\left(2\pi \left(\frac{ux}{N} + \frac{vy}{M}\right)\right) + j \sin\left(2\pi \left(\frac{ux}{N} + \frac{vy}{M}\right)\right) \right)$$

3. Fast Fourier Transform (FFT)

Metode Fast Fourier Transform (FFT) dibuat untuk mempercepat komputasi transformasi Fourier. Jika kompleksitas DFT untuk mentransformasikan sebuah piksel seperti yang tertuang dalam implementasi di depan sebesar $O(N^2)$, FFT memiliki kompleksitas sebesar $O(N \log_2 N)$.

Sebagai pembanding, jika N sama dengan 256 maka $N^2 = 65.536$, sedangkan $N \log_2 N = 256 \times 8 = 2048$.

Jadi, FFT lebih cepat 32 kali dibandingkan DFT.

J. Filter Lolos-Rendah (Low-pass Filter)

Filter lolos-bawah (low-pass filter) adalah filter yang mempunyai sifat dapat meloloskan yang berfrekuensi rendah dan menghilangkan yang berfrekuensi tinggi.

Efek filter ini membuat perubahan level keabuan menjadi lebih lembut. Filter ini berguna untuk menghaluskan derau atau untuk kepentingan interpolasi tepi objek dalam citra.

Latihan:

1. Transformasi Twirl

```
import matplotlib.pyplot as plt

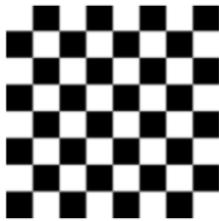
from skimage import data
from skimage.transform import swirl

image = data.checkerboard()
swirled = swirl(image, rotation=0, strength=10,
radius=120)

fig, (ax0, ax1) = plt.subplots(nrows=1,
ncols=2, figsize=(8, 3),
sharex=True,
sharey=True)

ax0.imshow(image, cmap=plt.cm.gray)
ax0.axis('off')
ax1.imshow(swirled, cmap=plt.cm.gray)
ax1.axis('off')

plt.show()
```



2. Translation

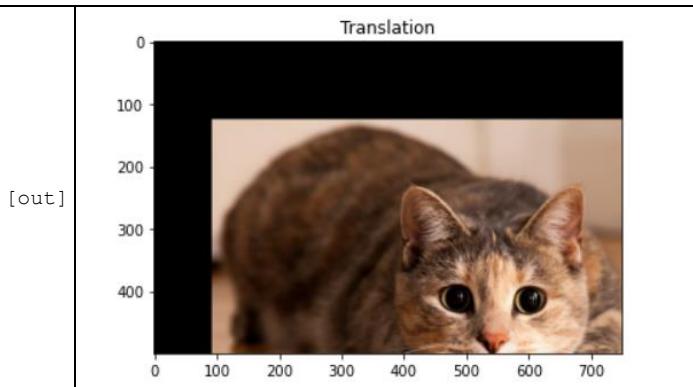
```
[in] import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

image = cv.imread("kucing.jpg")
h, w = image.shape[:2]

half_height, half_width = h//4, w//8
transition_matrix = np.float32([[1, 0,
half_width],
[0, 1,
half_height]])

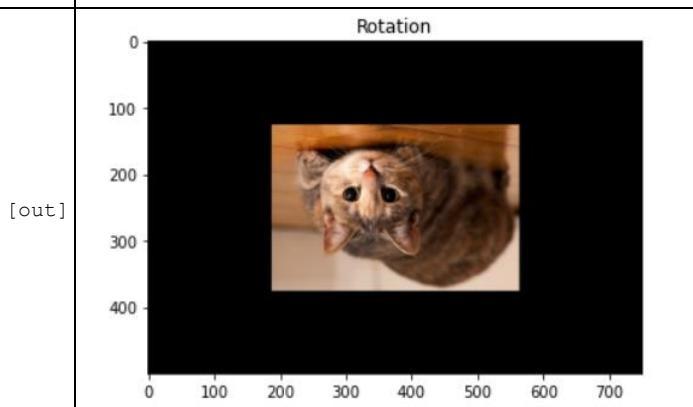
img_transition = cv.warpAffine(image,
transition_matrix, (w, h))

plt.imshow(cv.cvtColor(img_transition,
cv.COLOR_BGR2RGB))
plt.title("Translation")
plt.show()
```



3. Rotation

[in]	<pre> import cv2 as cv import numpy as np import matplotlib.pyplot as plt image = cv.imread("kucing.jpg") h, w = image.shape[:2] rotation_matrix = cv.getRotationMatrix2D((w/2,h/2), -180, 0.5) rotated_image = cv.warpAffine(image, rotation_matrix, (w, h)) plt.imshow(cv.cvtColor(rotated_image, cv.COLOR_BGR2RGB)) plt.title("Rotation") plt.show() </pre>
------	---



4. Interpolation

	<pre> import mahotas as mh import numpy as np from pylab import imshow, show regions = np.zeros((8,8), bool) [in] regions[3:,3:] = 1 regions[6:,6:] = 1 labeled, nr_objects = mh.label(regions) imshow(labeled, interpolation='nearest') show() </pre>
[out]	
[in]	<pre> labeled, nr_objects = mh.label(regions, np.ones((3,3), bool)) </pre>
[in]	<pre> sizes = mh.labeled.labeled_size(labeled) print('Background size', sizes[0]) print('Size of first region: {}'.format(sizes[1])) </pre>
[out]	<pre> Background size 51 Size of first region: 9 </pre>
[in]	<pre> array = np.random.random_sample(regions.shape) sums = mh.labeled.sum(array, labeled) print('Sum of first region: {}'.format(sums[1])) </pre>
[out]	<pre> Sum of first region: 5.760848502091579 </pre>

5. Skala Interpolasi Miring

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

image = cv.imread("kucing.jpg")

fig, ax = plt.subplots(1, 3, figsize=(16, 8))

# image size being 0.15 times of it's original
# size
image_scaled = cv.resize(image, None, fx=0.15,
fy=0.15)
ax[0].imshow(cv.cvtColor(image_scaled,
cv.COLOR_BGR2RGB))
ax[0].set_title("Linear Interpolation Scale")

[in]
# image size being 2 times of it's original
# size
image_scaled_2 = cv.resize(image, None, fx=2,
fy=2, interpolation=cv.INTER_CUBIC)
ax[1].imshow(cv.cvtColor(image_scaled_2,
cv.COLOR_BGR2RGB))
ax[1].set_title("Cubic Interpolation Scale")

# image size being 0.15 times of it's original
# size
image_scaled_3 = cv.resize(image, (200, 400),
interpolation=cv.INTER_AREA)
ax[2].imshow(cv.cvtColor(image_scaled_3,
cv.COLOR_BGR2RGB))
ax[2].set_title("Skewed Interpolation Scale")
```

[out]

```
Text(0.5, 1.0, 'Skewed Interpolation Scale')
```



Soal:

Pilih salah satu jawaban yang benar dengan memberi tanda pada lembar jawaban.

1. Suatu citra dapat diputar dengan sudut θ seiring arah jarum jam atau berlawanan arah jarum jam dengan pusat putaran pada koordinat (0,0).
Kode manakah yang sesuai untuk pernyataan diatas?
 - a. cv.warpAffine
 - b. cv.getRotationMatrix2D
 - c. cv.INTER_AREA
 - d. cv.cvtColor
 - e. cv.imread
2. Berikut ini kode manakah yang digunakan untuk pergeseran citra....
 - a. cv.warpAffine
 - b. cv.COLOR_BGR2RGB
 - c. cv.INTER_AREA
 - d. cv.cvtColor
 - e. cv.imread
3. Untuk menggunakan Interpolasi citra dapat digunakan pada perintah kode:
 - a. cv.warpAffine
 - b. cv.INTER_CUBIC
 - c. cv.INTER_AREA
 - d. cv.cvtColor
 - e. cv.imread
4. `np.float32([[1, 0, half_width],[0, 1, half_height]])`
Baris kode tersebut digunakan untuk perintah...
 - a. Transisi matriks
 - b. Transformasi affine
 - c. Pemutaran citra
 - d. Efek Ripple
 - e. Interpolasi piksel

5. Transformasi linear yang menyertakan penskalaan, pemutaran, penggeseran, dan pembengkokan, merupakan bagian dari...
 - a. Transisi matriks
 - b. Transformasi affine
 - c. Pemutaran citra
 - d. Teknik Pergeseran citra
 - e. Interpolasi piksel

BAB 7

OPERASI MORFOLOGI

A. Operasi Morfologi Untuk Pengolahan Citra

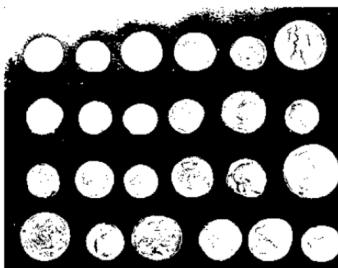
Operasi morfologi merupakan operasi yang umum dikenakan pada citra biner (hitam-putih) untuk mengubah struktur bentuk objek yang terkandung dalam citra. Morfologi adalah teknik pengolahan citra berdasarkan bentuk segmen citra. Tujuan dari operasi morfologi adalah untuk memperbaiki hasil segmentasi.

Contoh aplikasi/operasi morfologi:

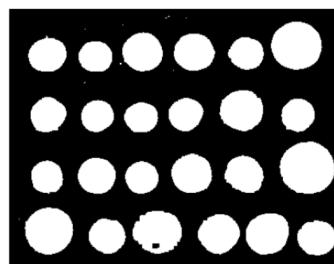
- Menutup lubang pada citra
- Memisahkan objek
- Membentuk filter spasial.
- Memperoleh skeleton (rangka) objek.
- Menentukan letak objek di dalam citra.
- Memperoleh bentuk struktur objek.



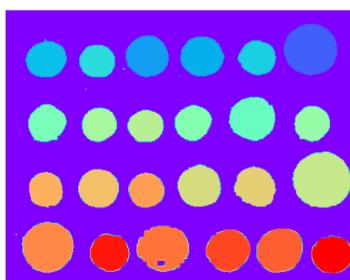
Gambar 7.1 Original koin



Gambar memisahkan objek



Gambar membentuk filter



Gambar pelabelan citra



Gambar Mengenali objek

Operasi morfologi umumnya melibatkan 2 (dua) masukan:

- Citra masukan yang akan dikenai operasi morfologi
- Kernel atau structuring element dengan sebuah titik pusat (hotspot)

Contoh kernel:

0	1	0
1	1	1
0	1	0

1	1	1
1	1	1
1	1	1

1	1
1	1

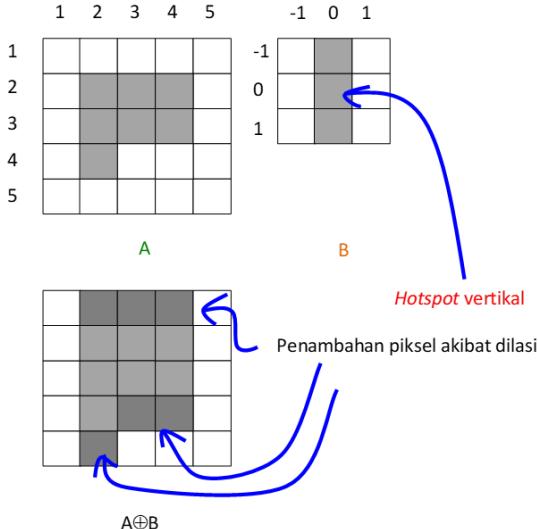
1	1	1	1	1
---	---	---	---	---

1	1	1	1
---	---	---	---

1
1
1
1

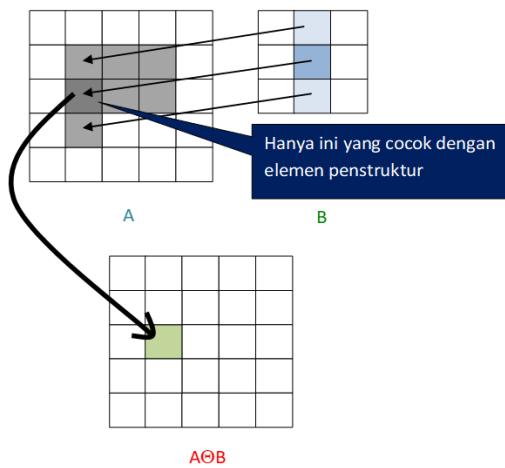
1. Operasi dilasi

Operasi dilasi biasa dipakai untuk mendapatkan efek pelebaran terhadap piksel yang bernilai 1.



2. Operasi Erosi

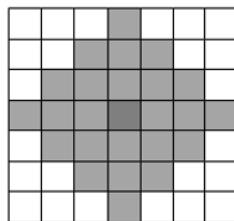
Operasi erosi mempunyai efek memperkecil struktur citra



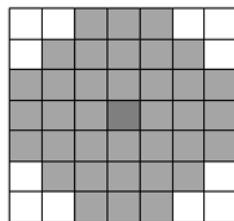
B. Elemen Penstruktur (Structure Element)

Ukuran dan bentuk elemen penstruktur (*structure element*) juga menentukan hasil operasi morfologi. Bentuk yang umum digunakan pada operasi morfologi adalah cakram dan lingkaran.

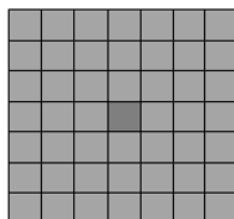
Elemen Penstruktur (Structure Element)



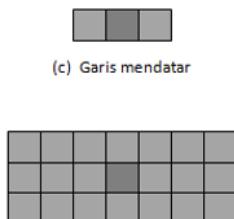
(a) Belah ketupat



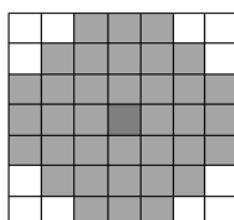
(b) Cakram



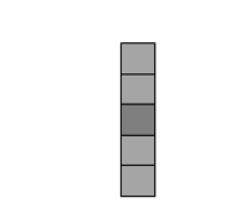
(h) Bujur sangkar



(c) Garis mendatar



(f) Oktagon



(e) Garis tegak

C. Operasi Opening dan Closing

Operasi opening adalah operasi erosi yang diikuti dengan dilasi dengan menggunakan elemen penstruktur yang sama. Operasi ini berguna untuk menghaluskan kontur objek dan menghilangkan seluruh piksel di area yang terlalu kecil untuk ditempati oleh elemen penstruktur.

Operasi closing berguna untuk menghaluskan kontur dan menghilangkan lubang-lubang kecil. Operasi closing dilaksanakan dengan melakukan operasi dilasi terlebih dahulu dan kemudian diikuti dengan operasi erosi.

Latihan:

- Contoh ini menunjukkan cara menggunakan fungsi `skimage.morphology` untuk menghasilkan elemen penataan. Judul setiap plot menunjukkan panggilan fungsi.

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

from skimage.morphology import (square,
                                rectangle, diamond, disk, cube, octahedron,
                                ball, octagon, star)

# Generate 2D and 3D structuring elements.
struc_2d = {
    "square(15)": square(15),
    "rectangle(15, 10)": rectangle(15, 10),
    "diamond(7)": diamond(7),
    "disk(7)": disk(7),
    "octagon(7, 4)": octagon(7, 4),
    "star(5)": star(5)
}

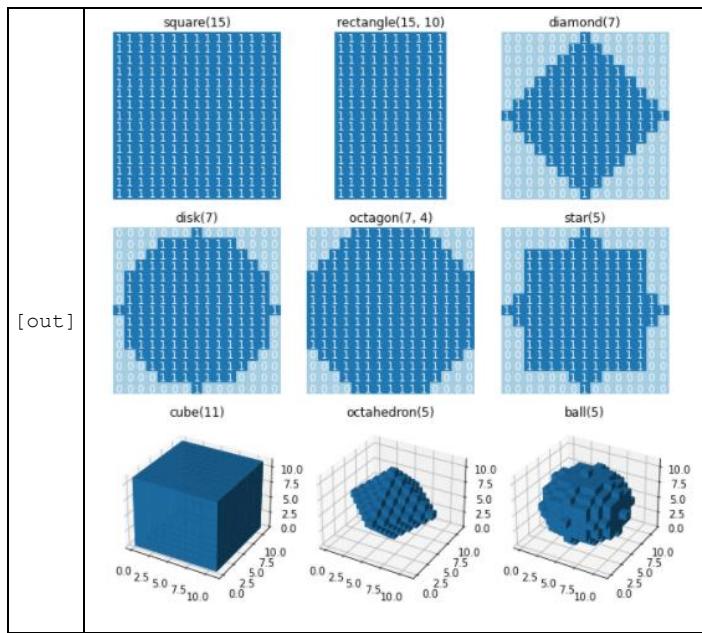
struc_3d = {
    "cube(11)": cube(11),
    "octahedron(5)": octahedron(5),
    "ball(5)": ball(5)
}

[in] # Visualize the elements.
fig = plt.figure(figsize=(8, 8))

idx = 1
for title, struc in struc_2d.items():
    ax = fig.add_subplot(3, 3, idx)
    ax.imshow(struc, cmap="Paired", vmin=0,
              vmax=12)
    for i in range(struc.shape[0]):
        for j in range(struc.shape[1]):
            ha="center", va="center", color="w")
            ax.text(j, i, struc[i, j],
                    ha="center", va="center", color="w")
    ax.set_axis_off()
    ax.set_title(title)
    idx += 1

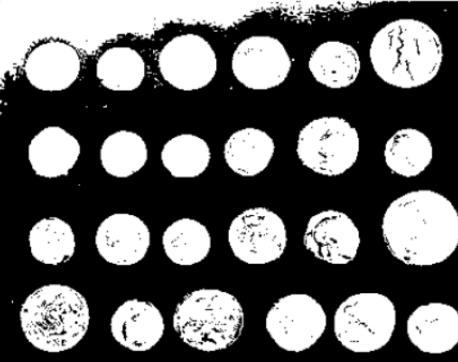
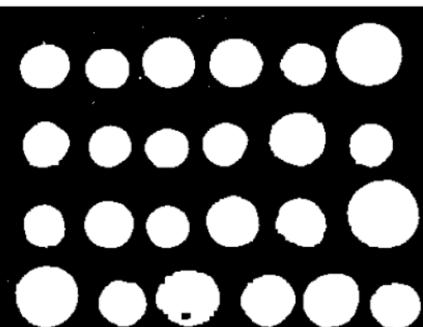
for title, struc in struc_3d.items():
    ax = fig.add_subplot(3, 3, idx,
                        projection=Axes3D.name)
    ax.voxels(struc)
    ax.set_title(title)
    idx += 1

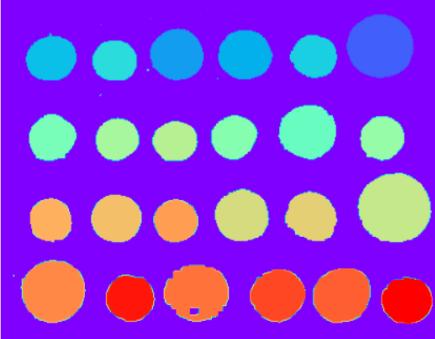
fig.tight_layout()
plt.show()
```

