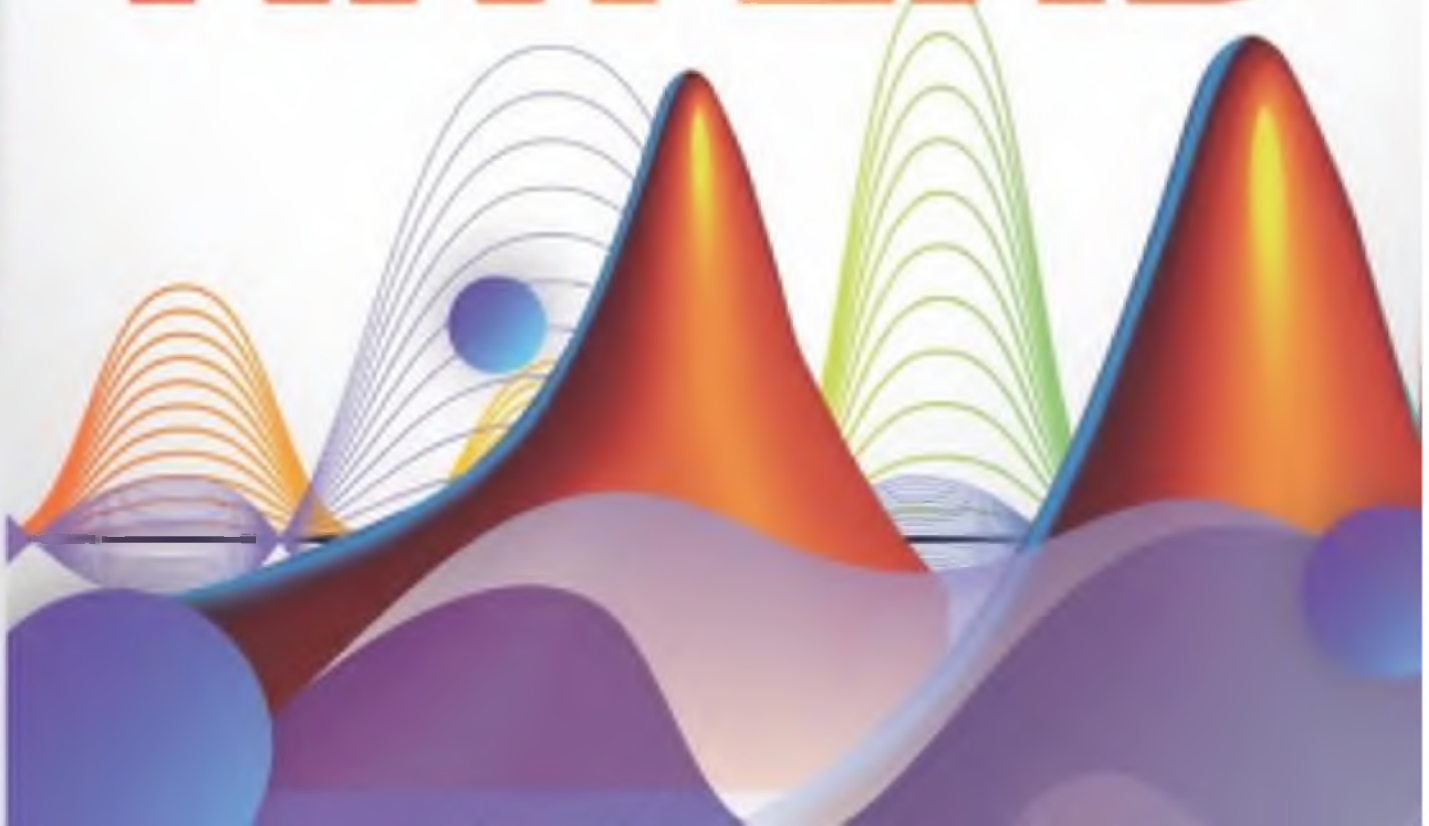
Ricardus Anggi Pramunendar | Pulung Nurtantio Andono
Moch. Arief Soeleman | Dwi Puji Prabowo | Dewi Pergiwati

PENGENALAN BERBASIS CITRA DUA DIMENSI

MENGGUNAKAN

MATLAB



PENGENALAN BERBASIS CITRA DUA DIMENSI MENGGUNAKAN MATLAB

PENGENALAN BERBASIS CITRA DUA DIMENSI MENGGUNAKAN MATLAB

Ricardus Anggi Pramunendar | Pulung Nurtantio Andono
Moch. Arief Soeleman | Dwi Puji Prabowo | Dewi Pergiwati

PENGENALAN BERBASIS CITRA DUA DIMENSI MENGGUNAKAN MATLAB

Istana Agency

PENGENALAN BERBASIS CITRA DUA DIMENSI MENGGUNAKAN MATLAB

Disusun oleh:

Ricardus Anggi Pramunendar, Pulung Nurtantio Andono, Moch. Arief Soeleman
Dwi Puji Prabowo, Dewi Pergiwati

Editor : Maulana Aenul Yaqin
Tata letak & cover : Ahmad Bahaudin

Cetakan I, Januari 2020
15,5 x 23 cm; xviii + 154 Halaman
ISBN 978-623-7313-32-8

Diterbitkan oleh:
CV. ISTANA AGENCY
Istana Publishing
Jl. Nyi Adi Sari Gg. Dahlia I, Pilahan KG.I/722 RT 39/12
Rejowinangun-Kotagede-Yogyakarta
085100523476 | whatsapp 0857-2902-2165
istanaagency09@gmail.com | percetakanistana09@gmail.com
istanaagency | istanaagency | www.istanaagency.com

PRAKATA

Buku ini merupakan salah satu luaran hasil Penelitian serta merupakan tugas dan tanggungjawab dosen dalam Tridharma Perguruan Tinggi. Dalam buku ini terdapat kompilasi materi yang sangat relevan saat ini bagi siswa dalam ilmu pencitraan, teknologi pencitraan, pemahaman citra, dan, pemrosesan citra. Beberapa bentuk komputasi dari citra adalah kata kunci yang digunakan dan muncul pertama kali. Anggap bahwa akan ada dan pada kenyataannya pemrosesan setelah pendeteksian informasi dari data mentah yang dikumpulkan oleh sistem sensor. Sensor dapat menampilkan sesuatu yang sedikit atau tidak masuk akal bagi pengamat. Namun pada akhirnya, dalam sistem produksi secara komersial, pemrosesan citra diimplementasikan oleh perangkat keras khusus. Sedangkan dalam tahap penelitian serta pengembangan sistem pencitraan, setiap prosesnya hampir pasti akan diimplementasikan menggunakan sebuah alat yang mendukung komputasi secara matematika. MATLAB merupakan salah satu alat untuk mendukung hal tersebut. Buku ini membahas fakta dengan menghubungkan secara langsung ke MATLAB melalui banyak operasi yang dijelaskan. Demikian buku ini disusun, semoga dapat bermanfaat bagi ber bagai pihak khususnya bagi perkembangan ilmu pengetahuan. Kami menyadari bahwa buku ini masih terdapat banyak kekurangan. Oleh karena itu,

kritik dan saran yang membangun tetap kami harapkan untuk lebih menyempurnakan buku ini.

KATA PENGANTAR

Prospek menggunakan komputer untuk meniru beberapa atribut dari sistem visual manusia telah menarik minat para ilmuwan, insinyur, dan ahli matematika lebih dari beberapa tahun yang lalu, menjadikan bidang pengolahan citra sebagai salah satu cabang penelitian ilmu komputer terapan yang paling cepat berkembang. Dalam beberapa tahun terakhir, bidang pemrosesan citra mengalami pertumbuhan luar biasa dan menjadi populer serta mudah diakses. Pertumbuhan ini telah didorong oleh beberapa factor seperti perangkat keras yang tersedia secara luas, mudah dan murah; berbagai alat perangkat lunak untuk mengolah, memanipulasi, dan memproses gambar dan video; mempopulerkan Web melalui informasi visual, membuat revolusi dalam fotografi yang menjadikan kamera berbasis film semuanya sudah usang; kemajuan dalam industri film; dan perubahan secara inovatif dalam cara kita melihat, merekam dalam berbagai program TV maupun video.

Buku ini sebagai pengantar praktis untuk topik paling penting dalam pemrosesan citra, menggunakan media pemprograman MATLAB sebagai alat untuk menunjukkan teknik dan algoritma yang relevan. Diharapkan dapat memungkinkan pembaca untuk

mengembangkan proyek-proyek praktis, yaitu, prototipe kerja, menggunakan pengetahuan yang diperoleh dari buku.

Buku ini disusun untuk pengenalan dalam pemrosesan Citra yang dimulai dengan pengantar dan tinjauan lapangan (Bab 1) yang digunakan memotivasi siswa untuk mencurahkan waktu dan upaya untuk materi dalam bab-bab selanjutnya. Bab 2 memperkenalkan konsep dasar, notasi, dan terminologi yang terkait dengan representasi gambar dan operasi pemrosesan citra dasar. Bab 3 dan 4 dikhususkan untuk MATLAB dan Image Processing Toolbox. Bab 5 melihat masalah ekstraksi fitur dan representasi dan mengarah secara alami ke Bab 6 di mana vektor fitur yang dihasilkan dapat digunakan untuk tujuan klasifikasi dan pengenalan.

Beberapa bahan disediakan dalam website (https://bit.ly/2Pd5CIH) berisi banyak bahan pelengkap bagi siswa dan instruktur: skrip kode MATLAB untuk semua tutorial dalam buku ini, skrip kode MATLAB untuk citra yang dipilih, citra uji, masalah pelengkap, tutorial, dan proyek-proyek (yang tidak dapat mencapai versi cetak), dan daftar situs web yang bermanfaat termasuk (tautan ke) konferensi pemrosesan gambar, perangkat lunak, perangkat keras, grup riset, database gambar uji, dan lebih banyak.

DAFTAR ISI

PRAKATA ..... **v**
KATA PENGANTAR ..... **vii**
DAFTAR ISI ..... **ix**
DAFTAR TABEL ..... **xi**
DAFTAR GAMBAR ..... **xiii**

BAB 1 PENDAHULUAN **1**
1.1. TUJUAN 2
1.2. MOTIVASI 2
1.3. KONSEP DASAR DAN TERMINOLOGI 4
1.3.1. Apa itu Citra? 4
1.3.2. Apa itu Citra Digital? 4
1.3.3. Apa itu Pengolahan Citra Digital? 4
1.3.4. Apa Lingkup Pengolahan Citra? 5
1.4. CONTOH OPERASI KHUSUS PENGOLAHAN CITRA 6
1.5. KOMPONEN SISTEM PENGOLAHAN CITRA DIGITAL 11
1.5.1. Perangkat Keras (hardware) 12
1.5.2. Perangkat lunak (software) 13
1.6. MACHINE VISION SYSTEMS 14
1.7. RANGKUMAN 17
1.8. LATIHAN PERMASALAHAN 19

BAB 2 DASAR PENGOLAHAN CITRA **21**
2.1. TUJUAN 22
2.2. REPRESENTASI CITRA DIGITAL 22
2.2.1. Citra Biner (1-bit) 24

2.2.2. Citra grayscale (8-Bit) 25
2.2.3. Citra Berwarna (24-Bit atau RGB) 26
2.2.4. Citra Warna Berindeks 27
2.2.5. Kompresi 28
2.3. FORMAT FILE CITRA 29
2.4. TERMINOLOGI DASAR 30
2.4.1. Topologi Citra 30
2.4.2. Ketetanggaan Piksel 30
2.4.3. Kedekatan 32
2.4.4. Jalur 32
2.4.5. Konektivitas 32
2.4.6. Komponen 32
2.4.7. Jarak antara Piksel 33
2.5. GAMBARAN UMUM OPERASI PENGOLAHAN CITRA 34
2.5.1. Operasi di Domain Spasial: 34
2.5.2. Operasi dalam domain transform 34
2.5.4. Operasi berorientasi kedekatan 35
2.5.5. Operasi Menggabungkan Banyak Citra 36
2.5.6. Operasi dalam Domain Transform 37
2.6. RANGKUMAN 37

BAB 3 DASAR MATLAB 39
3.1. TUJUAN 40
3.2. PENDAHULUAN UNTUK MATLAB 40
3.3. UNSUR DASAR MATLAB 41
3.3.1. Area Kerja 41
3.3.2. Jenis Data 42
3.3.3. Array dan Index Matriks dalam MATLAB 43
3.3.4. Array Standar 43
3.3.5. Operasi Command-Line 44
3.4. ALAT PEMROGRAMAN: SKRIP DAN FUNGSI 45
3.4.1. File-M 45
3.4.2. Operator 46
3.4.3. Variabel dan Konstanta 49
3.4.4. Representasi Angka 50
3.4.5. Kontrol Aliran (Flow Control) 50

3.4.6. Optimasi Kode 50
3.4.7. Input dan Output 51
3.5. GRAFIS DAN VISUALISASI 51
3.6. TUTORIAL STRUKTUR MATLAB 51
3.7. TUTORIAL PEMPROGRAMAN MATLAB 62
3.8. LATIHAN PERMASALAHAN 67

BAB 4 PROSES PENGOLAHAN CITRA DENGAN MATLAB TOOLBOX 69
4.1. TUJUAN 70
4.2. ALAT PENGOLAHAN CITRA: GAMBARAN UMUM 70
4.3. FUNGSI DAN FITUR PENTING 71
4.3.1. Menampilkan Informasi Tentang File Citra 71
4.3.2. Membaca File Citra 72
4.3.3. Kelas Data dan Konversi Data 72
4.3.4. Menampilkan Isi Citra 76
4.3.5. Menjelajahi Isi Citra 78
4.3.6. Menulis Citra yang dihasilkan ke File 78
4.4. TUTORIAL MANIPULASI CITRA DASAR 79
4.5. LATIHAN PERMASALAHAN 86

Bab 5 EKSTRAKSI FITUR DAN REPRESENTASI 87
5.1. TUJUAN 88
5.2. PENDAHULUAN: GAMBARAN UMUM 88
5.3. VEKTOR FITUR DAN RUANG VEKTOR 89
5.3.1 Invarian dan Robustness 91
5.4. FITUR OBJEK BINARY 92
5.4.1 Area 94
5.4.2 Centroid 94
5.4.3 Axis of Least Second Moment 95
5.4.4 Projections 95
5.4.5 Euler Number 96
5.4.6 Perimeter 96
5.4.7 Thinness Ratio 97
5.4.8 Eccentricity 98
5.4.9 Aspect Ratio 98
5.4.10 Momen 99

5.5. FITUR BERBASIS DESKRIPSI BATASAN 100
5.5.1 Chain Code, Freeman Code, and Shape Number 103
5.5.2 Signatures 105
5.5.3 Fourier Descriptors 105
5.6. FITUR BERBASIS HISTOGRAM (STATISTIK) 108
5.7. FITUR TEKSTUR 111
5.8. TUTORIAL FEATURE EXTRACTION DAN REPRESENTASI 115
5.9. LATIHAN PERMASALAHAN 119

BAB 6 PENGENALAN POLA VISUAL 121
6.1. TUJUAN 122
6.2. PENDAHULUAN: GAMBARAN UMUM 122
6.3. FUNDAMENTAL 123
6.3.1 Desain dan Implementasi Klasifikasi Pola Visual 124
6.3.2 Pola dan Pola kelas 126
6.3.3 Pengolahan Data Awal 128
6.3.4 Data Pelatihan dan Pengujian 129
6.3.5 Confusion Matrix 130
6.3.6 Sistem Kesalahan 131
6.3.7 Hit Rates, False Alarm Rates, and Kurva ROC 131
6.3.8 Precision dan Recall 133
6.3.9 Distance dan Similarity Measures 136
6.4. TEKNIK KLASIFIKASI POLA STATISTIK 138
6.4.1 Minimum Distance Classifier 140
6.4.2 Klasifikasi k-Nearest Neighbors 141
6.4.3 Klasifikasi Naïve Bayes 142
6.5. TUTORIAL KLASIFIKASI POLA 144
6.5.1 Refleksi 149
6.6. LATIHAN PERMASALAHAN 151

REFERENSI 153

DAFTAR TABEL

Tabel 3. 1	Tipe data dalam MATLAB	42
Tabel 3. 2	Operator Arithmetic MATLAB Array dan Matrix	46
Tabel 3. 3	Operator Matrix spesial MATLAB	47
Tabel 3. 4	Operator Matrix spesial MATLAB berdasar Toolbox Image Processing	48
Tabel 3. 5	Operator Relasi	48
Tabel 3. 6	Operator atau Fungsi Logika	49
Tabel 3. 7	Variabel dan Konstanta	49
Tabel 4. 1	Fungsi IPT untuk Melakukan Konversi Kelas Data Citra	72
Tabel 4. 2	Fungsi IPT untuk melakukan Konversi Kelas Data Citra	76
Tabel 5. 1	Tabel Fitur berupa Luas Area dan Perimeter	90
Tabel 5. 2	Properti Wilayah Berlabel	92
Tabel 5. 3	RST-Invariant Moments	100
Tabel 5. 4	Properti Wilayah Berlabel	112
Tabel 5. 5	Properti Wilayah Berlabel	117

DAFTAR GAMBAR

Gambar 1. 1 *Image sharpening*: (a) citra asli; (b) setelah sharpening 7
Gambar 1. 2. *Noise Removal*: (a) original (*noise*) image; (b) setelah menghapus noise 8
Gambar 1. 3 *Deblurring*: (a) original (*blurry*) image; (b) setelah menghapus blur 8
Gambar 1. 4 *Edge Extraction*: (a) citra asli; (b) setelah ekstraksi tepi 9
Gambar 1. 5 *Binarization*: (a) citra asli; (b) setelah konversi menjadi hitam putih 9
Gambar 1. 6 *Blurring*: (a) citra asli; (b) setelah *blurring* untuk menghapus detail yang tidak berguna 10
Gambar 1. 7 *Contrast Enhancement*: (a) citra asli; (b) peningkatan contrast 10
Gambar 1. 8 *Object Segmentation* dan *Labeling*: (a) citra asli; (b) hasil segmentasi dan pelabelan 11
Gambar 1. 9 Komponen sistem pengolahan citra digital (Gonzalez & Woods, 2008) 12
Gambar 1. 10 Kerangka diagram *machine vision system* (Gonzalez & Woods, 2008) 14
Gambar 1. 11 (a) Uji citra untuk segmentasi objek berbasis tekstur. (B) Uji citra untuk segmentasi objek berdasarkan "interpolasi" batas objek. 19
Gambar 2. 1. a dan b merupakan ilustrasi matrik dan citra monokrom dengan ketentuan yang digunakan untuk mewakili baris (x) dan kolom (y) 23
Gambar 2. 2. Merupakan ilustrasi matrik dalam citra dengan pemprograman MATLAB 23
Gambar 2. 3. Citra biner dan nilai piksel di area n × m 25

Gambar 2. 4. Citra grayscale dan nilai piksel di area n × m 26
Gambar 2. 5. Warna citra (a) dan komponen R (b), G (c), dan B (d) 27
Gambar 2. 6. Citra warna yang diindeks dan indeks. Sumber: MathWorks 28
Gambar 2. 7. Piksel dalam ketetanggaan 31
Gambar 2. 8. Konsep ketetanggaan piksel p (dari perspektif topologi citra): (a) 4-ketetanggaan; (b) ketetanggaan diagonal; (c) 8- ketetanggaan..... 31
Gambar 2. 9. Ukuran 3 × 3 kernel konvolusi, dengan bobot W1, ..., W9 ... 36
Gambar 2. 10. Operasi transformasi domain 37
Gambar 4. 1 Menampilkan citra: (a) tanpa penskalaan; (b) penskalaan (c) memilih piksel 77
Gambar 4. 2 Menampilkan citra dam menjelajahi dengan MATLAB, *Pixel Region*..... 78
Gambar 4. 3 Membaca dan menulis citra: (a) Citra asli (TIF); (B) citra terkompresi (JPG, q = 75, ukuran file = 7 kB); (c) citra terkompresi (JPG, q = 5, ukuran file = 2 kB); (d) citra terkompresi (JPG, q = 95, ukuran file = 17 kB). 79
Gambar 4. 4 Pembagian angka menggunakan subplot..... 83
Gambar 5.1 memperlihatkan hasil Fitur dalam vektor 2D 90
Gambar 5. 2 Contoh dua wilayah dengan nomor Euler masing-masing sama dengan 0 dan −1..... 96
Gambar 5. 3 Contoh wilayah kompak (a) dan tidak kompak (b)..... 97
Gambar 5. 4 Keanehan (A / B) suatu daerah..... 98
Gambar 5. 5 Penelusuran batas objek..... 102
Gambar 5. 8 Chain code, first differences, dan shape number..... 105
Gambar 5. 9 Fourier descriptor dari batas 106
Gambar 5. 10 Contoh rekonstruksi batas menggunakan deskriptor Fourier: (a) citra asli; (b –f) citra yang direkonstruksi masing-masing menggunakan 100%, 50%, 25%, 2,5%, dan 1% dari jumlah total poin 108
Gambar 5. 11 Contoh citra dengan tekstur halus (a), kasar (b), dan reguler (c). Citra dari set data tekstur 109
Gambar 5. 12 Contoh citra dengan tekstur halus (a), kasar (b), dan reguler (c). Citra dari set data tekstur 111

Gambar 5. 13 Contoh Citra (a) dan cooccurrence matrix untuk d = (0,1) (b). 114
Gambar 6. 1 Diagram klasifikasi pola secara statistik..... 123
Gambar 6. 2 Interaksi antara ekstraksi fitur, pemilihan fitur, dan klasifikasi pola sebagai fungsi dari aplikasi yang ada.... 124
Gambar 6. 3 Contoh dua kelas (pegulat sumo — lingkaran merah — dan pemain tenis meja — berlian biru) dijelaskan dengan dua pengukuran (berat dan tinggi)..... 127
Gambar 6. 4 confusion matriks ukuran 4 4x4..... 130
Gambar 6. 5 ROC Curve 132
Gambar 6. 6 Graph presisi recall 136
Gambar 6. 7 Fungsi diskriminasi untuk kelas tiga kelas dalam ruang fitur 2D 139
Gambar 6. 8 Contoh berdasarkan kelas 140
Gambar 6. 9 Contoh KNN dengan 5 kelas..... 142
Gambar 6. 10 Ruang Fitur untuk data Pelatihan 146
Gambar 6. 11 Matriks kebingungan dengan hasil klasifikasi KNN untuk fitur yang dipilih 149
Gambar 6. 12 Matriks Confusion dengan hasil klasifikasi KNN untuk kasus di mana nilai abu-abu gambar digunakan sebagai "fitur." 150
Gambar 6. 13 Kasus Matriks Confusion 151

Bab 1
PENDAHULUAN

1.1. TUJUAN

Tujuan dari bahasan ini adalah

- Mempelajari dasar pengolahan citra, algoritme dan operasi umum hingga penerapan pada pengolahan citra.
- Mengenalkan perangkat yang diperlukan dalam pengolahan citra.
- Mempelajari apa, bagaimana dan perbedaan kinerja *machine vision system* (MVS) terhadap *human visual system* (HVS).

1.2. MOTIVASI

Manusia mengandalkan visi mereka untuk tugas-tugas mulai dari keterampilan akan naluri bertahan hidup dalam melakukan analisis secara rinci dan rumit dari suatu karya seni. Kemampuan untuk membimbing tindakan dan melibatkan kemampuan kognitif berdasarkan input visual adalah sifat luar biasa dari spesies manusia. Namun seberapa tepat manusia dapat melakukan apa yang mereka lakukan dengan sangat baik masih harus diteliti.

Kebutuhan untuk mengekstrak informasi dari citra dan menafsirkan isinya menjadi salah satu faktor pendorong dalam pengembangan untuk pengolahan citra dan *computer vision* selama beberapa dekade terakhir. Aplikasi pengolahan citra mencakup berbagai aktivitas manusia, seperti berikut ini:

- Aplikasi medis: Diagnostic imaging modalities seperti digital radiography, PET (positron emission tomography), CAT (*computerized axial tomography)*, MRI (*magnetic resonance imaging*), and fMRI (functional magnetic resonance imaging), antara lain, telah diadopsi oleh komunitas medis dalam skala besar.

- Aplikasi industri: Sistem pengolahan citra telah berhasil digunakan dalam sistem manufaktur di beberapa tugas, seperti *safety systems*, *quality control*, dan *control of automated guided vehicles (AGVs)*.
- Aplikasi militer: Beberapa skenario yang paling menantang dan kritis terhadap kinerja untuk menyelesaikan proses penyelesaian harus mengembangkan kebutuhan untuk militer, mulai dari deteksi tentara atau kendaraan hingga panduan rudal dan pengenalan objek dan tugas pengintaian menggunakan kendaraan udara tak berawak atau unmanned aerial vehicles (UAVs). Selain itu, aplikasi militer sering kali memerlukan penggunaan sensor pencitraan yang berbeda, seperti *range cameras* dan *thermographic forward-looking infrared (*FLIR*) cameras*.
- Penegakan hukum dan keamanan: Aplikasi pengawasan telah menjadi salah satu bidang yang paling banyak diteliti dalam komunitas pengolahan video.
- Teknik biometrik (seperti fingerprint, face, iris, dan hand recognition), yang telah menjadi subjek penelitian pada pengolahan citra selama lebih dari satu dekade, baru-baru ini tersedia secara komersial.
- Elektronik konsumen: Kamera digital dan *camcorder*, dengan kemampuan pengolahan yang canggih, telah membuat film dan teknologi pita analog menjadi usang. Paket perangkat lunak untuk meningkatkan, mengedit, mengatur, dan mempublikasikan citra dan video telah berkembang dalam kecanggihan sambil menjaga antarmuka yang ramah pengguna. *High-definition TVs*, *monitors*, *DVD players*, *dan personal video recorders (PVRs)* semakin meningkat karena populer dan terjangkau. Pencitraan dan pembagian juga

telah berhasil membuat lompatan ke perangkat lain, seperti *personal digital assistants (PDAs)*, *cell phones*, dan *portable music (MP3) players*.

- Internet, khususnya *world wide web*: Ada banyak informasi visual yang tersedia di Web. Kolaboratif *video uploading*, *sharing*, dan *annotation (tagging)* menjadi semakin populer. Menemukan dan mengambil citra dan video di Web berdasarkan isinya tetap merupakan tantangan terbuka dalam penelitian.

1.3. KONSEP DASAR DAN TERMINOLOGI

1.3.1. Apa itu Citra?

Citra adalah representasi visual dari suatu objek, seseorang, atau pemandangan yang dihasilkan oleh perangkat optik seperti cermin, lensa, atau kamera. Representasi tersebut adalah dua dimensi (2D), meskipun sesuai dengan salah satu dari banyak proyeksi objek atau adegan dunia nyata, tiga dimensi (3D).

1.3.2. Apa itu Citra Digital?

Citra digital adalah penyajian citra dua dimensi menggunakan sejumlah titik, biasanya disebut sebagai elemen citra atau piksel. Setiap piksel diwakili oleh satu atau lebih nilai numerik: untuk citra monokrom (*grayscale*), nilai tunggal yang mewakili intensitas piksel (biasanya dalam kisaran [0, 255]) sudah cukup; untuk citra berwarna, biasanya diperlukan tiga nilai (seperti mewakili jumlah merah (R), hijau (G), dan biru (B)).

1.3.3. Apa itu Pengolahan Citra Digital?

Pengolahan citra digital dapat didefinisikan sebagai ilmu memodifikasi citra digital dengan menggunakan komputer digital.

Perubahan yang terjadi pada citra biasanya dilakukan secara otomatis dan mengandalkan algoritme yang dirancang dengan cermat. Ini sangat kontras dengan skenario lain, seperti menyentuh foto menggunakan alat air brush di perangkat lunak pengedit foto, di mana citra diproses secara manual dan keberhasilan tugas tergantung pada kemampuan manusia dan ketangkasan. Hal ini merupakan manipulasi citra untuk membuat perbedaan yang lebih eksplisit.

1.3.4. Apa Lingkup Pengolahan Citra?

Beberapa terminologi yang digunakan dalam pengolahan citra dilakukan dengan semua teknik dan aplikasi, yang bertujuan mendapatkan output yang dimodifikasi atau diproses versi citra input, versi yang disandikan dari atribut utamanya, atau deskripsi non-citra dari isinya.

Selain itu, terdapat tiga level operasi pengolahan citra:

- Tingkat Rendah: Operasi primitif (seperti *noise reduction*, *contrast enhancement*, dll.) dengan input dan outputnya merupakan citra.
- Tingkat Menengah: Ekstraksi atribut (seperti *edges*, *contours*, *regions*, dll.) dengan input yang berupa citra.
- Tingkat Tinggi: Analisis dan interpretasi isi dari adegan.

Buku ini tidak mencakup bidang grafis komputer atau sintesis citra, proses di mana citra 2D atau 3D dirender dari data numerik. Namun pada proses yang berlawanan (analisis citra), dimana data tekstual dan numerik dapat diekstraksi dari array piksel. Pengolahan citra adalah bidang multidisiplin, dengan kontribusi dari berbagai cabang ilmu (terutama matematika, fisika, dan ilmu komputer) dan komputer, optik, dan teknik listrik. Selain itu, tumpang tindih bidang

lain seperti *pattern recognition*, *machine learning*, *artificial intelligence*, dan *human vision research*.

1.4. CONTOH OPERASI KHUSUS PENGOLAHAN CITRA

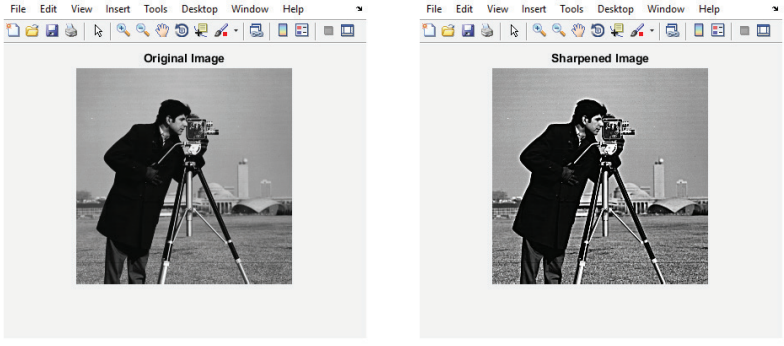
Pengolahan citra mencakup ide dan beragam aturan teknik dan algoritme. Secara umum, operasi pengolahan citra dapat dikategorikan ke dalam empat jenis, diantaranya:

- *Pixel operations*: keluaran pada piksel hanya bergantung pada input dari piksel tersebut, terlepas dari semua piksel lain dalam citra itu. Threshold atau ambang batas merupakan proses pembuatan piksel input yang sesuai di atas level threshold berwarna putih dan lainnya berwarna hitam, merupakan operasi piksel sederhana. Contoh lain termasuk *brightness addition/subtraction*, *contrast stretching*, *image inverting*, *log*, dan *power law*.
- *Local (neighborhood) operations*: keluaran pada pixel tergantung pada nilai input di area piksel itu. Beberapa contohnya adalah deteksi tepi, smoothing filters (seperti filter rata-rata dan filter median), dan filter penajaman (seperti filter Laplacian dan filter gradien). Operasi ini bisa adaptif karena hasilnya tergantung pada nilai piksel tertentu yang dijumpai di setiap wilayah citra.
- *Geometric operations*: keluaran pada piksel hanya bergantung pada level input di beberapa piksel lain yang ditentukan oleh transformasi geometrik. Operasi geometris berbeda dari operasi global, sehingga inputnya hanya dari beberapa piksel spesifik berdasarkan transformasi geometrik. Mereka tidak membutuhkan input dari semua piksel untuk melakukan transformasi.