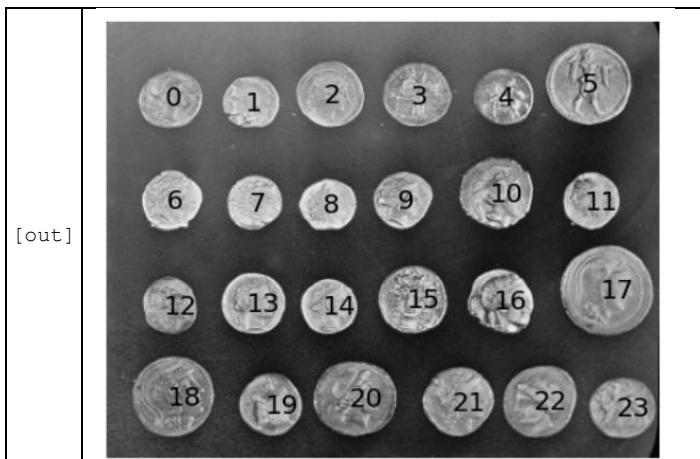


2. Segmentasi dan Morphology Gambar Koin

[in]	<pre>import numpy as np import matplotlib.pyplot as plt from skimage.data import coins from skimage.filters import threshold_otsu from skimage.segmentation import clear_border from skimage.morphology import label, closing, square from skimage.measure import regionprops from skimage.color import lab2rgb %matplotlib inline</pre>
[in]	<pre>def show(img, cmap=None): cmap = cmap or plt.cm.gray fig, ax = plt.subplots(1, 1, figsize=(8, 6)) ax.imshow(img, cmap=cmap) ax.set_axis_off() plt.show()</pre>
[in]	<pre>img = coins() show(img)</pre>

[out]	
[in]	<pre>threshold_otsu(img) 107 show(img > 107)</pre>
[out]	
[in]	<pre>img_bin = clear_border(closing(img > 120, square(5))) show(img_bin)</pre>
[out]	
[in]	<pre>labels = label(img_bin) show(labels, cmap=plt.cm.rainbow)</pre>

[out]	
[in]	regions = regionprops(labels)
[in]	boxes = np.array([label['BoundingBox'] for label in regions if label['Area'] > 100])
[in]	print(f"There are {len(boxes)} coins.")
[out]	There are 24 coins.
[in]	fig, ax = plt.subplots(1, 1, figsize=(8, 6)) ax.imshow(img, cmap=plt.cm.gray) ax.set axis off() # Get the coordinates of the boxes. xs = boxes[:, [1, 3]].mean(axis=1) ys = boxes[:, [0, 2]].mean(axis=1) # We reorder the boxes by increasing # column first, and row second. for row in range(4): # We select the coins in each of the four # rows. if row < 3: ind = ((ys[6 * row] <= ys) & (ys < ys[6 * row + 6])) else: ind = (ys[6 * row] <= ys) # We reorder by increasing x coordinate. ind = np.nonzero(ind)[0] reordered = ind[np.argsort(xs[ind])] xs_row = xs(reordered) ys_row = ys(reordered) # We display the coin number. for col in range(6): n = 6 * row + col ax.text(xs_row[col] - 5, ys_row[col] + 5, str(n), fontsize=20)



Latihan Mandiri:

1. Carilah sembarang gambar (boleh dari internet atau memotret sendiri), kemudian dari gambar tersebut tampilkan dengan **threshold_otsu**...
2. Berilah pelabelan angka pada setiap gambar yang telah di tampilkan dengan **threshold_otsu**...

Soal:

Pilih salah satu jawaban yang benar dengan memberi tanda pada lembar jawaban.

1. skimage.morphology biasanya sering digunakan untuk...
 - a. Menghasilkan pergeseran matriks
 - b. Membuat citra menjadi hitam-putih
 - c. Menghasilkan elemen penataan
 - d. Menduplikasi citra biner
 - e. Menghilangkan Sebagian citra berwarna
2. Berikut ini manakah perintah untuk memanggil proses segmentasi citra....
 - a. skimage.data
 - b. skimage.filters
 - c. skimage.morphology
 - d. skimage.segmentation
 - e. skimage.measure
3. Bentuk yang umum digunakan pada operasi morfologi adalah...
 - a. Bentuk Lingkaran dan bentuk Belah ketupat
 - b. Bentuk Oktagon dan bentuk Lingkaran
 - c. Bentuk Garis tegak dan bentuk Lingkaran
 - d. Bentuk Cakram dan bentuk Lingkaran
 - e. Bentuk Garis Mendatar dan bentuk persegi
4. Apa tujuan dari operasi morfologi untuk proses citra digital...
 - a. Memperbaiki hasil segmentasi citra
 - b. Mengubah struktur bentuk objek
 - c. Membuat rotasi citra
 - d. Mengubah piksel
 - e. Interpolasi piksel

5. Bagian Operasi Morfologi yang digunakan untuk menghaluskan kontur dan menghilangkan lubang-lubang adalah...
 - a. Operasi Dilasi
 - b. Operasi Erosi
 - c. Operasi Closing
 - d. Operasi Opening
 - e. Operasi dasar

BAB 8

PENGOLAHAN CITRA BINER

Representasi Bentuk

Representasi dari bentuk citra dapat berupa kontur, area, dan transformasi.

A. Deteksi Tepi

Konsep algoritma deteksi tepi pada dasarnya memanfaatkan 8-ketetanggaan. Jika sekeliling pixel P bernilai sama (semua=1 atau semua=0), maka diasumsikan bahwa pixel P tidak berada di tepi objek.

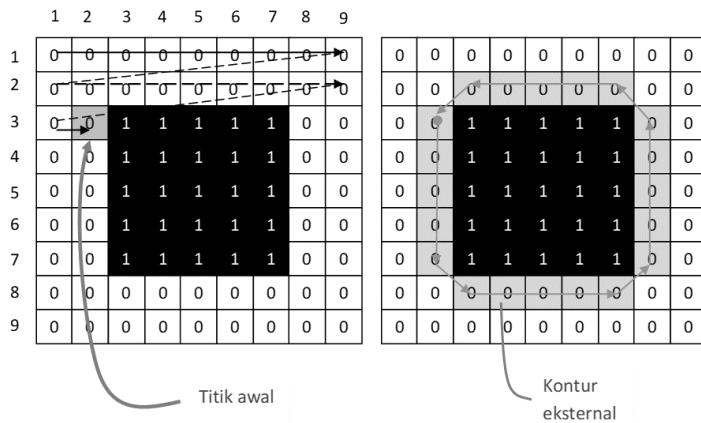
Mengikuti Kontur (Contour Following)

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0	
0	0	1	1	1	1	1	0	0	
0	0	1	1	1	1	1	0	0	
0	0	1	1	1	1	1	0	0	
0	0	1	1	1	1	1	0	0	
0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	

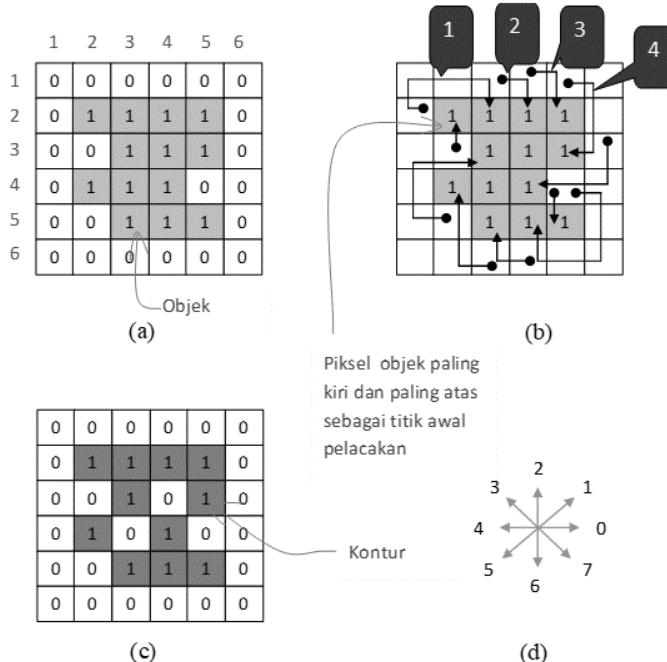
Kontur eksternal

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0	
0	0	1	1	1	1	1	1	0	
0	0	1	1	1	1	1	1	0	
0	0	1	1	1	1	1	1	0	
0	0	1	1	1	1	1	1	0	
0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	

Kontur internal



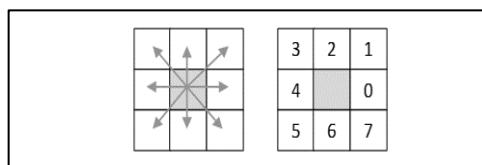
Gambar 8.1 Proses penelusuran countour



Gambar 8.2: Penjelasan pelacakan kountur dengan menggunakan Algoritma Moore

B. Rantai Kode (Code Chain)

Kode rantai merupakan suatu teknik pengolahan citra yang didasarkan pengkodean dengan berdasarkan arah mata angin pada suatu objek citra dua dimensi. Selama ini, kode rantai banyak yang digunakan dalam pengolahan citra untuk merepresentasikan garis, kurva atau batas tepi dari suatu area.



Gambar 8.3: Arah rantai kode

Δx	Δy	Kode Rantai	$\text{Indeks} = 3 \Delta y + \Delta x + 5$
0	+1	6	8
0	-1	2	2
-1	+1	5	7
-1	-1	3	1
+1	+1	7	9
+1	-1	1	3
-1	0	4	4
+1	0	0	6

$$\text{indeks} = 3 \Delta y + \Delta x + 5$$

Kelemahan rantai kode (chain code)

- Kode cenderung panjang.
- Sensitif terhadap distorsi dan segmentasi yang tidak sempurna.
- Sangat bergantung pada penyekalaan ataupun rotasi.

Latihan:

1. Deteksi tepi

Operator tepi digunakan dalam pemrosesan gambar dalam algoritma deteksi tepi. Operator tepi adalah operator diferensiasi diskrit dan menghitung perkiraan gradien fungsi intensitas gambar.

[in]	<pre>import numpy as np import matplotlib.pyplot as plt from skimage import filters from skimage.data import camera from skimage.util import compare_images image = camera() edge_roberts = filters.roberts(image) edge_sobel = filters.sobel(image) fig, axes = plt.subplots(ncols=2, sharex=True, sharey=True, figsize=(8, 4)) axes[0].imshow(edge_roberts, cmap=plt.cm.gray) axes[0].set_title('Roberts Edge Detection') axes[1].imshow(edge_sobel, cmap=plt.cm.gray) axes[1].set_title('Sobel Edge Detection') for ax in axes: ax.axis('off') plt.tight_layout() plt.show()</pre>
[out]	

2. Active Contour

Model Active Contour adalah metode segmentasi yang menggunakan model kurva tertutup yang dapat bergerak melebar ataupun mengecil

```
import numpy as np
import matplotlib.pyplot as plt
from skimage.color import rgb2gray
from skimage import data
from skimage.filters import gaussian
from skimage.segmentation import
active_contour
img = data.text()

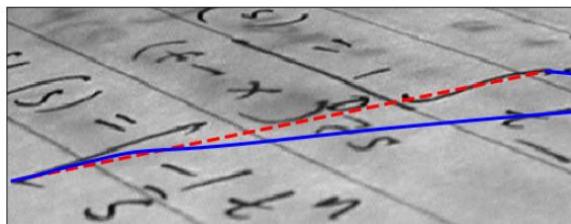
r = np.linspace(136, 50, 100)
c = np.linspace(5, 424, 100)
init = np.array([r, c]).T

[in]
snake = active_contour(gaussian(img, 1), init,
boundary_condition='fixed',
alpha=0.1, beta=1.0,
w_line=-5, w_edge=0, gamma=0.1)

fig, ax = plt.subplots(figsize=(9, 5))
ax.imshow(img, cmap=plt.cm.gray)
ax.plot(init[:, 1], init[:, 0], '--r', lw=3)
ax.plot(snake[:, 1], snake[:, 0], '-b', lw=3)
ax.set_xticks([]), ax.set_yticks([])
ax.axis([0, img.shape[1], img.shape[0], 0])

plt.show()
```

[out]



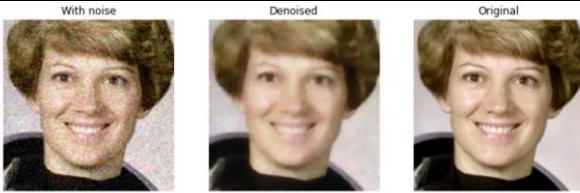
3. Gaussian noise

[in]	<pre>import numpy as np import matplotlib.pyplot as plt import skimage import skimage.color as skic import skimage.filters as skif import skimage.data as skid import skimage.util as sku %matplotlib inline</pre>
[in]	<pre>def show(img): fig, ax = plt.subplots(1, 1, figsize=(8, 8)) ax.imshow(img, cmap=plt.cm.gray) ax.set_axis_off() plt.show()</pre>
[in]	<pre>img = skic.rgb2gray(skid.astronaut()) show(img)</pre>
[out]	
[in]	<pre>show(skif.gaussian(img, 5.))</pre>
[out]	

[in]	<pre>sobimg = skif.sobel(img) show(sobimg)</pre>
[out]	
[in]	<pre>from ipywidgets import widgets @widgets.interact(x=(0.01, .2, .005)) def edge(x): show(sobimg < x)</pre>
[in]	<pre>img = skimage.img_as_float(skid.astronaut()) # We take a portion of the image to show the # details. img = img[50:200, 150:300] # We add Gaussian noise. img_n = sku.random_noise(img) show(img_n)</pre>
[out]	

```
[in] img_r =  
skimage.restoration.denoise_tv_bregman(  
    img_n, 5.)  
  
fig, (ax1, ax2, ax3) = plt.subplots(  
    1, 3, figsize=(12, 8))  
  
ax1.imshow(img_n)  
ax1.set_title('With noise')  
ax1.set_axis_off()  
  
ax2.imshow(img_r)  
ax2.set_title('Denoised')  
ax2.set_axis_off()  
  
ax3.imshow(img)  
ax3.set_title('Original')  
ax3.set_axis_off()
```

[out]



Soal:

Pilih salah satu jawaban yang benar dengan memberi tanda pada lembar jawaban.

1. Berikut ini manakah perintah untuk memanggil operasi deteksi tepi...
 - a. filters.sobel () dan filters.roberts()
 - b. skimage.filters dan skimage.data
 - c. skimage.segmentation dan active_contour
 - d. skimage.restoration.denoise_tv_bregman
 - e. skic.rgb2gray
2. Yang tidak termasuk kedalam Representasi dari bentuk citra adalah....
 - a. kontur, area, dan transformasi
 - b. area dan transformasi
 - c. kontur dan area
 - d. Kontur dan transformasi
 - e. Piksel dan kontur
3. Metode segmentasi yang menggunakan model kurva tertutup yang dapat bergerak melebar ataupun mengecil disebut model...
 - a. Model Gaussian
 - b. Model Active Contor
 - c. Model Treshold
 - d. Model Segmentasi
 - e. Model Morphology
4. Teknik pengolahan citra yang didasarkan pengkodean dengan berdasarkan arah mata angin pada suatu objek citra dua dimensi disebut sebagai teknik ...
 - a. Deteksi Tepi
 - b. Active Contor
 - c. Gaussian
 - d. Code chain
 - e. Otsu

5. Apakah kelemahan penggunaan Rantai kode(Code chain)...
 - a. Kode pendek
 - b. Sensitif terhadap distorsi
 - c. Segmentasi Sempurna
 - d. Bergantung pada rotasi
 - e. Tidak beraturan

BAB 9

PENGOLAHAN CITRA BERWARNA

A. Teori Dasar Warna

Warna yang dilihat manusia memiliki beberapa persepsi, sebenarnya manusia melihat warna karena adanya cahaya yang dipantulkan oleh objek. Dalam hal ini, spektrum cahaya kromatis berkisar antara 400-700 nanometer.

Warna berdasarkan persepsi mata manusia:

- Hue (warna sesuai panjang gelombang)
- Saturation (kemurnian)
- Brightness (kecerahan)

Hue merujuk ke warna yang dikenal manusia, seperti merah dan hijau, sesuai dengan panjang gelombangnya. Panjang gelombang antara 430 dan 480 nanometer biru. Panjang gelombang berkisar antara 570 sampai dengan 600 nm kuning.

Saturation menyatakan tingkat kemurnian warna atau seberapa banyak cahaya putih yang tercampur dengan hue. Setiap warna murni bersaturasi 100% dan tidak mengandung cahaya putih sama sekali.

Brightness atau kadang disebut lightness (kecerahan) menyatakan intensitas pantulan objek yang diterima mata. Intensitas dapat dinyatakan sebagai perubahan warna putih menuju abu-abu dan terakhir mencapai ke warna hitam, atau yang dikenal dengan istilah aras keabuan.

B. Ruang Warna

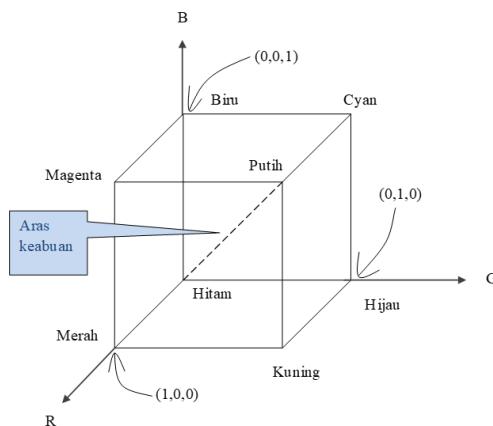
Ruang warna merupakan suatu spesifikasi sistem koordinat dan suatu sub-ruang dalam sistem dengan setiap warna dinyatakan dengan satu titik di dalamnya. Tujuan dibentuknya ruang warna adalah untuk memfasilitasi spesifikasi warna dalam bentuk suatu standar.

Ruang warna yang paling dikenal pada perangkat komputer adalah RGB.

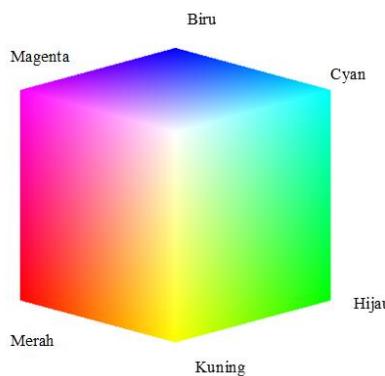
Ruang warna lain: HSI, CMY, LUV, dan YIQ.

C. Ruang Warna RGB

Ruang warna RGB biasa diterapkan pada monitor CRT dan kebanyakan sistem grafika komputer. Ruang warna ini menggunakan tiga komponen dasar yaitu merah (R), hijau (G), dan biru (B). Sistem RGB familiar bagi manusia, tapi sulit digunakan dalam mengidentifikasi objek.



Gambar 9.1: Skema ruang warna RGB dalam bentuk kubus

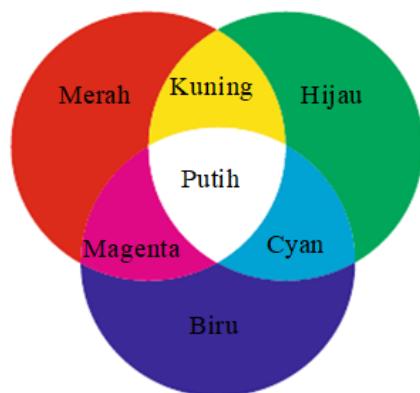


Gambar 9.2: Kubus Warna dengan 24 bit

D. Ruang warna CMYK

Model warna CMY (cyan, magenta, yellow) mempunyai hubungan dengan RGB (ternormalisasi 0-1) sebagai berikut:

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$



Gambar 9.3: Penjumlahan warna pada sistem RGB



Gambar 9.4: pengurangan warna pada sistem CMY

E. Ruang Warna YIQ

Ruang warna **YIQ**, yang juga dikenal dengan nama ruang warna **NTSC**, dirumuskan oleh NTSC ketika mengembangkan sistem televisi berwarna di Amerika Serikat. Pada model ini, Y disebut luma (yang menyatakan luminans); I dan Q disebut chroma.

Warna hitam terjadi saat nilai C=M=Y.

Pada printer ditambahkan warna hitam (black/K) menjadi CMYK tinta hitam lebih mudah diperoleh dan lebih murah.

RGB ke YIQ

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0,299 & 0,587 & 0,114 \\ 0,596 & -0,274 & -0,322 \\ 0,211 & -0,523 & 0,312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Octave: `rgb2ntsc()`

YIQ ke RGB

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1,000 & 0,956 & 0,621 \\ 1,000 & -0,272 & -0,647 \\ 1,000 & -1,106 & 1,703 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$

Octave: `ntsc2rgb()`

F. Ruang Warna YCbCr

Ruang warna YCbCr biasa digunakan pada video digital.

Komponen Y menyatakan intensitas, sedangkan C_b dan C_r menyatakan informasi warna.

Contoh penggunaan: kompresi JPEG dan MPEG

RGB ke YCbCr

$$\begin{aligned} Y &= 0,29900R + 0,58700G + 0,11400B \\ C_b &= -0,16874R - 0,33126G + 0,50000B \\ C_r &= +0,5000R - 0,41869G - 0,08131B \end{aligned}$$

Octave: `rgb2ycbcr()`

YCbCr ke RGB

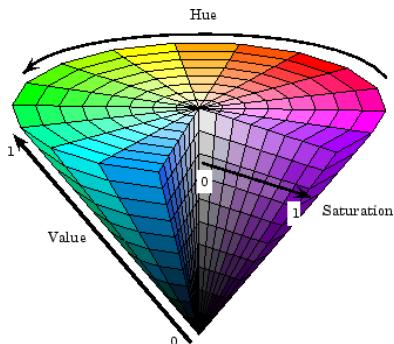
$$\begin{aligned} R &= Y + 1,40200C_r \\ G &= Y - 0,34414C_b - 0,71414C_r \\ B &= Y + 1,77200C_b \end{aligned}$$

Octave: `ycbcr2rgb()`

1. Ruang Warna HSI, HSV, dan HSL

HSI, HSV dan HSL merupakan contoh ruang warna yang merepresentasikan warna seperti yang dilihat oleh mata manusia.

H berasal dari kata “hue”, S berasal dari “saturation”, L berasal dari kata “luminance”, I berasal dari kata “intensity”, dan V berasal dari “value”.



Gambar 9.5: Ruang Warna Hue

2. Konversi RGB ke HSV

Konversi RGB ke HSV menurut Acharya & Ray (2005)

$$r = \frac{R}{(R+G+B)}, g = \frac{G}{(R+G+B)}, b = \frac{B}{(R+G+B)}$$

$$V = \max(r, g, b)$$

$$S = \begin{cases} 0, & \text{jika } V = 0 \\ 1 - \frac{\min(r, g, b)}{V}, & V > 0 \end{cases}$$

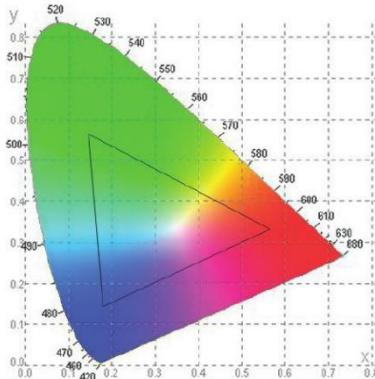
$$H = \begin{cases} 0, & \text{jika } S = 0 \\ \frac{60 * (g - b)}{S * V}, & \text{jika } V = r \\ 60 * \left[2 + \frac{b - r}{S * V} \right], & \text{jika } V = g \\ 60 * \left[4 + \frac{r - g}{S * V} \right], & \text{jika } V = b \end{cases}$$

$$H = H + 360 \text{ jika } H < 0$$

3. Ruang Warna CIELAB

CIELAB adalah nama lain dari CIE L*a*b*. Diagram kromasitas CIE (Commission Internationale de L'Eclairage) ditunjukkan pada Gambar. Setiap perpaduan x dan y menyatakan suatu warna. Namun, hanya warna yang berada dalam area ladam (tapal kuda) yang bisa terlihat. Angka yang berada di tepi menyatakan panjang gelombang cahaya.

Warna yang terletak di dalam segitiga menyatakan warna-warna umum di monitor CRT, yang dapat dihasilkan oleh komponen warna merah, hijau, dan biru.



Gambar 9.6: Ruang warna

4. Statistik Warna

Statistik pada citra berwarna dapat diekstraksi untuk berbagai keperluan, antara lain identifikasi warna / objek.

Beberapa statistik warna:

- rerata,
- deviasi standar,
- skewness (kecondongan), dan
- kurtosis (kelancipan).

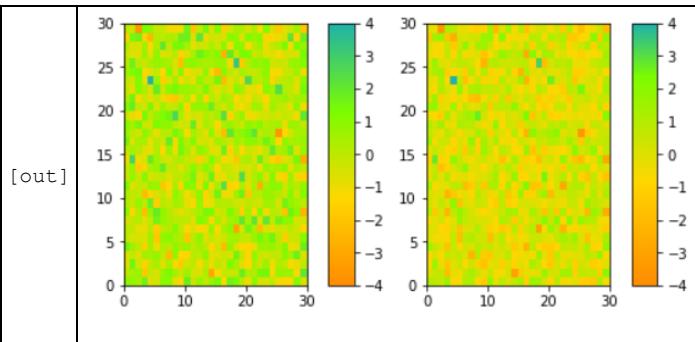
Latihan:

1. Membuat list peta warna

[in]	<pre>import numpy as np import matplotlib.pyplot as plt from matplotlib import cm from matplotlib.colors import ListedColormap, LinearSegmentedColormap viridis = cm.get_cmap('viridis', 8)</pre>
[in]	<pre>print(viridis(0.56))</pre>
[out]	<pre>(0.122312, 0.633153, 0.530398, 1.0)</pre>
[in]	<pre>def plot_examples(colormaps): """ Helper function to plot data with associated colormap. """ np.random.seed(19680801) data = np.random.randn(30, 30) n = len(colormaps) fig, axs = plt.subplots(1, n, figsize=(n * 2 + 2, 3), constrained_layout=True, squeeze=False) for [ax, cmap] in zip(axs.flat, colormaps): psm = ax.pcolormesh(data, cmap=cmap, rasterized=True, vmin=-4, vmax=4) fig.colorbar(psm, ax=ax) plt.show()</pre>
[in]	<pre>cmap = ListedColormap(["darkorange", "gold", "lawngreen", "lightseagreen"]) plot examples([cmap])</pre>
[out]	
[in]	<pre>viridis = cm.get_cmap('viridis', 256) newcolors = viridis(np.linspace(0, 1, 256)) pink = np.array([248/256, 24/256, 148/256, 1]) newcolors[:25, :] = pink newcmp = ListedColormap(newcolors) plot examples([viridis, newcmp])</pre>

	[out]	
	[in]	<pre>top = cm.get_cmap('Oranges_r', 128) bottom = cm.get_cmap('Blues', 128) newcolors = np.vstack((top(np.linspace(0, 1, 128)), bottom(np.linspace(0, 1, 128)))) newcmp = ListedColormap(newcolors, name='OrangeBlue') plot examples([viridis, newcmp])</pre>
	[out]	
	[in]	<pre>N = 256 vals = np.ones((N, 4)) vals[:, 0] = np.linspace(90/256, 1, N) vals[:, 1] = np.linspace(40/256, 1, N) vals[:, 2] = np.linspace(40/256, 1, N) newcmp = ListedColormap(vals) plot examples([viridis, newcmp])</pre>
	[out]	

	<pre> cdict = {'red': [[0.0, 0.0, 0.0], [0.5, 1.0, 1.0], [1.0, 1.0, 1.0]], 'green': [[0.0, 0.0, 0.0], [0.25, 0.0, 0.0], [0.75, 1.0, 1.0], [1.0, 1.0, 1.0]], 'blue': [[0.0, 0.0, 0.0], [0.5, 0.0, 0.0], [1.0, 1.0, 1.0]]} def plot_linearemap(cdict): newcmp = LinearSegmentedColormap('testCmap', segmentdata=cdict, N=256) rgba = newcmp(np.linspace(0, 1, 256)) fig, ax = plt.subplots(figsize=(4, 3), constrained_layout=True) col = ['r', 'g', 'b'] for xx in [0.25, 0.5, 0.75]: ax.axvline(xx, color='0.7', linestyle='--') for i in range(3): ax.plot(np.arange(256)/256, rgba[:, i], color=col[i]) ax.set_xlabel('index') ax.set_ylabel('RGB') plt.show() plot_linearemap(cdict) </pre>
[out]	
[in]	<pre> colors = ["darkorange", "gold", "lawngreen", "lightseagreen"] cmap1 = LinearSegmentedColormap.from_list("mycmap", colors) nodes = [0.0, 0.4, 0.8, 1.0] cmap2 = LinearSegmentedColormap.from_list("mycmap", list(zip(nodes, colors))) plot examples([cmap1, cmap2]) </pre>
[in]	<pre> colors = ["darkorange", "gold", "lawngreen", "lightseagreen"] cmap1 = LinearSegmentedColormap.from_list("mycmap", colors) nodes = [0.0, 0.4, 0.8, 1.0] cmap2 = LinearSegmentedColormap.from_list("mycmap", list(zip(nodes, colors))) plot examples([cmap1, cmap2]) </pre>