- *Global operations*: keluaran pada piksel tergantung pada semua piksel dalam suatu citra. Hal tersebut mungkin tidak bergantung pada nilai piksel dalam citra, atau mungkin mencerminkan statistik yang dihitung untuk semua piksel, tetapi bukan subset piksel lokal. Jarak transformasi yang populer pada suatu citra yang memberikan piksel minimum pada setiap jarak objek dari citra ke semua piksel latar belakang, milik operasi global.

Operasi pengolahan citra paling representatif yang ada, diantaranya:

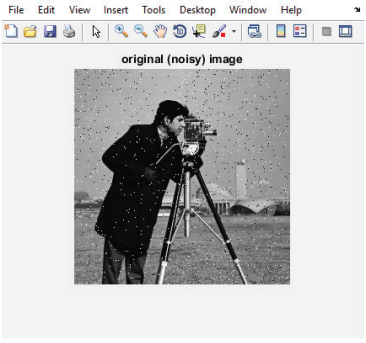
- *Sharpening*: Suatu teknik yang meningkatkan tepi dan detail citra untuk tampilan manusia. Metode ini dilakukan dalam domain spasial maupun domain frekuensi.

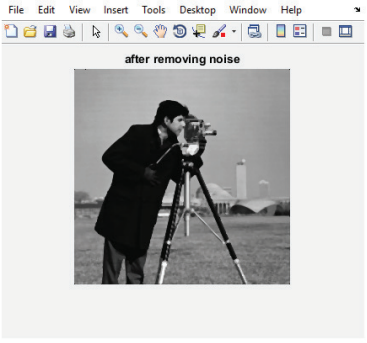


a b

Gambar 1. 1 *Image sharpening*: (a) citra asli; (b) setelah sharpening

- *Noise Removal*: Filter pengolahan citra dapat digunakan untuk mengurangi jumlah noise dalam citra sebelum memprosesnya lebih jauh. Bergantung pada jenis noise, berbagai teknik penghilangan noise digunakan.

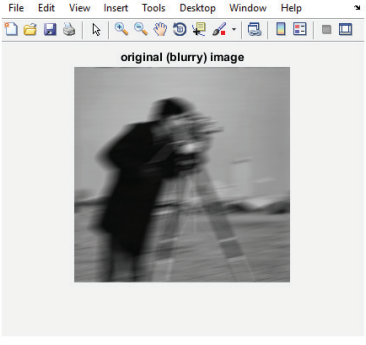




a b

Gambar 1. 2. *Noise Removal*: (a) original (*noise*) image; (b) setelah menghapus noise

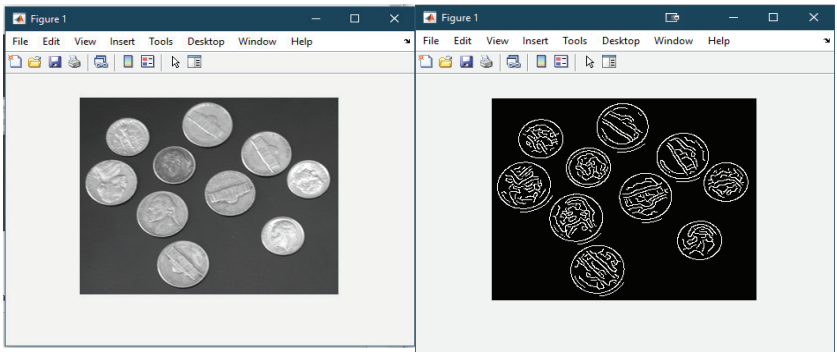
- *Deblurring*: Citra mungkin tampak buram karena berbagai alasan, mulai dari pemfokusan lensa yang tidak tepat hingga rana yang tidak memadai. kecepatan untuk objek yang bergerak cepat.



a b

Gambar 1. 3 *Deblurring*: (a) original (*blurry*) image; (b) setelah menghapus blur

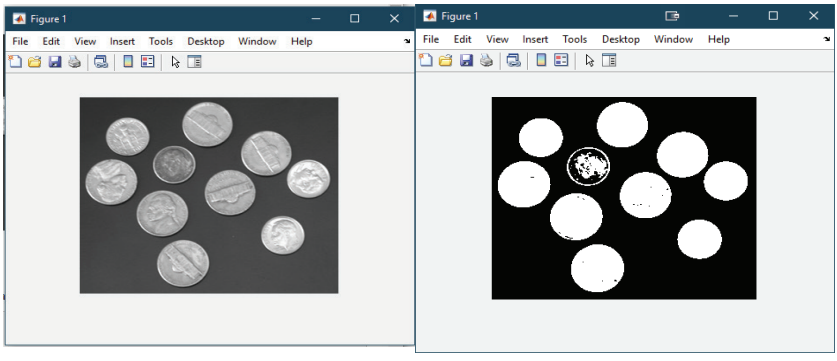
- *Edge Extraction*: Ekstraksi tepi dari citra merupakan langkah preprocessing mendasar yang digunakan untuk memisahkan objek satu sama lain sebelum mengidentifikasi isinya.



a b

Gambar 1. 4 *Edge Extraction*: (a) citra asli; (b) setelah ekstraksi tepi

- *Binarization*: Dalam banyak aplikasi analisis citra, seringkali perlu untuk mengurangi jumlah *grayscale* dalam citra monokrom untuk menyederhanakan dan mempercepat interpretasinya. Mengurangi citra *grayscale* menjadi hanya dua *grayscale* (hitam dan putih) biasanya disebut sebagai binarisasi.

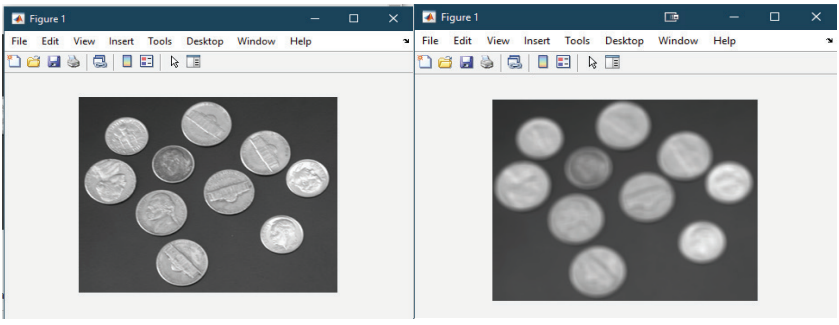


a b

Gambar 1. 5 *Binarization*: (a) citra asli; (b) setelah konversi menjadi hitam putih

- *Blurring*: Kadang-kadang perlu untuk mengaburkan citra untuk meminimalkan pentingnya tekstur dan detail halus dalam sebuah adegan, misalnya, dalam kasus di mana objek dapat

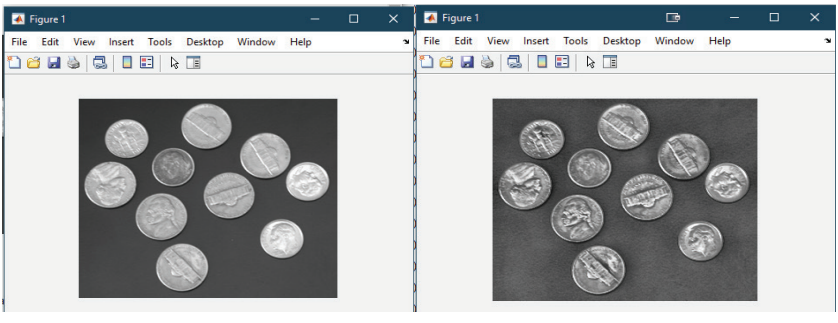
lebih dikenali oleh bentuknya. Teknik *Blurring* dapat dilakukan dalam domain spasial dan frekuensi.



a b

Gambar 1. 6 *Blurring*: (a) citra asli; (b) setelah *blurring* untuk menghapus detail yang tidak berguna

- *Contrast Enhancement*: Untuk meningkatkan citra untuk dilihat manusia serta mempermudah tugas pengolahan citra lainnya (seperti *edge extraction*), seringkali perlu untuk meningkatkan kontras citra. Teknik peningkatan kontras dapat menggunakan fungsi transformasi dan pengolahan analog.

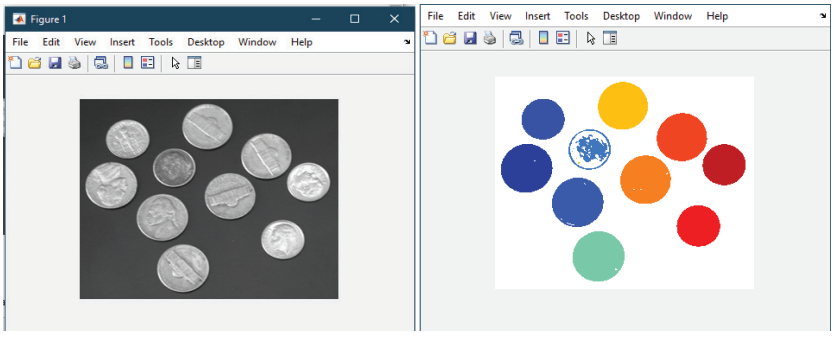


a b

Gambar 1. 7 *Contrast Enhancement*: (a) citra asli; (b) peningkatan contrast

- *Object Segmentation* dan *Labeling*: Tugas segmentasi dan pelabelan objek dalam adegan adalah prasyarat untuk sebagian besar pengenalan objek dan sistem klasifikasi. Setelah objek

yang relevan telah tersegmentasi dan diberi label, fitur-fitur yang relevan dapat diekstraksi dan digunakan untuk mengklasifikasikan, membandingkan, mengelompokkan, atau mengenali objek yang dimaksud.

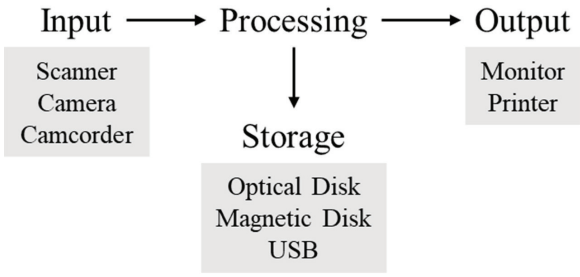


a b

Gambar 1. 8 *Object Segmentation* dan *Labeling*: (a) citra asli; (b) hasil segmentasi dan pelabelan

1.5. KOMPONEN SISTEM PENGOLAHAN CITRA DIGITAL

Sistem pengolahan citra digital secara umum, dijabarkan dalam blok diagram pada Gambar 1.9. Dimana sistem dibangun di sekitar komputer di mana sebagian besar tugas pengolahan citra dilakukan, tetapi juga mencakup perangkat keras dan perangkat lunak untuk akuisisi data, penyimpanan, dan menampilkan citra. Perangkat keras aktual yang terkait dengan setiap blok berubah seiring perkembangan teknologi.



Gambar 1. 9 Komponen sistem pengolahan citra digital (Gonzalez & Woods, 2008)

1.5.1. Perangkat Keras (hardware)

Komponen perangkat keras dari sistem pengolahan citra digital umumnya terdiri dari:

- *Acquisition devices*: bertanggung jawab untuk mengambil dan mendigitalkan citra atau urutan video. Beberapa contoh perangkat akuisisi secara umum diantaranya: pemindai, kamera, dan camcorder. Perangkat akuisisi dapat dihubungkan dengan komputer utama dalam beberapa cara, misalnya, *USB*, *firewire*, *camera link*, atau *ethernet*. Dalam kasus di mana kamera menghasilkan output video analog maupun digitizer citra, biasanya dikenal sebagai *frame grabber* yang digunakan untuk mengubahnya menjadi format digital.
- *Processing equipment*: komputer utama itu sendiri, dalam ukuran, bentuk, atau konfigurasi apa pun. Bertanggung jawab untuk menjalankan perangkat lunak yang memungkinkan pengolahan dan analisis citra yang diperoleh.

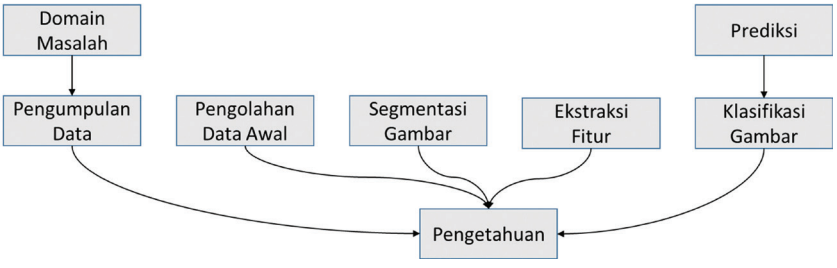
- *Display dan hardcopy devices*: bertanggung jawab untuk menampilkan konten citra untuk dilihat manusia. Contohnya termasuk monitor warna dan printer.
- *Storage device*: disk magnetik atau optik yang bertanggung jawab atas penyimpanan citra dalam jangka panjang.

1.5.2. Perangkat lunak (software)

Perangkat lunak dari sistem pengolahan citra digital biasanya terdiri dari modul untuk melakukan tugas-tugas khusus. Pengembangan dan penyesuaian perangkat lunak untuk solusi pengolahan citra bersifat iteratif. Akibatnya, para peneliti dan praktisi pengolahan citra bergantung pada bahasa pemrograman dan area pengembangan yang mendukung pengembangan perangkat lunak modular, gesit, dan berulang. Salah satu perangkat lunak yang membatu dalam hal tersebut adalah MATLAB (MATrix LABoratory), multiplatform, analisis data, prototyping, dan alat visualisasi yang mendukung operasi matriks dan matriks, kemampuan grafis yang kaya, serta bahasa pemrograman yang ramah dalam area pengembangan. MATLAB menawarkan kepada programmer kemampuan untuk mengedit dan berinteraksi dengan fungsi utama dan berbagai parameternya, yang bertujuan untuk penghematan waktu yang berharga dalam siklus pengembangan perangkat lunak. MATLAB telah menjadi sangat populer di kalangan insinyur, ilmuwan, dan peneliti di industri dan akademisi, karena banyak faktor, seperti ketersediaan kotak peralatan yang mengandung fungsi khusus untuk banyak area aplikasi, mulai dari akuisisi data hingga pengolahan citra.

1.6. MACHINE VISION SYSTEMS

Beberapa komponen utama dari *machine vision systems (MVS)* pada Gambar 1.10 menggunakan contoh pada aplikasi praktis: pengenalan jenis ikan bawah laut. Pengolahan berdasarkan data citra bukan proses yang dapat diselesaikan dalam satu langkah, sebagian besar solusi mengikuti skema pengolahan secara sekuensial. Diawali dengan domain masalah, dalam pengenalan jenis ikan bawah laut secara otomatis. Memiliki tujuan adalah untuk dapat mengekstraksi konten alfanumerik dari citra ikan yang didapatkan dari bawah laut dengan cara otomatis dan tanpa pengawasan atau tanpa perlu campur tangan manusia. Dengan beberapa persyaratan tambahan seperti kondisi di bawah pencahayaan buatan, dilakukan pada semua cuaca, tingkat keberhasilan minimal yang dapat diterima, dan jarak ikan secara minimum dan maksimum terhadap kamera.



Gambar 1. 10 Kerangka diagram *machine vision system* (Gonzalez & Woods, 2008)

Pada bagian akuisisi data yang bertanggung jawab untuk memperoleh satu atau lebih citra yang mengandung tampilan objek ikan yang memiliki latar belakang bawah laut. Ini dapat diimplementasikan menggunakan kamera CCD dan mengendalikan kondisi pencahayaan untuk memastikan bahwa citra akan cocok untuk diproses lebih lanjut. Output bagian ini adalah citra digital yang berisi tampilan (sebagian) objek ikan dan beberapa faktor

harus dipertimbangkan dalam membuat desain akuisisi. Selain itu juga mempertimbangkan kemungkinan dampak yang diberikan pada pada kualitas citra yang dihasilkan terhadap kinerja keseluruhan sistem, seperti kondisi bawah air yang memiliki risiko mengaburkan citra, aspek pencahayaan (misalnya, jumlah, jenis, dan posisi sumber cahaya), pilihan lensa, dan spesifikasi (resolusi dan kecepatan) dari perangkat keras digitizer citra.

Oleh karena itu, tujuan dari tahap preprocessing adalah untuk meningkatkan kualitas citra yang diperoleh. Algoritme yang mungkin digunakan selama tahap ini meliputi *contrast improvement*, *brightness correction*, dan *noise removal*.

Pada bagian segmentasi bertanggung jawab untuk membagi citra ke dalam komponen utamanya: objek latar depan dan latar belakang yang relevan. Ini menghasilkan pada keluarannya sejumlah daerah berlabel atau "*subimages*". Segmentasi citra otomatis merupakan salah satu tugas paling menantang dalam *machine vision system*.

Pada bagian ekstraksi fitur (representasi dan deskripsi) terdiri dari algoritme yang bertanggung jawab untuk menyandikan konten citra dengan cara yang ringkas dan deskriptif. Fitur yang umum termasuk ukuran distribusi warna (atau intensitas), tekstur, dan bentuk objek yang paling relevan (sebelumnya tersegmentasi) dalam citra. Fitur-fitur ini biasanya dikelompokkan ke dalam vektor fitur yang kemudian dapat digunakan sebagai indikator numerik dari konten citra (objek) untuk tahap selanjutnya, di mana konten tersebut akan dikenali (diklasifikasikan).

Setelah fitur yang paling relevan dari citra (atau objek yang relevan) didapatkan dengan diekstraksi dan dikodekan ke dalam vektor fitur. Langkah selanjutnya adalah menggunakan representasi numerik k-dimensi ini sebagai input ke klasifikasi pola (tahap

pengakuan dan interpretasi). Pada titik ini, pengolahan citra dengan pengenalan pola klasik mendapatkan manfaat dari banyak tekniknya yang telah terbukti, seperti distance *classifiers*, *probabilistic classifiers*, *neural networks*, dan banyak lagi. Tujuan akhir dari diagram tersebut adalah untuk mengklasifikasikan (seperti menetapkan label untuk) setiap karakter individu, menghasilkan string (atau file ASCII) pada bagian prediksi, yang berisi nama atau jenis citra.

Basis pengetahuan merupakan inti dari keseluruhan langkah diagram *MVS*. Hal ini menunjukkan bahwa solusi yang berhasil untuk masalah pengenalan jenis ikan akan tergantung pada seberapa banyak pengetahuan tentang domain masalah yang telah dikodekan dan disimpan dalam MVS. Peran basis pengetahuan tersebut pada tahap terakhir cukup jelas (seperti pengetahuan bahwa karakter pertama harus berupa angka dapat membantu membedakan antara tekstur objek A dan objek B pada tahap klasifikasi pola). Dengan cara yang kurang jelas, basis pengetahuan harus (idealnya) membantu semua tugas dalam MVS. Misalnya, segmentasi dapat memberikan objek citra yang bebas dari latar belakang laut yang beraneka ragam.

HSV dan MVS memiliki kekuatan dan keterbatasan yang berbeda dan perancang MVS harus mampu menyadari perbedaan tersebut. Analisis yang cermat terhadap perbedaan-perbedaan ini memberikan wawasan mengapa sangat sulit untuk meniru kinerja HSV menggunakan kamera dan komputer. Terdapat tiga tantangan terbesar menonjol:

- HVS dapat mengandalkan database citra yang sangat besar yang berhubungan dengan konsep pada saat data diambil, diproses, dan disimpan selama proses pengenalan. Penyimpanan citra yang dilakukan dapat dikatakan mudah, namun memetakan konsep semantik ke tingkat tinggi dan

menempatkan kelasnya masing-masing merupakan tugas yang sangat sulit dalam melakukan MVS.

- MVS mengandalkan kemampuan komputer dengan kecepatan sangat tinggi, di mana HVS membuat keputusan berdasarkan input visual. Beberapa tugas yang memanfaatkan pengolahan citra dan *machine vision* dapat diimplementasikan pada perangkat keras khusus atau komputer super, namun banyak implementasi tersebut masih tidak dapat menyamai kecepatan manusia dalam membandingkan, sehingga masih tidak dapat diterapkan secara *real-time*.
- Kemampuan HVS yang luar biasa untuk bekerja dalam berbagai kondisi, dari pencahayaan yang kurang hingga perspektif yang kurang ideal dalam melakukan berbagai kegiatan. Hal ini menjadi hambatan terbesar dalam membuat citraan MVS. Untuk menghindari permasalahan tersebut, sebagian besar MVS harus menerapkan banyak parameter seperti pencahayaan yang dikontrol dengan cermat hingga menghilangkan gangguan yang tidak sesuai dan dapat menyesatkan sistem.

1.7. RANGKUMAN

- Pengolahan citra digital adalah ilmu memodifikasi citra digital menggunakan komputer digital.
- Pengolahan citra digital terkait erat dengan bidang lain seperti *computer vision* dan pengenalan pola.
- Algoritme, teknik, dan aplikasi pengolahan citra digital biasanya mengambil citra sebagai input dan menghasilkan salah satu dari output berikut: citra yang dimodifikasi (diproses), atribut yang

disandikan dari atribut utama yang ada pada citra input, atau deskripsi dari isi citra input.

- Pengolahan citra digital telah menemukan aplikasi di hampir setiap bidang kehidupan modern, dari perangkat pencitraan medis hingga kontrol kualitas dalam sistem manufaktur, dan dari elektronik konsumen hingga penegakan hukum dan keamanan.
- Citra adalah representasi visual dari suatu objek, seseorang, atau pemandangan yang dihasilkan oleh perangkat optik seperti cermin, lensa, atau kamera. Representasi ini berbentuk dua dimensi, meskipun sesuai dengan salah satu dari proyeksi tak terbatas dari objek atau adegan tiga dimensi dunia nyata.
- Citra digital adalah representasi citra dua dimensi menggunakan jumlah piksel terbatas, yang masing-masing menunjukkan *grayscale* atau isi warna citra pada titik itu.
- Teknik manipulasi citra terdiri dari memodifikasi secara manual konten citra menggunakan alat yang sudah ada sebelumnya.
- Operasi pengolahan citra yang representatif meliputi penajaman citra, penghilangan derau, ekstraksi tepi, peningkatan kontras, dan segmentasi dan pelabelan objek.
- Sistem pengolahan citra digital biasanya dibangun di sekitar komputer serba guna yang dilengkapi dengan perangkat keras untuk akuisisi, penyimpanan, dan tampilan citra dan video. Bagian perangkat lunak sistem biasanya terdiri dari modul yang melakukan tugas-tugas khusus. Dalam buku ini akan menggunakan MATLAB (dan *Image Processing Toolbox*) sebagai perangkat lunak pilihan.
- MVS adalah kombinasi perangkat keras dan perangkat lunak yang dirancang untuk menyelesaikan masalah yang melibatkan analisis adegan visual menggunakan algoritme

cerdas. Komponen utamanya adalah akuisisi, pengolahan awal, segmentasi, ekstraksi fitur, dan klasifikasi.

1.8. LATIHAN PERMASALAHAN

1. Dalam diskusi tentang MVS, diindikasikan bahwa berikut ini adalah tiga kesulitan terbesar dalam meniru HVS: basis datanya yang besar (citra dan konsep yang ditangkap, diproses, dan direkam selama pelatihan), tingginya kecepatan untuk memproses data visual dan membuat keputusan atas data yang dimiliki, dan kemampuan untuk bekerja di berbagai kondisi. Jelaskan masing-masing tantangan ini dengan kata-kata Anda sendiri, dan komentari tantangan mana yang lebih mungkin untuk diperkecil atau diselesaikan.
2. Menurut Anda siapa yang akan melakukan lebih baik pada tugas-tugas berikut: manusia (HVS) atau komputer (MVS)? jelaskan mengapa.

a b

Gambar 1. 11 (a) Uji citra untuk segmentasi objek berbasis tekstur. (B) Uji citra untuk segmentasi objek berdasarkan "interpolasi" batas objek.

a. Menentukan garis mana yang terpendek pada Citra a.
b. Menentukan lingkaran mana yang terkecil pada Citra b.
c. Mengelompokkan citra yang mengandung huruf “F” dari latar belakang di Citra a.
d. Mengelompokkan segitiga putih pada Citra b.

Bab 2

DASAR PENGOLAHAN CITRA

2.1. TUJUAN

Tujuan dari bahasan ini adalah

- Mempelajari bagaimana cara citra digital direpresentasikan dan disimpan dalam memori.
- Mempelajari berbagai jenis utama dari representasi citra digital.
- Mempelajari tentang format file citra yang paling populer?
- Mempelajari berbagai jenis operasi pemrosesan citra yang paling umum dan bagaimana pengaruhnya terhadap nilai piksel?

2.2. REPRESENTASI CITRA DIGITAL

Citra digital diperoleh sebagai hasil pengambilan sampel dan kuantisasi citra analog yang dibuat dalam bentuk digital dan direpresentasikan sebagai matriks dua dimensi (2D) dari bilangan real. Biasanya citra disimbulkan dengan *f (x, y)* yang merujuk pada citra monokrom berukuran *M* × *N*, di mana x menunjukkan nomor baris (dari 0 hingga M - 1) dan y mewakili nomor kolom (antara 0 dan N - 1).

$$f(x,y)=\begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ . & . & & . \\ . & . & & . \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{bmatrix}$$

a

b

Gambar 2. 1. a dan b merupakan ilustrasi matrik dan citra monokrom dengan ketentuan yang digunakan untuk mewakili baris (x) dan kolom (y)

Nilai fungsi dua dimensi $f(x,y)$ pada piksel koordinat tertentu (x_0, y_0), dilambangkan dengan $f(x_0, y_0)$ disebut intensitas atau *grayscale* dari citra pada piksel tersebut. Nilai maksimum dan minimum yang dapat diasumsikan oleh intensitas piksel akan bervariasi tergantung pada tipe data dan ketentuan yang digunakan. Rentang umum adalah sebagai berikut: 0,0 (hitam) hingga 1,0 (putih) untuk tipe data *double* dan 0 (hitam) hingga 255 (putih) untuk representasi uint8 (unsigned integer, 8 bit).

Ketentuan yang dinyatakan pada Gambar 2.1.a konsisten dengan bahasa pemrograman yang menggunakan notasi array berbasis 0 (seperti Java, C, C ++), tetapi tidak dengan MATLAB yang menggunakan notasi array berbasis 1. Setiap kali situasi menuntut disambiguasi eksplisit antara dua ketentuan yang saling bertentangan, maka notasi $f(p,q)$ berikut merujuk pada representasi MATLAB dari $f(x,y)$ (di mana p menunjukkan baris dan q menunjukkan kolom).

$$f(p,q) = \begin{bmatrix} f(1,1) & f(1,2) & \dots & f(1,N) \\ f(2,1) & f(2,2) & \dots & f(2,N) \\ . & . & & . \\ . & . & & . \\ f(M,1) & f(M,2) & \dots & f(M,N) \end{bmatrix}$$

Gambar 2. 2. Merupakan ilustrasi matrik dalam citra dengan pemprograman MATLAB

Citra monokrom pada dasarnya adalah matriks 2D (atau array). MATLAB memperlakukan matriks sebagai tipe data bawaan, lebih mudah untuk memanipulasinya tanpa menggunakan konstruksi

bahasa pemrograman yang umum (seperti dobel untuk loop untuk mengakses setiap elemen individu dalam array).

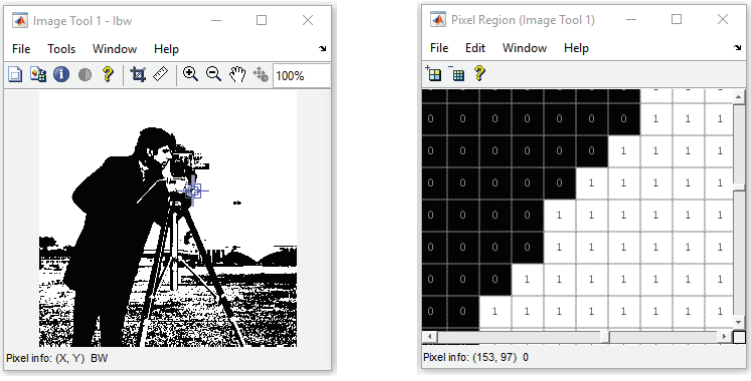
Citra direpresentasikan dalam format digital dalam berbagai cara. Pada tingkat paling dasar, ada dua cara pengkodean konten citra 2D dalam format digital: raster (juga dikenal sebagai bitmap) dan vektor. Representasi bitmap menggunakan satu atau lebih array dua dimensi piksel, sedangkan representasi vektor menggunakan serangkaian perintah mengcitra untuk mewakili citra. Setiap metode pengkodean memiliki pro dan kontra, dimana keuntungan terbesar dari grafik bitmap adalah kualitas dan kecepatan tampilan mereka, sedangkan kerugian utamanya termasuk persyaratan penyimpanan memori yang lebih besar dan ketergantungan ukuran (seperti memperbesar citra bitmap dapat menyebabkan artefak yang terlihat). Representasi vektor membutuhkan lebih sedikit memori dan memungkinkan pengubahan ukuran dan manipulasi geometris tanpa memperkenalkan artefak, tetapi perlu dirasterisasi untuk sebagian besar citraan perangkat.

Dalam kedua kasus, baik raster maupun vektor, tidak ada yang namanya citraan digital secara sempurna dari suatu citra. Terbentuknya artefak karena resolusi yang terbatas, pemetaan warna, dan banyak lainnya. Kunci untuk memilih citraan yang memadai adalah untuk menemukan kompromi yang cocok antara ukuran (dalam byte), kualitas subjektif, dan interoperabilitas dari format atau standar yang diadopsi.

2.2.1. Citra Biner (1-bit)

Citra biner dikodekan sebagai array 2D, biasanya menggunakan 1 bit per piksel, di mana 0 biasanya berarti “hitam” dan 1 berarti “putih”. Keuntungan utama dari citraan ini biasanya cocok untuk citra yang mengandung grafik sederhana, teks, atau garis seni adalah

ukurannya yang kecil. Citra 2.2 menunjukkan citra biner dan wilayah detail dalam ukuran $n x m$, di mana piksel dengan nilai 1 sesuai dengan tepi dan piksel dengan nilai 0 sesuai dengan latar belakang. Dalam MATLAB citra biner dilambangkan dalam pemprograman MATLAB menggunakan array logis dari 0 dan 1. Konversi dari numerik ke array logis dapat dicapai menggunakan fungsi *logical*.