2. Colormap Normalization

```

[in]   import numpy as np
       import matplotlib.pyplot as plt
       import matplotlib.colors as colors
       import matplotlib.cbook as cbook
       from matplotlib import cm

       N = 100
       X, Y = np.mgrid[-3:3:complex(0, N), -
                        2:2:complex(0, N)]

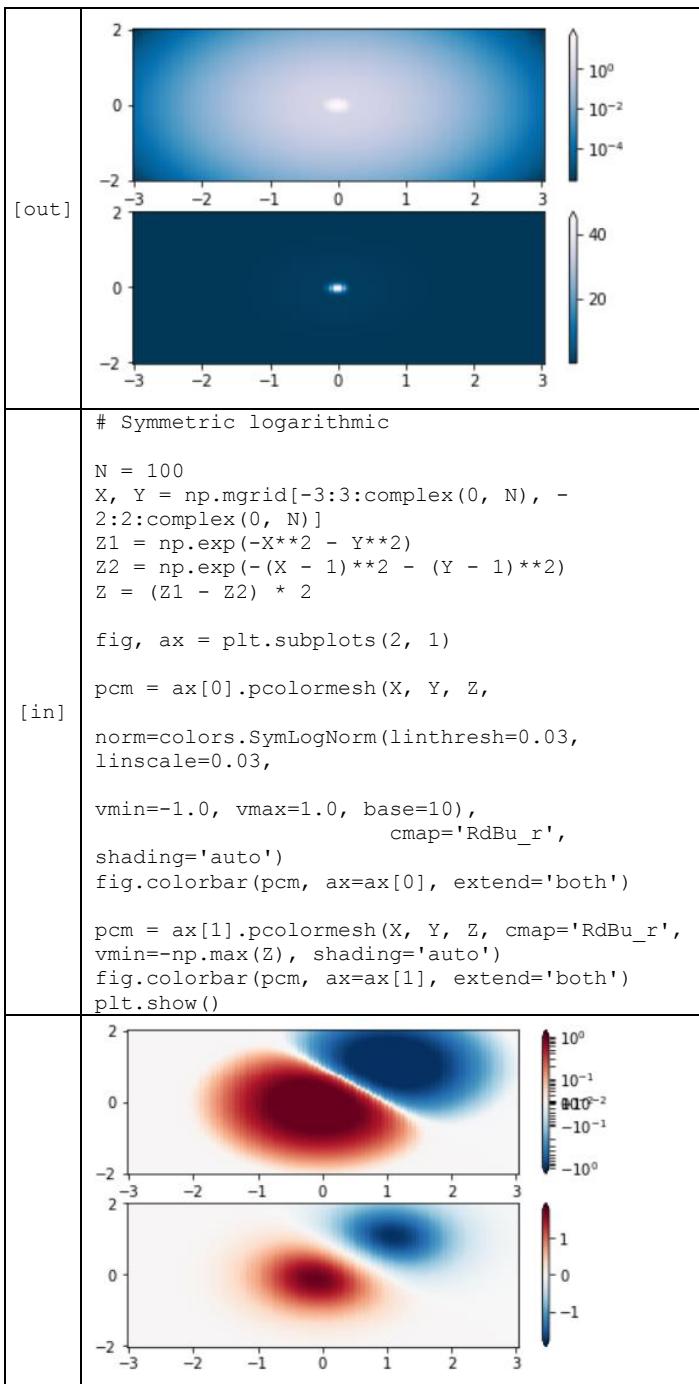
       # Logarithmic
       Z1 = np.exp(-X**2 - Y**2)
       Z2 = np.exp(-(X * 10)**2 - (Y * 10)**2)
       Z = Z1 + 50 * Z2

[out]  fig, ax = plt.subplots(2, 1)

       pcm = ax[0].pcolor(X, Y, Z,
                           norm=colors.LogNorm(vmin=Z.min(),
                           vmax=Z.max()),
                           cmap='PuBu_r',
                           shading='auto')
       fig.colorbar(pcm, ax=ax[0], extend='max')

       pcm = ax[1].pcolor(X, Y, Z, cmap='PuBu_r',
                           shading='auto')
       fig.colorbar(pcm, ax=ax[1], extend='max')
       plt.show()

```



```

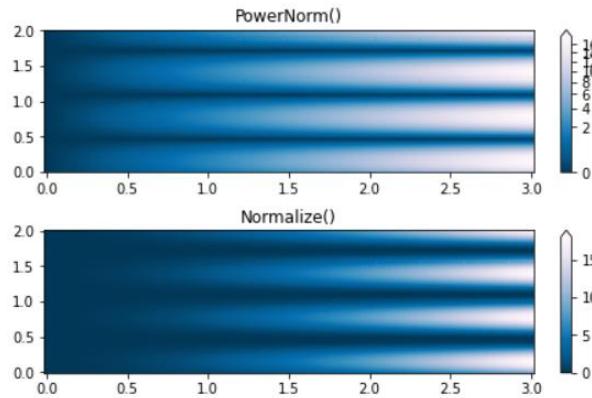
N = 100
X, Y = np.mgrid[0:3:complex(0, N),
0:2:complex(0, N)]
Z1 = (1 + np.sin(Y * 10.)) * X**2

fig, ax = plt.subplots(2, 1,
constrained_layout=True)

pcm = ax[0].pcolormesh(X, Y, Z1,
norm=colors.PowerNorm(gamma=0.5),
cmap='PuBu_r',
shading='auto')
fig.colorbar(pcm, ax=ax[0], extend='max')
ax[0].set_title('PowerNorm()')

pcm = ax[1].pcolormesh(X, Y, Z1,
cmap='PuBu_r', shading='auto')
fig.colorbar(pcm, ax=ax[1], extend='max')
ax[1].set_title('Normalize()')
plt.show()

```



```

import matplotlib.colors as colors

bounds = np.array([-0.25, -0.125, 0, 0.5, 1])
norm = colors.BoundaryNorm(boundaries=bounds,
ncolors=4)
print(norm([-0.2, -0.15, -0.02, 0.3, 0.8,
0.99]))

```

```
[0 0 1 2 3 3]
```

```

# Discrete bounds
N = 100
X, Y = np.meshgrid(np.linspace(-3, 3, N),
np.linspace(-2, 2, N))
Z1 = np.exp(-X**2 - Y**2)
Z2 = np.exp(-(X - 1)**2 - (Y - 1)**2)
Z = ((Z1 - Z2) * 2)[::-1, ::-1]

fig, ax = plt.subplots(2, 2, figsize=(8, 6),
constrained_layout=True)

```

```

ax = ax.flatten()

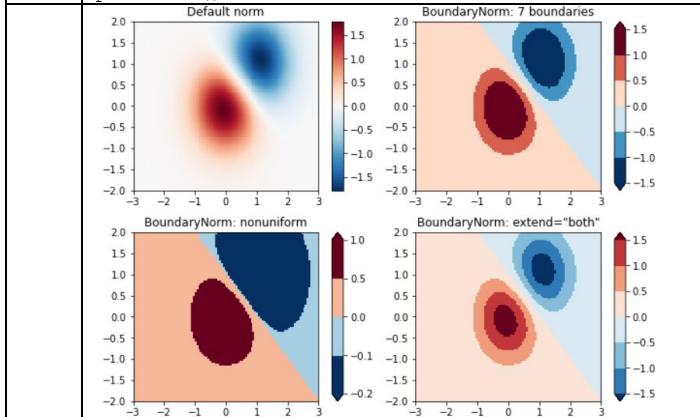
# Default norm:
pcm = ax[0].pcolormesh(X, Y, Z, cmap='RdBu_r')
fig.colorbar(pcm, ax=ax[0],
             orientation='vertical')
ax[0].set_title('Default norm')

# Even bounds give a contour-like effect:
bounds = np.linspace(-1.5, 1.5, 7)
norm = colors.BoundaryNorm(boundaries=bounds,
                            ncolors=256)
pcm = ax[1].pcolormesh(X, Y, Z, norm=norm,
                        cmap='RdBu_r')
fig.colorbar(pcm, ax=ax[1], extend='both',
             orientation='vertical')
ax[1].set_title('BoundaryNorm: 7 boundaries')

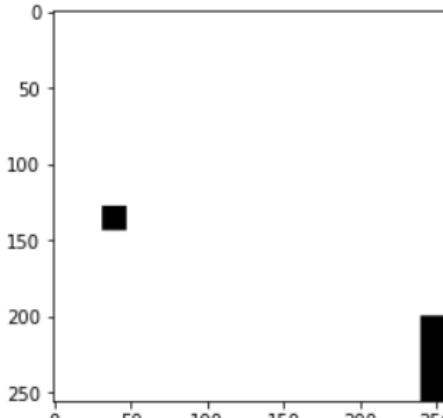
# Bounds may be unevenly spaced:
bounds = np.array([-0.2, -0.1, 0, 0.5, 1])
norm = colors.BoundaryNorm(boundaries=bounds,
                            ncolors=256)
pcm = ax[2].pcolormesh(X, Y, Z, norm=norm,
                        cmap='RdBu_r')
fig.colorbar(pcm, ax=ax[2], extend='both',
             orientation='vertical')
ax[2].set_title('BoundaryNorm: nonuniform')

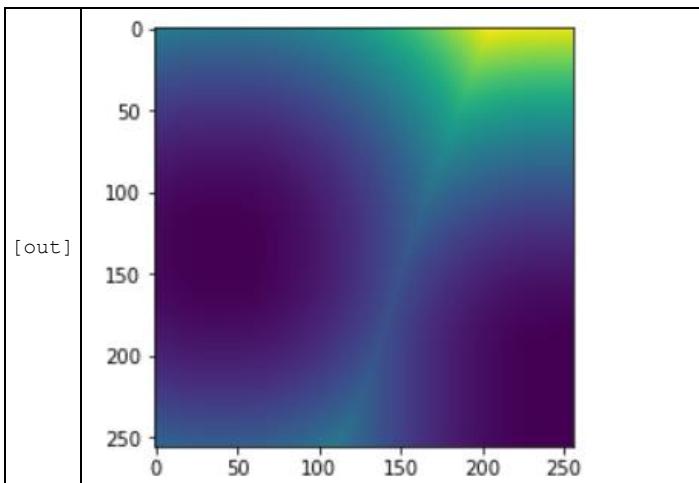
# With out-of-bounds colors:
bounds = np.linspace(-1.5, 1.5, 7)
norm = colors.BoundaryNorm(boundaries=bounds,
                            ncolors=256, extend='both')
pcm = ax[3].pcolormesh(X, Y, Z, norm=norm,
                        cmap='RdBu_r')
# The colorbar inherits the "extend" argument
# from BoundaryNorm.
fig.colorbar(pcm, ax=ax[3],
             orientation='vertical')
ax[3].set_title('BoundaryNorm: extend="both"')
plt.show()

```

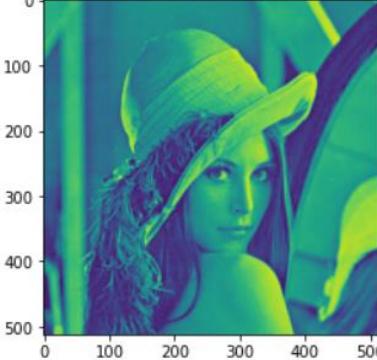


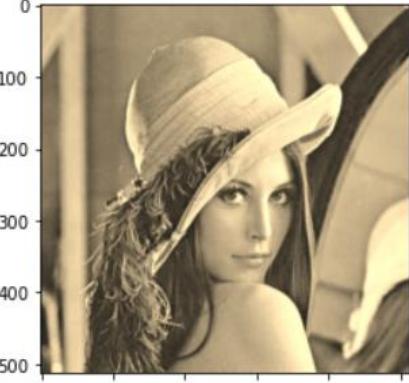
3. Transformasi Jarak

[in]	<pre>import numpy as np import mahotas f = np.ones((256,256), bool) f[200:,240:] = False f[128:144,32:48] = False</pre>
[in]	<pre>from pylab import imshow, gray, show import numpy as np f = np.ones((256,256), bool) f[200:,240:] = False f[128:144,32:48] = False gray() imshow(f) show()</pre>
[out]	
[in]	<pre>import mahotas dmap = mahotas.distance(f)</pre>
[in]	<pre>from __future__ import print_function import pylab as p import numpy as np import mahotas f = np.ones((256,256), bool) f[200:,240:] = False f[128:144,32:48] = False dmap = mahotas.distance(f) p.imshow(dmap) p.show()</pre>



4. Konversi Ruang Warna

[in]	<pre>import mahotas as mh lena = mh.demos.load('lena') print(lena.shape)</pre>
[out]	(512, 512, 3)
[in]	<pre>import mahotas as mh lena = mh.demos.load('lena') lenag = mh.colors.rgb2grey(lena)</pre>
[in]	<pre>from pylab import imshow import mahotas as mh lena = mh.demos.load('lena') lenag = mh.colors.rgb2grey(lena) imshow(lenag)</pre>
[out]	<matplotlib.image.AxesImage at 0x206ef0aff40> 

[in]	<pre>lenas = mh.colors.rgb2sepia(lena) from pylab import imshow import mahotas as mh lena = mh.demos.load('lena') lenas = mh.colors.rgb2sepia(lena) imshow(lenas)</pre>
[out]	<matplotlib.image.AxesImage at 0x206ef1aea30>  A sepia-toned photograph of a woman with dark hair, wearing a light-colored hat and a dark top. She is looking slightly to her left. The image is displayed as a plot with numerical scales on both the horizontal and vertical axes, ranging from 0 to 500.

Soal:

Pilih salah satu jawaban yang benar dengan memberi tanda pada lembar jawaban.

1. Ruang warna yang paling dikenal pada perangkat komputer adalah ...
 - a. Hue, Saturation, Brightness
 - b. RGB dan CMYK
 - c. CMYK dan YIQ
 - d. YCbCr dan RGB
 - e. CIELAB dan RGB
2. Pada ruang warna CMYK, Inisiak K berasak dari warna....
 - a. Dark Brown
 - b. Hunter
 - c. Pink
 - d. Black
 - e. Cyan
3. Ruang Warna YCbCr komponen Y menyatakan...
 - a. Saturation
 - b. Intensitas
 - c. Brightness
 - d. Hue
 - e. Color
4. Perintah untuk memanggil daftar peta warna adalah...
 - a. LinearSegmentedColormap
 - b. ListedColormap
 - c. newcolors
 - d. fig.colorbar
 - e. matplotlib.colors
5. Yang tidak termasuk statistic warna adalah...
 - a. Rerata
 - b. Deviasi standar
 - c. Skewness
 - d. Kurtosis
 - e. Median

BAB 10

SEGMENTASI CITRA BERWARNA

A. Pengantar Segmentasi citra

Segmentasi citra merupakan tahapan penting dalam proses pengenalan pola. Dari objek yang berhasil tersegmentasi, maka selanjutnya dilakukan proses ekstraksi. Segmentasi citra merupakan proses yang ditujukan untuk mendapatkan objek-objek yang terkandung di dalam citra atau membagi citra ke dalam beberapa objek berdasarkan kemiripan dari objek tersebut. Pada citra yang mengandung hanya satu objek, objek dibedakan dari latar belakangnya.