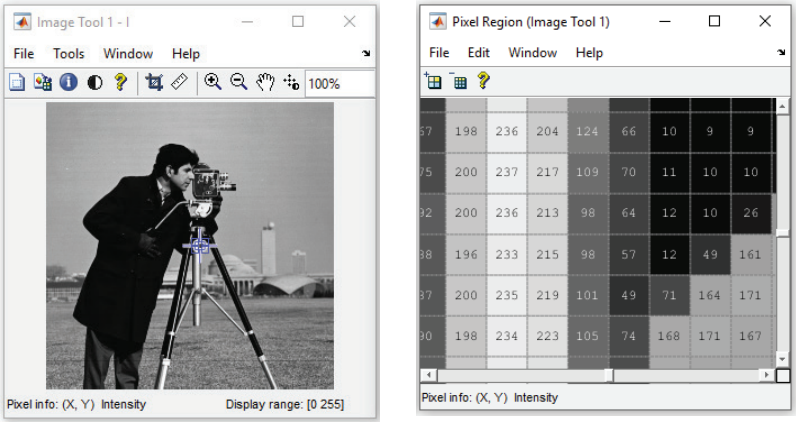


a b

Gambar 2. 3. Citra biner dan nilai piksel di area n × m

2.2.2. Citra grayscale (8-Bit)

Citra *grayscale* (monokrom) juga dikodekan sebagai array 2D piksel, biasanya dengan 8 bit per piksel, di mana nilai piksel 0 sesuai dengan "hitam" nilai piksel 255 berarti "putih" dan nilai-nilai antara menunjukkan berbagai variasi warna *grayscale*, yang diperlihatkan pada Gambar 2.4. Jumlah total *grayscale* lebih besar daripada persyaratan HVS, menjadikan format ini bekerja yang baik antara kualitas visual subjektif dan pengcitraan maupun penyimpanan yang relatif kompak.



a b

Gambar 2. 4. Citra grayscale dan nilai piksel di area n × m

2.2.3. Citra Berwarna (24-Bit atau RGB)

Representasi citra berwarna lebih kompleks dan bervariasi. Citra berwarna ini dapat ditunjukkan menggunakan tiga array 2D dengan ukuran yang sama, satu untuk setiap lapisan warna: merah (R), hijau (G), dan biru (B), yang diperlihatkan pada Gambar 2.5. Setiap larik elemen mengandung nilai 8-bit, yang menunjukkan jumlah merah, hijau, atau biru pada titik itu dalam skala [0, 255]. Kombinasi dari tiga nilai 8-bit ke dalam angka 24-bit memungkinkan 2^{24} (16.777.216, biasanya disebut sebagai kombinasi warna 16 juta). Representasi alternatif menggunakan 32 bit per piksel dan termasuk lapisan keempat, yang disebut lapisan alpha, yang menyediakan ukuran transparansi untuk setiap piksel dan banyak digunakan dalam efek pengeditan citra.

a b

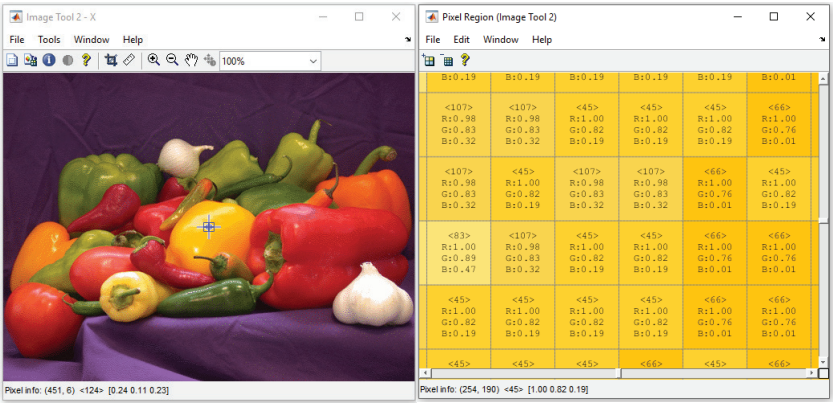
c d

Gambar 2. 5. Warna citra (a) dan komponen R (b), G (c), dan B (d)

2.2.4. Citra Warna Berindeks

Masalah dengan representasi warna 24-bit adalah kompatibilitas ke belakang dengan perangkat keras lama yang mungkin tidak dapat menampilkan 16 juta warna secara bersamaan. Sebuah solusi dibuat sebelum tampilan warna 24-bit dan kartu video tersedia secara luas yang terdiri dari representasi yang diindeks, di mana array 2D dengan ukuran yang sama dengan citra berisi indeks

ke peta warna dari ukuran maksimum tetap (biasanya 256 warna). Peta warna hanyalah daftar warna yang digunakan dalam citra itu. Contoh citra warna berindeks yang ditunjukkan pada Gambar 2.6.



a b

Gambar 2. 6. Citra warna yang diindeks dan indeks. Sumber: MathWorks

2.2.5. Kompresi

Representasi citra mentah biasanya memerlukan sejumlah besar ruang penyimpanan (dan waktu pengiriman yang lama dalam hal mengunggah / mengunduh file), sebagian besar format file citra menggunakan beberapa jenis kompresi. Metode kompresi dapat bersifat lossy ketika tingkat penurunan kualitas visual citra yang dihasilkan dapat diterima, atau lossless ketika citra dikodekan dalam kualitas penuhnya. Hasil keseluruhan dari proses kompresi dalam hal penghematan penyimpanan biasanya dinyatakan dalam rasio kompresi atau bit per pixel (bpp) dan hilangnya kualitas yang dihasilkan (untuk kasus teknik lossy) dapat bervariasi tergantung pada teknik, format, opsi (seperti pengaturan kualitas untuk JPEG), dan konten citra aktual. Sebagai pedoman umum, kompresi lossy harus digunakan untuk citra fotografi, sedangkan kompresi lossless

harus lebih disukai ketika berhadapan dengan seni garis, citra, faksimili, atau citra di mana tidak ada kehilangan detail yang dapat ditoleransi (terutama citra ruang) dan citra medis).

2.3. FORMAT FILE CITRA

Sebagian besar format file citra yang digunakan untuk mewakili citra bitmap terdiri dari header file diikuti oleh data piksel. Header file citra menyimpan informasi tentang citra, seperti tinggi dan lebar citra, jumlah bit per piksel, dan beberapa tanda yang menunjukkan jenis file. Dalam format file yang lebih kompleks, header juga dapat berisi informasi tentang jenis kompresi yang digunakan dan parameter lain yang diperlukan untuk memecahkan citra.

Format file paling sederhana adalah format BIN dan PPM. Format BIN hanya terdiri dari data piksel mentah, tanpa header apa pun. Akibatnya, pengguna file BIN harus mengetahui parameter citra yang relevan (seperti tinggi dan lebar) sebelumnya untuk menggunakan citra. Format PPM dan variannya (PBM untuk citra biner, PGM untuk citra *grayscale*, PPM untuk citra berwarna, dan PNM untuk lainnya) banyak digunakan dalam penelitian pemrosesan citra dan banyak alat untuk melakukan konversi format.

Format Microsoft Windows bitmap (BMP) adalah format lain yang banyak digunakan dan cukup sederhana, terdiri dari header diikuti oleh data piksel mentah. Format JPEG adalah format file paling populer untuk representasi citra dengan kualitas foto. Format ini mampu melakukan kompresi tingkat tinggi dengan kehilangan kualitas persepsi yang minimal.

GIF (Graphics Interchange Format) dan TIFF (Tagged Image File Format) merupakan dua format file citra lainnya sangat banyak digunakan dalam tugas pemrosesan citra. GIF menggunakan

representasi yang diindeks untuk citra berwarna (peta warna maksimum 256 warna), algoritme kompresi LZW (Lempel-Ziv-Welch), dan header 13-byte. TIFF adalah format yang lebih canggih dengan banyak opsi dan kemampuan, termasuk kemampuan untuk mewakili warna asli (24 bpp) serta dukungan untuk lima skema kompresi yang berbeda. Selain GIF dan TIFF, Portable Network Graphics (PNG) merupakan format file yang semakin populer dan mendukung citra berwarna yang diindeks.

Beberapa paket pemrosesan citra mengadopsi formatnya sendiri (terkadang eksklusif). Contohnya termasuk XCF (format citra asli dari program editing citra GIMP) dan RAW (yang merupakan format yang paling banyak diadopsi oleh produsen kamera).

2.4. TERMINOLOGI DASAR

Bagian ini memperkenalkan konsep dan terminologi penting yang digunakan untuk memahami dan mengekspresikan sifat-sifat suatu citra.

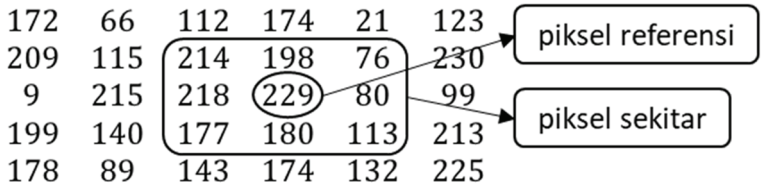
2.4.1. Topologi Citra

Melibatkan penyelidikan properti dasar citra yang biasanya dilakukan pada citra biner dan dengan bantuan operator morfologis seperti jumlah kemunculan objek tertentu, jumlah wilayah terpisah (tidak terhubung), dan jumlah lubang di suatu objek.

2.4.2. Ketetanggaan Piksel

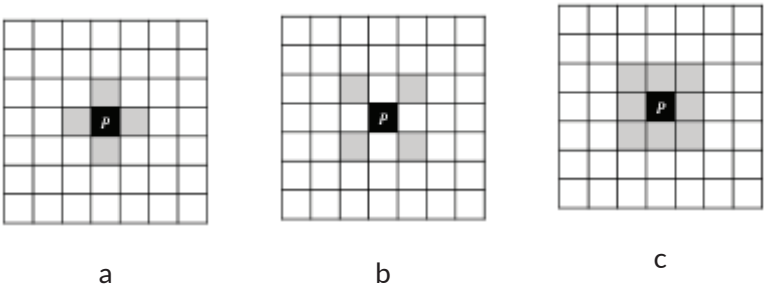
Pixel yang mengelilingi piksel tertentu dalam ketetanggaan, dapat diartikan sebagai matriks lebih kecil yang berisi (dan biasanya berpusat di sekitar) piksel referensi. Sebagian besar ketetanggaan yang digunakan dalam algoritme pemrosesan citra adalah array

persegi kecil dengan jumlah piksel ganjil, misalnya, ketetanggaan 3 × 3 yang ditunjukkan pada Gambar 2.7.



Gambar 2. 7. Piksel dalam ketetanggaan

Dalam konteks topologi citra, ketetanggaan mengambil makna yang sedikit berbeda. Hal ini umum, merujuk ke 4-ketetanggaan dari suatu piksel sebagai himpunan piksel yang terletak di atas, di bawah, ke kanan, dan di sebelah kiri piksel referensi (p), sedangkan himpunan semua tetangga terdekat p disebut sebagai sebagai 8-ketetanggaan seperti yang ditunjukkan pada Gambar 2.8. Pixel yang termasuk dalam 8-ketetanggaan, tetapi bukan 4-ketetanggaan, membentuk ketetanggaan diagonal p .



Gambar 2. 8. Konsep ketetanggaan piksel p (dari perspektif topologi citra): (a) 4-ketetanggaan; (b) ketetanggaan diagonal; (c) 8- ketetanggaan.

2.4.3. Kedekatan

Dalam konteks topologi citra, dua piksel p dan q berbatasan 4 jika 4 tetangga satu sama lain dan 8 berdekatan jika 8 tetangga satu sama lain. Tipe ketiga dari kedekatan dikenal sebagai kedekatan campuran (atau hanya adjacency-m) yang kadang-kadang digunakan untuk menghilangkan ambiguitas yang mungkin muncul ketika 8-ketetanggaan digunakan.

2.4.4. Jalur

Dalam konteks topologi citra, 4-jalur antara dua piksel p dan q adalah urutan piksel yang dimulai dengan p dan berakhir dengan q sedemikian rupa sehingga setiap piksel dalam urutan berdekatan dengan pendahulunya dalam urutan. Demikian pula, 8-jalur menunjukkan bahwa setiap piksel dalam urutan 8-berdekatan dengan pendahulunya.

2.4.5. Konektivitas

Jika ada 4 jalur antara piksel p dan q, konon ada 4 yang terhubung. Demikian pula, keberadaan 8 jalur di antara mereka berarti bahwa mereka terhubung 8.

2.4.6. Komponen

Satu set piksel yang terhubung satu sama lain disebut komponen. Jika piksel terhubung 4-ketetanggaan, ekspresi 4 komponen digunakan; Namun, jika pixel-nya terhubung-8, perangkat itu disebut komponen-8. Komponen diberi label dengan cara yang unik, menghasilkan citra berlabel, $L(x,y)$, yang nilai pikselnya adalah simbol dari alfabet yang dipilih. Nilai simbol suatu piksel biasanya menunjukkan hasil keputusan yang dibuat untuk piksel.

Penggunaan dalam MATLAB menggunakan IPT yang berisi *bwlabel* fungsi untuk memberi label komponen yang terhubung dalam citra biner. Fungsi terkait lainnya, *label2rgb*, yang membantu memvisualisasikan hasil dengan mengecat setiap wilayah dengan warna yang berbeda.

2.4.7. Jarak antara Piksel

Ada banyak aplikasi pemrosesan citra yang membutuhkan jarak pengukuran antar piksel. Ukuran jarak paling umum antara dua piksel *p* dan *q*, masing-masing koordinat $(x0, y0)$ dan $(x1, y1)$.

Jarak *Euclidean*

$$D_e(p,q) = \sqrt{(x1 - x0)^2 + (y1 - y0)^2} \quad (2.1)$$

Jarak *city-block*

$$D_e(p,q) = \left|(x1 - x0)^2\right| + \left|(y1 - y0)^2\right| \quad (2.2)$$

Jarak *Chebychef*

$$D_e(p,q) = max\left(\left|(x1 - x0)^2\right| + \left|(y1 - y0)^2\right|\right) \quad (2.3)$$

Penting untuk dicatat bahwa jarak antara dua piksel hanya bergantung pada koordinatnya, bukan nilainya. Satu-satunya pengecualian adalah jarak yang didefinisikan sebagai "jalur-m terpendek antara dua piksel yang terhubung m".

2.5. GAMBARAN UMUM OPERASI PENGOLAHAN CITRA

Bagian ini memperlihatkan kategori utama operasi pemrosesan citra. Meskipun tidak ada kesepakatan tentang taksonomi untuk bidang pengolahan citra, pengkategorian dilakukan sebagai berikut:

2.5.1. Operasi di Domain Spasial:

Merupakan perhitungan aritmatika dan/atau operasi logis dilakukan pada nilai piksel asli. Operasi ini menjadi menjadi tiga jenis:

- Operasi Global:

  merupakan operasi titik, di mana seluruh citra diperlakukan secara seragam dan nilai yang dihasilkan untuk piksel yang diproses adalah fungsi dari nilai aslinya, terlepas dari lokasinya di dalam citra. Contoh: penyesuaian kontras.

- Operasi Berorientasi kedekatan:

  Merupakan operasi lokal atau area, di mana citra input diperlakukan pada basis piksel-demi-piksel dan nilai yang dihasilkan untuk piksel yang diproses adalah fungsi dari nilai aslinya dan nilai tetangga terdekat. Contoh: filter spasial-domain.

- Operasi menggabungkan beberapa Citra:

  Dua citra atau lebih digunakan sebagai input dan hasilnya diperoleh dengan menerapkan (seri) aritmatika atau operator logika. Contoh: mengurangi satu citra dari yang lain untuk mendeteksi perbedaan di antara mereka.

2.5.2. Operasi dalam domain transform

Dalam operasi berdomain transform citra mengalami transformasi matematis seperti *transformasi fourier* (FT) atau *discrete*

cosine transform (DCT) dan algoritme pemrosesan citra berfungsi dalam domain transformasi. Contoh: teknik filter domain frekuensi.

2.5.3. Operasi global (titik)

Operasi titik menerapkan fungsi matematika yang sama, atau merupakan fungsi transformasi untuk semua piksel, terlepas dari lokasi dalam citra atau nilai tetangga. Fungsi transformasi dalam domain spasial dapat dinyatakan sebagai

$$g(x,y)=T\left[f(x,y)\right]$$

di mana g (x, y) adalah citra yang diproses, f (x, y) adalah citra asli, dan T adalah operator pada f (x, y). Karena koordinat aktual tidak memainkan peran apa pun dalam cara fungsi transformasi memproses citra asli, notasi dapat ditulis menggunakan:

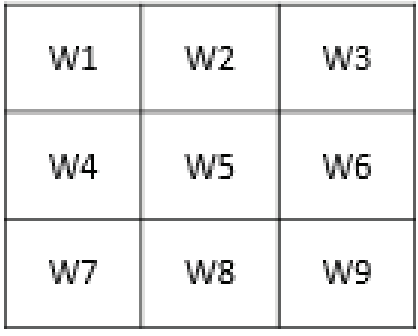
$$s=T\left[r\right]$$

di mana r adalah *grayscale* asli dan s adalah *grayscale* yang dihasilkan setelah pemrosesan.

2.5.4. Operasi berorientasi kedekatan

Operasi berorientasi kedekatan (lokal atau area) terdiri dari menentukan nilai piksel yang dihasilkan pada koordinat (x, y) sebagai fungsi dari nilai aslinya dan nilai (beberapa) tetangganya, biasanya menggunakan operasi konvolusi. Konvolusi citra sumber dengan array 2D kecil (window, template, mask, atau kernel) menghasilkan citra tujuan di mana setiap nilai piksel tergantung pada nilai aslinya dan nilai (beberapa) tetangganya. Kernel konvolusi menentukan tetangga mana yang digunakan serta bobot relatif dari nilai aslinya. Ukuran kernel biasanya 3 × 3 seperti Gambar 2.9 dengan setiap koefisien kernel (W1, ..., W9) diartikan sebagai bobot. Kernel dapat dianggap sebagai window kecil yang dilapis pada citra untuk

melakukan perhitungan pada satu piksel pada suatu waktu. Setiap piksel yang diproses akan bergerak ke piksel berikutnya dalam citra sumber dan proses diulangi hingga piksel terakhir diproses. Operasi konvolusi banyak digunakan dalam pemrosesan citra. Bergantung pada pilihan kernel, operasi dasar yang sama dapat digunakan untuk mengaburkan citra, menyempurnakan, menemukan tepin, atau menghilangkan noise.



Gambar 2. 9. Ukuran 3 × 3 kernel konvolusi, dengan bobot W1, ..., W9

2.5.5. Operasi Menggabungkan Banyak Citra

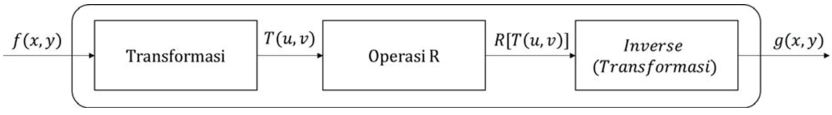
Ada banyak aplikasi pemrosesan citra yang menggabungkan dua citra, piksel demi piksel, menggunakan operator aritmatika atau logis, menghasilkan citra ketiga *Z*.

$$X\ opn\ Y = Z$$

di mana X dan Y dapat berupa citra (array) atau skalar, Z tentu merupakan array, dan opn adalah operator matematika biner (+, -, ×, /) atau logis (DAN, ATAU, XOR).

2.5.6. Operasi dalam Domain Transform

Transformasi adalah alat matematika yang memungkinkan konversi satu set nilai ke set nilai lain, sehingga menciptakan cara baru untuk merepresentasikan informasi yang sama. Di bidang pemrosesan citra, domain asli disebut sebagai domain spasial, sedangkan hasilnya dikatakan terletak pada domain transformasi. Motivasi untuk menggunakan transformasi matematis dalam pemrosesan citra berasal dari kenyataan bahwa beberapa tugas paling baik dilakukan dengan mentransformasikan citra input, menerapkan algoritme yang dipilih dalam domain transformasi, dan akhirnya menerapkan transformasi terbalik pada hasilnya (seperti pada Gambar 2.10).



Gambar 2. 10. Operasi transformasi domain

2.6. RANGKUMAN

- Citra direpresentasikan dalam format digital dalam berbagai cara. Representasi Bitmap (juga dikenal sebagai raster) menggunakan satu atau lebih array dua dimensi piksel (elemen citra), sedangkan representasi vektor menggunakan serangkaian perintah mengcitra untuk mewakili citra.
- Citra biner dikodekan sebagai array 2D, menggunakan 1 bit per piksel, di mana biasanya 0 tidak selalu berarti "hitam" dan 1 berarti "putih"
- Citra monokrom (*grayscale*) dikodekan sebagai array 2D piksel, menggunakan 8 bit per piksel, di mana nilai piksel 0 biasanya

berarti “hitam” dan nilai piksel 255 berarti “putih” dengan nilai menengah yang sesuai dengan berbagai nuansa *grayscale*.

- Dua cara paling umum untuk menyimpan konten citra warna adalah representasi RGB di mana setiap piksel biasanya diwakili oleh angka 24-bit yang mengandung jumlah komponen merah (R), hijau (G), dan biru (B) dan di representasi secara terindeks di mana array 2D berisi indeks ke peta warna.
- Beberapa format file citra paling populer yang digunakan saat ini adalah BMP, GIF, JPEG, TIFF, dan PNG.
- Fungsi bawaan MATLAB untuk membaca citra dari file dan menulis citra ke file (masing-masing *imread* dan *imwrite*) mendukung sebagian besar format file.
- Topologi citra adalah bidang pemrosesan citra yang berkaitan dengan investigasi properti citra mendasar (seperti jumlah komponen yang terhubung dan jumlah lubang pada suatu objek) menggunakan konsep seperti kedekatan dan konektivitas.
- Operasi pemrosesan citra dapat dibagi menjadi dua kelompok besar: domain spasial dan domain transformasi. Teknik spasial-domain dapat dibagi lagi menjadi operasi piksel-demi-piksel (titik) atau berorientasi area.

Bab 3

DASAR MATLAB

3.1. TUJUAN

Tujuan dari bahasan ini adalah

- mempelajari cara membuat, menginisialisasi, dan mengakses beberapa struktur data paling berguna yang tersedia di MATLAB untuk mengeksplorasi konsep pemrograman menggunakan skrip dan fungsi.
- mempelajari cara menggunakan MATLAB untuk perhitungan dasar yang melibatkan variabel, array, matriks, dan vektor serta menjelajahi array multidimensi dan cell.
- mempelajari cara menggunakan struktur MATLAB dengan menjelajahi fungsi-fungsi bermanfaat yang dapat digunakan dengan struktur data MATLAB
- mempelajari cara menulis dan menjalankan skrip dengan menjelajahi Editor MATLAB, fungsi dan cara menulisnya.
- mempelajari dasar-dasar vektorisasi loop.

3.2. PENDAHULUAN UNTUK MATLAB

MATLAB (MATrix LABoratory) adalah alat analisis data, dan visualisasi yang mendukung operasi matriks, kemampuan grafis, dan bahasa pemrograman tingkat tinggi serta area pengembangan. MATLAB telah menjadi sangat populer di berbagai kalangan seperti di kalangan insinyur, ilmuwan, dan peneliti baik peneliti di industri dan akademisi. Hal ini karena banyak faktor, seperti, ketersediaan beragam fungsi khusus yang dikemas dalam library atau *toolbox* MATLAB untuk banyak bidang penting mulai dari jaringan saraf hingga keuangan dan pengolahan citra. MATLAB memiliki dokumentasi lengkap yang berisi deskripsi fungsi utama MATLAB, contoh *code*, demo yang sesuai, dan halaman bantuan umum. Dokumentasi MATLAB dapat diakses dengan berbagai cara, mulai

dari bantuan hanya teks-baris perintah atau *command-line* hingga halaman HTML *hyperlink* (web MathWorks: http://www.mathworks.com).

Tipe data dasar MATLAB adalah matriks atau array. MATLAB tidak memerlukan dimensi atau alokasi memori sebelum penggunaan sesungguhnya. Semua data dianggap semacam matriks. Nilai tunggal dianggap oleh MATLAB sebagai matriks 1 × 1. Pengembangan algoritme MATLAB menyediakan antarmuka *command-line*, serangkaian fungsi untuk manipulasi numerik dan string, fungsi ploting 2D dan 3D, dan kemampuan untuk membangun antarmuka atau *graphical user interface*. Bahasa pemrograman MATLAB menginterpretasikan perintah, yang mempersingkat waktu pemrograman dengan menghilangkan butuhkan untuk kompilasi.

3.3. UNSUR DASAR MATLAB

Beberapa elemen dasar MATLAB seperti environment dari tool MATLAB, tipe data, dan operasi *command-line*.

3.3.1. Area Kerja

Area kerja MATLAB terdiri dari yang berikut:

- MATLAB Desktop:

  Ini biasanya berisi lima subwindows: *command window*, *workspace browser*, current directory window, command, history window, dan satu atau lebih figure windows, yang terlihat ketika pengguna menampilkan grafik, seperti plot atau citra.

- Editor MATLAB:

  Ini digunakan untuk membuat dan mengedit file-M, termasuk sejumlah fungsi yang berguna untuk menyimpan, melihat, dan men-debug file-M.

- Sistem Bantuan:

  Browser bantuan, yang ditampilkan dalam bentuk dokumen HTML dan berisi sejumlah opsi pencarian dan tampilan.

3.3.2. Jenis Data

MATLAB mendukung tipe data yang tercantum dalam Tabel 3.1. Terdapat delapan tipe data pertama yang tercantum dalam tabel dikenal sebagai kelas numerik. Nilai numerik dalam MATLAB adalah dari kelas *double* kecuali ditentukan lain. Konversi antara kelas data (juga dikenal sebagai *typecasting*) dan seringkali perlu. Sebuah string dalam MATLAB hanyalah array karakter 1 × n.

Tabel 3. 1 Tipe data dalam MATLAB

Tipe Data	Keterangan
uint8	8-bit unsigned intergers (1 byte per element)
uint16	16-bit unsigned intergers (2 byte per element)
uint32	32-bit unsigned intergers (4 byte per element)
int8	8-bit signed intergers (1 byte per element)
int16	16-bit signed intergers (2 byte per element)
int32	32-bit unsigned intergers (4 byte per element)
single	single-precision floating numbers (4 byte per element)
double	double-precesion floating numbers (8 byte per element)
logical	values are 0 (false) or 1 (true) (1 byte per element)
char	characters (2 bytes per element)

3.3.3. Array dan Index Matriks dalam MATLAB

Array MATLAB dapat vektor dan matriks yang diindeks menggunakan ketentuan berbasis 1. Oleh karena itu, *a*(1) adalah sintaks untuk merujuk ke elemen pertama dari array satu dimensi dan *f*(1,1) adalah sintaks untuk merujuk ke elemen pertama dari array dua dimensi, seperti kiri atas piksel dalam citra *f*. Operator titik dua (:) memiliki kemampuan pengindeksan yang kuat. MATLAB tidak membatasi jumlah dimensi array, tetapi memiliki batas jumlah maksimum elemen yang diizinkan dalam array. Jika Anda mengetikkan

```
[c, maxsize] = computer
```

Variabel *maxsize* akan berisi jumlah maksimum elemen yang diizinkan dalam sebuah matriks di komputer dan versi MATLAB yang digunakan.

3.3.4. Array Standar

MATLAB memiliki sejumlah array standar yang berguna seperti:

- $zeros(m,n)$ membentuk ukuran $m \times n$ matrix yang bernilai nol.
- $ones(m,n)$ membentuk ukuran $m \times n$ matrix yang bernilai satu.
- $true(m,n)$ membentuk ukuran $m \times n$ matrix yang bernilai logic satu.
- $false(m,n)$ membentuk ukuran $m \times n$ matrix yang bernilai logic nol.
- $eye(n)$ mengembalikan ukuran $n \times n$ matrix identitas.

- $magic(m)$ mengembalikan array dimana jumlah baris, kolom atau diagonal utama menghasilkan nilai sama pada ordo m.
- $rand(m,n)$ membentuk ukuran $m \times n$ matriks yang isinya adalah nomor pseudorandom yang didistribusikan secara merata dalam interval $[0,1]$.
- $randn(m,n)$ creates an $m \times n$ matriks yang isinya adalah nomor pseudorandom yang mengikuti normal (Gaussian) distribusi dengan rata-rata 0 dan varian 1.

3.3.5. Operasi Command-Line

Banyak fungsi yang tersedia di MATLAB dapat diakses dari *command-line*. Operasi *command-line* sama dengan mengeksekusi satu baris s, biasanya memanggil fungsi bawaan dengan mengetikkan nama dan parameternya pada *prompt* (>>). Jika baris tidak menyertakan variabel yang hasilnya harus ditetapkan, baris tersebut ditugaskan ke variabel ans (default sebagai variable jawaban). Jika garis tidak diakhiri dengan titik koma, hasilnya akan kembali ke *command-line*. Mempertimbangkan banyaknya fungsi yang tersedia dalam MATLAB, antarmuka *command-line* adalah cara yang sangat efektif untuk mengakses fungsi-fungsi ini tanpa harus menggunakan serangkaian menu dan kotak dialog yang kompleks. Waktu yang dihabiskan untuk mempelajari nama, sintaks, dan parameter fungsi yang berguna akan dihabiskan dengan baik, karena apa pun yang bekerja pada mode *command-line* (satu perintah pada satu waktu) juga akan berfungsi sebagai bagian dari program yang lebih besar atau fungsi yang ditentukan pengguna. Selain itu, MATLAB menyediakan fitur tambahan untuk membuat interaksi *command-line* lebih efektif, seperti kemampuan untuk dengan mudah mengakses (dengan tombol panah) operasi

sebelumnya (dan menghemat waktu pengetikan). Operasi *command-line* yang berhasil dapat dengan mudah dipilih dari *command-history* dan dapat digabungkan menjadi sebuah skrip.

3.4. ALAT PEMROGRAMAN: SKRIP DAN FUNGSI

Area pengembangan MATLAB menginterpretasikan perintah yang ditulis dalam bahasa pemrograman MATLAB, yang sangat nyaman ketika tujuan utamanya adalah membuat prototipe suatu algoritme dengan cepat. Setelah suatu algoritme stabil, ia dapat dikompilasi dengan kompiler MATLAB untuk eksekusi yang lebih cepat, yang sangat penting untuk set data besar. Kompiler MATLAB Coder untuk mengubah kode asli MATLAB menjadi kode C ++, mengkompilasi kode C ++, dan menautkannya dengan library MATLAB. Kode yang dikompilasi mungkin hingga 10 kali lebih cepat dari coding aslinya, tetapi faktor percepatan bergantung pada seberapa vektor kode aslinya; kode vektor sangat dioptimalkan mungkin tidak mengalami percepatan apa pun sama sekali. Untuk perhitungan lebih cepat, programmer dapat secara dinamis menghubungkan C sebagai fungsi MATLAB melalui utilitas MEX.

3.4.1. File-M

File-M dalam MATLAB dapat berupa skrip yang hanya menjalankan serangkaian perintah MATLAB (atau pernyataan) atau dapat berupa fungsi yang menerima argumen (parameter) dan menghasilkan satu atau beberapa nilai output. Fungsi yang dibuat pengguna memperluas kemampuan MATLAB dan *toolbox* untuk memenuhi kebutuhan. File-M yang berisi skrip terdiri dari urutan perintah yang harus ditafsirkan dan dieksekusi. Selain panggilan ke fungsi bawaan, skrip juga dapat berisi deklarasi variabel, panggilan ke fungsi yang dibuat pengguna (yang dapat disimpan dalam file

terpisah), pernyataan dan pengulangan. Skrip biasanya dibuat menggunakan editor teks (seperti Editor MATLAB bawaan) dan disimpan dengan nama yang bermakna dan ekstensi ".m". Setelah skrip dibuat dan disimpan, skrip dapat dipanggil dari *command-line* hanya dengan mengetik namanya.

3.4.2. Operator

Operator MATLAB seperti pada Tabel 3.2 dapat dikelompokkan ke dalam tiga kategori utama :

- Operator Aritmatika: Melakukan perhitungan numerik pada matriks.
- Operator Relasional: Bandingkan operasi.
- Operator Logika: Melakukan fungsi logis standar (seperti DAN, TIDAK, dan ATAU)

Tabel 3. 2 Operator Arithmetic MATLAB Array dan Matrix

Operator	Nama	Fungsi MATLAB
+	Penjumlahan array dan matrix	plus(a,b)
–	Pengurangan array dan matrix	minus(a,b)
.*	Perkalian array elemen demi elemen	times(a,b)
*	Perkalian Matrix	mtimes(a,b)
./	Pembagi array kanan	rdivide(a,b)
.\	Pembagi array kiri	ldivide(a,b)
/	Pembagi matrix kanan	mrdivide(a,b)
\	Pembagi matrix kiri	mldivide(a,b)

Operator	Nama	Fungsi MATLAB
.^	Pangkat pada array	power(a,b)
.^	Pangkat pada matrix	mpower(a,b)
.'	Transpose vector dan matrix	transpose(a)
'	Complex conjugate transpose pada vector and matrix	ctranspose(a)
+	Tambah Unary	uplus(a)
−	Kurang Unary	uminus(a)
:	Colon	colon(a,b) or colon(a,b,c)

MATLAB mempertimbangkan sebuah matriks dan tipe data standar, sehingga jumlah array dan operator matriks yang tersedia di MATLAB jauh melebihi operator tradisional yang ditemukan dalam bahasa pemrograman konvensional. Selain itu, MATLAB mencakup seperangkat operator logika standar. MATLAB juga mendukung fungsi logika dan sejumlah besar fungsi yang mengembalikan 1 (benar) atau 0 (salah) tergantung pada apakah nilai atau kondisi dalam argumen mereka benar atau salah, misalnya, isempty (a), tidak sama (a, b), isnumerik (a), dan banyak lainnya.

Tabel 3. 3 Operator Matrix spesial MATLAB

Operator atau Fungsi MATLAB	Keterangan
Apostrophe (') operator	Matrix transpose
inv function	Inversion
det function	Matrix determinant
flipud function	Flip up and down

Operator atau Fungsi MATLAB	Keterangan
fliplr function	Flip left and right
rot90 function	Matrix rotation
reshape function	Matrix reshape
trace function	Sum of the diagonal elements

Tabel 3. 4 Operator Matrix spesial MATLAB berdasar Toolbox Image Processing

Operator atau Fungsi MATLAB	Keterangan
imadd	Menambahkan dua citra atau konstanta ke sebuah citra
imsubtract	Mengurangkan dua citra atau konstanta ke sebuah citra
immultiply	Mengalikan dua citra (tiap elemen) atau konstanta ke sebuah citra
imdivide	Membagi dua citra (tiap elemen) atau konstanta ke sebuah citra
imabsdiff	Menghitung perbedaan absolut antara dua citra
imcomplement	Melengkapi citra
imlincomb	Menghitung kombinasi linear dari dua citra atau lebih

Tabel 3. 5 Operator Relasi

Operator MATLAB	Nama
<	Kurang dari
<=	Kurang dari atau sama dengan
>	Lebih besar dari
>=	Lebih besar dari atau sama dengan
==	Sama dengan

Operator MATLAB	Nama
~=	Tidak sama dengan

Tabel 3. 6 Operator atau Fungsi Logika

Operator atau Fungsi MATLAB	Keterangan
&	AND
|	OR
~	NOT
xor	exclusive-or (XOR)
all	Mengembalikan 1 jika semua elemen dalam vektor bukan nol atau 0 sebaliknya. Beroperasi secara kolom pada matriks
Any	Mengembalikan 1 jika salah satu elemen dalam vektor bukan nol atau 0 sebaliknya. Beroperasi secara kolom pada matriks

3.4.3. Variabel dan Konstanta

Beberapa variable dan konstanta yang dimiliki oleh MATLAB.

Tabel 3. 7 Variabel dan Konstanta

Nama	Keterangan
ans	Jawaban terbaru
eps	Akurasi relatif floating-point
$i(or\ j)$	Unit imajine
$NaN\ (or\ nan)$	Bukan-angka (seperti hasil 0/0)

Nama	Keterangan
Inf	Infinity (seperti hasil pembagian oleh 0)

3.4.4. Representasi Angka

MATLAB dapat mewakili angka dalam notasi desimal konvensional (dengan titik desimal yang dapat memilih dan tanda tambah atau minus) serta dalam notasi ilmiah (menggunakan huruf e untuk menentukan eksponen pangkat dari 10). Bilangan kompleks direpresentasikan menggunakan *i* atau *j* sebagai akhiran untuk bagian imajiner.

3.4.5. Kontrol Aliran (Flow Control)

Bahasa pemrograman MATLAB mendukung pernyataan kontrol aliran yang biasa ditemukan di sebagian besar bahasa pemrograman tingkat tinggi kontemporer lainnya: pernyataan pemilihan keputusan dengan *if* (dengan opsional *else* dan *elseif*) dan *switch*, perulangan *for* dan *while* terkait dengan *break* dan *continue*, dan *try... catch* untuk penanganan kesalahan.

3.4.6. Optimasi Kode

Sebagai hasil dari sifat MATLAB yang berorientasi matriks, bahasa pemrograman MATLAB adalah bahasa yang divektorkan, yang berarti dapat melakukan banyak operasi pada angka yang dikelompokkan sebagai vektor atau matriks tanpa pernyataan loop secara eksplisit. Kode vektor lebih kompak, lebih efisien, dan dapat diparalelkan. Selain menggunakan loop vektor, programmer MATLAB didorong untuk menggunakan trik optimasi lainnya, seperti alokasi awal memori yang digunakan oleh array. Hal ini memberikan pengaruh terhadap kecepatan eksekusi kode MATLAB.

3.4.7. Input dan Output

Fungsi input dan output dasar dapat diperoleh melalui fungsi input (untuk meminta input pengguna dan membaca data dari keyboard) dan disp (untuk menampilkan teks atau array di layar). MATLAB juga mengandung banyak fungsi pendukung untuk membaca dan menulis ke file.

3.5. GRAFIS DAN VISUALISASI

MATLAB memiliki kumpulan fungsi dasar untuk merencanakan grafik 2D dan 3D. MATLAB juga mencakup sejumlah fungsi bawaan untuk menampilkan (dan memeriksa isi) citra.

3.6. TUTORIAL STRUKTUR MATLAB

Pada latihan berikut, setiap pertanyaaan terdapat prosedur yang harus dilakukan, lakukan prosedur tersebut dan jawab pertanyaan yang muncul.

Prosedur 1

Jalankan baris kode berikut satu per satu di Command Window untuk melihat bagaimana MATLAB dapat digunakan sebagai kalkulator.

```
1+2
1*2 + 3*4 + 5*6;
```

Pertanyaan 1 **:** Untuk apa variabel *ans* digunakan?

Pertanyaan 2 **:** Apa tujuan menggunakan tanda titik koma (;) di akhir pernyataan?

Prosedur 2

Lakukan perhitungan menggunakan variabel.

```
buah_per_box = 20;
jumlah_dalam_box = 5;
total_jumlah_buah = buah_per_box * jumlah_dalam_box
```

Pertanyaan 3 : Eksperimen dengan membuat variabel Anda sendiri. apakah variabel case sensitive?

Pertanyaan 4 : Apa nilai / tujuan dari variabel-variabel ini: pi, eps, inf, i? Apakah mungkin untuk menimpa variabel-variabel ini? Jika demikian, bagaimana bisa dibatalkan?

Prosedur 3

Jalankan perintah *who* dan *whos*, satu per satu, untuk melihat fungsinya dan apa perbedaan di antara mereka?

Catatan : Ada beberapa perintah yang akan menjaga area MATLAB tetap bersih. Gunakan mereka setiap kali Anda merasa command window atau workspace Anda terasa penuh dengan pernyataan dan variabel.

Prosedur 4

Bersihkan variabel di ruang kerja. Setelah eksekusi, perhatikan bagaimana variabel menghilang dari ruang kerja.

```
clear buah_per_box
```

Prosedur 5

Kosongkan command window dan semua variabel dengan baris kode berikut (satu per satu untuk melihat pengaruhnya secara individual)

```
clc
clear all
```

Prosedur 6

Create a 3×3 matrix by executing the following line of code.

```
A = [123;456;789]
```

Pertanyaan 5 : Apa gunanya titik koma dalam pernyataan ini?

Prosedur 7

Operator yang sangat berguna di MATLAB adalah titik dua (:). Dapat digunakan untuk membuat vektor angka.

```
1 : 5
```

Prosedur 8

Parameter ketiga menentukan cara menghitung antara angka awal dan akhir. Ini ditentukan antara nilai awal dan akhir.

```
1: 1: 5
5: -1: 1
1: 2: 9
9: -2: 1
```

Pertanyaan 6 : Tulis pernyataan yang akan menghasilkan vektor nilai mulai dari 0 hingga π dalam kelipatan π / 4.

Prosedur 9

Operator titik dua juga dapat digunakan untuk mengembalikan seluruh baris atau kolom dari sebuah matriks.

```
A = [123;456;789]
A(:,1) A(1,:)
```

Pertanyaan 7 :

Tulis satu baris kode yang akan menghasilkan matriks 3 × 3 yang sama seperti pada variabel A di atas, tetapi menggunakan operator titik dua untuk menghasilkan urutan angka di setiap baris alih-alih menuliskannya secara eksplisit.

Prosedur 10

Operator titik dua dapat diganti dengan fungsi titik dua, yang melakukan operasi yang sama.

```
colon(1,5)
```

Catatan : Seperti terlihat pada langkah-langkah di atas, membuat vektor angka-angka yang diberi spasi secara merata mudah dilakukan dengan operator titik dua (:) ketika mengetahui di mana vektor dimulai, berakhir, dan seberapa besar ruang di antara masing-masing nilai. Dalam kasus tertentu, mungkin ingin membuat vektor angka yang berkisar antara dua angka, tetapi hanya tahu jumlah nilai yang dibutuhkan (misalnya, membuat vektor yang terdiri dari 4 nilai antara π / 4 dan π). Untuk melakukan ini, menggunakan fungsi linspace.

Prosedur 11

Jalankan perintah ini untuk melihat bagaimana fungsi linspace beroperasi.

```
linspace(pi / 4, pi, 4)
```

Prosedur 12

Bandingkan hasil dari langkah sebelumnya dengan nilai-nilai ini

```
pi / 4
pi / 2
3*pi / 4
pi
```

Catatan: MATLAB memiliki beberapa fungsi bawaan yang akan menghasilkan matriks yang sering digunakan secara otomatis.

Prosedur 13

Jalankan baris kode berikut satu per satu.

```
zeros(3,4)

ones(3,4)

ones(3,4) *10

rand(3,4)

randn(3,4)
```

Pertanyaan 8 : Apa perbedaan antara fungsi rand (M, N) dan randn (M, N)?

Catatan: Pada penggabungan matriks, rangkaian matriks dilakukan dengan tanda kurung ([]) atau menggunakan fungsi cat.

A = [1 2 3; 4 5 6; 7 8 9].

Pada pernyataan tersebut, kurung menggabungkan tiga baris. Alih-alih secara eksplisit mendefinisikan setiap baris sekaligus, mereka dapat didefinisikan secara individual sebagai vektor dan kemudian digabungkan ke dalam matriks menggunakan tanda kurung.

Prosedur 14

Gabungkan tiga vektor individu ke dalam matriks 3 × 3.

```
X = [1 2 3]; Y = [4 5 6]; Z = [7 8 9];
A = [X; Y; Z]
B = cat (1, X, Y, Z)
```

Catatan: tanda kurung dapat digunakan untuk menghapus deretan matriks.

Prosedur 15

Hapus baris terakhir (baris 3) dari matriks A. Perhatikan bagaimana operator titik dua digunakan untuk menentukan seluruh baris.

```
A (3, :) = []
```

Catatan: Vektor dengan elemen N adalah array dengan satu baris dan kolom N. Unsur vektor dapat diakses dengan mudah dengan mengatasi jumlah elemen, seperti pada X (5), yang akan mengakses elemen kelima vektor X. Unsur matriks dua dimensi diakses dengan terlebih dahulu menentukan baris, maka kolom, seperti pada X (2,5), yang akan mengembalikan elemen pada baris 2, kolom 5. Matriks dimensi yang lebih tinggi dari 2 dapat diakses dengan cara yang sama. Merupakan hal yang relevan jika array dalam MATLAB adalah berbasis 1 elemen pertama dari sebuah array ditugaskan atau diakses menggunakan 1, sebagai lawan dari 0, yang merupakan standar dalam banyak bahasa pemrograman.

Prosedur 16

Gunakan yang dan fungsi rand untuk membuat array multidimensi.

```
A = yang(4,3,2);
B = rand(5,2,3);
size(A)
size(B)
disp(A)
disp(B)
```

Pertanyaan 9 ***:*** *Apa fungsi ukuran lakukan?*

Pertanyaan 10 ***:*** *Apa fungsi fungsi disp?*

Catatan: Pada operasi yang melibatkan matriks, aritmatika dapat dicapai dengan operator + – * / . Default untuk beberapa operator (*) dan divide (/) adalah perkalian matriks dan pembagian matriks. Untuk melakukan operasi aritmatika pada elemen individu dari sebuah matriks, awali operator dengan titik (.).

Prosedur 17

Lakukan perkalian matriks pada dua matriks.

```
X = [12 –2; 0 –34; 730]
Y = [10 –1; 23 –5; 135]
X * Y
```

Prosedur 18

Lakukan penggandaan elemen demi elemen.

$$X. * Y$$

Prosedur 19

Lakukan perkalian matriks lain pada dua matriks.

$$X = \text{eye}(3,4)$$
$$Y = \text{rand}(4,2)$$
$$X * Y$$
$$Y * X$$

Pertanyaan 11 **:**
Mengapa operasi terakhir gagal?

Prosedur 20

Gunakan fungsi diag dan lacak untuk melakukan operasi pada diagonal matriks.

$$Y = \text{rand}(3,3) * 4$$
$$Y_{\text{diag}} = \text{diag}(Y)$$
$$Y_{\text{trace}} = \text{trace}(Y)$$

Pertanyaan 12 **:** *Apa kegunaan fungsi diag?*

Pertanyaan 13 **:** *Apa kegunaan fungsi trace?*

Pertanyaan 14 **:** *Tulis pernyataan alternatif yang akan menghasilkan hasil yang sama dengan fungsi trace.*

Prosedur 21

Hitung transpose dari sebuah matriks.

$$Y$$
$$Y_t = Y$$

Prosedur 22

Hitung kebalikan dari sebuah matriks dan perlihatkan bahwa $YY-1=Y-1Y=I$, di mana I adalah matriks identitas.

$$Y_{inv} = \text{inv}(Y)$$
$$Y * Y_{inv}$$
$$Y_{inv} * Y$$

Prosedur 23

Hitung determinant dari matriks.

$$Y\_\det = \det(Y)$$

Catatan: Untuk penggunaan *cell array*, matriks adalah tipe data mendasar dalam MATLAB. Ini menyerupai definisi klasik dari array sebagai struktur data yang homogen, yaitu, di mana semua komponennya memiliki tipe yang sama. Cell Array, di sisi lain, adalah jenis array lain di mana setiap cell dapat berupa tipe data apa pun yang diizinkan oleh MATLAB. Setiap cell tidak bergantung dari yang lain, dan dapat berisi tipe data apa pun yang didukung MATLAB. Saat menggunakan cell array, seseorang harus berhati-hati saat mengakses atau menetapkan nilai, alih-alih parentheses, kurung kurawal ({}) harus digunakan.

Prosedur 24

Jalankan baris kode berikut satu per satu untuk melihat bagaimana *cell* array ditangani di MATLAB.

```
X{1} = [123;456;789]; %Cell1 is a matrix
X{2} = 2+3i; %Cell2 is complex
X{3} = 'String'; %Cell3 is a string
X{4} = 1:2:9; %Cell4 is a vector
X
celldisp(X)
X(1)
X{1}
```

Pertanyaan 15 *: Apa kegunaan dari fungsi celldisp?*

Pertanyaan 16 *: Apa yang dilakukan persen (%) pada karakter?*

Pertanyaan 17 *: Apa perbedaan antara dua baris terakhir dalam kode di atas (X (1) dibandingkan dengan X {1})?*

Catatan: Terdapat cara lain untuk menetapkan nilai ke cell array yang secara sintaksis berbeda, tetapi menghasilkan hasil yang sama. Perhatikan pada langkah selanjutnya bagaimana cell index terlampir di dalam tanda kurung normal (()), tetapi data yang akan disimpan ke *cell* dienkapsulasi oleh kurung kurawal ({}).

Prosedur 25

Jalankan baris ini untuk melihat cara lain untuk menetapkan nilai *cell array*.

```
X(1) = {[123; 456; 789]};
```

Prosedur 26

Beberapa baris kode berikutnya akan menunjukkan cara penugasan cell array yang tepat dan tidak tepat saat berurusan dengan string.

$$X(3) = \text{'Ini menghasilkan kesalahan'}$$

$$X(3) = \{\text{'Ini baik} - \text{baik saja '}\}$$

$$X\{3\} = \text{' Ini juga tidak apa} - \text{apa'}$$

Catatan: Struktur adalah cara lain untuk menyimpan data di MATLAB. Sintaks untuk struktur mirip dengan bahasa pemrograman lain. MATLAB menggunakan operator titik (.) untuk merujuk ke bidang yang berbeda dalam suatu struktur. Struktur dengan tata letak yang identik (jumlah bidang, nama, ukuran, dan makna) dapat digabungkan dalam array (struktur).

Prosedur 27

Buat array dari dua struktur yang mewakili dua citra dan ukurannya.

$$my\_images(1).\,imagename = \text{'Gambar1';}$$

$$my\_images(1).\,width = 256;$$

$$my\_images(1).\,height = 256;$$

$$my\_images(2).\,imagename = \text{'Gambar2';}$$

$$my\_images(2).\,width = 128;$$

$$my\_images(2).\,height = 128;$$

Prosedur 28

Lihat detail tentang struktur dan tampilkan konten bidang.

$$my\_images(1)$$

$$my\_images(2).imagename$$

Prosedur 29

Tampilkan informasi tentang struktur.

$$num\_of\_images = prod(size(my\_images))$$
$$fieldnames(my\_images)$$
$$class(my\_images)$$
$$isstruct(my\_images)$$
$$isstruct(num\_of\_images)$$

Pertanyaan 18 *: Apa kegunaan dari fungsi fieldnames?*

Pertanyaan 19 *: Apa artinya ketika hasil dari fungsi isstruct adalah 1? dan apa artinya ketika 0?*

Pertanyaan 20 *: Gunakan sistem bantuan untuk menentukan fungsi apa yang dapat digunakan untuk menghapus bidang dari suatu struktur.*

3.7. TUTORIAL PEMPROGRAMAN MATLAB

Pada latihan berikut, setiap pertanyaaan terdapat prosedur yang harus dilakukan, lakukan prosedur tersebut dan jawab pertanyaan yang muncul.

Catatan: Meskipun command window sederhana dan nyaman untuk digunakan, tetapi tidak menyediakan cara untuk menyimpan atau mengedit code. Namun, perintah yang disimpan di *command history* dapat dengan mudah dibuat menjadi file skrip. Script adalah file teks dengan ekstensi '.m' dan digunakan untuk menyimpan kode MATLAB. Script dapat dibuat dan dimodifikasi menggunakan editor bawaan.

Prosedur 1

Untuk memulai skrip baru, navigasikan ke File > New > M-File.

Catatan: Editor MATLAB dapat terbuka di jendela yang terpisah atau mungkin merapat di dalam area MATLAB, tergantung pada bagaimana tampilan area Anda diatur.

Prosedur 2

Buka file anda yang berekstensi '.m'. Jika file terletak di direktori saat ini, Anda dapat membukanya dari daftar direktori saat ini dengan mengklik ganda nama file.

Catatan: File-file adalah kode warna sintaks untuk membantu membacanya. Komentar dapat ditambahkan ke skrip apa pun menggunakan dua metode: tanda persen (%), atau balut komentar dengan% {dan%}. Metode kedua digunakan untuk informasi header dalam skrip.

Pertanyaan 1 : Berdasarkan skrip file '.m', apa perbedaan antara menggunakan tanda persen (%) dan menggunakan % {*and* %}?

Catatan: Untuk menjalankan satu atau beberapa baris kode dalam sebuah skrip, sorot kode tersebut dan tekan F9.

Prosedur 3

Pastikan bahwa file '.m' terletak di direktori yang merupakan bagian dari set path atau direktori saat ini.

Prosedur 4

Lihat informasi bantuan untuk fungsi dengan mengetik pernyataan berikut ke dalam *command window*:

```
help
```

Pertanyaan 2 : Bandingkan dokumentasi bantuan dengan komentar di File. Bagaimana MATLAB menentukan komentar mana yang akan ditampilkan untuk bantuan dan mana yang hanya komentar dalam kode fungsi?

Catatan: Dalam memanipulasi citra menggunakan MATLAB, penting untuk dilakukan dalam memanfaatkan CPU dan praktik pengkodean yang efisien-memori, khususnya dalam vektorisasi loop.

```
MAX_CNT = 10000
for i = 1 toMAX_CNT
do x(i) = i ^ 2
```

Catatan: Hal tersebut dilakukan dengan mengisi array di mana setiap elemen adalah indeks dari elemen yang kuadrat. Kode MATLAB berikut mengimplementasikan pseudocode di atas menggunakan teknik pemrograman khas. Dalam contoh-contoh ini, perintah *tic* dan *toc* digunakan untuk menghitung waktu eksekusi.

Prosedur 5

Laksanakan pseudocode di atas dengan pernyataan berikut.

```
tic
MAX_CNT = 10.000
```

```
for i = 1: MAX_CNT
x (i) = i ^ 2;
end
toc
```

Catatan: Hal ini sangat memengaruhi kecepatan loop ini dengan hanya mengalokasikan memori yang digunakan oleh array. Ini efektif karena setiap kali menambahkan data ke matriks, memori baru harus dialokasikan untuk menampung variabel yang lebih besar jika tidak ada ruang untuk itu. Jika jumlah memori yang benar sudah dialokasikan, maka MATLAB hanya perlu mengubah data di setiap *cell*.

Prosedur 6

Terapkan loop sebelumnya, tetapi dengan preallokasi.

```
tic
MAX_CNT = 10000
x = zeros (1, MAX_CNT);
for i = 1: MAX_CNT
x (i) = i ^ 2;
end
toc
```

Pertanyaan 3 : Dengan faktor apa alokasi awal array x dapat meningkatkan kinerja?

Catatan: Dalam MATLAB, ini masih dianggap teknik pemrograman yang buruk. MATLAB bertindak sebagai juru bahasa antara kode yang ditulis dengan perangkat keras komputer. Ini berarti bahwa setiap kali sebuah pernyataan ditemui, itu ditafsirkan ke bahasa tingkat yang lebih rendah yang dipahami oleh perangkat keras. Karena itu, loop, seperti yang di atas, menyebabkan MATLAB untuk

menginterpretasikan pernyataan dalam loop, namun, berkali-kali loop akan dijalankan dalam kasus di atas atau 10.000 kali. Interpretasi akan lambat jika dibandingkan dengan kecepatan perangkat keras komputer. Karena MATLAB beroperasi secara negatif, sehingga dapat melakukan operasi yang sama menggunakan *loop vectorization*, sehingga menghilangkan loop.

Prosedur 7

Terapkan versi loop yang lebih efisien.

```
tic
MAX_CNT = 10000
i = 1: MAX_CNT;
x = i. ^ 2;
toc
```

Pertanyaan 7 : Dalam kode tersebut, mengapa menggunakan operator .^ bukan hanya ^ ?

Pertanyaan 8 : Seberapa cepat loop vectorization dari dua implementasi tersebut?

Catatan: Tidak perlu secara eksplisit memberitahu MATLAB untuk melakukan operasi pada setiap elemen karena itu adalah sifat dari area MATLAB. *Loop vectorization* juga menangani masalah pra-alokasi yang dilihat dalam implementasi pertama.

Pertanyaan 9 : Pertimbangkan psedocode berikut. Tuliskan persamaan vektor.

```
i=0
for t = 0 to 2*pi in steps of pi/4
do i = i + 1
x(i) = sin(t)
```

Prosedur 8

Untuk membuat skrip dari *command history*, cari empat pernyataan terakhir yang dimasukkan. Untuk memilih keempat pernyataan, tahan tombol ctrl dan pilih masing-masing pernyataan, lalu klik kanan pernyataan yang disorot, dan pilih buat File-M.

3.8. LATIHAN PERMASALAHAN

1. Dengan menggunakan MATLAB sebagai kalkulator, lakukan perhitungan berikut:
   a. 24.4 * 365
   b. $cos\left(\pi / 4\right)$
   c. $\sqrt[3]{45}$
   d. $e^{-0.6}$
   e. 4.6^5
   f. $y = \sum_{i=1}^{6} i^2 - 3$
2. Gunakan fungsi format untuk menjawab pertanyaan-pertanyaan berikut:
   a. Apa presisi yang digunakan oleh MATLAB untuk *floating-point calculations*?
   b. Berapakah jumlah default tempat desimal yang digunakan untuk menampilkan hasil *floating-point calculations* di *command window*?
   c. Bagaimana Anda bisa mengubahnya ke tempat desimal yang berbeda?
3. Diketahui matrix sebagai berikut:

$$X = \begin{bmatrix} 4 & 5 & 1 \\ 0 & 2 & 4 \\ 3 & 4 & 1 \end{bmatrix}; \quad Y = \begin{bmatrix} 0 & 7 & 1 \\ 0 & 2 & -5 \\ 3 & 3 & -1 \end{bmatrix};$$

Gunakan kedua matrix tersebut untuk menghitung:

a. $3X + 2Y$

b. $X^2 - Y^3$

c. $X^T \; Y^T$

d. $X \; Y^{-1}$

4. Apa perbedaan antara fungsi-fungsi MATLAB berikut:
   a. log dan log10
   b. rand dan randn
   c. power dan mpower
   d. uminus dan minus
5. Apa tujuan dari fungsi *lookfor*? Berikan contoh yang bisa bermanfaat.
6. Apa tujuan dari fungsi *which*? Berikan contoh yang bisa bermanfaat.

Bab 4

PROSES PENGOLAHAN CITRA DENGAN MATLAB TOOLBOX

4.1. TUJUAN

Tujuan dari bahasan ini adalah

- Mempelajari cara membaca citra, konversi citra, menampilkan citra dan menulis citra ke disk dalam aplikasi MATLAB.
- Mempelajari berbagai jenis citra yang didukung oleh MATLAB dan IPT.
- mengeksplorasi teknik manipulasi citra dasar menggunakan MATLAB dan IPT

4.2. ALAT PENGOLAHAN CITRA: GAMBARAN UMUM

Image Processing Toolbox (IPT) adalah kumpulan fungsi yang memperluas kemampuan dasar area MATLAB untuk mengaktifkan operasi pengolahan sinyal dan citra khusus, seperti berikut ini:

- Transformasi spasial
- Analisis dan peningkatan citra
- Operasi area dan blokir
- Desain saringan linier dan filter
- Transformasi matematis
- Deblurring
- Operasi morfologis
- Pengolahan citra berwarna

Sebagian besar fungsi toolbox adalah M-file MATLAB, yang kode sumbernya dapat diperiksa dengan membuka file tersebut menggunakan editor MATLAB atau cukup mengetikkan function_ name saat diminta. Direktori di mana file file telah disimpan dapat ditemukan dengan mengetikkan function_name mana pada prompt.

Untuk menentukan versi IPT mana yang diinstal pada sistem Anda, ketikkan ver saat diminta. Anda dapat memperluas kemampuan IPT dengan menulis file Anda sendiri, memodifikasi dan memperluas file yang ada, menulis fungsi pembungkus di sekitar yang sudah ada, atau menggunakan IPT dalam kombinasi dengan toolbox lainnya.

4.3. FUNGSI DAN FITUR PENTING

Bagian ini menyajikan fungsi IPT yang digunakan untuk melakukan operasi citra esensial, seperti membaca konten citra dari file, mengkonversi antara kelas data yang berbeda yang digunakan untuk menyimpan nilai piksel, menampilkan citra di layar, dan menyimpan citra ke file.

4.3.1. Menampilkan Informasi Tentang File Citra

IPT MATLAB memiliki fungsi bawaan untuk menampilkan informasi tentang file citra (tanpa membukanya dan menyimpan kontennya di ruang kerja) *imfinfo*.

Contoh 1 **:** Kode MATLAB di bawah ini membaca informasi tentang citra bawaan file 'pout.tif'.

```
imfinfo ('pout.tif');
```

Struktur yang dihasilkan (disimpan dalam variabel dan) akan berisi beberapa informasi. Hasil yang diperlihatkan terlalu teknis dan sebagian lagi bergantung pada format file. Meskipun demikian, masih harus dapat menemukan informasi tentang ukuran citra (240 × 291 piksel), ukuran file (69.004 byte), jenis citra (*grayscale*), jumlah bit per piksel (8), dan nilai minimum dan maksimum (0 dan 255).

4.3.2. Membaca File Citra

IPT MATTAB memiliki fungsi bawaan untuk membuka dan membaca konten file citra dalam format paling populer (seperti TIFF, JPEG, BMP, GIF, dan PNG), *imread*. Fungsi *imread* memungkinkan membaca file citra dari hampir semua jenis, hampir semua format, yang terletak di mana saja. Hal ini berpotensi besar terhadap masalah yang terkait dengan header file, alokasi memori, ketentuan format file, dan sebagainya, dan memungkinkan untuk fokus pada apa yang ingin dilakukan terhadap citra setelah citra dibaca dan disimpan ke dalam variabel di ruang kerja MATLAB. IPT juga berisi fungsi-fungsi khusus untuk membaca file DICOM (Digital Imaging and Communications in Medicine), file (*dicomread*), file NITF (National Imagery Transmission Format) (*nitfread*), dan file HDR (high dynamic range) (*hdrread*).