Gambar 10.1: segmentasi citra objek Cat & Dog

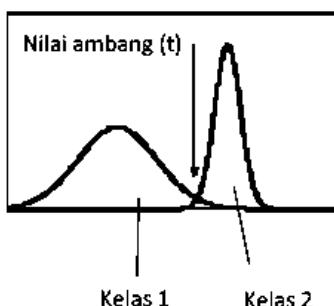
Tujuan segmentasi citra adalah untuk mempartisi gambar menjadi beberapa wilayah yang tidak tumpang tindih dengan karakteristik yang homogen, seperti intensitas, warna, dan tekstur.

B. Thresholding Metode Otsu

Dalam pengolahan citra digital, metode Otsu digunakan untuk pengambangan (*thresholding*) citra otomatis. Algoritma ini mengembalikan nilai ambang intensitas tunggal yang membagi piksel-piksel menjadi dua kelas, yaitu latar depan dan latar belakang. Nilai ambang ini ditentukan

dengan meminimalkan ragam intensitas dalam kelas atau memaksimalkan ragam intensitas antarkelas.

Metode Otsu dipublikasikan oleh *Nobuyuki Otsu* pada tahun 1979. Metode ini menentukan nilai ambang dengan cara membedakan dua kelompok, yaitu objek dan latar belakang, yang memiliki bagian yang saling bertumpukan, berdasarkan histogram.



Gambar 10.2: Penentuan nilai ambang untuk memperoleh hasil yang optimal

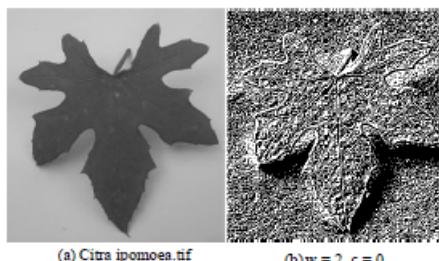
Metode Otsu cukup baik jika histogram dianggap memiliki persebaran dua puncak serta memiliki lembah yang curam dan dalam di antara dua puncak. Namun, bila luas objek (latar depan) berukuran kecil dibandingkan luas latar belakang, histogram tidak lagi memiliki sifat dua puncak. Jika ragam objek dan ragam latar belakang cukup besar dibanding selisih rata-rata atau jika citra sangat rusak akibat derau aditif, kecuraman lembah pada histogram menurun. Akibatnya, nilai ambang yang dihasilkan menyebabkan kesalahan segmentasi.

C. Multilevel Thresholding

Pada peng-ambangan beraras-jamak (*multilevel thresholding*), citra dibagi menjadi beberapa bagian dengan menggunakan beberapa nilai ambang.

1. Adaptive Thresholding

Peng-ambangan adaptif (*adaptive thresholding*) merupakan peng-ambangan yang menggunakan nilai ambang lokal, yang dihitung secara adaptif berdasarkan statistika piksel-piksel tetangga. Hal ini didasarkan kenyataan bahwa bagian-bagian kecil dalam citra mempunyai iluminasi yang sama, sehingga lebih tepat kalau nilai ambang dihitung berdasarkan bagian-bagian kecil dalam citra dan bukan berdasarkan seluruh piksel dalam citra.



Nilai threshold ditentukan berdasarkan nilai piksel tetangga pada “window” dengan ukuran tertentu. Nilai threshold untuk setiap “window” dapat berbeda-beda (adaptive).

Algoritma penentuan *threshold*:

- Rerata (mean)
- Max-min
- Nilai tengah (median)

D. Segmentasi Chan-vese

Algoritma segmentasi Chan-Vese dirancang untuk mengelompokkan objek tanpa batasan yang jelas. Algoritme ini didasarkan pada set level yang dikembangkan secara iteratif untuk meminimalkan energi, yang ditentukan oleh nilai berbobot yang sesuai dengan jumlah perbedaan intensitas dari nilai rata-rata di luar wilayah tersegmentasi, jumlah perbedaan dari nilai rata-rata di dalam wilayah

tersegmentasi dan suku yang bergantung pada panjang batas daerah tersegmentasi.

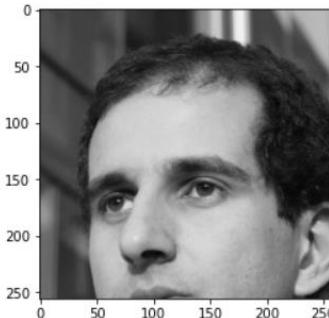
Algoritma ini pertama kali diusulkan oleh Tony Chan dan Luminita Vese, dalam sebuah publikasi yang berjudul "*An Active Contour Model Without Edges*".

Implementasi algoritma ini agak disederhanakan dalam arti bahwa faktor area 'nu' yang dijelaskan dalam makalah asli tidak diterapkan, dan hanya cocok untuk gambar skala abu-abu. Nilai tipikal untuk lambda1 dan lambda2 adalah 1. Jika latar belakang objek sangat berbeda dari objek tersegmentasi dalam hal distribusi (misalnya, gambar hitam seragam dengan figur dengan intensitas bervariasi), maka nilai ini harus berbeda satu sama lain.

Nilai tipikal untuk muadalah antara 0 dan 1, meskipun nilai yang lebih tinggi dapat digunakan ketika berhadapan dengan bentuk dengan kontur yang sangat tidak jelas. Algoritma juga mengembalikan daftar nilai yang sesuai dengan energi pada setiap iterasi. Ini dapat digunakan untuk menyesuaikan berbagai parameter yang dijelaskan di atas.

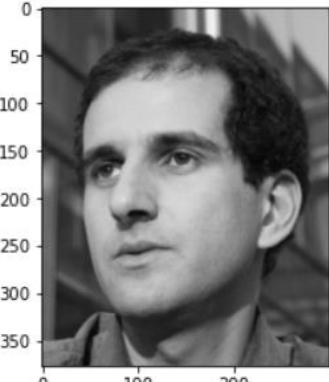
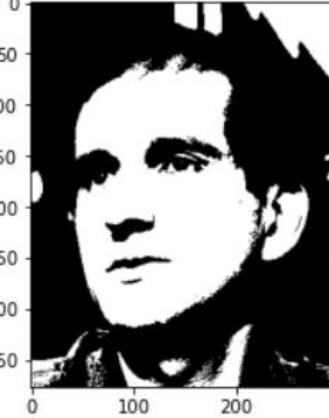
Latihan:

1. Penggunaan **mahotas.thresholding**

[in]	<pre>import numpy as np import mahotas import mahotas.demos from mahotas.thresholding import soft_threshold from matplotlib import pyplot as plt from os import path f = mahotas.demos.load('luispedro', as_grey=True) f = f[:256,:256] plt.gray() # Show the data: print("Fraction of zeros in original image: {0}".format(np.mean(f==0))) plt.imshow(f) plt.show()</pre>
[out]	<p>Fraction of zeros in original image: 0.0</p> 

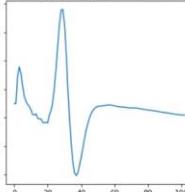
2. Penggunaan **mahotas.otsu**

[in]	<pre>import mahotas import mahotas.demos import numpy as np from pylab import imshow, gray, show from os import path photo = mahotas.demos.load('luispedro', as_grey=True) photo = photo.astype(np.uint8) gray() imshow(photo) show()</pre>
------	---

	[out]	
	[in]	<pre>T_otsu = mahotas.otsu(photo) print(T_otsu) imshow(photo > T_otsu) show()</pre>
	[out]	
	[in]	<pre>import mahotas import mahotas.demos import numpy as np from pylab import imshow, gray, show from os import path photo = mahotas.demos.load('luispedro', as_grey=True) photo = photo.astype(np.uint8) T_otsu = mahotas.otsu(photo) print(T_otsu) gray() imshow(photo > T_otsu) show()</pre>

	116	
[out]	 A binary thresholded image of a man's face. The image is black and white, showing high contrast between the subject and the background. The man has dark hair and is looking slightly to the right. The image is displayed within a coordinate system where the x-axis ranges from 0 to 200 and the y-axis ranges from 0 to 350.	
[in]	<pre>T_rc = mahotas.rc(photo) print(T_rc) imshow(photo > T_rc) show()</pre>	
[out]	115.95273822325237	 A binary thresholded image of a man's face, identical to the one above. It shows high contrast between the subject and the background. The man has dark hair and is looking slightly to the right. The image is displayed within a coordinate system where the x-axis ranges from 0 to 200 and the y-axis ranges from 0 to 350.

3. Segmentasi Chan Vase

	<pre> import matplotlib.pyplot as plt from skimage import data, img_as_float from skimage.segmentation import chan_vese image = img_as_float(data.camera()) # Feel free to play around with the parameters # to see how they impact the result cv = chan_vese(image, mu=0.25, lambda1=1, lambda2=1, tol=1e-3, max_iter=200, dt=0.5, init_level_set="checkerboard", extended_output=True) fig, axes = plt.subplots(2, 2, figsize=(8, 8)) ax = axes.flatten() ax[0].imshow(image, cmap="gray") ax[0].set_axis_off() ax[0].set_title("Original Image", fontsize=12) ax[1].imshow(cv[0], cmap="gray") ax[1].set_axis_off() title = "Chan-Vese segmentation - {}" iterations".format(len(cv[2])) ax[1].set_title(title, fontsize=12) ax[2].imshow(cv[1], cmap="gray") ax[2].set_axis_off() ax[2].set_title("Final Level Set", fontsize=12) ax[3].plot(cv[2]) ax[3].set_title("Evolution of energy over iterations", fontsize=12) fig.tight_layout() plt.show() </pre>
[in]	
[out]	   

Soal:

Pilih salah satu jawaban yang benar dengan memberi tanda pada lembar jawaban.

1. Membagi citra ke dalam beberapa objek berdasarkan kemiripan dari objek merupakan proses segmentasi citra, namun pada citra yang mengandung hanya satu objek Bagaimanakah cara membedakannya?
 - a. Objek dibedakan dari kemiripannya
 - b. Objek dibagi dari jenisnya.
 - c. Objek dibagi dari tingkat warnanya
 - d. Objek dibedakan dari latar belakangnya
 - e. Objek dibedakan dari pikselnya
2. Bagaimana cara kerja Adaptive thresholding
 - a. Menghituing nilai piksel terdekat pada *window* dengan ukuran tertentu
 - b. Menggunakan nilai ambang lokal, dan dihitung secara adaptif dari piksel terdekat
 - c. Menyusun warna berdasarkan ukuran latar objek
 - d. Menghitung piksel tetangga berdasarkan level warna
 - e. Menggunakan nilai terdekat dari objek
3. Apakah dungsi dari mahotas.thresholding...
 - a. Memanggil fungsi nilai pengambangan
 - b. Mencetak hasil fungsi nilai pengambangan
 - c. Memanipulasi gambar objek
 - d. Meningkatkan kontras objek
 - e. Memanggil filter hitam putih
4. Fungsi dari mengonversi **np.uint8** adalah ...
 - a. Mengembalikan gambar titik *thresholding*
 - b. Memanipulasi fungsi ambang batas
 - c. Memanipulasi gambar objek
 - d. Meningkatkan kontras objek
 - e. Memanggil filter hitam putih

5. Bagaimana cara kerja segmentasi Chan-Vese ...
 - a. Mengelompokkan objek tanpa batasan
 - b. Mengelompokan piksel dari nilai ambang
 - c. Memanipulasi fungsi ambang batas
 - d. Memanipulasi gambar objek
 - e. Meningkatkan kontras objek

BAB 11

EKSTRAKSI FITUR BENTUK DAN KONTUR

A. Ekstraksi fitur

Ciri atau fitur merupakan sesuatu yang dapat membedakan suatu objek dengan objek yang lain. Penggunaan ciri sebagai pembeda antara citra satu dengan yang lainnya sangat bergantung dengan tujuan dan kebutuhan proses pengolahan citra. Sebagai contoh untuk mendeteksi lampu lalu lintas yang sedang menyala di suatu perempatan, ciri warna dan bentuk sangat tepat untuk digunakan. Namun untuk keperluan mendeteksi pejalan kaki di suatu keramaian, ciri warna tidak terlalu tepat untuk digunakan. Ekstraksi fitur merupakan proses atau metode untuk mendapatkan fitur-fitur yang diperlukan dari sebuah citra.

Ekstraksi fitur terbagi menjadi tiga macam yaitu ekstraksi fitur bentuk, ekstraksi fitur tekstur, ekstraksi fitur warna.

1. Ekstraksi fitur bentuk

Bentuk dari suatu objek adalah karakter konfigurasi permukaan yang diwakili oleh garis dan kontur. Fitur bentuk dikategorikan bergantung pada teknik yang digunakan. Kategori tersebut adalah berdasarkan batas (boundary-based) dan berdasarkan daerah (region-based). Teknik berdasarkan batas (boundary-based) menggambarkan bentuk daerah dengan menggunakan karakteristik eksternal, contohnya adalah piksel sepanjang batas objek. Sedangkan teknik berdasarkan daerah (region-based) menggambarkan bentuk wilayah dengan menggunakan karakteristik internal, contohnya adalah piksel yang berada dalam suatu wilayah.

Fitur bentuk yang biasa digunakan adalah

- Wilayah (area) yang merupakan jumlah piksel dalam wilayah digambarkan oleh bentuk (foreground).
- Lingkar (perimeter) adalah jumlah dari piksel yang berada pada batas dari bentuk. perimeter didapatkan dari hasil deteksi tepi.
- Kekompakan (compactness).
- Euler number atau faktor E adalah perbedaan antara jumlah dari connected component (C) dan jumlah lubang (H) pada citra.

2. Ekstraksi fitur tekstur

Pada ekstraksi fitur ini, fitur pembeda adalah tekstur yang merupakan karakteristik penentu pada citra. Teknik statistik yang terkenal untuk ekstraksi fitur adalah matriks gray level co-occurrence. Teknik tersebut dilakukan dengan melakukan pemindaian untuk mencari jejak derajat keabuan setiap dua buah piksel yang dipisahkan dengan jarak d dan sudut θ yang tetap. Biasanya sudut yang digunakan adalah $0, 45, 90$, dan 135 .

3. Ekstraksi fitur warna

Pada ekstraksi fitur warna, ciri pembeda adalah warna. Biasanya ekstraksi fitur ini digunakan pada citra berwarna yang memiliki komposisi warna RGB (red, green, blue).

B. Sifat bundar

Sifat bundar (circularity) adalah perbandingan antara rerata jarak Euclidean dari sentroid terhadap tepi area dan deviasi standar jarak dari sentroid ke tepi area.