Tabel 4. 1 Fungsi IPT untuk Melakukan Konversi Kelas Data Citra

Nama	Keterangan
$im2single$	single
$im2double$	double
$im2uint8$	uint8
$im2uint16$	uint16
$im2int16$	int16

4.3.3. Kelas Data dan Konversi Data

Kemampuan IPT MATLAB untuk membaca citra jenis apa pun, menyimpan kontennya ke dalam array, dan membuatnya tersedia untuk diproses lebih lanjut dan tampilan tidak menghalangi

kebutuhan untuk memahami bagaimana konten citra diwakili dalam memori. Kelas data yang paling umum untuk citra adalah sebagai berikut:

- *uint8*: 1 byte per piksel, dalam rentang [0, 255]
- *double*: 8 bytes per piksel, biasanya dalam kisaran [0,0, 1.0]
- *logical*: 1 byte per piksel, mewakili nilainya sebagai true (1 atau putih) atau false (0 atau hitam)

Setelah konten citra dibaca dan disimpan dalam satu atau lebih variabel, memeriksa kelas data dari variabel-variabel dan rentang nilainya untuk memahami bagaimana konten piksel direpresentasikan dan berapa kisaran nilai yang diizinkan. Dengan memastikan bahwa kelas data dari variabel kompatibel dengan kelas input data yang diharapkan oleh fungsi IPT yang akan diterapkan pada variabel itu. Jika menulis fungsi dan skrip, harus memastikan kompatibilitas kelas data, atau melakukan konversi yang diperlukan.

MATLAB memungkinkan konversi kelas data (*typecasting*) dilakukan secara langsung, tetapi jenis konversi tidak menangani masalah rentang. Kelas data input untuk fungsi-fungsi tersebut dapat berupa *logical*, *uint8*, *uint16*, *int16*, *single*, atau *double*.

Contoh 2 : Dimungkinkan untuk mengubah susunan 2 × 2 dari kelas *double* (A) menjadi *uint8*(B) hanya dengan mengetikkannya

$$A = \left[-12; 0\,0.5\right]$$

$$B = \text{uint8}\left(A\right)$$

```
>> A=[-1 2;0 0.5]

A =

   -1.0000    2.0000
         0    0.5000

>>  B = uint8(A)

B =

  2×2 uint8 matrix

   0   2
   0   1
```

Konversi terdiri dari pemotongan (semua nilai negatif menjadi nol) dan pembulatan (0,5 menjadi 1). Menggunakan im2uint8 akan menghasilkan hasil dalam rentang yang dinormalisasi ([0, 255]).

$$C = \text{im2uint8}(A)$$

```
>> C = im2uint8(A)

C =

  2×2 uint8 matrix

     0   255
     0   128
```

Berdasarkan hal tersebut dapat disimpulkan bahwa nilai-nilai asli seolah-olah berada dalam kisaran [0,1] untuk data kelas double, yaitu 0,5 menjadi 128 (titik rentang tengah), apa pun yang kurang dari atau sama dengan 0 terpotong menjadi 0, dan apapun yang lebih besar dari atau sama dengan 1 terpotong menjadi 255.

Sekarang coba untuk menerapkan mat2gray ke A dan periksa hasilnya:

$$D = \text{mat2gray}(A)$$

```
>> D = mat2gray(A)

D =

         0    1.0000
    0.3333    0.5000
```

Hasil ini mengilustrasikan poin penting: dengan A yang berua kelas data *double*, tidak ada ketentuan kelas data, tetapi hanya konversi rentang nilai terkecil (−8,0) menjadi 0,0, nilai terbesar (4,0) menjadi 1,0, dan semua nilai menengah diskalakan dalam rentang baru.

Coba gunakan *im2bw* untuk mengonversi citra menjadi setara biner.

$$E = \text{im2bw}(D, 0.4)$$

```
>> E = im2bw(D, 0.4)

E =

  2×2 logical array

   0   1
   0   1
```

Hasilnya berfungsi seperti yang diharapkan: nilai apa pun yang lebih besar dari 0,4 menjadi 1 (benar), jika tidak maka menjadi 0 (salah). Perhatikan bahwa karena im2bw mengharapkan tingkat pencahayaan ambang batas sebagai parameter, dan

mengharuskannya menjadi bilangan non-negatif antara 0 dan 1, ia secara implisit mengasumsikan bahwa variabel input (D, dalam hal ini) adalah citra *grayscale* yang dinormalisasi dari kelas *double*. Dengan kata lain, jika Anda mencoba menggunakan A sebagai citra input ($E = im2bw(A, 0, 4)$), fungsi akan bekerja tanpa kesalahan atau peringatan, tetapi hasilnya mungkin tidak masuk akal.

IPT MATLAB mencakup fungsi untuk mengkonversi antara RGB (truecolor), citra indek, dan citra grayscale.

Tabel 4. 2 Fungsi IPT untuk melakukan Konversi Kelas Data Citra

Nama	Keterangan
ind2gray	Citra indeks
gray2ind	Citra grayscale
rgb2gray	Citra RGB
rgb2ind	Citra RGB
ind2rgb	Indeks citra berwarna

4.3.4. Menampilkan Isi Citra

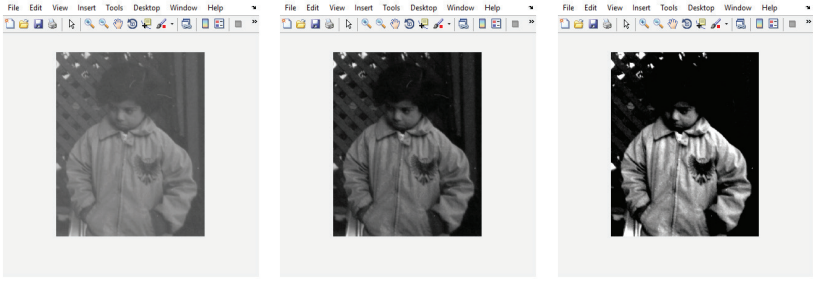
MATLAB memiliki beberapa fungsi untuk menampilkan citra, seperti:

- *image*: menampilkan citra menggunakan peta warna saat ini.
- *imagesc*: menskala data citra hingga jangkauan penuh dari peta warna saat ini dan menampilkan citra.

- *imshow*: menampilkan citra dan memuat sejumlah optimasi dan parameter opsional untuk pengaturan properti yang terkait dengan tampilan citra.
- *imtool*: menampilkan citra dan berisi sejumlah alat terkait yang dapat digunakan untuk menjelajahi konten citra.

Contoh 3 : Kode MATLAB berikut digunakan untuk membuka file citra dan menampilkannya menggunakan *imshow* yang berbeda (sesuai Gambar 4.1):

```
I = imread('pout.tif');
figure, imshow(I)
figure, imshow (I, [])
figure, imshow (I, [100 160])
```



(a) (b) (c)

Gambar 4. 1 Menampilkan citra: (a) tanpa penskalaan; (b) penskalaan (c) memilih piksel

Pernyataan pada baris pertama untuk menampilkan citra dalam keadaan aslinya. Baris berikut membuka citra baru dan menampilkan versi skala citra. Baris terakhir menentukan kisaran *grayscale*, sehingga semua nilai yang lebih rendah dari 100 ditampilkan sebagai hitam dan semua nilai lebih besar dari 160 ditampilkan sebagai putih.

4.3.5. Menjelajahi Isi Citra

Peneliti dan praktisi pada topik pengolahan citra seringkali perlu memeriksa lebih dekat isi citra. Dalam MATLAB dapat dilakukan menggunakan fungsi *imtool*, yang menyediakan semua kemampuan tampilan citra *imshow* serta akses ke alat lain untuk menavigasi dan menjelajahi citra, seperti *Pixel Region tool*, *Image Information tool*, dan *Adjust Contrast tool*.

(a) (b) (c)

Gambar 4. 2 Menampilkan citra dam menjelajahi dengan MATLAB, *Pixel Region*.

Alat-alat ini juga dapat diakses secara langsung menggunakan fungsi pustaka masing-masing *impixelinfo*, *imageinfo*, dan *imcontrast*.

4.3.6. Menulis Citra yang dihasilkan ke File

MATLAB IPT memiliki fungsi *imwrite*, yang berguna untuk menulis konten citra dalam salah satu format file citra paling popular. Jika format file keluaran menggunakan kompresi lossy (seperti JPEG), *imwrite* memungkinkan spesifikasi parameter kualitas, digunakan sebagai menghasilkan kualitas subjektif dari citra yang dihasilkan dan ukuran file.

Contoh 4 : Kode MATLAB berikut digunakan untuk membaca citra dari file PNG dan menyimpannya ke file JPG menggunakan tiga parameter kualitas yang berbeda: 75 (default), 5 (kualitas buruk, ukuran kecil), dan 95 (kualitas lebih baik, ukuran lebih besar). Hasil contoh 4 dapat dilihat pada Gambar 4.3.

```
I = imread ('pout.tif);
imwrite (I, 'pout75.jpg');
imwrite (I, 'pout95.jpg ',' quality ', 95);
```

(a) (b) (c)

Gambar 4. 3 Membaca dan menulis citra: (a) Citra asli (TIF); (B) citra terkompresi (JPG, q = 75, ukuran file = 7 kB); (c) citra terkompresi (JPG, q = 5, ukuran file = 2 kB); (d) citra terkompresi (JPG, q = 95, ukuran file = 17 kB).

4.4. TUTORIAL MANIPULASI CITRA DASAR

Pada latihan berikut, setiap pertanyaaan terdapat prosedur yang harus dilakukan, lakukan prosedur tersebut dan jawab pertanyaan yang muncul. IPT mendukung citra dengan tipe biner, indeks, intensitas, dan truecolor. Sebelum citra diproses dalam

MATLAB, citra tersebut harus terlebih dahulu dimuat ke dalam memori. Untuk membaca dalam citra, digunakan fungsi imread.

Prosedur 1

Langkah 1. Baca citra coin.png dengan menjalankan pernyataan berikut:

```
I = imread('coins.png');
```

Pertanyaan 1 : Apa jenis citra itu coin.png?

Pertanyaan 2 : Mengapa menggunakan operator titik koma (;) setelah pernyataan yang sudah dibaca? Apa yang terjadi jika operator tersebut dihilangkan?

Citra biner, intensitas, dan truecolor semuanya dapat dibaca dengan fungsi *imread* seperti yang ditunjukkan di atas. Saat membaca dalam citra yang diindeks, pertama harus menentukan variabel untuk citra dan mapping warna. Ini diilustrasikan dalam langkah berikut:

Prosedur 2

Baca citra trees.tif.

```
[X,map] = imread('trees.tif');
```

Beberapa operasi mungkin mengharuskan mengonversi citra dari satu jenis ke jenis lainnya. Misalnya, melakukan penyesuaian citra pada citra yang diindeks mungkin tidak memberikan hasil yang ingin dicapai karena perhitungan dilakukan pada nilai indeks dan bukan nilai RGB yang representatif. Untuk itu dapat dilakukan dengan mengonversi citra yang diindeks menjadi citra RGB menggunakan *ind2rgb*.

Prosedur 3

Konversikan citra yang diindeks X dengan peta peta berwarna menjadi citra RGB, X_rgb.

```
X_rgb = ind2rgb (X, peta);
```

Pertanyaan 3 : Berapa dimensi yang dimiliki variabel X_rgb dan berapa ukurannya?

Prosedur 4

Konversikan citra yang diindeks X dengan peta peta berwarna menjadi citra intensitas.

```
X_gray = ind2gray (X, peta);
```

Pertanyaan 4 : Jenis kelas apa X_gray?

Prosedur 5

memverifikasi bahwa citra intensitas baru terdiri dari nilai-nilai piksel dalam kisaran [0, 255].

```
maks (X_gray (:)) min (X_gray (:))
```

Pertanyaan 5 : Mengapa diharuskan menggunakan operator titik dua (:) ketika menentukan variabel X_gray? Apa yang terjadi jika operator tersebut dihilangkan?

Itu ditunjukkan pada langkah sebelumnya bahwa citra X_gray mengandung nilai dalam kisaran [0, 255] (dalam citra khusus ini, mereka kebetulan tepat 0 dan 255, yang hanya kebetulan). Mari kita lihat apa yang terjadi ketika kita mengonversi citra ke kelas *double*.

Prosedur 6

Konversikan variabel X_gray ke kelas *double*.

```
X_gray_dbl = im2double (X_gray);
```

Pertanyaan 6 : Berapa kisaran nilai untuk variabel baru X_gray_dbl?

Mengonversi ke tipe kelas lain dengan menggunakan *im2uint8* dan *im2uint16* dapat dilakukan.

MATLAB hadir dengan fungsi tampilan citra bawaan. Fungsi *image* dapat digunakan untuk menampilkan data citra, dan fungsi *imagesc* akan melakukan operasi yang sama tetapi selain itu akan menskala data citra dengan rentang nilai penuh. IPT menyediakan fungsi tampilan citra yang disempurnakan yang mengoptimalkan pengaturan pada sumbu citra untuk memberikan tampilan data citra yang lebih baik: *imshow*.

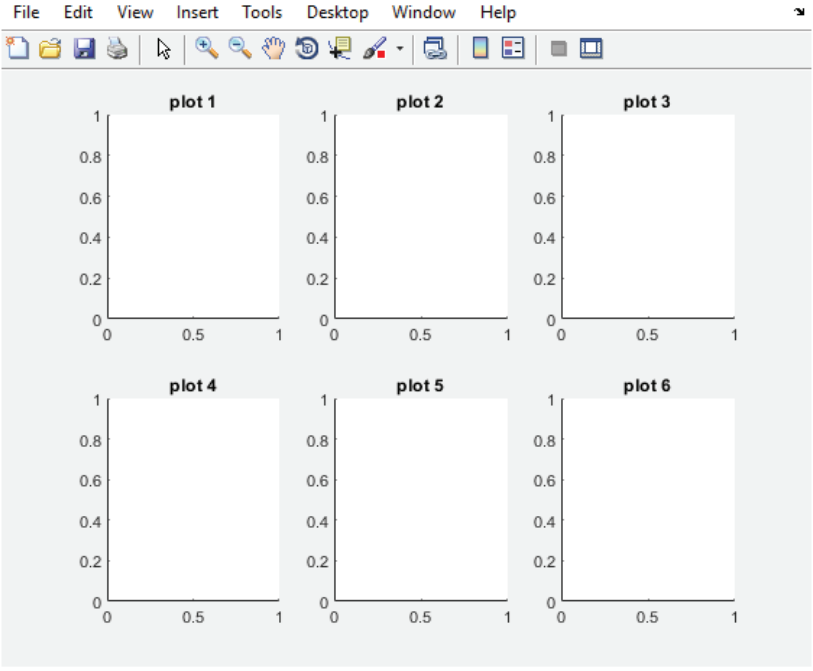
Fungsi *imtool* adalah fungsi IPT terbaru dan terkaya untuk menampilkan citra. Ini memberikan semua kemampuan tampilan citra *imshow* serta akses ke alat lain untuk menavigasi dan menjelajahi citra, *seperti Pixel Region tool*, *Image Information tool*, dan *Adjust Contrast tool.*

Prosedur 7

Gunakan fungsi *imtool* untuk menampilkan citra yang saat ini dimuat dalam variabel X_rgb. Perhatikan bahwa jendela sekunder akan terbuka. Jelajahi fungsionalitas tambahan, termasuk kemungkinan mengukur jarak antara dua titik di dalam citra.

```
imtool (X_rgb)
```

MATLAB dapat menampilkan banyak citra dalam satu citra menggunakan fungsi *subplot*. Saat menggunakan fungsi ini, dua parameter pertama menentukan jumlah baris dan kolom untuk membagi angka. Parameter ketiga menentukan subdivisi mana yang akan digunakan. Gambar 4.4 merupakan hasil penggunaan *subplot*.



Gambar 4. 4 Pembagian angka menggunakan subplot

Prosedur 8

Jalankan pernyataan berikut untuk membuat subplot dengan dua citra.

Prosedur 9

Tutup semua citra terbuka (dengan menggunakan '*close all*').

```
A = imread('pout.tif');
B = imread('cameraman.tif');
figure
```

```
subplot(1,2,1), imshow(A)
subplot(1,2,2), imshow(B)
```

Pertanyaan 7 *: Berapa kisaran nilai untuk citra A dan citra B?*

Sebagai langkah awal, akan menampilkan dua citra, keduanya citra intensitas. Meskipun tidak ada peta warna yang terkait dengan citra intensitas, MATLAB menggunakan peta warna *grayscale* untuk menampilkan citra intensitas.

Prosedur 10

Tutup semua jendela citra yang terbuka.

Prosedur 11

Tampilkan coin.png (dimuat dalam variabel I) dan tree.tif (dimuat dalam variabel X dan peta warna dalam variabel map) citra dalam *subplot*. Jalankan setiap pernyataan sekaligus untuk melihat efek pada citra keduanya ditampilkan.

```
figure
subplot(1,2,1), imshow(I)
subplot(1,2,2), imshow(X,map)
```

Pertanyaan 8 *: Apa yang terjadi pada citra koin setelah citra pohon ditampilkan? Jelaskan.*

Untuk menampilkan citra dengan baik, gunakan fungsi *subimage*.

Prosedur 12

Gunakan fungsi *subimage* untuk menampilkan banyak citra dengan peta warna yang berbeda.

```
figure
subplot(1,2,1), subimage(I), axis off
```

$$\text{subplot}(1,2,2),\ \text{subimage}(\text{X},\ \text{peta}),\ \text{axis off}$$

Fungsi subimage mengubah citra menjadi citra RGB yang setara dan kemudian menampilkan citra itu.

Prosedur 13

Secara manual mengkonversi koin intensitas citra (dimuat dalam variabel I) ke citra yang diindeks dan kemudian ke RGB. Perhatikan bahwa citra pohon (dimuat X tidak berubah-ubah dengan peta warnanya dalam variable map) telah dikonversi ke RGB (disimpan dalam variabel X_rgb).

$$[\text{I\_ind}, \text{I\_map}] = \text{gray2ind}(\text{I}, 256);$$

$$\text{I\_rgb} = \text{ind2rgb}(\text{I\_ind}, \text{I\_map});$$

Pertanyaan 9 *: Apa isi variabel I_ind dan I_map?*

Prosedur 14

Tampilkan citra truecolor menggunakan fungsi imshow

```
figure
subplot (1,2,1), imshow (I_rgb)
subplot (1,2,2), imshow (X_rgb)
```

Prosedur 15

Gunakan *imwrite* untuk menyimpan dua citra yang dimodifikasi ke file untuk penggunaan lebih lanjut.

Gunakan format JPG untuk salah satunya dan ekstensi PNG untuk yang lain.

```
imwrite (X_rgb, 'rgb_trees.jpg');
imwrite (X_gray, 'gray_trees.png');
```

4.5. LATIHAN PERMASALAHAN

1. Buat array 2 × 2 tipe double A = [1,5 -2; 0,5 0] gunakan untuk melakukan operasi berikut:
   a. Konversikan ke *uint8* menggunakan *typecasting* dan menginterpretasikan hasilnya.
   b. Konversikan ke citra *grayscale* kelas data *uint8* yang dinormalisasi ([0, 255]) menggunakan *im2uint8* dan interpretasikan hasilnya.
   c. Konversikan ke citra *grayscale* kelas data yang dinormalisasi ([0,0, 1.0]) menggunakan *mat2gray* dan tafsirkan hasilnya.
   d. Konversi hasil dari langkah sebelumnya ke citra biner menggunakan *im2bw* (dengan tingkat ambang 0,5) dan menafsirkan hasilnya.
2. Buat array 2 × 2 tipe double A = [1 4; 5 3] dan tulis pernyataan MATLAB untuk melakukan operasi berikut:
   a. Konversikan ke citra *grayscale* kelas data yang dinormalisasi ([0,0,1.0]).
   b. Konversi ke citra biner dari kelas data *logical*, sehingga setiap nilai lebih dari 2 (dalam citra asli) akan ditafsirkan sebagai 1 (benar).
   c. Ulangi langkah sebelumnya, kali ini menghasilkan hasil data kelas *double*.
3. Bagaimana fungsi IPT *rgb2ind* menangani kemungkinan bahwa citra RGB asli mengandung lebih banyak warna daripada ukuran peta warna (65.536 warna)?
4. Pilih lima citra yang tersedia di MATLAB. Buka masing-masing menggunakan *imread*, simpan (gunakan *imwrite*) ke (setidaknya tiga) format file yang berbeda, dan bandingkan ukuran file yang dihasilkan (dalam byte) untuk setiap format output.

Bab 5

EKSTRAKSI FITUR DAN REPRESENTASI

5.1. TUJUAN

Tujuan dari bahasan ini adalah

- Mempelajari maksud dan dasar ekstraksi fitur yang merupakan langkah penting dalam kebanyakan komputer
- Mempelajari solusi akhir dalam pemrosesan citra?
- Mempelajari berbagai jenis fitur yang dapat diekstraksi dari citra dan cara melakukan?
- Mempelajari cara fitur yang diekstraksi biasanya direpresentasikan untuk diproses lebih lanjut?

5.2. PENDAHULUAN: GAMBARAN UMUM

Bagian ini membahas metode dan teknik untuk mewakili dan mengcitrakan citra dan objek atau wilayah yang diamati. Sebagian besar teknik yang dijelaskan dalam bab ini mengasumsikan bahwa citra telah mengalami segmentasi. Tujuan umum dari teknik ekstraksi fitur dan representasi adalah untuk mengubah objek tersegmentasi menjadi representasi yang lebih baik mengcitrakan fitur dan atribut utama mereka. Jenis dan kompleksitas representasi yang dihasilkan tergantung pada banyak faktor, seperti jenis citra (misalnya, biner, *grayscale*, atau warna), tingkat rincian (seluruh citra atau wilayah individu) yang diinginkan, dan konteks aplikasi yang diinginkan (misalnya, pengklasifikasi pola dua kelas yang memberi tahu objek melingkar dari yang tidak berbentuk lingkaran atau sistem pengambilan citra yang mengambil citra yang dinilai mirip dengan contoh citra).

Ekstraksi fitur adalah proses di mana fitur tertentu yang menarik dalam suatu citra terdeteksi dan diwakili untuk diproses lebih lanjut. Ini adalah langkah penting dalam sebagian besar *computer vision* dan

solusi pemrosesan citra karena menandai transisi dari representasi data bercitra ke bukan citra (alfanumerik, biasanya kuantitatif).

Representasi yang dihasilkan selanjutnya dapat digunakan sebagai input ke sejumlah teknik pengenalan pola dan klasifikasi, yang kemudian akan melabeli, mengklasifikasikan, atau mengenali konten semantik dari citra atau objek-objeknya. Ada banyak cara suatu citra (dan objeknya) dapat direpresentasikan untuk keperluan analisis citra, dan beberapa cara tersebut disajikan dalam pembahasan di Bab ini.

5.3. VEKTOR FITUR DAN RUANG VEKTOR

Vektor fitur adalah array n × 1 yang mengkodekan fitur n (atau pengukuran) dari suatu citra atau objek. Konten array mungkin simbolis (seperti string yang berisi nama warna dominan pada citra), numerik (seperti bilangan bulat yang mengekspresikan area objek, dalam piksel), atau keduanya.

Secara matematis, vektor fitur numerik x diberikan oleh $x=\left(x1,x2,\ldots,xn\right)^{T}$ di mana n adalah jumlah total fitur dan T menunjukkan operasi transpos.

Vektor fitur adalah representasi dari suatu citra (atau objek di dalam citra), yang dapat dikaitkan dengan gagasan tentang ruang fitur, sebuah *n-dimensi hyperspace* memungkinkan visualisasi (untuk n <4) dan interpretasi konten vektor fitur tersebut.

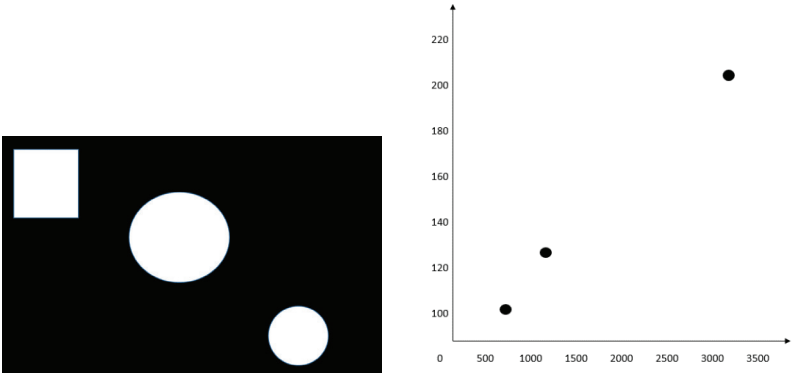
Contoh 1

Misalkan objek pada citra telah diwakili oleh luas area dan perimeternya, yang telah dihitung dan diperlihatkan pada Tabel 5.1.

Tabel 5. 1 Tabel Fitur berupa Luas Area dan Perimeter

Objek	Luas Area	Perimeter
Kotak (Sq)	1024	124
Lingkaran Besar (LC)	3209	211
Lingkaran Kecil (SC)	797	105

Gambar 5.1 memperlihatkan hasil Fitur dalam vektor 2D .



(a) (b)

Gambar 5. 1 Citra test (a) and hasil 2D fitur vektor (b)

Vektor fitur yang dihasilkan adalah sebagai berikut:

$$Sq = \left(1024, 124\right)^T$$

$$LC = \left(3209, 211\right)^T$$

$$SC = \left(797, 105\right)^T$$

Citra menunjukkan tiga vektor fitur yang diplot dalam grafik 2D yang sumbunya adalah fitur yang dipilih, yaitu luas area

dan perimeter. Dimana vektor fitur biasanya direpresentasikan menggunakan susunan dan struktur sel pada MATLAB.

5.3.1 Invarian dan Robustness

Persyaratan umum untuk ekstraksi fitur dan teknik representasi adalah fitur yang digunakan untuk mewakili citra tidak berubah dalam rotasi, penskalaan, dan terjemahan, secara kolektif dikenal sebagai RST. Invarian RST memastikan bahwa MVS masih akan dapat mengenali objek bahkan ketika muncul pada ukuran yang berbeda, posisi di dalam citra, dan sudut (relatif terhadap referensi horisontal). Sehingga persyaratan ini hanya tergantung pada aplikasi. Misalnya, jika tujuan dari VMS adalah untuk memastikan bahwa sirkuit terintegrasi (IC) hadir pada posisi dan orientasi yang benar pada papan sirkuit cetak, invarian rotasi tidak lagi menjadi persyaratan; sebaliknya, dapat mengatakan bahwa IC terbalik adalah bagian integral dari fungsionalitas sistem.

Seperti tujuan dari MVS (mesin) adalah meniru sebaik mungkin kemampuan HVS (manusia) untuk mengenali objek dan adegan dalam berbagai keadaan. Untuk mencapai kemampuan seperti itu, tahap ekstraksi fitur dan representasi MVS harus idealnya memberikan representasi yang tidak hanya invarian RST tetapi juga kuat untuk aspek lain, seperti resolusi spasial yang buruk, pencahayaan yang tidak seragam, distorsi geometris (disebabkan oleh sudut pandang yang berbeda), dan noise. Ini adalah tantangan besar bagi perancang MVS, yang mungkin memerlukan pemilihan fitur yang cermat serta teknik pra dan pasca pemrosesan yang harus dilakukan sepenuhnya.

5.4. FITUR OBJEK BINARY

Pada bagian ini, beberapa fitur digunakan untuk objek biner. Objek biner, dalam hal ini, adalah wilayah yang terhubung dalam citra biner $f(x,y)$, yang akan dilambangkan sebagai $Oi, i>0$. Secara matematis, fungsi $Oi(x,y)$ didefinisikan dalam persamaan 5.1.

$$Oi = \begin{cases} 1 & jika\ f(x,y) \in Oi \\ 0 & lainnya \end{cases} \quad (5.1)$$

Dalam Fungsi IPT MATLAB *bwlabel* mengimplementasikan operasi $i \times Oi(x,y)$ dengan memberi label wilayah piksel yang terhubung dalam citra biner: piksel berlabel 0 sesuai dengan latar belakang; piksel berlabel 1 sesuai dengan komponen yang terhubung dalam citra. Adapun fungsi lain seperti *bwperim* dapat digunakan untuk menghitung perimeter objek dalam citra biner menggunakan kriteria 4 atau 8 konektivitas. Selain itu, proses mendapatkan fitur dari objek dengan data binary, IPT MATLAB memiliki fungsi *regionprops* yang dapat digunakan untuk mengukur serangkaian properti untuk setiap wilayah berlabel.

Tabel 5. 2 Properti Wilayah Berlabel

Properti	Deskripsi
'Area'	Jumlah aktual piksel di wilayah tersebut
'BoundingBox'	Kotak terkecil yang berisi wilayah, ditentukan oleh koordinat sudut kiri atas dan panjang setiap dimensi
'Centroid'	Vektor yang menentukan pusat wilayah
'ConvexHull'	*convex polygon* terkecil yang dapat berisi wilayah tersebut

Properti	Deskripsi
'ConvexImage'	Citra biner yang menentukan *convex hull*, yaitu citra yang semua pikselnya berada di dalam lambung ditetapkan menjadi true
'ConvexArea'	Jumlah piksel dalam 'ConvexImage'
'Eccentricity'	Eksentrisitas dari elips memiliki momen kedua yang sama dengan wilayah. Eksentrisitas adalah nilai dalam rentang [0,1] (0 untuk lingkaran, 1 untuk segmen garis). Ini didefinisikan sebagai rasio jarak antara fokus elips dan panjang sumbu utamanya.
'EquivDiameter'	Diameter lingkaran dengan area yang sama dengan wilayah, dihitung sebagai $\sqrt{(4 * Area / \pi)}$
'EulerNumber'	Perbedaan antara jumlah objek di wilayah dan jumlah lubang di objek ini
'Extent'	Proporsi piksel dalam kotak pembatas yang juga ada di wilayah tersebut, yaitu rasio antara area wilayah dan area kotak pembatas
'Extrema'	Matriks 8 × 2 yang berisi koordinat titik ekstrem di wilayah tersebut
'FilledArea'	Jumlah piksel dalam *FilledImage*
'FilledImage'	Citra biner dengan ukuran yang sama dengan kotak pembatas wilayah, tempat semua piksel sebenarnya sesuai dengan wilayah dan semua lubang telah terisi.
'Image'	Citra biner dengan ukuran yang sama dengan kotak pembatas wilayah, tempat semua piksel asli sesuai dengan wilayah tersebut, dan semua piksel lainnya dianggap salah.

Properti	Deskripsi
'MajorAxisLength'	Panjang (dalam piksel) dari sumbu utama elips yang memiliki momen pusat kedua yang sama dengan wilayah
'MaxIntensity'	Nilai dari piksel dengan intensitas terbesar di wilayah
'MeanIntensity'	Mean dari semua nilai intensitas di wilayah tersebut
'MinIntensity'	Nilai piksel dengan intensitas terendah di wilayah tersebut
'MinorAxisLength'	Panjang (dalam piksel) dari sumbu minor elips yang memiliki momen pusat kedua yang sama dengan wilayah

5.4.1 Area

Luas objek *i* pada Oi, diukur dalam piksel, dapat dilakukan dengan cara menghitung jumlah piksel pada objek tersebut dengan persamaan 5.2.

$$Ai = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} Oi(x,y) \tag{5.2}$$

5.4.2 Centroid

Koordinat dari centroid (pusat area) dari objek Oi, dilambangkan ($\overline{xi}, \overline{yi}$) dinyatakan dalam persamaan 5.3 dan persamaan 5.4 dengan Ai merupakan area objek Oi.

$$\overline{xi} = \frac{1}{Ai}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} x\,Oi(x,y) \tag{5.3}$$

$$\overline{yi} = \frac{1}{Ai}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} y\,Oi(x,y) \tag{5.4}$$

5.4.3 Axis of Least Second Moment

Axis of least second moment digunakan untuk memberikan informasi tentang orientasi objek relatif terhadap koordinat citra. Ini dapat dicitrakan sebagai sumbu inersia paling rendah, yaitu garis yang membutuhkan paling sedikit energi untuk memutar objek.

Metode ini dapat dilakukan dengan cara konvensi, sudut θ mewakili sudut antara sumbu vertikal dan *axis of least second moment*, diukur berlawanan arah jarum jam. Asal sistem koordinat dipindahkan ke pusat area objek dan simatematis dapat dilakukan dengan menghitung menggunakan persamaan 5.5.

$$\tan(2\theta i) = 2 \times \frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} xOi(x,y)}{\left(\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} x^2 Oi(x,y) - \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} y^2 Oi(x,y)\right)} \quad (5.5)$$

5.4.4 Projections

Projection adalah deskriptor bentuk yang sangat berguna dan kompak. Misalnya, tinggi dan lebar objek tanpa lubang dapat dihitung sebagai nilai maksimum proyeksi vertikal dan horizontal objek. Proyeksi horizontal dan vertikal dari objek biner atau hi (x) dan vi (y), masing-masing diperoleh dengan persamaan

$$hi(x) = \sum_{x=0}^{M-1} Oi(x,y) \quad (5.6)$$

$$vi(y) = \sum_{y=0}^{N-1} Oi(x,y) \quad (5.7)$$

Sedangkan, persamaan untuk koordinat pusat area objek sebagai fungsi dari proyeksi horisontal dan vertikal diperoleh dengan persamaan 5.8 dan persamaan 5.9.

$$\overline{xi} = \frac{1}{Ai} \sum_{x=0}^{M-1} x\, hi(x) \quad (5.8)$$

$$\overline{yi} = \frac{1}{Ai}\sum_{x=0}^{M-1} y\,vi(y) \tag{5. 9}$$

5.4.5 Euler Number

Euler number dari suatu objek atau citra (E) didefinisikan sebagai jumlah komponen yang terhubung (C) dikurangi jumlah lubang (H) dalam citra (sesuai dengan persamaan 5.10).

$$E = C - H \tag{5. 10}$$

Dalam kata lain *euler number* dapat dinyatakan sebagai perbedaan antara jumlah bentuk cembung dan jumlah bentuk cekung dalam citra atau area. *Euler number* dianggap sebagai deskriptor topologi, karena tidak terpengaruh oleh perubahan bentuk, asalkan tidak mengisi lubang atau memecah komponen yang terhubung secara terpisah.

Gambar 5. 2 Contoh dua wilayah dengan nomor Euler masing-masing sama dengan 0 dan −1

5.4.6 Perimeter

Perimeter objek biner Oi dapat dihitung dengan menghitung jumlah piksel objek (yang nilainya 1) yang memiliki satu atau lebih

piksel latar belakang (yang nilainya 0) sebagai tetangganya. Metode alternatif terdiri dari mengekstraksi tepi (kontur) objek, dilanjutkan dengan menghitung jumlah piksel di perbatasan yang dihasilkan. Namun, beberapa ketidaksempurnaan yang tak terhindarkan dalam proses digitalisasi (seperti kurva bergerigi menguraikan dan tepi bergerigi) menyebabkan nilai perimeter yang dihitung menggunakan metode mana pun tidak 100% akurat; sehingga disarankan bahwa nilai-nilai tersebut harus dikalikan dengan π / 4 untuk akurasi yang lebih baik.

5.4.7 Thinness Ratio

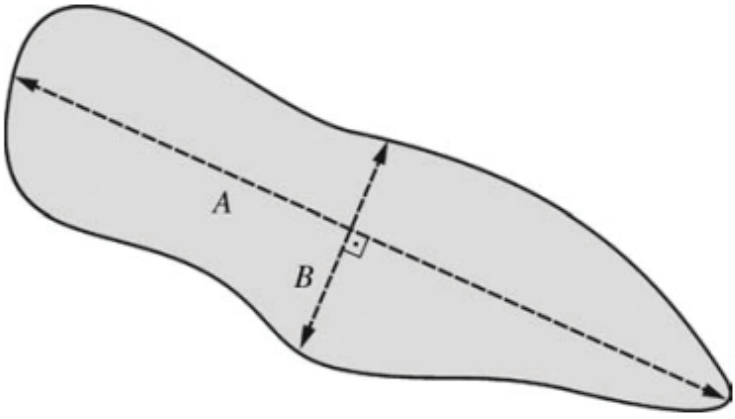
Thinness Rasio Ti dari objek biner Oi adalah angka pangkat yang menghubungkan area objek dan perimeternya. *Thinness Rasio* ini digunakan untuk mengukur bentuk bulat (roundness). Nilai maksimum yang digunakan Ti adalah 1 (lingkaran sempurna), maka untuk objek generik semakin tinggi rasio ketipisannya, semakin bulat objek tersebut. Angka rasio ini dapat digunakan sebagai ukuran keteraturan dan kebalikannya, $1/Ti$ kadang-kadang disebut rasio ketidakteraturan atau kekompakan (diperlihatkan pada Gambar 5.3).

(a) (b)

Gambar 5. 3 Contoh wilayah kompak (a) dan tidak kompak (b)

5.4.8 Eccentricity

Eccentricity (Keanehan) suatu objek didefinisikan sebagai rasio sumbu utama dan minor dari objek tersebut (sesuai Gambar 5.4).



Gambar 5. 4 Keanehan (A / B) suatu daerah.

5.4.9 Aspect Ratio

Aspect Ratio (AR) adalah ukuran hubungan antara dimensi kotak pembatas suatu objek. Dengan koordinat sudut kiri atas (xmin, ymin) dan kanan bawah (xmax, ymax) dari kotak pembatas yang mengelilingi objek, maka nilai AR dapat diperoleh dengan menggunakan persamaan 5.11.

$$AR = \frac{xmax - xmin + 1}{ymax - ymin + 1} \tag{5. 11}$$

Dalam penggunaan AR, dinyatakan bahwa data tidak invarian terhadap rotasi dan tidak dapat digunakan untuk membandingkan objek yang diputar terhadap satu sama lain kecuali jika itu dinormalisasi. Salah satu cara melakukan normalisasi adalah dengan

menghitung AR setelah menyelaraskan *axis of least second moment* ke arah horizontal. Representasi AR yang dinormalisasi tersebut juga disebut sebagai objek yang memanjang.

5.4.10 Momen

Momen orde 2D (p + q) dari citra digital f (x, y) didefinisikan dalam persamaan 5.12. Dimana *M* dan *N* adalah tinggi dan lebar citra masing-masing, sedangkan *p* dan *q* adalah bilangan bulat positif bukan-nol.

$$m_{pq} = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} x^p y^q f(x,y) \qquad (5.12)$$

Pusat momen adalah perwujudan yang setara dengan momen yang didefinisikan berdasarkan persamaan 5.13 dan $\bar{x} = m10 / m00$ dan $\bar{y} = m01 / m00$

$$\mu_{pq} = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} (x-\bar{x})^p (y-\bar{y})^q f(x,y) \qquad (5.13)$$

Normalisasi pusat momen dilakukan dengan persamaan 5.14 dan y didapatkan dari persamaan 5.15.

$$\eta_{pq} = \mu_{pq} / \mu_{00}^{y} \qquad (5.14)$$

$$y = \frac{p+q}{2} + 1 \ldots\ldots\ldots (\mathrm{p} + \mathrm{q}) > 1 \qquad (5.15)$$

Satu set tujuh momen invarian RST, ϕ1 – ϕ7, dapat diturunkan dari pusat momen kemudian dinormalisasi orde dua dan ketiga sesuai dengan Tabel 5.3.

Tabel 5. 3 RST-Invariant Moments

$$\emptyset_1 = \eta_{20} + \eta_{02}$$
$$\emptyset_2 = (\eta_{20} - \eta_{02})^2 + (2\eta_{02})^2$$
$$\emptyset_3 = (\eta_{30} - 3\eta_{12})^2 + (\eta_{03} - 3\eta_{21})^2$$
$$\emptyset_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{03} + \eta_{21})^2$$
$$\emptyset_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (\eta_{03} - 3\eta_{12})(\eta_{03} + \eta_{21})[(\eta_{03} + \eta_{12})^2 - 3(\eta_{12} + \eta_{30})^2]$$
$$\emptyset_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{03} + \eta_{21})$$
$$\emptyset_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[(\eta_{03} + \eta_{21})^2 - 3(\eta_{30} + \eta_{12})^2]$$

5.5. FITUR BERBASIS DESKRIPSI BATASAN

Deskripsi yang diuraikan pada bagian ini secara kolektif disebut sebagai batasan yang berbasis wilayah. Beberapa teknik representasi dan deskripsi berbasis kontur disampaikan dalam bagian ini. Teknik-teknik mengasumsikan bahwa kontur (atau batas) suatu objek dapat direpresentasikan dalam sistem koordinat yang sesuai (diagram cartesian yang paling umum, kutub, atau tangensial) dan bergantung secara eksklusif pada piksel batas untuk mengcitrakan wilayah atau objek. Batas-batas objek dapat diwakili oleh teknik yang berbeda, mulai dari metode pendekatan poligonal sederhana hingga teknik yang lebih rumit yang melibatkan interpolasi polinomial untuk setiap bagian seperti kurva B-spline.

Teknik mengasumsikan bahwa piksel yang termasuk dalam batas objek (atau wilayah) dapat ditelusuri. Proses penelusuran berawal dari piksel yang dianggap sebagai latar belakang, dilanjutkan segera setelah bug konseptual melintasi menjadi piksel batas, belok kiri dan pindah ke piksel berikutnya; jika piksel itu adalah piksel batas, bug membuat belokan ke kiri lagi, jika tidak belok kanan;

proses ini diulangi sampai bug kembali ke titik awal. Ketika bug konseptual mengikuti kontur, ia membangun daftar koordinat piksel batas yang dikunjungi. Proses penelusuran tersebut dikenal sebagai penelusuran bug.

Konsep penelusuran bug dapat dilakukan dengan menggunakan fungsi *bwtraceboundary* yang terdapat dalam dalam IPT MATLAB yang bertujuan untuk melacak objek dalam citra biner. Koordinat titik awal dan arah pencarian awal (N, NE, E, SE, S, SW, W, atau NW) harus dilewati sebagai parameter. Secara opsional, jenis konektivitas (4 atau 8) dan arah pelacakan (searah jarum jam atau berlawanan arah jarum jam) juga dapat dilewatkan sebagai argumen. Fungsi *bwtraceboundary* mengembalikan array yang berisi koordinat baris dan kolom dari piksel batas.

Beberapa contoh penggunaan *bwtraceboundary* untuk melacak 50 piksel pertama dari batas objek paling kiri pada citra input, mulai dari sudut kiri atas dan menuju ke selatan, mencari ke arah berlawanan arah jarum jam (diperlihatkan pada Gambar 5.5.a).

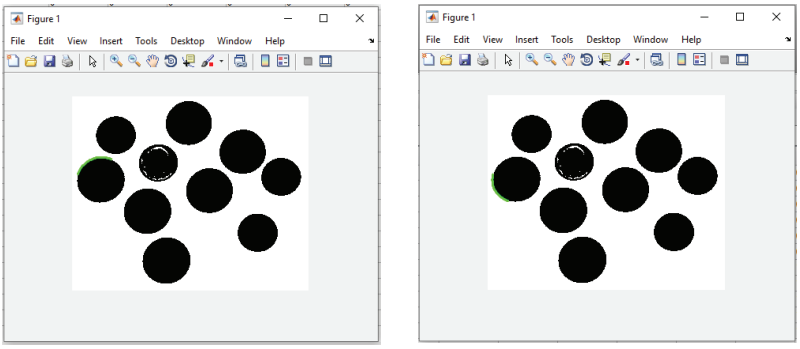
Hal tersebut dilakukan dengan sintaks

$$[\,\text{contour}\,] = \text{bwtraceboundary}(\text{BW}, [\text{r c}], \text{'W'}, 8, 50, \text{'counterclockwise'});$$

Sedangkan pada contoh yang menggunakan penelusuran sebanyak 40 piksel pertama dari batas semua objek dalam citra, menuju ke timur, dan mencari searah jarum jam (diperlihatkan pada Gambar 5.5.b). Batas ditunjukkan dengan warna hijau.

Hal tersebut dilakukan dengan sintaks

$$[\,\text{contour}\,] = \text{bwtraceboundary}(\text{BW}, [\text{r c}], \text{'W'}, 8, 40, \text{'clockwise'});$$



(a) (b)

Gambar 5. 5 Penelusuran batas objek

Fungsi *bwboundaries* di IPT MATLAB, berguna untuk penelusuran batas wilayah dilakukan dalam citra biner. Fungsi tersebut dapat menelusuri batas luar objek, serta batas lubang di dalam objek. Selain itu juga dapat menelusuri batas objek kecil dari objek utama dalam suatu citra. Fungsi *bwboundaries* mengembalikan array sel (B) di mana setiap sel berisi koordinat baris dan kolom dari piksel batas. Secara opsional, ini juga mengembalikan matriks label L di mana setiap objek dan lubang diberi label unik, jumlah total objek yang ditemukan (N), dan matriks *adjacency* yang digunakan untuk mewakili dependensi.

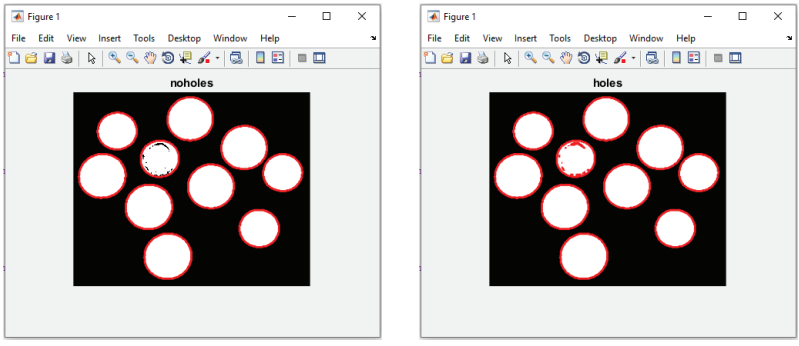
Beberapa citra yang menunjukkan penggunaan *bwboundaries* untuk melacak objek (diwakili dalam warna merah) dan lubang (diwakili dalam warna merah) pada citra sesuai Gambar 5.6.a yang didapatkan dengan menggunakan input sintaks

$$[\mathrm{B},\ \mathrm{L},\ \mathrm{N}] = \mathrm{bwboundaries}(\mathrm{BW});$$

Proses serupa dilakukan dengan menggunakan 4-konektivitas dan hanya mencari batas-batas objek, yaitu, tidak ada lubang sesuai

dengan Gambar 5.6.b yang didapatkan dengan menggunakan input sintaks

$$[\mathrm{B},\ \mathrm{L},\ \mathrm{N}] = \mathrm{bwboundaries}(\mathrm{BW}, 4, \text{'noholes'});$$



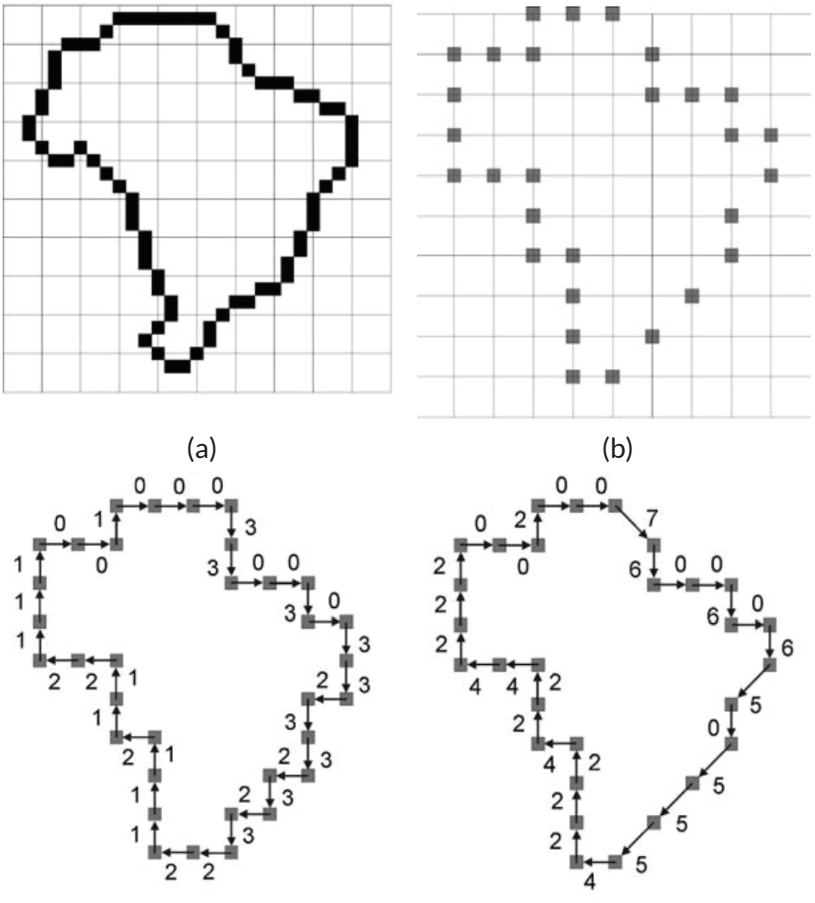
(a) (b)

Gambar 5. 6 Penelusuran batas objek dan lubang

5.5.1 Chain Code, Freeman Code, and Shape Number

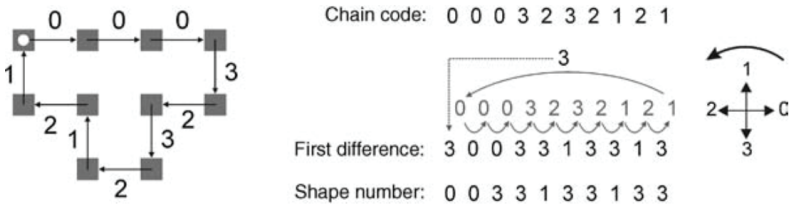
Code chain (kode rantai) adalah metode alternatif untuk melacak dan mengcitrakan kontur. Kode rantai adalah teknik representasi batas dengan kontur yang direpresentasikan sebagai urutan segmen garis lurus dengan panjang tertentu (biasanya 1) dan arah. Mekanisme kode rantai paling sederhana, juga dikenal sebagai kode crack, terdiri dari menugaskan nomor ke arah yang diikuti oleh algoritme pelacakan bug sebagai berikut: kanan (0), turun (1), kiri (2), dan atas (3). Dengan asumsi bahwa jumlah total poin batas adalah p (perimeter kontur), array C (ukuran p), di mana C (p) = 0, 1, 2, 3, berisi kode rantai batas. Versi modifikasi dari kode rantai dasar, yang dikenal sebagai Freeman Code (kode Freeman), menggunakan delapan arah, bukan empat. Setelah kode rantai untuk batas telah dihitung, dimungkinkan untuk mengubah array yang dihasilkan

menjadi setara rotasi-invarian, yang dikenal sebagai perbedaan pertama, yang diperoleh dengan mengkodekan jumlah perubahan arah, dinyatakan dalam kelipatan 90◦, antara dua elemen berturut-turut dari kode Freeman. Perbedaan pertama dari besaran terkecil diperoleh dengan memperlakukan array yang dihasilkan sebagai array melingkar dan memutarnya berdasarkan siklus hingga pola numerik yang dihasilkan menghasilkan angka terkecil yang mungkin dikenal sebagai *shape number* (jumlah bentuk) kontur. Jumlah bentuk invarian rotasi dan tidak sensitif terhadap titik awal yang digunakan untuk menghitung urutan asli.



(c) (d)

Gambar 5. 7 Kode rantai dan kode Freeman untuk kontur: (a) kontur asli; (B) versi kontur subsampled; (c) representasi kode rantai; (d) Representasi kode freeman.



Gambar 5. 8 Chain code, first differences, dan shape number

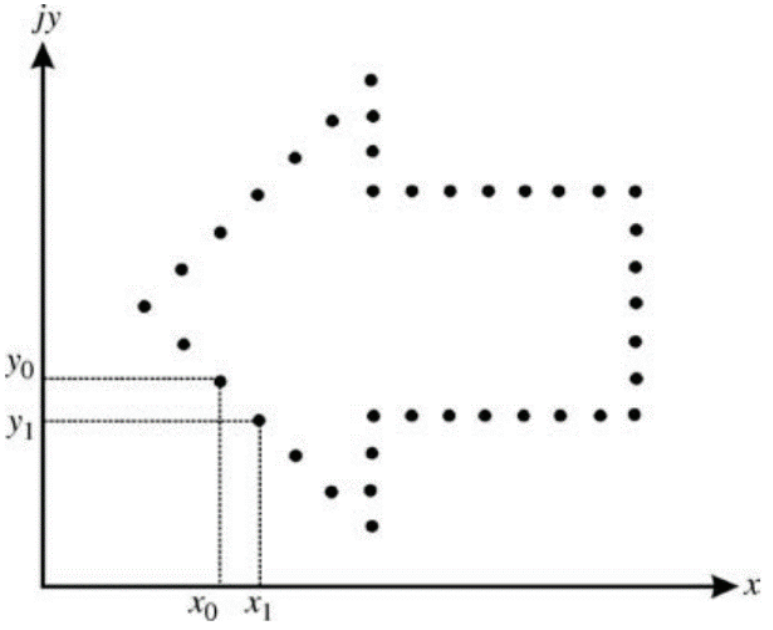
5.5.2 Signatures

Tanda tangan adalah representasi 1D dari batas, biasanya diperoleh dengan mewakili batas dalam sistem koordinat polar dan menghitung jarak r antara setiap piksel di sepanjang batas dan pusat wilayah, dan sudut θ disubstitusi antara garis lurus yang menghubungkan pixel batas ke pusat dan referensi horisontal. Plot yang dihasilkan dari semua nilai yang dihitung untuk $0 \leq \theta \leq 2\pi$ memberikan representasi singkat dari batas yang wujud invarian yang dibuat rotasi invarian (jika titik awal yang sama selalu dipilih), tetapi tidak menskalakan invarian.

5.5.3 Fourier Descriptors

Gagasan di balik Fourier deskriptor adalah untuk melintasi piksel batas, mulai dari titik arbitrer, dan merekam koordinatnya. Setiap nilai dalam daftar pasangan koordinat yang dihasilkan

$(x0, y0), (x1, y1), ..., (x_K - 1, y_K - 1)$ kemudian ditafsirkan sebagai bilangan kompleks $x_k + jy_k$, untuk k = 0, 1, ..., K - 1. *Transformasi Fourier diskrit* (DFT) dari daftar bilangan kompleks ini adalah deskriptor Fourier dari batas. DFT terbalik mengembalikan batas asli. Citra dibawah ini menunjukkan batas digital K-point pada bidang xy dan dua pasangan koordinat pertama, $(x0, y0)$ dan $(x1, y1)$.

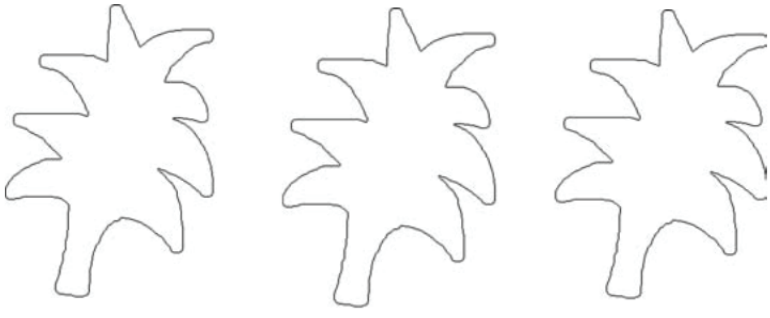


Gambar 5. 9 Fourier descriptor dari batas

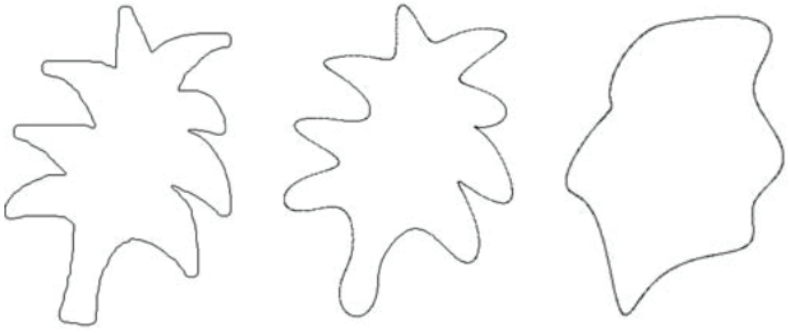
Dalam fungsi IPT MATLAB, menghitung deskriptor Fourier dari batas tidak disediakan, tetapi tidak terlalu sulit untuk membuat fungsi deskriptor Fourier sendiri. Prosedur dasarnya adalah menghitung batas wilayah, mengubah koordinat yang dihasilkan menjadi bilangan kompleks (seperti menggunakan fungsi kompleks), dan menerapkan fungsi fft ke array 1D yang dihasilkan dari bilangan

kompleks. Salah satu keuntungan utama menggunakan deskriptor Fourier adalah kemampuan mereka untuk mewakili esensi dari batas yang sesuai menggunakan sangat sedikit koefisien. Properti ini secara langsung berkaitan dengan kemampuan koefisien orde rendah dari DFT untuk mempertahankan aspek-aspek utama batas, sedangkan koefisien orde tinggi mengkodekan detail halus.

Beberapa contoh citra yang menunjukkan batas dengan jumlah 1467 poin (Gambar 5.10.a) dan hasil merekonstruksinya (menerapkan DFT diikuti oleh IDFT) menggunakan sejumlah variabel. Gambar 5.10.b menunjukkan hasil merekonstruksi dengan jumlah poin yang sama dalam verifikasi. Gambar 5.10.c hingga Gambar 5.10.f menunjukkan hasil untuk rekonstruksi dengan semakin sedikit poin: 734, 366, 36, dan 15, masing-masing. Nilai yang dipilih untuk Gambar 5.10.c hingga Gambar 5.10.f masing-masing sesuai dengan sekitar 50%, 25%, 2,5%, dan 1% dari total jumlah poin. Pemeriksaan yang teliti terhadap hasil menunjukkan bahwa citra yang direkonstruksi menggunakan 25% titik hampir identik dengan yang diperoleh dengan 50% atau 100% titik. Batas yang direkonstruksi menggunakan 2,5% dari titik masih mempertahankan sebagian besar bentuk keseluruhannya, dan hasilnya menggunakan hanya 1% dari jumlah total poin dapat dianggap tidak dapat diterima.



(a) (b) (c)

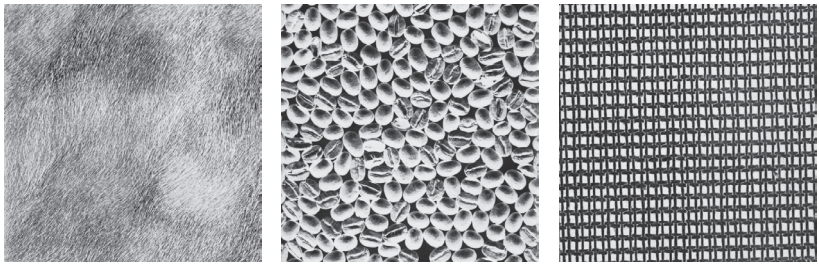


(d) (e) (f)

Gambar 5. 10 Contoh rekonstruksi batas menggunakan deskriptor Fourier: (a) citra asli; (b –f) citra yang direkonstruksi masing-masing menggunakan 100%, 50%, 25%, 2,5%, dan 1% dari jumlah total poin

5.6. FITUR BERBASIS HISTOGRAM (STATISTIK)

Histogram memberikan representasi ringkas dan berguna pada tingkat intensitas dalam citra *grayscale*. Histogram dapat digunakan sebagai teknik untuk peningkatan citra. Namun, pada bagian ini histogram digunakan sebagai fitur yang mengcitrakan citra (atau objeknya). Fitur berbasis histogram dan variannya biasanya digunakan sebagai deskriptor tekstur, seperti yang diperlihatkan pada Gambar 5.11.