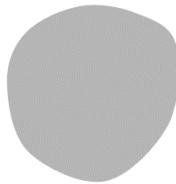
$$c = \frac{\mu_R}{\sigma_R}$$

$$\mu_R = \frac{1}{N} \sum_{i=1}^N |(y_i, x_i) - (\bar{y}_c, \bar{x}_c)|$$

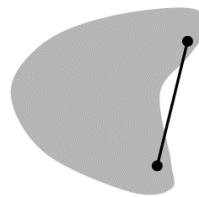
$$\sigma_R = \frac{1}{N} \sum_{i=1}^N [|(y_i, x_i) - (\bar{y}_c, \bar{x}_c)| - \mu_R]^2$$

C. Convex hull dan soliditas

Convex hull dapat diibaratkan seperti karet gelang yang dililitkan pada objek. Suatu objek dikatakan convex jika seluruh titik tepi objek berada pada garis/tepi convex (tidak ada ruang antara lilitan karet dan objek)



Himpunan Konveks



Bukan Himpunan Konveks

Ada 2 fitur yang dapat dihasilkan dari convex hull:

- Konveksitas

$$\text{Konveksitas} = \frac{\text{perimeter konveks}}{\text{perimeter objek}}$$

- Soliditas

$$\text{Soliditas} = \frac{\text{luas objek}}{\text{luas konveks}}$$

Contoh Kode:

```
[in] import matplotlib.pyplot as plt

from skimage.morphology import convex_hull_image
from skimage import data, img_as_float
from skimage.util import invert

# The original image is inverted as the object
# must be white.
image = invert(data.horse())

chull = convex_hull_image(image)

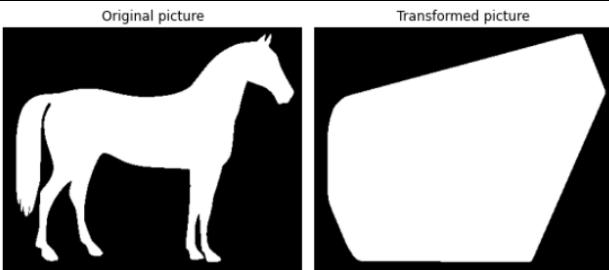
fig, axes = plt.subplots(1, 2, figsize=(8, 4))
ax = axes.ravel()

ax[0].set_title('Original picture')
ax[0].imshow(image, cmap=plt.cm.gray)
ax[0].set_axis_off()

ax[1].set_title('Transformed picture')
ax[1].imshow(chull, cmap=plt.cm.gray)
ax[1].set_axis_off()

plt.tight_layout()
plt.show()
```

[out]

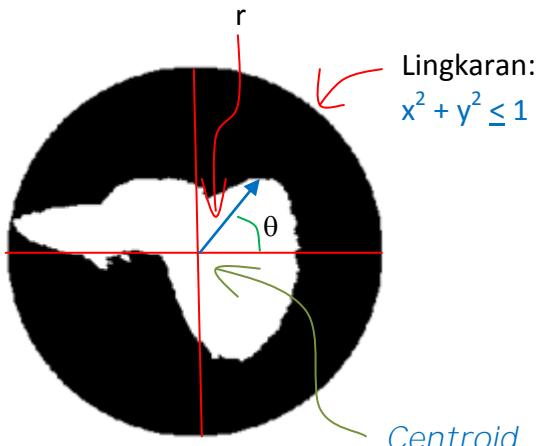


D. Momen invariant

Fitur momen invariant bermanfaat untuk menyatakan objek dengan memperhitungkan area objek. Fitur ini menggunakan dasar momen pusat yang ternormalisasi. Momen yang dihasilkan dapat digunakan untuk menangani translasi, penyekalaan, dan rotasi gambar. Moment invariant yang diusulkan oleh Hu, ada 7 momen.

E. Momen Zernike

Momen Zernike diperkenalkan oleh F. Zernike dalam bukunya berjudul *Physica* yang diterbitkan pada tahun 1934. Penerapan momen Zernike untuk pengolahan citra diperkenalkan pertama kali oleh M.R. Teague pada tahun 1980 (Chen, dkk., 2005). Hasilnya berupa Zernike Moment Descriptors (ZMD). Momen Zernike didasarkan pada polinomial Zernike yang bersifat ortogonal terhadap lingkaran $x^2 + y^2 \leq 1$.



Kelebihan Momen Zernike:

- bersifat independen terhadap pemutaran (rotasi);
- andal terhadap derau dan variasi minor dalam bentuk objek;
- memiliki redundansi informasi yang minimum.

Kelemahan Momen Zernike:

- perlu normalisasi ruang koordinat (dalam hal ini, harus dilakukan pengubahan ke bentuk lingkaran $x^2 + y^2 < 1$).
- perlu penggunaan hampiran penjumlahan mengingat aslinya menggunakan integral. Hal ini berkontribusi dalam memberikan kesalahan numerik, yang memberikan pengaruh terhadap sifat ketidakbergantungan pada rotasi.

- perlu dilakukan normalisasi terhadap translasi dan penyekalaan mengingat momen Zernike tidak bebas dari penggeseran dan penyekalaan.

Latihan:

1. Menggunakan **skeletonize(image)** Skeletonisasi mengurangi objek biner menjadi representasi lebar 1 piksel. Fungsi ini dapat berguna untuk ekstraksi fitur, dan/atau mewakili topologi objek. Skeletonize bekerja dengan membuat lintasan gambar yang berurutan pada setiap lintasan, piksel batas diidentifikasi dan dihapus dengan syarat bahwa piksel tersebut tidak memutus koneksiitas objek terkait.

```
[in]   from skimage.morphology import skeletonize
        from skimage import data
        import matplotlib.pyplot as plt
        from skimage.util import invert

        # Invert the horse image
        image = invert(data.horse())

        # perform skeletonization
        skeleton = skeletonize(image)

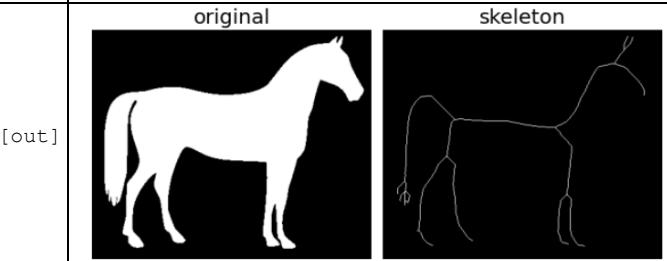
        # display results
        fig, axes = plt.subplots(nrows=1, ncols=2,
                                figsize=(8, 4),
                                sharex=True,
                                sharey=True)

        ax = axes.ravel()

        ax[0].imshow(image, cmap=plt.cm.gray)
        ax[0].axis('off')
        ax[0].set_title('original', fontsize=20)

        ax[1].imshow(skeleton, cmap=plt.cm.gray)
        ax[1].axis('off')
        ax[1].set_title('skeleton', fontsize=20)

        fig.tight_layout()
        plt.show()
```



2. active contour

```
import numpy as np
import matplotlib.pyplot as plt
from skimage.color import rgb2gray
from skimage import data
from skimage.filters import gaussian
from skimage.segmentation import
active_contour

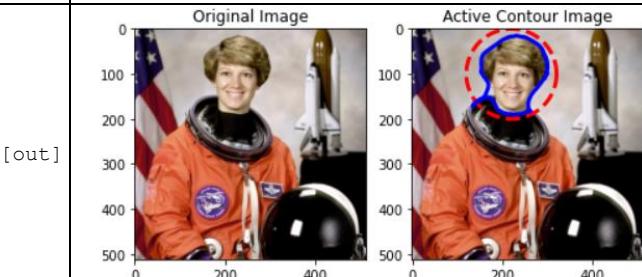
img = data.astronaut()

# Data for circular boundary
s = np.linspace(0, 2*np.pi, 400)
x = 220 + 100*np.cos(s)
y = 100 + 100*np.sin(s)
init = np.array([x, y]).T

[in]

# formation of the active contour
cntr = active_contour(gaussian(img, 3), init,
alpha=0.015, beta=10, gamma=0.001)
fig, ax = plt.subplots(1, 2, figsize=(7, 7))
ax[0].imshow(img, cmap=plt.cm.gray)
ax[0].set_title("Original Image")

ax[1].imshow(img, cmap=plt.cm.gray)
# circular boundary
ax[1].plot(init[:, 0], init[:, 1], '--r',
lw=3)
ax[1].plot(cntr[:, 0], cntr[:, 1], '-b', lw=3)
ax[1].set_title("Active Contour Image")
```



3. Deteksi Wajah

```
[in] import io
import zipfile
import requests
import numpy as np
import cv2
import matplotlib.pyplot as plt
%matplotlib inline

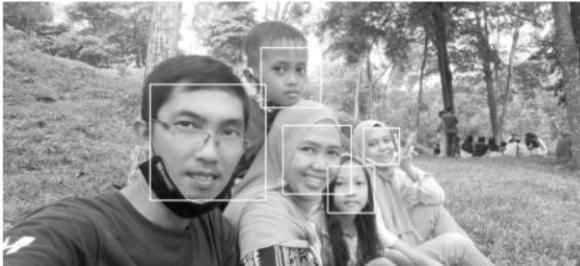
img = cv2.imread('keluarga.jpg')

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

path =
'data/haarcascade_frontalface_default.xml'
face_cascade = cv2.CascadeClassifier(path)

for x, y, w, h in
face_cascade.detectMultiScale(
    gray, 1.3):
    cv2.rectangle(
        gray, (x, y), (x + w, y + h), (255, 0,
0), 2)
fig, ax = plt.subplots(1, 1, figsize=(8, 6))
ax.imshow(gray, cmap=plt.cm.gray)
ax.set_axis_off()
```

[out]



BAB 12

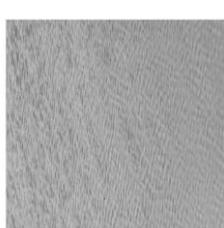
EKSTRAKSI FITUR KONTUR DAN BENTUK

A. Fitur tekstur

Tekstur adalah ukuran dan susunan (jaringan) bagian suatu benda dikutip dari kamus besar Bahasa Indonesia. Menurut Kulkarni tekstur adalah Hubungan mutual antara nilai intensitas piksel-piksel yang bertetangga yang berulang di suatu area yang lebih luas daripada jarak hubungan tersebut.

Contoh penggunaan fitur tekstur citra:

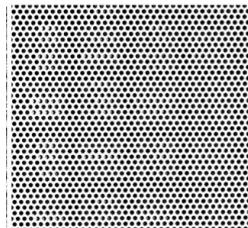
Inspeksi secara otomatis pada industri tekstil, pengecatan mobil, pemakaian karpet. Analisis citra medis. Misalnya, tekstur digunakan untuk klasifikasi penyakit paru-paru, diagnosis leukemia, dan pembedaan tipe-tipe sel darah putih. Analisis penginderaan jarak-jauh. Misalnya, tekstur dipakai untuk kepentingan klasifikasi area tanah.



Halus



Kasar



Teratur

1. Fitur Tekstur Citra

Keteraturan:

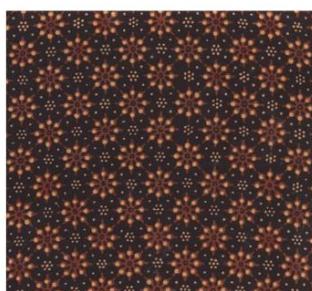
- Teratur atau buatan
- Tidak teratur atau alamiah

Kekasaran:

- Mikrotekstur
- Makrotekstur

Matematis:

- Stokastis (statistik) atau derau
- Regular (struktural) atau warna/intensitas



Teratur atau buatan



Tidak teratur atau
alamiah



Mikrotekstur



Makrotekstur

2. Metode Ekstrasi Ciri Tekstur

metode statistis : statistik citra/objek

- GLCM
- Tamura

metode struktural : struktur pembentuk citra/objek

- Shape Grammar

metode spectral : struktur frekuensi-spasial citra/objek

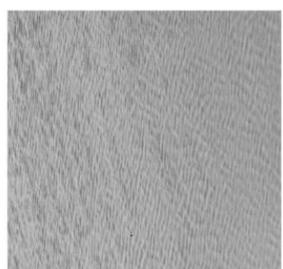
- Fourier

- Gabor

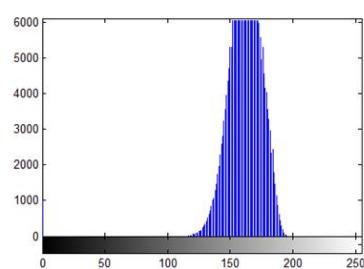
- Filter Laws

B. Tekstur berbasis histogram

Ternyata histogram citra dapat mencirikan tekstur citra,
Perhatikan gambar di bawah ini:



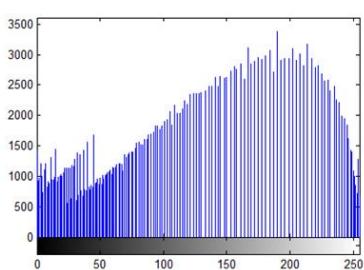
(a) Tekstur halus



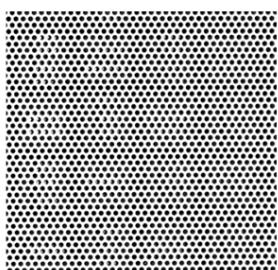
(b) Histogram tekstur halus



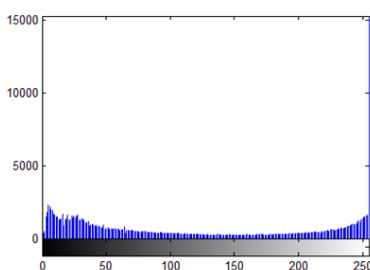
(c) Tekstur kasar



(d) Histogram tekstur kasar



(e) Tekstur periodik



(f) Histogram tekstur periodik

1. Fitur Tekstur Berbasis Histogram Citra

Rerata intensitas (μ) → orde 1

$$m = \sum_{i=0}^{L-1} i \cdot p(i)$$

Deviasi standar (σ) / varians → momen orde 2
mencerminkan kekontrasan citra

$$\sigma = \sqrt{\sum_{i=1}^{L-1} (i - m)^2 p(i)}$$

Skewness → momen orde 3

ukuran ketidaksimetrisan terhadap rerata intensitas

- Nilai negatif → kecerahan condong ke kiri
- Nilai positif → kecerahan condong ke kanan

$$skewness = \sum_{i=1}^{L-1} (i - m)^3 p(i)$$

2. Fitur Tekstur Berbasis Histogram Citra

Energi (keseragaman)

- Distribusi intensitas piksel terhadap jangkauan aras keabuan
- Nilai energy semakin mendekati 1 → semakin seragam intensitasnya.

Entropi (kompleksitas citra)

- Semakin tinggi nilai entropi, semakin kompleks citra tersebut
- Entropi berlawanan dengan energy

$$entropi = - \sum_{i=0}^{L-1} p(i) \log_2(p(i))$$

Kehalusan (R)

- Ukuran tingkat kehalusan/kekasaran intensitas citra
- R rendah → kasar

$$R = 1 - \frac{1}{1 + \sigma^2}$$

C. Gray Level Co-occurrence Matrices (GLCM)

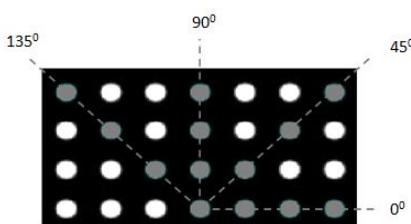
GLCM pertama kali diusulkan oleh Haralick (1973) yang terdiri dari 28 fitur untuk menjelaskan pola spasial (Kulkarni, 1994)

GLCM menggunakan perhitungan tekstur pada orde kedua. Pengukuran tekstur pada orde pertama menggunakan perhitungan statistika didasarkan pada nilai piksel citra asli.