(a) (b) (c)

Gambar 5. 11 Contoh citra dengan tekstur halus (a), kasar (b), dan reguler (c). Citra dari set data tekstur

Deskriptor berbasis histogram yang paling sederhana adalah nilai *grayscale* rata-rata dari suatu citra berupa nilai m, dengan r_j adalah *grayscale* ke-j (dari total nilai yang mungkin L), dengan probabilitas munculnya r_j adalah p (rj). Maka nilai m dapat diperoleh persamaan 5.16.

$$m = \sum_{j=0}^{L-1} r_j p\left(r_j\right) \tag{5. 16}$$

Nilai rata-rata *grayscale* juga dapat dihitung langsung dari nilai piksel dari citra asli $f\left(x,y\right)$ dengan ukuran *M* × *N*. Dimana mean merupakan deskriptor yang sangat kompak (satu nilai acuan per citra atau objek) yang menyediakan ukuran kecerahan keseluruhan dari citra atau objek yang sesuai, dan merupakan invarian RST. Namun dilihat dari sisi negatif, nilai rata-rata cenderung memiliki ekspresi yang sangat terbatas dan sangat diskriminatif. Nilai rata-rata dapat diperoleh dengan persamaan 5.17.

$$m = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f\left(x,y\right) \tag{5. 17}$$

Kuadrat dari standar deviasi adalah varians, yang juga dikenal sebagai momen orde dua yang dinormalisasi dari citra. Standar deviasi memberikan representasi singkat dari keseluruhan kontras. Menyerupai dengan rata-rata, standar deviasi kompak dan invarian RST, tetapi memiliki ekspresi terbatas dan sangat diskriminatif. Standar deviasi bergantung pada nilai m yang merupakan nilai rata-rata citra, sehingga standar deviasi dapat diperoleh dengan persamaan 5.18.

$$\acute{o} = \sqrt{\sum_{j=0}^{L-1} \left(r_j - m\right)^2 p\left(r_j\right)} \qquad (5.18)$$

Kemiringan histogram adalah ukuran simetri tentang tingkat rata-rata. Tanda dari kemiringan menunjukkan apakah ekor histogram menyebar ke kanan (positif) atau ke kiri (negatif). Kemiringan ini juga dikenal sebagai momen orde ketiga dari citra yang dinormalisasi, yang didapatkan dengan melalui persamaan 5.19.

$$skew = \frac{1}{\acute{o}^3} \sum_{j=0}^{L-1} \left(r_j - m\right)^3 p\left(r_j\right) \qquad (5.19)$$

Namun, jika nilai rata-rata citra (m), standar deviasi (σ), dan mode didefinisikan sebagai puncak tertinggi pada histogram telah diketahui, kemiringannya dapat dihitung dengan persamaan 5.20.

$$\text{skew} = \frac{\text{m} - \text{mode}}{\acute{o}} \qquad (5.20)$$

Deskripsi energi memberikan ukuran lain tentang bagaimana nilai piksel didistribusikan sepanjang rentang *grayscale*: citra dengan nilai konstan tunggal memiliki energi maksimum (seperti energi = 1). Namun, citra dengan beberapa *grayscale* akan memiliki energi lebih

tinggi daripada yang memiliki banyak *grayscale*. Deskriptor energi dapat dihitung berdasarkan persamaan 5.21.

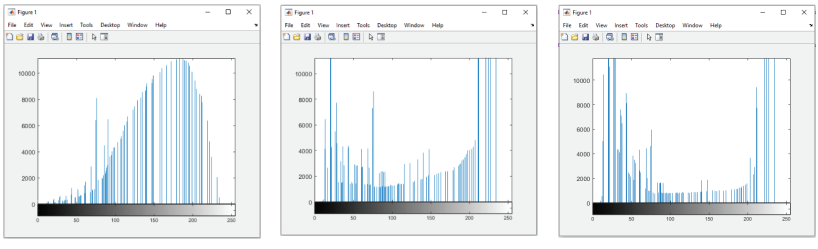
$$energi = \sum_{j=0}^{L-1} \left[p\left(r_j\right) \right]^2 \tag{5. 21}$$

Histogram juga dapat memberikan informasi tentang kompleksitas citra, dalam bentuk entropi. Semakin tinggi entropi, semakin kompleks citra. Entropi dan energi cenderung berbeda secara terbalik satu sama lain. Deskripto entrodi dapat diperoleh dengan persamaan 5.22.

$$entropi = -\sum_{j=0}^{L-1} p\left(r_j\right) \log_2 \left[p\left(r_j\right) \right] \tag{5. 22}$$

5.7. FITUR TEKSTUR

Tekstur bisa menjadi deskriptor yang kuat dari suatu citra (atau salah satu wilayah dalam citra). Meskipun tidak ada definisi tekstur yang disepakati secara universal, teknik pemrosesan citra biasanya mengaitkan gagasan tekstur dengan properti citra (atau wilayah) seperti halus (atau kebalikannya, kekasaran), kasar, dan teratur.



(a) (b) (c)

Gambar 5. 12 Contoh citra dengan tekstur halus (a), kasar (b), dan reguler (c). Citra dari set data tekstur

Ada tiga pendekatan utama untuk mengcitrakan properti tekstur dalam pemrosesan citra: struktural, spektral, dan statistik. Pendekatan statistic banyak digunakan karena popularitas, kegunaan, dan kemudahan dalam komputasi. Salah satu rangkaian fitur statistik paling sederhana untuk deskripsi tekstur terdiri dari deskriptor berbasis histogram citra (atau wilayah): mean, varian (atau akar kuadratnya, standar deviasi), kemiringan, energi (digunakan sebagai ukuran keseragaman), dan entropi. Varians kadang-kadang digunakan sebagai deskriptor normalisasi kekasaran (R), yang dipengaruhi oleh σ_2 yang merupakan varians setelah dinormalisasi (ke interval [0, 1]). Nilai R yang merupakan nilai terkecil pada area konstan, memberikan makna bahwa area tersebut memiliki tekstur halus. Sehingga R dapat diperoleh dengan persamaan 5.23.

$$R = 1 - \left(\frac{1}{\text{ϕ} + \quad ^2} \right) \tag{5. 23}$$

Penjelasan tekstur statistik untuk tiga objek pada Gambar 5.12, diperlihatkan pada Tabel 5.4. Memperlihatkan bahwa Tekstur reguler memiliki keseragaman tertinggi (dan entropi terendah) dari ketiganya. Selain itu, tekstur kasar menunjukkan nilai yang lebih tinggi untuk Roughness yang dinormalisasi daripada tekstur halus.

Tabel 5. 4 Properti Wilayah Berlabel

Texture	Mean	Standard deviation	Roughness	Skew	Uniformity	Entropy
Halus	147.1459	47.9172	0.0341	-0.4999	0.0190	5.9223
Kasar	138.8249	81.1479	0.0920	-1.9095	0.0306	5.8405
Regular	77.2129	64.4364	0.0600	4.4096	0.1100	4.1316

Contoh deskriptor tekstur berbasis histogram dibatasi oleh fakta bahwa histogram tidak membawa informasi tentang hubungan

spasial antar piksel. Salah satu cara untuk menghindari batasan ini adalah dengan menggunakan representasi alternatif untuk nilai-nilai piksel yang mengkodekan posisi relatif mereka terhadap satu sama lain. Salah satu representasi tersebut adalah *gray-level cooccurrence matrix* G, didefinisikan sebagai matriks yang elemennya $g(i,j)$ mewakili berapa kali pasangan piksel dengan intensitas zi dan zj terjadi pada citra $f(x,y)$ pada posisi yang ditentukan oleh operator d. Vektor d dikenal sebagai vektor perpindahan atau *displacement vector*, sesuai dengan persamaan 5.24.

$$d = (d_x, d_y) \tag{5. 24}$$

di mana dx dan dy adalah perpindahan dalam piksel, di sepanjang baris dan kolom masing-masing dari citra.

Gambar 5.13 menunjukkan contoh *gray-level cooccurrence matrix* untuk d = (0,1). Array di sebelah kiri adalah gambar $f(x,y)$ ukuran 4 × 4 dan L = 8 (L adalah jumlah total tingkat abu-abu). Array di sebelah kanan adalah *gray-level cooccurrence matrix* G, menggunakan konvensi $0 \le i, j < L$. Setiap elemen G sesuai dengan jumlah kemunculan piksel tingkat abu-abu saya terjadi di sebelah kiri piksel tingkat abu-abu j. Sebagai contoh, karena nilai 6 muncul di sebelah kiri dari nilai 3 pada gambar asli empat kali, nilai $g(6,3)$ sama dengan 4.

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

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

Gambar 5. 13 Contoh Citra (a) dan cooccurrence matrix untuk d = (0,1) (b).

Gray-level cooccurrence matrix dapat dinormalisasi sebagai berikut

$$N_g(i,j) = \left(\frac{g(i,j)}{\sum_i \sum_j g(i,j)} \right) \tag{5.25}$$

di mana $N_g(i,j)$ adalah *gray-level cooccurrence matrix* yang dinormalisasi. Karena semua nilai $N_g(i,j)$ terletak antara 0 dan 1, mereka dapat dianggap sebagai probabilitas bahwa sepasang poin yang memenuhi d akan memiliki nilai (z_i, z_j). Matriks cooccurrence dapat digunakan untuk mewakili properti tekstur dari suatu gambar. Alih-alih menggunakan seluruh matriks, deskriptor yang lebih kompak lebih disukai. Ini adalah fitur berbasis tekstur yang paling populer yang dapat dihitung dari *gray-level cooccurrence matrix* yang dinormalisasi $N_g(i,j)$

$$maximum\ probability = \max\left(N_g(i,j)\right) \tag{5.26}$$

$$energy = \sum_i \sum_j N_g^2(i,j) \tag{5.27}$$

$$entropy = -\sum_i \sum_j N_g(i,j) \log_2 N_g(i,j) \tag{5.28}$$

$$contrast = \sum_i \sum_j (i-j)^2 N_g(i,j) \quad (5.29)$$

$$homogeneity = \sum_i \sum_j \frac{N_g(i,j)}{1+|i-j|} \quad (5.30)$$

$$Correlation = \sum_i \sum_j \frac{(i-\mu_i)(j-\mu_j) N_g(i,j)}{\sigma_i \sigma_j} \quad (5.31)$$

di mana $\mu i, \mu j$ adalah rata-rata dan $\sigma i, \sigma j$ adalah standar deviasi dari jumlah baris dan kolom $Ng(i)$ dan $Ng(j)$, didefinisikan sebagai

$$N_g(i) = \sum_i N_g(i,j) \quad (5.32)$$

$$N_g(j) = \sum_j N_g(i,j) \quad (5.33)$$

5.8. TUTORIAL FEATURE EXTRACTION DAN REPRESENTASI

Prosedur 1

Langkah 1. Baca citra coin.png dengan menjalankan pernyataan berikut:

```
I = imread('coins.png');
Imshow(I)
```

Langkah 2. Gunakan batas-batas untuk menampilkan batas-batas objek pada gambar uji

```
[B, L] = bwboundaries(J);
figure; imshow(J);
hold on;
for k = 1: length(B),
    boundary = B{k};
```

```
    plot (boundary (: 2), boundary (:1), 'g', 'LineWidth', 2);
end
```

Langkah 3. Gunakan bwlabel untuk memberi label pada wilayah yang terhubung (misal., Objek) pada gambar uji, dan tampilkan masing-masing dengan label numerik terkait.

```
[L, N] = bwlabel(J);
RGB = label2rgb (L,'hsv', [.5 .5 .5], 'shuffle');
figure; imshow(RGB); hold on;
for k=1: length(B), boundary = B{k};
    plot (boundary (:, 2), boundary(:, 1),'w','LineWidth',2);
   text (boundary (1,2)-11, boundary (1,1) + 11, num2str(k),'Color','y',
'FontSize', 14, 'FontWeight', 'bold');
end
```

Pertanyaan 1: Berapa nilai N yang dikembalikan oleh bwlabel?

Langkah 4. Gunakan regionprops untuk mengekstrak fitur biner berikut untuk setiap objek dalam gambar (kotak kiri atas, kotak kanan atas, lingkaran kecil, lingkaran besar): area, centroid, orientation, Euler number, eccentricity, aspect ratio, perimeter, and thinness ratio.

Langkah 5. Atur nilai-nilai fitur dan nama-nama objek dalam sebuah tabel (lihat Tabel 5.5), untuk analisis komparatif yang lebih mudah.

```
stats = regionprops(L,'all');
temp = zeros(1,N);
for k = 1:N % Compute thinness ratio
temp(k) = 4*pi*stats(k,1).Area / (stats(k,1).Perimeter)^2;
stats(k,1).ThinnessRatio = temp(k); % Compute aspect ratio
temp(k) = (stats(k,1).BoundingBox(3))/(stats(k,1).BoundingBox(4));
stats(k,1).AspectRatio = temp(k);
end
```

Tabel 5. 5 Properti Wilayah Berlabel

Object	Area	Centroid	Orien-tation	Euler Number	Eccen-tricity	Aspect ratio	Perimeter	Thiness ratio
Top left square								
Big circle								
Small circle								
Top right square								

Pertanyaan 2 : Apakah hasil yang diperoleh untuk fitur yang diekstraksi sesuai dengan harapan Anda?

Pertanyaan 3 : Manakah dari fitur yang diekstrak memiliki kekuatan diskriminatif terbaik untuk membantu membedakan kotak dari lingkaran?

Pertanyaan 4 : Manakah dari fitur yang diekstraksi memiliki kekuatan diskriminatif terburuk untuk membantu membedakan kotak dari lingkaran?

Pertanyaan 5 : Manakah dari fitur yang diekstraksi yang ST invariant, yaitu, kuat untuk perubahan ukuran?

Pertanyaan 6 : Jika Anda harus menggunakan hanya satu fitur untuk membedakan kotak dari lingkaran, dengan cara ST-invariant, fitur apa yang akan Anda gunakan? Mengapa?

Langkah 6. Plot vektor fitur 2D yang diperoleh dengan menggunakan area dan rasio ketipisan masing-masing objek.

```
areas = zeros(1,N);
for k = 1:N
areas(k) = stats(k).Area;
end
TR = zeros(1,N);
```

```
for k = 1:N
TR(k) = stats(k).ThinnessRatio;
end
cmap = colormap(lines(16))
for k = 1:N
scatter(areas(k), TR(k), [], cmap(k,:), 'filled'), ylabel('Thinness Ratio'),
xlabel('Area')
hold on
end
```

Langkah 7. Ulangi langkah 1–6 untuk gambar uji yang berbeda

Langkah 8. Tulis kode MATLAB untuk mengimplementasikan klasifikasi tiga kelas heuristik yang mampu membedakan kotak dari lingkaran dari bentuk yang tidak diketahui.

Petunjuk: Gunakan subset fitur dengan kekuatan diskriminatif yang cukup dan enkode solusi Anda menggunakan pernyataan if-else-if. Gunakan potongan kode di bawah ini untuk memulai

```
name = cell(1,N);
for k = 1:N
if (TR(k) > 0.9)
        name{1,k}='circle';
else
        if (TR (k) > 0.8)
                name{1,k} = 'square';
        else
                name{1,k} = 'other';
        end
end
end
```

Langkah 9. Uji solusi Anda menggunakan 2 gambar tes.

Langkah 10. Uji solusi Anda menggunakan gambar uji yang berbeda.

Langkah 11. Perluas klasifikasi Anda untuk dapat memproses gambar berwarna.

5.9. LATIHAN PERMASALAHAN

1. Hitung perbedaan pertama untuk kode rantai: 0103212111021103.
2. Sketsa plot tanda tangan untuk gambar geometris berikut:
   (a) Persegi Panjang
   (b) Segitiga Sama Kaki
   (c) Elips
   (d) Bintang 6-titik
3. Tulis fungsi MATLAB untuk menghitung *graylevel cooccurrence matrixs* untuk gambar f (x, y) dan vektor perpindahan d, yang harus dilewatkan sebagai parameter.

Bab 6

PENGENALAN POLA VISUAL

6.1. TUJUAN

Tujuan dari bahasan ini adalah

- Mempelajari maksud dan dasar pengenalan dalam *computer vision*.
- Mempelajari pola secara visual dan kelas pola.
- Mempelajari langkah-langkah yang biasa digunakan dalam merancang, membangun dan menguji klasifikasi pola kelas.
- Mempelajari mengevaluasi kinerja dan klasifikasi pola visual.

6.2. PENDAHULUAN: GAMBARAN UMUM

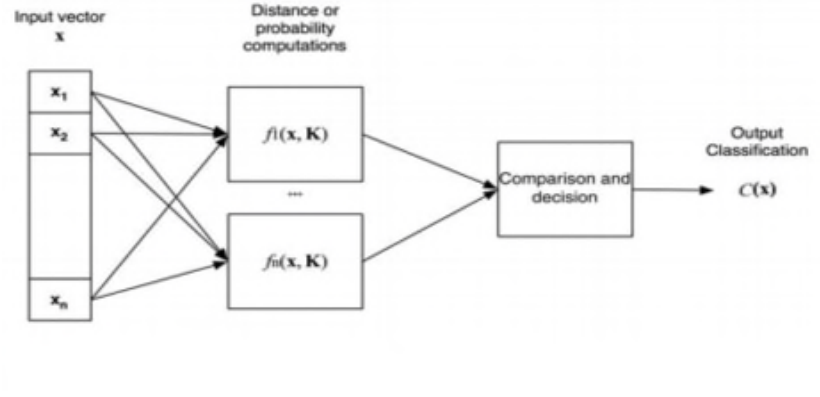
Dalam bagian ini menjelaskan konsep dasar dari Pattern Recognition atau disebut Pengenalan Pola dan sering disebut dengan Pattern Classification atau Klasifikasi Pola. Selain konsep dasar, juga memperkenalkan beberapa teknik representative dalam *Computer Vision*. Teknik-teknik klasifikasi ini beranggapan bahwa suatu gambar berserta kontennya dapat diproses dengan menggunakan satu atau beberapa teknik klasifikasi yang ada. Teknik klasifikasi pola bertujuan untuk menentukan kelas untuk masing-masing gambar (objek dalam gambar) berdasarkan pada representasi dari gambar atau obyek tersebut dengan menentukan property yang paling cocok untuk masalah yang dihadapi.

Teknik klasifikasi pola dibagi ke dalam dua kelompok utama yaitu statistik dan sintatik. Fokus utama pada bagian ini adalah teknik pengenalan pola statistic, yang menganggap bahwa setiap setiap objek atau kelas dapat direpresentasikan sebagai fitur vektor dan memberi keputusan untuk menentukan kelas dari bentuk pola yang terbentuk berdasarkan perhitungan jarak dan model probabilistic. Teknik berkerja dengan fitur vektor numerik pada gambar. Teknik-

teknik dapat diterapkan untuk penentuan kelas pada masalah lain selain pemrosesan image atau gambar. Meskipun demikian, fokus bagian ini adalah pengenalan pola visual sehingga akan menyajikan beberapa contoh dan mendiskusikan tentang pemrosesan gambar dan objek dalam gambar serta memperkenalkan metode ekstraksi fitur.

6.3. FUNDAMENTAL

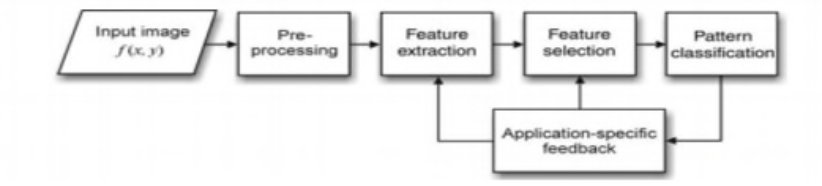
Pada bagian ini, membahas tentang konsep dan terminologi paling penting yang terkait dengan klasifikasi pola. Tujuan umum dari teknik klasifikasi pola adalah untuk menetapkan kelas ke pola yang belum diketahui sebelumnya berdasarkan pengetahuan tentang objek dan kelas tempat mereka berada. Gambar 6.1 menunjukkan sebuah diagram skematik yang menunjukkan bagaimana proses klasifikasi pola statistik informasi numerik dari fitur vektor, menghitung serangkaian jarak atau probabilitas, dan menggunakan hasil perhitungan tersebut untuk membuat keputusan tentang label kelas C (x) yang mana akan digunakan untuk menentukan pola dari setiap inputan x.



Gambar 6. 1 Diagram klasifikasi pola secara statistik

6.3.1 Desain dan Implementasi Klasifikasi Pola Visual

Desain dan implementasi dari sistem pengenalana pola secara visual biasanya proses interaktif yang melibatkan pemilihan dan perhitungan fitur dari image (gambar) atau objek yang akan kita klasifikasikan. Selain itu, sebagian besar tugas di *computer vision*, keputusan sering tergantung pada aplikasi. Gambar 6.2 menggambarkan proses dalam melakukan klasifikasi pola.



Gambar 6. 2 Interaksi antara ekstraksi fitur, pemilihan fitur, dan klasifikasi pola sebagai fungsi dari aplikasi yang ada

Desain klasifikasi pola visual secara statistik biasanya terdiri dari langkah-langkah seperti berikut ini:

1. Definisikan masalah dan tentukan jumlah kelas yang terlibat.
   Di sinilah semuanya dimulai. Pertanyaan yang sah untuk diajukan pada tahap ini meliputi berikut ini: Ada berapa kelas? Seberapa baik kelas-kelas ini menggambarkan benda atau gambar? Apakah kelas tertentu subkategori yang lain? Apakah ada reject class dalam hal ini?
2. Ekstrak fitur yang paling cocok untuk menggambarkan gambar dan memungkinkan classifier untuk memberi label yang sesuai.

   Ini adalah langkah di mana proses interaktif dan tergantung pada aplikasi ekstraksi fitur awal dan pada saat seleksi berlangsung. Itu juga merupakan langkah di mana

perancang *machine vision solution* dihadapkan dengan banyak opsi atau pilihan jenis fitur (misal., berbasis warna, berbasis tekstur, berorientasi batas, dll.) dan metode yang spesifik untuk mengekstraksinya (misal., histogram warna, deskriptor Tamura, deskriptor Fourier, dll.).

3. Pilih metode atau algoritme klasifikasi.

   Pada titik ini, kita dapat mengambil manfaat dari beragam alat dan teknik di bidang penambangan data dan pembelajaran mesin dan pilih satu yang paling cocok dengan kebutuhan kita, berdasarkan kompleksitasnya, biaya komputasi, kemampuan pelatihan, dan properti lainnya. Teknik yang dipilih bisa classifier jarak sesederhana mungkin atau serumit Support Vectoe Mechine (SVM).

4. Pilih satu set data.

   Pada tahap ini, mengumpulkan gambar representatif yang dapat digunakan sebagai data pelatihan dan data uji. Mempertimbangkan penggunaan data yang juga digunakan secara umum, karena itu akan jadi tolok ukur dari hasil pekerjaan mu.

5. Pilih subset gambar dan gunakan untuk melatih classifier.

   Banyak strategi klasifikasi pola memerlukan tahap pelatihan, di mana himpunan dari bagian gambar digunakan untuk "mengajar" pengklasifikasi tentang kelas yang seharusnya mengenali serta dapat menyesuaikan beberapa parameter classifier.

6. Uji penggolongan.

   Pada langkah ini, mengukur tingkat keberhasilan dan kesalahan dalam menghitung angka yang relevan (misasl., presisi dan penarikan kembali) dan kompilasi hasil numerik utama menjadi plot representatif (misal., kurva ROC).

7. Perbaiki dan tingkatkan solusi.

   Setelah menganalisis hasil yang dihitung pada langkah 6, mungkin harus kembali ke langkah sebelumnya, beberapa perubahan proses (misal., pilih fitur yang berbeda, memodifikasi parameter classifier, mengumpulkan gambar tambahan, dll.), dan menguji solusi yang dimodifikasi.

6.3.2 Pola dan Pola kelas

Suatu patterns atau pola dapat didefinisikan sebagai pengaturan fitur. Biasanya sebuah pola dikodekan dalam bentuk vektor fitur, strings atau trees. Vektor fitur adalah array n x 1 angka yang sesuai dengan fitur yang digunakan untuk mewakili sebuah pola tertentu (sesuai dengan persamaan 6.1). Dimana pada persamaan (6.1), n adalah jumlah total fitur dan T menunjukkan operasi perubahan urutan.

$$x = (x_1, x_2, x_3, \ldots\ldots\ldots\ldots x_n)^T \qquad (6.1)$$

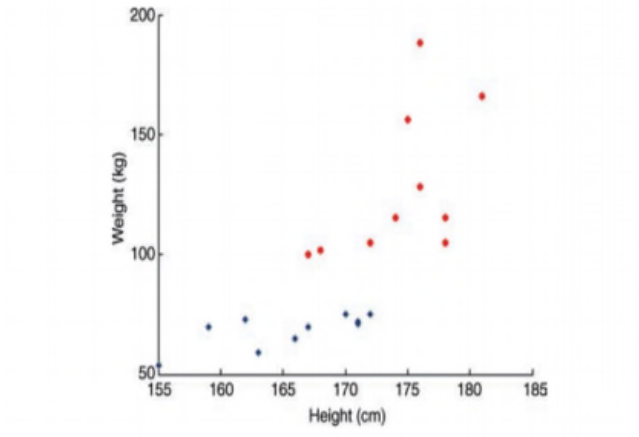
Jumlah total fitur akan tergantung pada properti dan teknik representasi yang dipilih dari objek yang digunakan untuk menggambarkan fitur tersebut. Ini jumlahnya dapat bervariasi dari satu (jika properti tertentu cukup diskriminatif untuk memungkinkan sebuah keputusan klasifikasi) hingga beberapa ribu (misal., jumlah poin untuk batas pengkodean menggunakan deskriptor Fourier).

Kelas adalah seperangkat pola yang memiliki beberapa sifat umum. Kelas yang ideal adalah salah satu di mana anggotanya sangat mirip satu sama lain (kelas memiliki tingkat kesamaan intraclass) namun sangat berbeda dari anggota kelas lain (perbedaan antar kelas adalah signifikan). Kelas pola akan direpresentasikan sebagai berikut: $\omega_1, \omega_2, \ldots, \omega_K$, di mana K adalah jumlah total kelas.

Gambar 6.3 menunjukkan contoh sederhana dari plot 2D yang mewakili fitur vektor tinggi dan berat sekelompok pemain tenis meja dan sekelompok pegulat sumo.

$$x = \left(x_1, x_2\right)^T \tag{6. 2}$$

Dalam contoh ini, $\omega 1$ adalah kelas pegulat sumo, sedangkan $\omega 2$ adalah kelas pemain tenis meja. Inspeksi visual yang sederhana memungkinkan kita untuk melihat pemisahan yang jelas antara keduanya, dua kelompok titik data yang biasanya diterjemahkan menjadi tugas yang relatif mudah untuk pengklasifikasi pola yang bertugas membedakan kedua kelompok dan mengklasifikasikan yang baru misalnya sesuai dengan kelompok yang paling mungkin dimiliki. Pengamatan lain yang dapat diturunkan dari angka yang sama adalah fitur bobot (x_1) lebih banyak diskriminatif daripada fitur ketinggian (x_2) dalam masalah khusus ini. Dalam hal ini x_1 merupakan berat dan x_2 merupakan tinggi.



Gambar 6. 3 Contoh dua kelas (pegulat sumo — lingkaran merah — dan pemain tenis meja — berlian biru) dijelaskan dengan dua pengukuran (berat dan tinggi)

6.3.3 Pengolahan Data Awal

Sebelum data numerik (misal., Kumpulan fitur vektor) dapat dimasukkan ke sebuah pola klasifikasi, seringkali diperlukan untuk melakukan langkah tambahan yang dikenal sebagai data preprocessing. Teknik preprocessing yang umum meliputi:

1. Penghapusan *Noise* Removal sering juga dikenal sebagai Outlier Removal:

   Merupakan langkah preprocessing dimana sampel data yang menyimpang terlalu jauh dari nilai rata-rata untuk suatu kelas dihapus, dengan alasan bahwa (a) mungkin ada kesalahan saat mengukur (atau mengekstraksi) sampel tertentu dan (b) sampel tersebut adalah contoh yang buruk dari struktur yang mendasari kelas.

2. Normalisasi:

   Merupakan vektor fitur yang mungkin perlu dinormalisasi sebelum jarak, kesamaan, dan perhitungan probabilitas terjadi. Ini beberapa representatif dari teknik normalisasi

   - Normalisasi Vektor Satuan atat *Unit Vector Normalization*

     Ini menegaskan bahwa semua vektor fitur memiliki besaran 1.

   - Standard Normal Density (SND)

     setiap elemen dari vektor fitur $x\left(x_1, x_2, ..., x_n\right)$ digantikan oleh nilai normal $\tilde{x}_i$, yang didapat dari persamaan (6.3) dengan di mana μ dan σ adalah rata-rata dan standar deviasi elemen dalam x.

$$\tilde{x}_i = \frac{x_i - \mu}{\sigma} \qquad (6.3)$$

   - Teknik Linear dan Nonlinier Lainnya

seperti min-max normalisasi dan softmax scaling, dengan tujuan untuk membatasi nilai fitur ke kisaran spesifik, misalnya [0, 1].

3. Penyisipan data yang hilang atau *Insertion of Missing Data*:

   Dalam langkah preprocessing terakhir ini tidak wajib ada, hanya tambahan item data, asalkan mereka mengikuti distribusi probabilistik yang serupa dan lakukan tidak bias hasilnya ditambahkan ke kumpulan data

6.3.4 Data Pelatihan dan Pengujian

Proses pengembangan dan pengujian algoritme klasifikasi pola biasanya mensyaratkan bahwa data set dibagi menjadi dua subkelompok: data set pelatihan yang digunakan untuk pengembangan algoritme dan fine-tuning, dan data set tes digunakan untuk mengevaluasi kinerja algoritme. Set pelatihan berisi kecil (biasanya 20% atau kurang) tetapi subsampel representatif dari kumpulan data yang dapat dipilih secara manual atau otomatis (misal., secara acak). Ukuran set pelatihan dan metode yang digunakan untuk membangunnya adalah sering tergantung pada teknik klasifikasi pola yang dipilih. Tujuan memiliki dua set terpisah yaitu satu untuk merancang, meningkatkan, dan menyempurnakan algoritme dan yang lain untuk evaluasi kuantitatif yang sistematis yaitu untuk menghindari bias dalam melaporkan tingkat keberhasilan pendekatan. Lagi pula, jika desainer diperbolehkan untuk bekerja pada hal yang sama mengatur data sepanjang waktu, sangat mungkin untuk mengubah solusi yang cukup untuk diproduksi kinerja yang hampir sempurna pada koleksi gambar tertentu. Mungkin akan ada tidak ada jaminan, bagaimanapun bahwa metode dan pilihan parameter yang sama akan bekerja dengan baik untuk gambar dan kumpulan data lainnya.

6.3.5 Confusion Matrix

Confusion Matriks adalah array 2D ukuran K × K (dengan K adalah jumlah total kelas) yang digunakan untuk melaporkan hasil mentah dari percobaan klasifikasi. Nilai dalam baris i, Kolom j menunjukkan berapa kali sebuah objek yang kelas aslinya adalah i, diberi label milik kelas j. Diagonal utama dari confusion matriks menunjukkan jumlah kasus di mana pengklasifikasi berhasil dan sebuah pengklasifikasi sempurna akan metampilkan semua elemen pada diagonal sama dengan nol.

Contoh:

Gambar 6.4 menunjukkan contoh confusion matriks untuk empat kelas generik $\omega_1, \ldots, \omega 4$. Analisis terhadap confusion matriks menunjukkan bahwa input yang berlabel kelas class3 diklasifikasikan dengan benar setiap saat. Kesalahan klasifikasi tertinggi (11%) untuk input berlabel class2. Confusion matrix yang paling umum ditimbulkan oleh klasifikasi adalah memberi label pada input kelas ω_2 sebagai kelas ω_3 (10% dari waktu). Selain itu, kinerja kelas untuk kelas ω_1 juga layak dikomentari: walaupun tiga input yang berlabel kelas ω_1 secara keliru diklasifikasi (dua sebagai kelas ω_3 dan satu sebagai kelas ω_4), kelas tidak memberi label input apa pun dari kelas lain sebagai kelas ω_1 (yaitu , nilai yang tersisa untuk kolom ω_1 semuanya nol).