GLCM terdiri dari 4

arah:

1. 00°
2. 450°
3. 900°
4. 1350°



Penentuan awal matriks GLCM:

(a) Citra asli

	0	1	2	3
0	[0, 0]	1	1	
1	0, 0	1	1	
2	0, 2	2	2	2
3	2, 2	[3, 3]	3	3

(b) Komposisi Piksel

	0	1	2	3
0	0,0	0,1	0,2	0,3
1	1,0	1,1	1,2	1,3
2	2,0	2,1	2,2	2,3
3	3,0	3,1	3,2	3,3

(c) Jumlah pasangan piksel

	0	1	2	3
0	2	2	1	0
1	0	2	0	0
2	0	0	3	1
3	0	0	0	1

Komposisi piksel 0 dengan 0
Komposisi piksel 3 dengan 3

Pembentukan matriks simetris

$$\begin{bmatrix} 2 & 2 & 1 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 2 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 \\ 1 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 2 & 1 & 0 \\ 2 & 4 & 0 & 0 \\ 1 & 0 & 6 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix}$$

Transpos

GLCM sebelum
dinormalisasi

Normalisasi

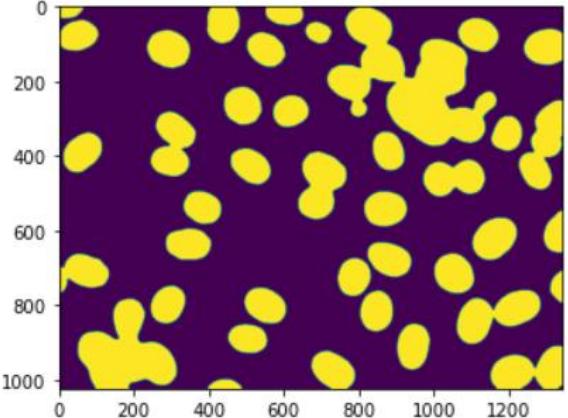
$$\begin{bmatrix} \frac{4}{24} & \frac{2}{24} & \frac{1}{24} & \frac{0}{24} \\ \frac{2}{24} & \frac{4}{24} & \frac{0}{24} & \frac{0}{24} \\ \frac{1}{24} & \frac{0}{24} & \frac{6}{24} & \frac{1}{24} \\ \frac{0}{24} & \frac{0}{24} & \frac{1}{24} & \frac{2}{24} \end{bmatrix}$$

GLCM terdiri dari 28 fitur, namun yang sering digunakan hanya 5 (fitur):

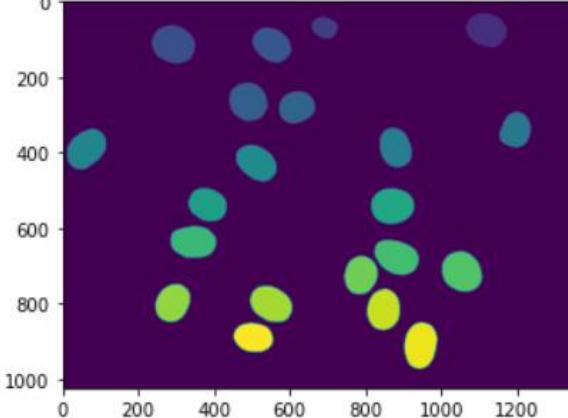
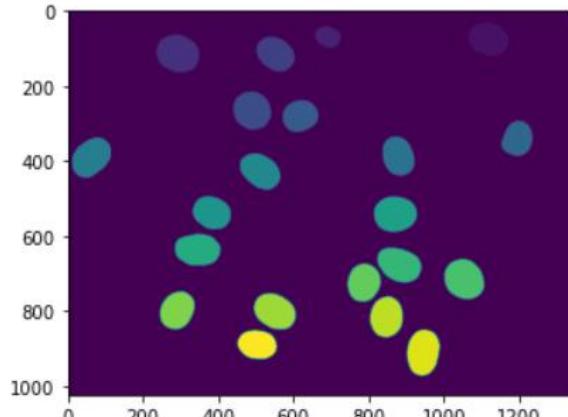
1. ASM → ukuran homogenitas citra
2. Contrast → ukuran keberadaan variasi aras keabuan piksel citra
3. Inverse Different Moment (IDM) → untuk mengukur homogenitas
4. Entropi → ukuran ketidakteraturan aras keabuan di dalam citra
5. Korelasi → ukuran ketergantungan linear antarnilai aras keabuan dalam citra

Latihan:

1. Penyaringan dan Pelabelan ulang

[in]	<pre>import mahotas as mh import mahotas.demos import numpy as np from pylab import imshow, show f = mh.demos.nuclear_image() f = f[:, :, 0] imshow(f) show()</pre>
[out]	
[in]	<pre>f = mh.gaussian_filter(f, 4) f = (f > f.mean())</pre>
[in]	<pre>f = mh.gaussian_filter(f, 4) f = (f > f.mean()) imshow(f) show()</pre>
[in]	<pre>labeled, n_nucleus = mh.label(f) print('Found {} nuclei.'.format(n_nucleus))</pre>
[out]	Found 40 nuclei.
[in]	<pre>labeled, n_nucleus = mh.label(f) print('Found {} nuclei.'.format(n_nucleus)) imshow(labeled) show()</pre>
[out]	Found 40 nuclei.

[in]	<pre>sizes = mh.labeled.labeled_size(labeled) too_big = np.where(sizes > 10000) labeled = mh.labeled.remove_regions(labeled, too_big)</pre>
[in]	<pre>sizes = mh.labeled.labeled_size(labeled) too_big = np.where(sizes > 10000) labeled = mh.labeled.remove_regions(labeled, too_big) imshow(labeled) show()</pre>
[out]	
[in]	<pre>labeled = mh.labeled.remove_bordering(labeled)</pre>
[in]	<pre>labeled = mh.labeled.remove_bordering(labeled) imshow(labeled) show()</pre>

	[out]	
	[in]	<pre> relabeled, n_left = mh.labeled.relabel(labeled) print('After filtering and relabeling, there are {} nuclei left'.format(n_left)) </pre>
	[out]	After filtering and relabeling, there are 21 nuclei left.
	[in]	<pre> relabeled, n_left = mh.labeled.relabel(labeled) print('After filtering and relabeling, there are {} nuclei left'.format(n_left)) imshow(relabeled) show() </pre>
	[out]	After filtering and relabeling, there are 21 nuclei left. 

2. berikut ini adalah contoh untuk Filter ridge, Filter ridge yang berbeda mungkin cocok untuk mendeteksi struktur yang berbeda, misalnya, tergantung pada kontras atau noise level.

Kelas filter ridge saat ini bergantung pada nilai eigen dari matriks Hessian intensitas gambar untuk mendeteksi struktur ridge di mana intensitas berubah tegak lurus tetapi tidak sepanjang struktur. Yang perlu diperhatikan bahwa, efek tepi, hasil untuk filter Meijering dan Frangi dipangkas sebesar 4 piksel di setiap tepi untuk mendapatkan rendering yang tepat.

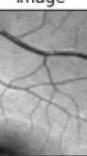
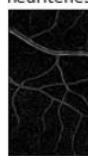
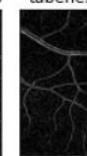
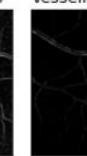
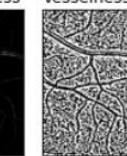
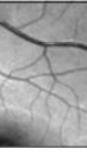
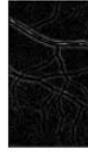
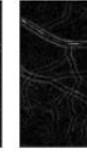
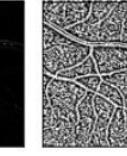
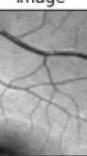
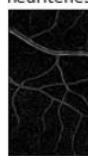
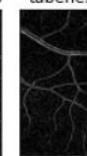
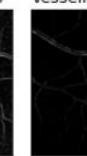
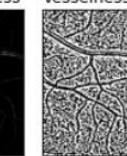
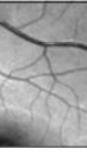
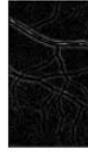
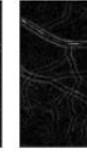
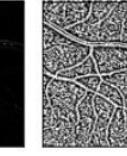
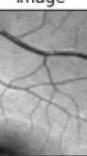
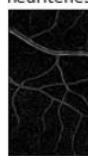
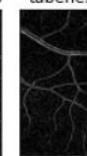
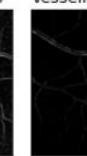
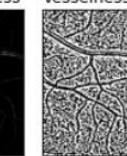
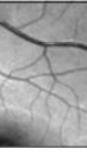
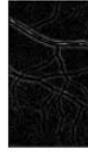
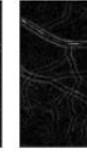
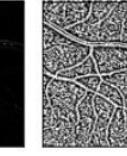
```
[in]  from skimage import data
from skimage import color
from skimage.filters import meijering, sato,
frangi, hessian
import matplotlib.pyplot as plt

def identity(image, **kwargs):
    """Return the original image, ignoring any
    kwargs."""
    return image

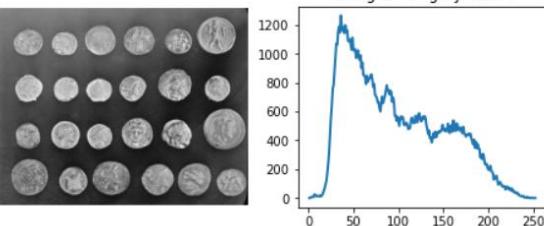
image = color.rgb2gray(data.retina())[300:700,
700:900]
cmap = plt.cm.gray

kwargs = {'sigmas': [1], 'mode': 'reflect'}

fig, axes = plt.subplots(2, 5)
for i, black_ridges in enumerate([1, 0]):
    for j, func in enumerate([identity,
meijering, sato, frangi, hessian]):
        kwargs['black_ridges'] = black_ridges
        result = func(image, **kwargs)
        axes[i, j].imshow(result, cmap=cmap,
aspect='auto')
        if i == 0:
            axes[i,
j].set_title(['Original\nimage',
'Meijering\nneuriteness',
'Sato\nntubeness', 'Frangi\nvesselness',
'Hessian\nvesselness'][j])
            if j == 0:
                axes[i,
j].set_ylabel('black ridges = ' +
```

	<pre> str(bool(black_ridges)) axes[i, j].set_xticks([]) axes[i, j].set_yticks([]) plt.tight_layout() plt.show() </pre>															
[out]	<table style="width: 100%; text-align: center;"> <thead> <tr> <th>Original image</th> <th>Meijering neuriteness</th> <th>Sato tubeness</th> <th>Frangi vesselness</th> <th>Hessian vesselness</th> </tr> </thead> <tbody> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>	Original image	Meijering neuriteness	Sato tubeness	Frangi vesselness	Hessian vesselness										
Original image	Meijering neuriteness	Sato tubeness	Frangi vesselness	Hessian vesselness												
																
																

3. Berikut ini merupakan contoh kode untuk segmentasi dan pelabelan

[in]	<pre> import numpy as np import matplotlib.pyplot as plt from skimage import data from skimage.exposure import histogram coins = data.coins() hist, hist_centers = histogram(coins) fig, axes = plt.subplots(1, 2, figsize=(8, 3)) axes[0].imshow(coins, cmap=plt.cm.gray) axes[0].axis('off') axes[1].plot(hist_centers, hist, lw=2) axes[1].set_title('histogram of gray values') </pre>
[out]	<p>Text(0.5, 1.0, 'histogram of gray values')</p> 
[in]	<pre> fig, axes = plt.subplots(1, 2, figsize=(8, 3), sharey=True) axes[0].imshow(coins > 100, cmap=plt.cm.gray) </pre>

```

axes[0].set_title('coins > 100')

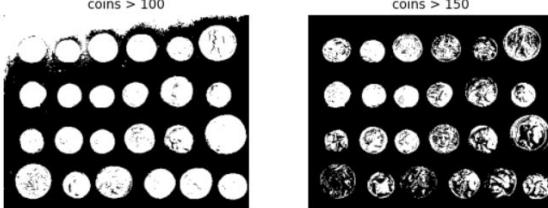
axes[1].imshow(coins > 150, cmap=plt.cm.gray)
axes[1].set_title('coins > 150')

for a in axes:
    a.axis('off')

plt.tight_layout()

```

[out]



[in]

```

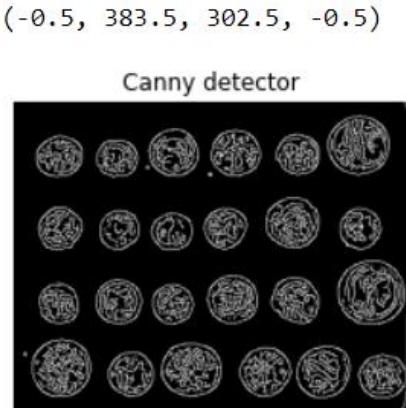
from skimage.feature import canny

edges = canny(coins)

fig, ax = plt.subplots(figsize=(4, 3))
ax.imshow(edges, cmap=plt.cm.gray)
ax.set_title('Canny detector')
ax.axis('off')

```

[out]



[in]

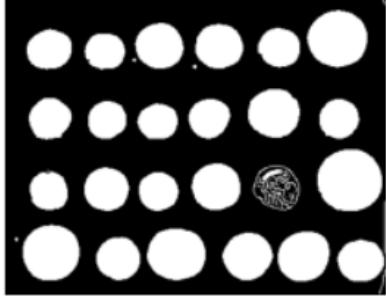
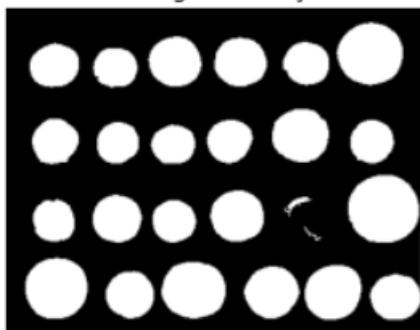
```

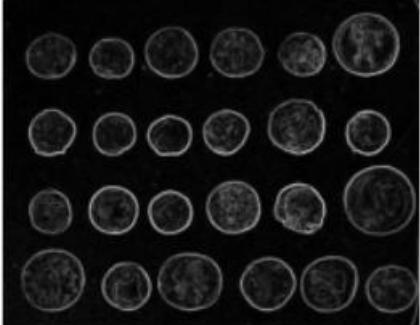
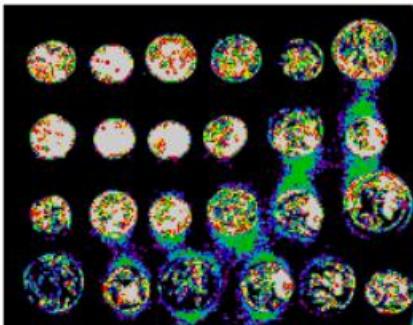
from scipy import ndimage as ndi

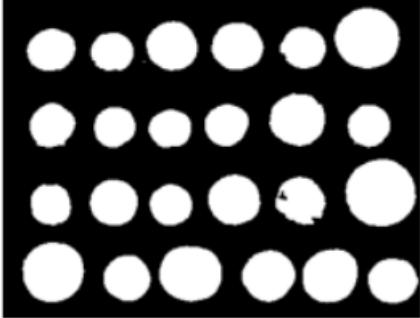
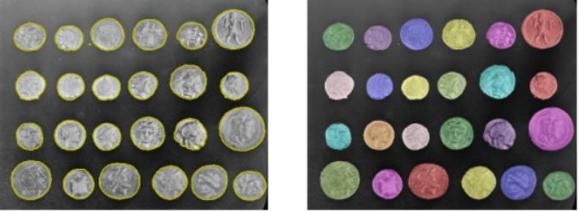
fill_coins = ndi.binary_fill_holes(edges)

fig, ax = plt.subplots(figsize=(4, 3))
ax.imshow(fill_coins, cmap=plt.cm.gray)
ax.set_title('filling the holes')
ax.axis('off')

```

[out]	<p>(-0.5, 383.5, 302.5, -0.5)</p> <p>filling the holes</p> 
[in]	<pre>from skimage import morphology coins_cleaned = morphology.remove_small_objects(fill_coins, 21) fig, ax = plt.subplots(figsize=(4, 3)) ax.imshow(coins_cleaned, cmap=plt.cm.gray) ax.set_title('removing small objects') ax.axis('off')</pre>
[out]	<p>(-0.5, 383.5, 302.5, -0.5)</p> <p>removing small objects</p> 
[in]	<pre>from skimage.filters import sobel elevation_map = sobel(coins) fig, ax = plt.subplots(figsize=(4, 3)) ax.imshow(elevation_map, cmap=plt.cm.gray) ax.set_title('elevation map') ax.axis('off')</pre>

	(-0.5, 383.5, 302.5, -0.5)
[out]	<p style="text-align: center;">elevation map</p> 
[in]	<pre>markers = np.zeros_like(coins) markers[coins < 30] = 1 markers[coins > 150] = 2</pre> <pre>fig, ax = plt.subplots(figsize=(4, 3)) ax.imshow(markers, cmap=plt.cm.nipy_spectral) ax.set_title('markers') ax.axis('off')</pre>
	(-0.5, 383.5, 302.5, -0.5)
[out]	<p style="text-align: center;">markers</p> 
[in]	<pre>from skimage import segmentation</pre> <pre>segmentation_coins = segmentation.watershed(elevation_map, markers)</pre> <pre>fig, ax = plt.subplots(figsize=(4, 3)) ax.imshow(segmentation_coins, cmap=plt.cm.gray) ax.set_title('segmentation') ax.axis('off')</pre>

[out]	<pre>(-0.5, 383.5, 302.5, -0.5)</pre> <p style="text-align: center;">segmentation</p> 
[in]	<pre>from skimage.color import label2rgb segmentation_coins = ndi.binary_fill_holes(segmentation_coins - 1) labeled_coins, _ = ndi.label(segmentation_coins) image_label_overlay = label2rgb(labeled_coins, image=coins, bg_label=0) fig, axes = plt.subplots(1, 2, figsize=(8, 3), sharey=True) axes[0].imshow(coins, cmap=plt.cm.gray) axes[0].contour(segmentation_coins, [0.5], linewidths=1.2, colors='y') axes[1].imshow(image_label_overlay) for a in axes: a.axis('off') plt.tight_layout() plt.show()</pre>
[out]	

4. Perbandingan algoritma segmentasi dan superpixsel

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

from skimage.data import astronaut
from skimage.color import rgb2gray
from skimage.filters import sobel
from skimage.segmentation import felzenszwalb,
slic, quickshift, watershed
from skimage.segmentation import
mark_boundaries
from skimage.util import img_as_float

img = img_as_float(astronaut()[:, :, ::2])

segments_fz = felzenszwalb(img, scale=100,
sigma=0.5, min_size=50)
segments_slic = slic(img, n_segments=250,
compactness=10, sigma=1,
start_label=1)
segments_quick = quickshift(img,
kernel_size=3, max_dist=6, ratio=0.5)
gradient = sobel(rgb2gray(img))
segments_watershed = watershed(gradient,
markers=250, compactness=0.001)

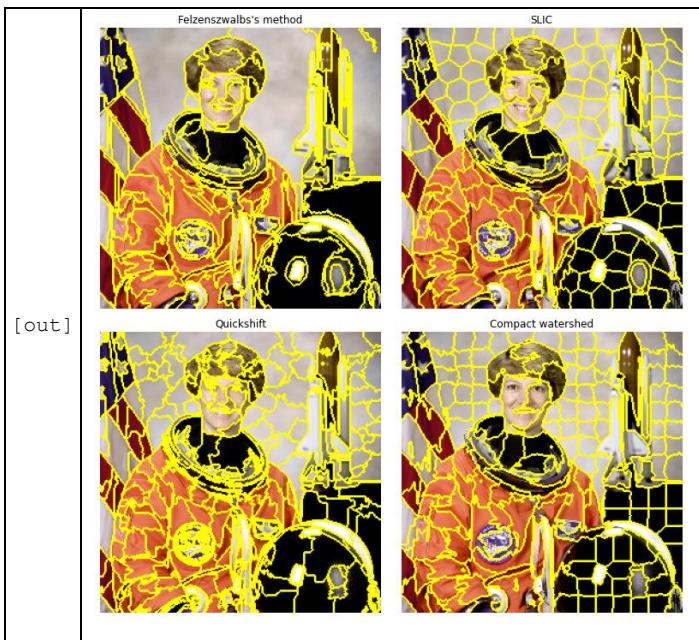
print(f"Felzenszwalb number of segments:
{len(np.unique(segments_fz))}")
print(f"SLIC number of segments:
{len(np.unique(segments_slic))}")
print(f"Quickshift number of segments:
{len(np.unique(segments_quick))}")

fig, ax = plt.subplots(2, 2, figsize=(10, 10),
sharex=True, sharey=True)

ax[0, 0].imshow(mark_boundaries(img,
segments_fz))
ax[0, 0].set_title("Felzenszwalbs's method")
ax[0, 1].imshow(mark_boundaries(img,
segments_slic))
ax[0, 1].set_title('SLIC')
ax[1, 0].imshow(mark_boundaries(img,
segments_quick))
ax[1, 0].set_title('Quickshift')
ax[1, 1].imshow(mark_boundaries(img,
segments_watershed))
ax[1, 1].set_title('Compact watershed')

for a in ax.ravel():
    a.set_axis_off()

plt.tight_layout()
plt.show()
```



Daftar Pustaka

- Ballard, D. H. (1982). CM Brown Computer Vision. NY: *Prentice Hill*.
- Basuki, A. (2005). Metode Numerik dan Algoritma Komputasi.
- Campbell, S., Chancelier, J. P., & Nikoukhah, R. (2006). Modeling and simulation in Scilab/Scicos. Springer, New York. *Modeling and simulation in Scilab/Scicos*. Springer, New York.
- Chan, T., & Vese, L. (1999). An active contour model without edges In: Nielsen M, Johansen P, Olsen OF, Weickert J, editors. Scale-Space Theories in Computer Vision.
- Cohen, R. (2011). The chan-vese algorithm. *arXiv preprint arXiv:1107.2782*.
- Frangi, A. F., Niessen, W. J., Vincken, K. L., & Viergever, M. A. (1998, October). Multiscale vessel enhancement filtering. In *International conference on medical image computing and computer-assisted intervention* (pp. 130-137). Springer, Berlin, Heidelberg.
- Freeman, W. (2008). William Freeman · Pietro Perona · Bernhard Schölkopf. *Int J Comput Vis*, 77, 1.
- Forsyth, D., & Ponce, J. (2011). *Computer vision: A modern approach* (p. 792). Prentice hall.
- Getreuer, P., Segmentation, C. V., & IPOL, J. (2012). Image Process. *Online*, 2, 214-224.
- Goehner, K., & Desell, T. Computer Vision to Aid in Wildlife Surveillance.
- Huang, D. Y., & Wang, C. H. (2009). Optimal multi-level thresholding using a two-stage Otsu optimization approach. *Pattern Recognition Letters*, 30(3), 275-284.
- Huang, T. S. (1996). Computer vision: Evolution and promise. *CERN European Organization for Nuclear Research-Reports-CERN*, 21-26.
- Jahne, B. (Ed.). (2000). *Computer vision and applications: a guide for students and practitioners*. Elsevier.

- Jianzhuang, L., Wenqing, L., & Yupeng, T. (1991, June). Automatic thresholding of gray-level pictures using two-dimension Otsu method. In *China., 1991 international conference on circuits and systems* (pp. 325-327). IEEE.
- Kass, M., Witkin, A., & Terzopoulos, D. (1988). Snakes: Active Contour Models, *Int'l J. Computer Vision*, 1, 312-331.
- Kanade, T. (2012). *Three-dimensional machine vision* (Vol. 21). Springer Science & Business Media.
- Kittler, J., & Illingworth, J. (1985). On threshold selection using clustering criteria. *IEEE transactions on systems, man, and cybernetics*, (5), 652-655.
- Lee, S. U., Chung, S. Y., & Park, R. H. (1990). A comparative performance study of several global thresholding techniques for segmentation. *Computer Vision, Graphics, and Image Processing*, 52(2), 171-190.
- Liao, P. S., Chen, T. S., & Chung, P. C. (2001). A fast algorithm for multilevel thresholding. *J. Inf. Sci. Eng.*, 17(5), 713-727.
- Marleny, F. D. (2019). OPTIMASI GENETIC ALGORITHM DENGAN JARINGAN SYARAF TIRUAN UNTUK KLASIFIKASI CITRA. *Jurnal Teknologi Informasi Universitas Lambung Mangkurat (JTIULM)*, 4(1), 1-6.
- Meijering, E., Jacob, M., & Sarria, J. C. (2004). F., Steiner P., Hirling H., and Unser M. *Design and validation of a tool for neurite tracing and analysis in fluorescence microscopy images. Cytometry A*, 58, 167.
- Ng, C. C., Yap, M. H., Costen, N., & Li, B. (2014, November). Automatic wrinkle detection using hybrid hessian filter. In *Asian Conference on Computer Vision* (pp. 609-622). Springer, Cham.
- Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1), 62-66.
- Putra, D. (2010). *Pengolahan citra digital*. Penerbit Andi.
- Priyanto, H. (2017). Pengolahan CItra Digital Teori dan Aplikasi Nyata. Bandung: *Informatika*.

- Rossant, C. (2018). *IPython Interactive Computing and Visualization Cookbook: Over 100 hands-on recipes to sharpen your skills in high-performance numerical computing and data science in the Jupyter Notebook*. Packt Publishing Ltd.
- Sari, Y., Marleny, F. D., Ansari, R., Izzana, M., Ricardus, A. P., & Lareno, B. (2015). Optimasi Conjugate Gradient Pada Backpropagation Neural Network untuk Deteksi Kualitas Daun Tembakau. *Proceedings Konferensi Nasional Sistem dan Informatika (KNS&I)*.
- Sato, Y., Nakajima, S., Shiraga, N., Atsumi, H., Yoshida, S., Koller, T., ... & Kikinis, R. (1998). Three-dimensional multi-scale line filter for segmentation and visualization of curvilinear structures in medical images. *Medical image analysis*, 2(2), 143-168.
- _____, Nakajima, S., Atsumi, H., Koller, T., Gerig, G., Yoshida, S., & Kikinis, R. (1997, March). 3D multi-scale line filter for segmentation and visualization of curvilinear structures in medical images. In *CVRMed-MRCAS'97* (pp. 213-222). Springer, Berlin, Heidelberg.
- Siahaan, V., & Sianipar, R. H. (2020). *Pemrograman MATLAB Untuk Komputasi Numerik dan Pengolahan Sinyal Digital*. SPARTA PUBLISHING.
- Sonka, M., Hlavac, V., & Boyle, R. (2014). *Image processing, analysis, and machine vision*. Cengage Learning.
- Sutojo, S. (2004). Membangun citra perusahaan. Jakarta: Damar Mulia Pustaka, 160.
- Sutoyo, T. D., Mulyanto, E., Suhartono, V., & Nurhayati, O. D. (2009). Teori pengolahan citra digital. Yogyakarta: Andi.
- Szeliski, R. (2010). *Computer vision: algorithms and applications*. Springer Science & Business Media.

Website

<https://britishmachinevisionassociation.github.io/>
<http://www.kitainformatika.com/2016/02/operasi-ketetanggaan-piksel-pada-filter.html>
<https://numpy.org/devdocs/user/quickstart.html>
https://scikit-image.org/docs/dev/auto_examples/
<https://achmatim.net/pengolahan-citra-digital-image-processing/>
<https://matplotlib.org/stable/tutorials/index.html>
<https://matplotlib.org/stable/tutorials/introductory/images.html#sphx-glr-tutorials-introductory-images-py>
https://d2l.ai/chapter_computer-vision/anchor.html
<https://learnopencv.com/edge-detection-using-opencv/#how-are-edges-detected>
<https://blog.docotel.com/computer-vision-apa-dan-mengapa-itu-penting/>
towardsdatascience.com
https://id.wikipedia.org/wiki/Pengolahan_citra_digital
https://id.wikipedia.org/wiki/Penglihatan_komputer
<https://myteks.wordpress.com/2011/02/19/pengetahuan-pola-pattern-recognition/>
<https://www.kajianpuptaka.com/2016/04/pengolahan-citra-digital.html>
<https://myteks.wordpress.com/2012/09/13/ppt-materi-kuliah-pcd/>
<https://dosen.itats.ac.id/farida/2020/02/10/mengenal-pengolahan-citra-digital-menggunakan-python-yuuukkk/>
<http://wahyunur.blog.um.ac.id/2021/02/24/instalasi-opencv-pada-jupyter-anaconda/>
<https://jupyter.brynmawr.edu/services/public/dblank/CS110%20Intro%20to%20Computing/2015-Spring/Lectures/Image%20Processing.ipynb>
<https://ipython-books.github.io/113-segmenting-an-image/>
<https://neptune.ai/blog/image-processing-techniques-you-can-use-in-machine-learning>

https://scikit-image.org/docs/dev/auto_examples/applications/plot_color_segmentation.html

<https://simpleitk.readthedocs.io/en/master/filters.html>

<https://neptune.ai/blog/image-processing-python-libraries-for-machine-learning>

<https://pillow.readthedocs.io/en/stable/reference/ImageFilter.html>

<https://pillow.readthedocs.io/en/stable/reference/Image.html>

<http://insightsoftwareconsortium.github.io/SimpleITK-Notebooks/>

<https://neptune.ai/blog/image-processing-in-python-algorithms-tools-and-methods-you-should-know>

<https://mahotas.readthedocs.io/en/latest/distance.html>

https://docs.opencv.org/master/d9/df8/tutorial_root.html

https://colab.research.google.com/github/danylaksono/OpenCV-PCD/blob/main/Minggu%202_%20Dasar%20Pengolahan%20Citra%20dengan%20OpenCV.ipynb#scrollTo=DMTUbKf4NUM9

https://en.wikipedia.org/wiki/Image_retrieval

<https://id.wikipedia.org/wiki/MATLAB>

<https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/>

https://www.mathworks.com/help/matlab/creating_guis/what-is-an-app.html

<https://id.wikipedia.org/wiki/Scilab>

<https://www.gnu.org/software/octave/about>

<https://docs.anaconda.com/anaconda/navigator/overview/>

<https://jupyter.org/>

INDEKS

A

Adaptive, vi, 141, 147
Anaconda Navigator, iv, 14,
16, 17, 18
Artificial Inteligent, 1

C

CIELAB, 126, 138
CMYK, vi, 32, 33, 124, 125,
138
Computer vision, iii, 1, 3, 32,
175, 177
Convex, vii, 38, 151

D

Dilasi, 111

E

Ekstraksi, vii, 60, 149, 150
Erosi, vi, 102, 111

G

Geometrik, v, 87
GNU Octave, iv, 16
Grayscale, iv, 9, 33, 49, 54

H

Histogram, v, 56, 57, 58, 59,
65, 161
HSL, 125
HSV, 84, 125, 126

I

Interpolasi, v, 89, 97, 98, 99,
110

J

Jupyter Nootebook, 14, 16

M

Mahotas, iv, 38, 46
Matlab, iv, 14, 15, 16
Mean, v, 70
Median, v, 71, 138

O

OpenCV, iv, 34, 35, 45, 46, 84,
179

P

Pgmagick, iv, 39

S

Scikit-Image, iv, 35, 46
Scilab, iv, 14, 15, 16, 175, 179
Scipy, 36, 45, 46, 77
Skeletonize, 155

Y

YIQ, vi, 123, 125, 138

Z

Zernike, vii, 38, 153, 154

Kunci Jawaban Soal:

Bab 1	Bab 2	Bab 3	Bab 4	Bab 5
1. A	1. A	1. C	1. D	1. C
2. D	2. E	2. D	2. A	2. D
3. C	3. C	3. D	3. E	3. B
4. D	4. D	4. B	4. B	4. E
5. E	5. E	5. E	5. D	5. D

Bab 6	Bab 7	Bab 8	Bab 9	Bab 10
1. A	1. C	1. A	1. B	1. D
2. B	2. D	2. E	2. D	2. B
3. B	3. D	3. B	3. B	3. A
4. A	4. A	4. D	4. B	4. A
5. B	5. C	5. B	5. E	5. A

PROFIL PENULIS



Finki Dona Marleny lahir di Kijang (*kepulauan Riau*). Ketika Masih Kecil ia tinggal di Muara Enim Sumatera selatan dan melanjutkan sekolah dasar di Talawi, Sawahlunto Sumatera Barat hingga kelas 3 SD. Kemudian pindah ke kota kelahirannya di Kijang Pulau Bintan Kepulauan Riau dan menamatkan Pendidikan SD serta melanjutkan sekolah ke tingkat SMP hingga kelas 1. Pada tahun 2002 ia pindah kembali ke Kabupaten Balangan Kalimantan Selatan dan pada tahun 2006 melanjutkan studi di STMIK INDONESIA Banjarmasin Jurusan Sistem Informasi. Kemudian Ia mendapatkan beasiswa di salah satu Yayasan dan melanjutkan studi S2 di jurusan Teknik Informatika UDINUS Semarang lulus pada tahun 2012. Sampai sekarang masih aktif sebagai *Content Creator,blogger* dan Dosen di salah satu Universitas Swasta di kota Banjarmasin.