	ω_1	ω_2	ω_3	ω_4
ω_1	97	0	2	1
ω_2	0	89	10	1
ω_3	0	0	100	0
ω_4	0	3	5	92

Gambar 6. 4 confusion matriks ukuran 4 4x4

6.3.6 Sistem Kesalahan

Analisis kinerja kuantitatif dari pengklasifikasi pola yang biasanya melibatkan pengukuran tingkat kesalahan. Hal-hal lain, seperti kecepatan dan kompleksitas komputasi, mungkin penting tetapi tingkat kesalahan lebih penting. Berapa kali tingkat kesalahan mengukur classifier yang terjadi dalam kesalahan klasifikasi yaitu mengklasifikasikan objek input sebagai class ωp ketika kelas yang benar adalah ωq, p ≠ q. Tingkat kesalahan klasifikasi biasanya ditentukan secara empiris, yaitu dengan mengukur jumlah (persentase) kesalahan yang diamati saat menguji classifier terhadap data set tes.

Contoh:

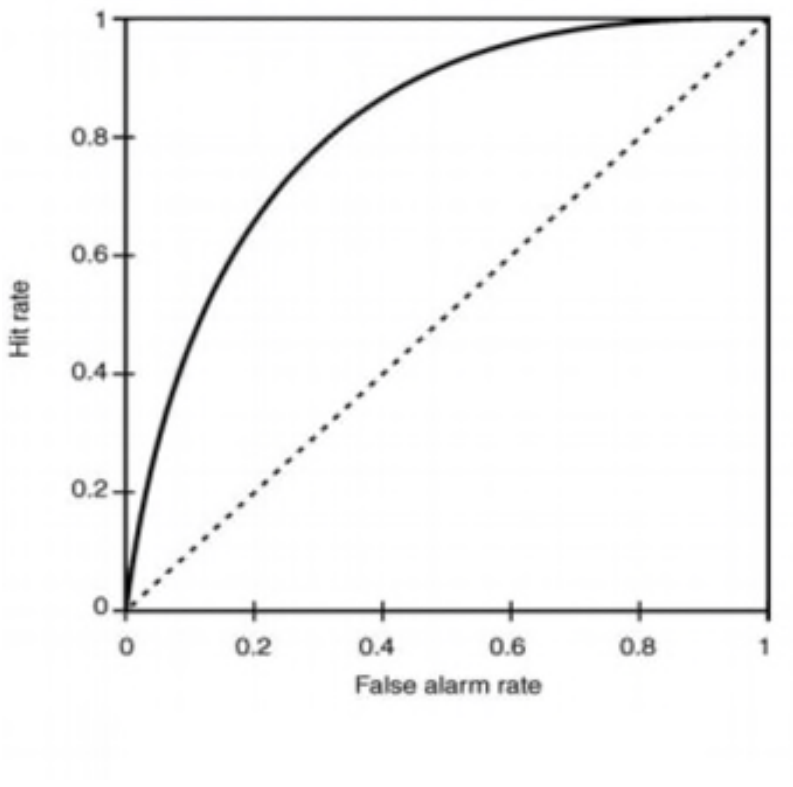
Confusion matriks pada Gambar 6.4 dengan asumsi bahwa semua kelas memiliki jumlah objek yang sama, tingkat kesalahan keseluruhan kelas akan menjadi (3 + 11 + 0 + 8) / (4 × 100) = 5,5%.

6.3.7 Hit Rates, False Alarm Rates, and Kurva ROC

Banyak masalah klasifikasi pola secara visual menggunakan dua kelas classifier. Contoh kasus klasik pada *computer vision* adalah tugas deteksi objek, di mana algoritme *computer vision* disajikan dengan gambar dan pertanyaan: "Apakah objek hadir dalam gambar ini atau tidak? "Jika algoritme berhasil menjawab ya (dan menunjuk ke bagian mana dalam gambar objek itu terletak) ketika objek itu hadir, itu disebut benar positif. Jika algoritme menjawab dengan benar tidak ketika objek tidak ada, itu disebut benar negatif.

Ada dua kemungkinan kesalahan yang dapat dilakukan algoritme: menjawab ya saat tidak ada dari suatu objek (ini disebut alarm palsu atau false positif) atau menjawab tidak ketika objek hadir, yaitu kehilangan objek (ini disebut false negative).

Biaya false positif atau false negatif tergantung pada aplikasi dan dapat mengarah pada hasil yang sangat berbeda. Dalam aplikasi pengawasan, misalnya, akan mungkin yang terbaik adalah memiliki false positif sesekali (mis., memperingatkan operator manusia untuk memprediksi keberadaan objek yang ada di layar dimana sesungguhnya tidak ada objek seperti itu) daripada melewatkan prediksi objek yang nyata. Dalam hal demikian, sebuah tujuan yang sah adalah untuk meminimalkan tingkat false negatif, bahkan jika itu hanya dapat dicapai dengan mentolerir tingkat false positif yang relatif tinggi.



Gambar 6. 5 ROC Curve

Kurva karakteristik operasi penerima (atau hanya ROC) adalah plot yang menunjukkan hubungan antara deteksi yang benar (true positive) rate (juga dikenal sebagai hit rate) dan alarm palsu (false positive) rate. Gambar 6.5 menunjukkan contoh generik Kurva ROC. Ini juga menunjukkan garis lurus putus-putus yang sesuai dengan kinerja sebuah classifier beroperasi secara kebetulan (misal., menebak ya atau tidak setiap kali). ROC yang ideal adalah salah satu di mana "lutut" kurva adalah dekat dengan sudut kiri atas grafik, menunjukkan tingkat hit mendekati 100% dengan tingkat alarm palsu mendekati nol.

6.3.8 Precision dan Recall

Aplikasi pemrosesan gambar tertentu, terutama pengambilan gambar memiliki tujuan untuk mengambil gambar yang relevan dan tidak mengambil yang tidak relevan. Ukuran kinerja yang digunakan dalam pengambilan gambar dari bidang informasi (dokumen) dan didasarkan pada dua hal utama yakni ketepatan dan daya ingat. Presisi adalah jumlah dokumen yang relevan yang diambil oleh sistem dan dibagi dengan jumlah total dokumen yang diambil (misal., positif asli ditambah alarm palsu). Recall adalah nomor dokumen yang relevan diambil oleh sistem dibagi dengan jumlah total yang relevan dokumen dalam basis data (yang seharusnya diambil).

Presisi dapat diartikan sebagai ukuran ketepatan, sedangkan recall menyediakan sebuah ukuran kelengkapan. Skor presisi sempurna dari 1,0 berarti bahwa setiap dokumen diambil (atau gambar dalam kasus ini) relevan, tetapi tidak memberikan wawasan apa pun tentang apakah semua dokumen yang relevan diambil. Skor recall sempurna 1,0 berarti bahwa semua gambar yang relevan diambil, tetapi tidak mengatakan mungkin juga telah diambil berapa gambar banyak yang tidak relevan.

Pengukuran presisi (P) dan recall (R) juga dapat diadaptasi dan digunakan dalam tugas-tugas klasifikasi dan diekspresikan dalam bentuk true positive (tp), false positive (fp), dan false negative (fn) pada persamaan (6.4) dan persamaan (6.5).

$$P = \frac{tp}{tp + fp} \tag{6.4}$$

$$R = \frac{tp}{tp + fn} \tag{6.5}$$

Dalam hal ini, skor presisi 1.0 untuk kelas ω_i berarti setiap item diberi label sebagai milik kelas ω_i memang milik kelas ω_i, tetapi tidak mengatakan apa - apa tentang jumlah item dari kelas ωi yang tidak diberi label dengan benar. Penarikan 1,0 berarti bahwa setiap item dari kelas ω_i diberi label sebagai milik kelas ωi, tetapi tidak mengatakan apa-apa tentang berapa banyak item lain yang juga salah diberi label sebagai milik kelas ω_i.

Contoh:

Confusion matriks pada Gambar 6.4, presisi dan recall per kategori dapat dihitung sebagai berikut

$$P1 = \frac{97}{97 + 0 + 0 + 0} = 100\%$$

$$P2 = \frac{89}{0 + 89 + 0 + 3} = 96.74\%$$

$$P3 = \frac{100}{2 + 10 + 100 + 5} = 85.47\%$$

$$P4 = \frac{92}{1 + 1 + 0 + 92} = 97.87\%$$

$$R1 = \frac{97}{97+0+2+1} = 97\%$$

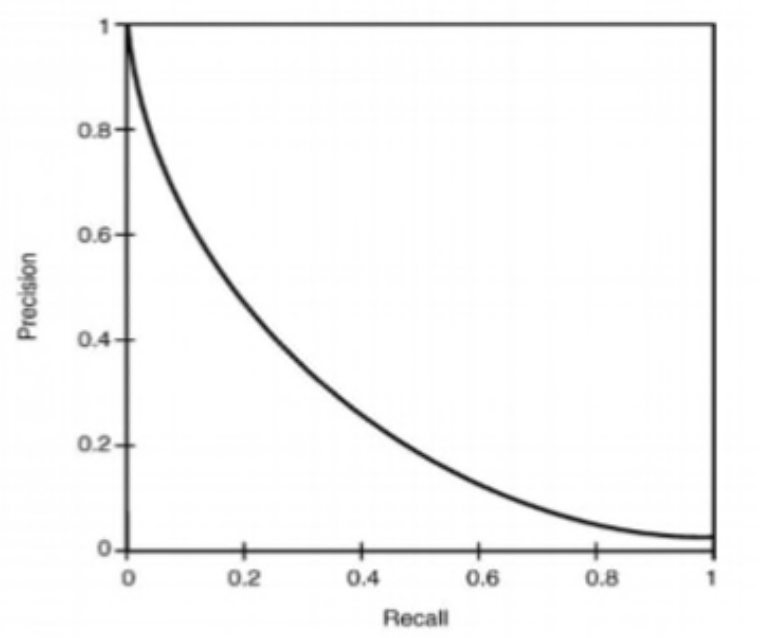
$$R2 = \frac{89}{0+89+10+1} = 89\%$$

$$R3 = \frac{100}{0+0+100+0} = 100\%$$

$$R4 = \frac{92}{0+3+5+92} = 92\%$$

Dalam hal ini, hasil memperlihatkan bahwa classifier menunjukkan presisi yang sempurna untuk kelas ω_1 dan recall yang sempurna untuk kelas ω_3.

Presisi dan recall biasanya saling berkaitan dan peningkatan biaya salah satunya adalah penurunan yang tidak diinginkan di sisi lainnya. Dalam hal sistem pengambilan dokumen (atau gambar), pilihan pengambilan dokumen yang lebih sedikit dapat meningkatkan presisi dengan memberikan nilai recall yang rendah, sedangkan pengambilan dokumen yang terlalu banyak recall memberikan nilai presisi yang lebih rendah. Pertukaran ini sering dinyatakan dalam plot, yang dikenal sebagai grafik presisi-recall (atau hanya PR). Grafik PR diperoleh dengan menghitung presisi pada berbagai recall. Grafik PR ideal menunjukkan nilai presisi sempurna pada setiap tingkat penarikan hingga titik di mana semua dokumen yang relevan (dan hanya itu) telah diambil; dari titik ini jatuh secara monoton sampai titik di mana daya ingat mencapai 1. Gambar 6.6 menunjukkan contoh grafik PR secara umum.



Gambar 6. 6 Graph presisi recall

Representasi yang lebih sesuai dari sifat presisi dan recall dari suatu sistem menyederhanakan kedua nilai tersebut ke dalam satu persamaan sebagai berikut, seperti ukuran F (juga dikenal sebagai ukuran F1) yang menghitung nilai rata-rata dari presisi dan recall. Persamaan hubungan antara presisi dan recall dapat di lihat pada persamaan (6.4) dengan P adalah presisi dan R adalah recall.

$$F1 = 2 \times \frac{P \times R}{P + R} \qquad (6.6)$$

6.3.9 Distance dan Similarity Measures

Dua fitur vektor dapat dibandingkan satu sama lain dengan cara menghitung jarak (misal., mengukur) antara distance dan

similarity Measures atau sebaliknya menetapkan derajat kesamaan. Ada banyak ukuran jarak yang digunakan dalam klasifikasi pola secara visual. Diberikan dua vektor fitur $a=(a_1,a_2,\ldots,a_n)^T$ dan $b=(b_1,b_2,\ldots,b_n)^T$. Beberapa persamaan untuk pengukuran jarak yang paling banyak digunakan seperti Euclidiean distance (persamaan (6.7)), Mahattan atau city block distance (persamaan (6.8)), dan Minskowski distance (persamaan (6.9)). Dengan r adalah bilangan bulat positif maka jarak Minkowski untuk r = 1 dan r = 2 sama dengan jarak Manhattan dan Euclidean.

$$d_E=\sqrt{\sum_{i=1}^{n}(a_i-b_i)^2} \tag{6.7}$$

$$d_C=\sum_{i=1}^{n}|a_i-b_i| \tag{6.8}$$

$$d_M=\left[\sum_{i=1}^{n}|a_i-b_i|^r\right]^{\frac{1}{r}} \tag{6.9}$$

Dalam MATLAB

Menghitung jarak dalam MATLAB sangat mudah, berkat kemampuan penanganan matriks MATLAB. Untuk jarak Euclidean antara dua vektor x dan y, kita dapat menggunakan fungsi norma sebagai alternative atau $d_E=norm(x-y)$. Meskipun ukuran jarak berbanding terbalik dengan gagasan (dan ukuran) kesamaan, ada ukuran kesamaan lainnya dalam literatur, seperti *vector inner product* (persamaan (6.10))dan *Tanimoto metric* (persamaan (6.11)).

$$\sum_{i=1}^{n}a_ib_i=a_1b_1+a_2b_2+\ldots+a_nb_n \tag{6.10}$$

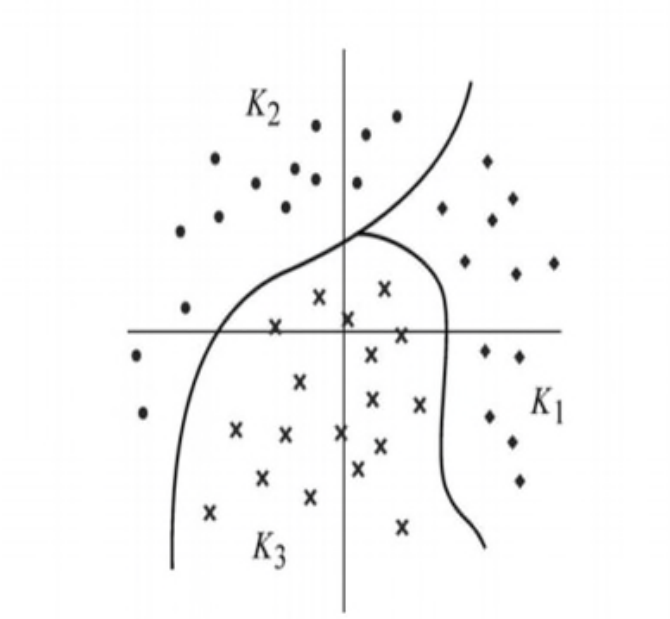
$$\frac{\sum_{i=1}^{n}a_ib_i}{\sum_{i=1}^{n}a_i^2+\sum_{i=1}^{n}b_i^2-\sum_{i=1}^{n}a_ib_i} \tag{6.11}$$

6.4. TEKNIK KLASIFIKASI POLA STATISTIK

Terdapat beberapa teknik klasifikasi polas secara statistik, seperti: minimum distance classifier, the k-nearest neighbors (KNN) classifier, and the maximum likelihood (or Bayesian) classifier. Tujuan bagian ini adalah untuk menyajikan beberapa alternatif untuk membangun pengenalan pola visual.

Properti objek dapat direpresentasikan menggunakan fitur vektor yang diproyeksikan ke ruang fitur. Jika fitur yang digunakan untuk mewakili objek (dan kelas tempat mereka berasal) dipilih dengan benar, hasilnya poin dalam ruang fitur n-dimensi akan didistribusikan dengan cara yang berkorelasi kedekatan dalam ruang fitur dengan kesamaan di antara objek yang sebenarnya. Dengan kata lain, vektor fitur yang terkait dengan objek dari kelas yang sama akan muncul berdekatan sebagai cluster di ruang fitur.

Tugas teknik klasifikasi pola statistik adalah menemukan diskriminasi kurva (atau *hypersurface*, dalam kasus ruang fitur n-dimensi) yang dapat memberitahu kelompok (dan kelas yang sesuai) terpisah. Gambar 6.7 menggambarkan konsep untuk classifier tiga kelas dalam ruang fitur 2D. Pengklasifikasi pola statistik memiliki n input (fitur objek yang akan diklasifikasikan, dikodekan ke vektor fitur $x=\left(x_1, x_2, ..., x_n\right)^T$) dan satu output (kelas yang mana objek milik, $C\ (x)$, diwakili oleh salah satu simbol $\omega_1, \omega_2, ..., \omega_W$, di mana W adalah jumlah total kelas). Simbol ω_w disebut pengidentifikasi kelas.



Gambar 6. 7 Fungsi diskriminasi untuk kelas tiga kelas dalam ruang fitur 2D

Pengklasifikasi membuat perbandingan antara representasi objek yang tidak dikenal dan kelas yang dikenal. Perbandingan ini memberikan informasi untuk membuat keputusan tentang kelas mana yang akan ditetapkan ke pola yang tidak diketahui. Keputusan menugaskan sebuah pola input ke kelas ω_i daripada kelas lain ω_j didasarkan pada sisi mana dari pembeda antara kelas objek yang tidak diketahui pasti. Secara matematis, tugas pengklasifikasi adalah menerapkan serangkaian aturan keputusan yang membagi ruang fitur menjadi subset W secara terpisah, $K_w, w=1,2, \ldots, W$, masing-masing yang mencakup vektor fitur x dimana $d(x)=\omega_w$, di mana $d(\cdot)$ adalah aturan keputusan.

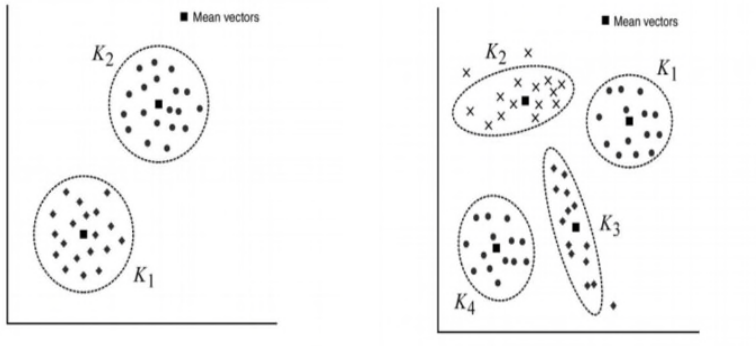
6.4.1 Minimum Distance Classifier

Minimum distance classifier atau klasifikasi jarak minimum dikenal sebagai nearest-class mean classifier yang bekerja dengan cara menghitung jarak matrik antara fitur vektor yang tidak dikenal dan centroids (mis., vektor rata-rata) dari setiap kelas sesuai dengan persamaan (6.12). Dimana dj adalah matrik jarak (antara kelas j dan fitur vektor yang tidak diketahui x) dan m_j adalah vektor rata-rata untuk kelas j yang didefinisikan dalam persamaan (6.13). Dengan N_j adalah jumlah pola vektor dari kelas ω_j.

$$d_j(x) = x - m_j \tag{6. 12}$$

$$m_j = \frac{1}{N_j}\sum_{x\in\omega_j} x_j \tag{6. 13}$$

Gambar 6.8a menunjukkan contoh dua kelas dan vektor rata-ratanya. Klasifikasi jarak minimum bekerja dengan baik untuk beberapa kelompok kelas, tetapi tidak bias menangani kasus yang lebih kompleks, seperti yang digambarkan dalam Gambar 6.8b.



(a) Dua kelas (b) Empat kelas

Gambar 6. 8 Contoh berdasarkan kelas

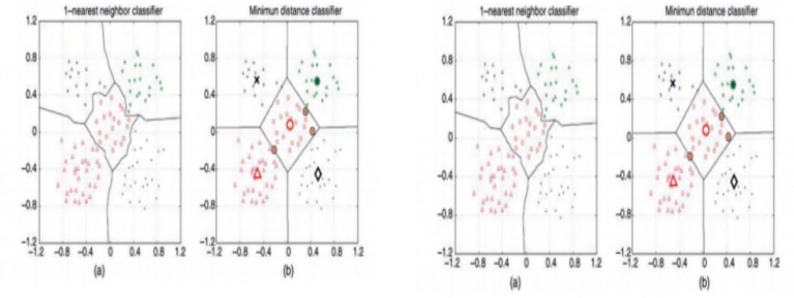
Pada kasus ini, ada dua masalah yang layak disebutkan: (i) kelas A (kluster K1 dan K4) bersifat multimodal, yaitu sampelnya terletak pada dua kluster yang terpisah (walaupun sama), dan vektor rata-rata terletak pada titik di ruang fitur yang sebenarnya di luar kedua kelompok; (ii) kelas B (kluster K2) dan C (kluster K3) memiliki bentuk yang tidak beraturan yang berpotensi menyebabkan keputusan klasifikasi yang berbeda untuk dua pola yang tidak diketahui yang terletak pada jarak yang sama dari vektor rata-rata mereka. Masalah yang terakhir bisa diatasi dengan menggunakan jarak matrik yang dimodifikasi, yang dikenal sebagai skala Euclidean berskala (persamaan (6.14)), sedangkan masalah sebelumnya biasanya membutuhkan skema pengklasifikasi lebih kompleks. Dengan d_E merupakan jarak Euclide yang diskalakan antara pola yang tidak diketahui dengan kelas j, x merupakan fitur vektor dari pola yang tidak diketahui, x_j merupakan vektor rata-rata kelas j, σ_i merupakan standar deviasi kelas j sepanjang dimensi i, dan n merupakan dimensi ruang fitur.

$$d_E = x - x_j = \sqrt{\sum_{i=1}^{n} \left| \frac{\left(x[i] - x_j[i] \right)}{\sigma_i} \right|^2} \qquad (6.14)$$

6.4.2 Klasifikasi k-Nearest Neighbors

Klasifikasi k-Nearest Neighbors (KNN) bekerja dengan menghitung jarak antara sebuah fitur vektor pola yang tidak diketahui x dan k terdekat di ruang fitur, dan kemudian menetapkan pola yang tidak diketahui ke kelas yang mayoritas k dari poin sampel. Keuntungan utama dari pendekatan ini adalah kesederhanaannya (misal., tidak perlu dibuat asumsi tentang distribusi probabilitas dari setiap kelas) dan keserbagunaan (misal., ia menangani kelas yang tumpang tindih atau kelas dengan struktur yang kompleks).

Kerugian utamanya adalah biaya komputasi yang terlibat dalam jarak komputasi antara sampel yang tidak diketahui dan banyak (berpotensi semua, jika pendekatan brute force adalah digunakan) poin yang disimpan dalam ruang fitur.



(a) Penggunaan KNN dengan k = 1 (b) Hasil klasifikasi jarak terkecil

Gambar 6. 9 Contoh KNN dengan 5 kelas

Gambar 6.9a mengilustrasikan konsep classifier lima kelas dalam fitur ruang 2D, dimana k = 1. Jelas menunjukkan bahwa pengklasifikasi KNN mampu memperoleh fungsi diskriminasi berbentuk tidak teratur di dalam sebuah kelas. Ini berbeda dengan klasifikasi jarak minimum, yang akan dibatasi hanya menggunakan garis lurus sebagai fungsi diskriminasi, seperti yang ditunjukkan pada Gambar 6.9b, yang juga menyoroti vektor untuk setiap kelas dan tiga titik data yang akan ditinggalkan kelas mereka.

6.4.3 Klasifikasi Naïve Bayes

Alasan di balik klasifikasi Bayesian adalah bahwa keputusan klasifikasi dapat dibuat berdasarkan distribusi probabilitas sampel pelatihan untuk setiap kelas; itu merupakan, objek yang tidak

dikenal ditugaskan ke kelas yang lebih mungkin untuk dimiliki pada fitur yang diamati.

Perhitungan matematis yang dilakukan oleh classifier Bayesian membutuhkan tiga distribusi probabilitas:

- Probabilitas sebelumnya untuk setiap kelas ω_k, dilambangkan dengan $P(\omega_k)$.
- Distribusi tanpa syarat dari fitur vektor mewakili yang diukur pola x, dilambangkan dengan $p(x)$.
- Distribusi bersyarat kelas yaitu probabilitas x yang diberikan kelas k, dilambangkan dengan $p(x|\omega_k)$.

Ketiga distribusi ini kemudian digunakan menerapkan aturan Bayes, untuk menghitung sebuah probabilitas posteriori bahwa pola x berasal dari kelas ω_k, direpresentasikan sebagai $p(\omega_k|x)$ pada persamaan (6.15).

$$p(\omega_k|x) = \frac{p(x|\omega_k)P(\omega_k)}{p(x)} = \frac{p(x|\omega_k)P(\omega_k)}{\sum_{k=1}^{W} p(x|\omega_k)P(\omega_k)} \qquad (6.15)$$

Desain classifier Bayes mensyaratkan bahwa probabilitas sebelumnya untuk setiap kelas $P(\omega_k)$ dan distribusi bersyarat kelas $p(x|\omega_k)$ telah diketahui. Probabilitas sebelumnya mudah untuk dihitung, berdasarkan jumlah sampel per kelas dan total jumlah sampel. Memperkirakan masalah distribusi kelas bersyarat yang jauh lebih sulit, dan seringkali ditangani oleh pemodelan fungsi densitas probabilitas (PDF) dari setiap kelas sebagai distribusi Gaussian (normal).

6.5. TUTORIAL KLASIFIKASI POLA

Tujuan dari tutorial ini adalah

1. mempelajari bagaimana membangun solusi optical character recognition (OCR) kecil untuk mengklasifikasikan digit (antara 0 dan 9) menggunakan regionprops dan klasifikasi KNN.
2. Mempelajari bagaimana mempersiapkan dan memproses pelatihan dan menguji set data.
3. Mempelajari cara melakukan pemilihan fitur.
4. Mempelajari cara mempresentasikan hasil klasifikasi menggunakan matriks kebingungan.

Perlengkapan yang diperlukan

- Skrip kode dan gambar uji tersedia dari situs web buku (ocr_example.zip).

Prosedur 1

Langkah 1. Unduh file ocr_example.zip dan unzip isinya, dengan mempertahankan struktur folder. File ini berisi semua skrip dan data yang Anda butuhkan, disusun dalam folder yang bermakna.

Langkah 2. Jalankan skrip load_data.m. Script ini akan memuat 1000 gambar pelatihan dan 1000 estimasi (10 gambar per karakter antara 10 dan 9) untuk mencetak folder.

```
load_data
```

Langkah 3. Periksa konten dari scriptload_data.m untuk lebih memahami bagaimana skip ini melakukan apa yang dilakukan.

Langkah 4. Jalankan skrip preprocess_images.m. dan script ini akan memproses pelatihan dan menguji gambar dan melakukan penghapusan outlier.

```
preprocess_images
```

Pertanyaan 1 : Apa jenis operasi preprocessing yang diterapkan untuk semua pelatihan dan gambar uji?

Pertanyaan 2 : Bagaimana outlier diidentifikasi dan ditangani?

Langkah 5. Ekstrak fitur dari gambar latihan yang telah diproses menggunakan alat peraga wilayah dan memilih dua properti dengan daya diskriminatif potensial: eksentrisitas dan nomor Euler.

Langkah 6. Simpan hasilnya ke dalam array 2 × 1000 yang sesuai dengan 1000 vektor fitur (masing-masing berukuran 2 × 1).

```
fv = zeros(2,Ntrain);
for k = 1:Ntrain
        class = trn_data.y(k);
        I = imread([out_dir, sprintf('train_image_%02d_%02d_
        bw.png',class,rem(k,100))]);
        [L, N] = bwlabel(I); % compute features and build feature vector using
        2 properties
        stats = regionprops(L,'all');
        fv(1,k) = stats.Eccentricity;
        fv(2,k) = stats.EulerNumber;
end
```

Langkah 7. Simpan fitur vektor dalam bentuk mat file.

```
save([out_dir,'train_fv.mat'], 'fv');
```

Langkah 8. Format fitur vektor sehingga klasifikasi KNN dapat mengerti.

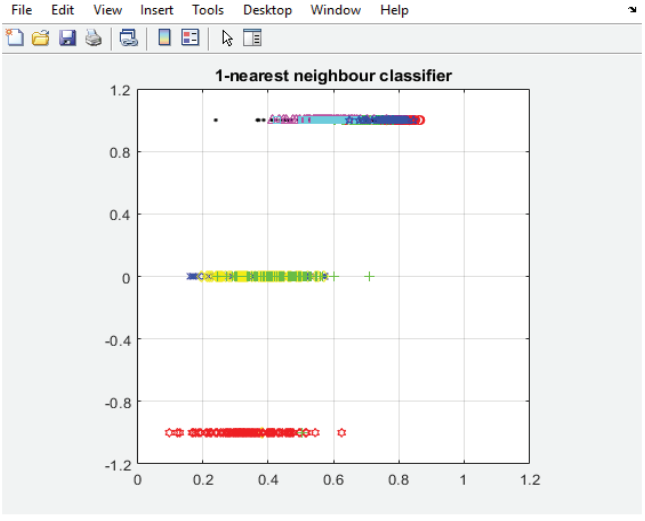
```
trn_data_binary.X = fv;
trn_data_binary.y = trn_data.y;
```

Langkah 9. Buat klasifikasi KNN dan atur data untuk klasifikasi.

```
model = knnrule(trn_data_binary, 1);
```

Langkah 10. Plot ruang fitur 2D dan periksa dengan cermat. Anda akan melihat plot yang identik dengan Gambar 6.10.

```
plot_feature_space
```



Gambar 6. 10 Ruang Fitur untuk data Pelatihan

Pertanyaan 3 : Fitur apa yang diwakili pada setiap sumbu plot?

Pertanyaan 4 : Setelah memeriksa plot (dan membandingkannya dengan yang ditunjukkan pada Gambar 6.11), apa yang dapat Anda simpulkan sejauh ini?

Langkah 11. Ekstrak fitur dari gambar uji preprocessed menggunakan regionprops dan memilih dua properti yang sama yang digunakan untuk set pelatihan, yaitu, eksentrisitas dan nomor Euler.

Langkah 12. Simpan hasilnya ke dalam array 2 × 1000 yang sesuai dengan 1000 vektor fitur (masing-masing berukuran 2 × 1).

```
fv_test = zeros(2,Ntest);
for k = 1:Ntest
        class = tst_data.y(k);
        I = imread([out_dir, sprintf('test_image_%02d_%02d_
        bw.png',class,rem(k,100))]);
        [L, N] = bwlabel(I);
        stats = regionprops(L,'all');
        fv_test(1,k) = stats.Eccentricity;
        fv_test(2,k) = stats.EulerNumber;
end
```

Langkah 13. Simpan fitur vektor menjadi mat file.

```
save([out_dir,'test_fv.mat'], 'fv_test');
```

Langkah 14. Buat format vektor fitur dengan cara yang akan dimengerti oleh klasifikasi KNN.

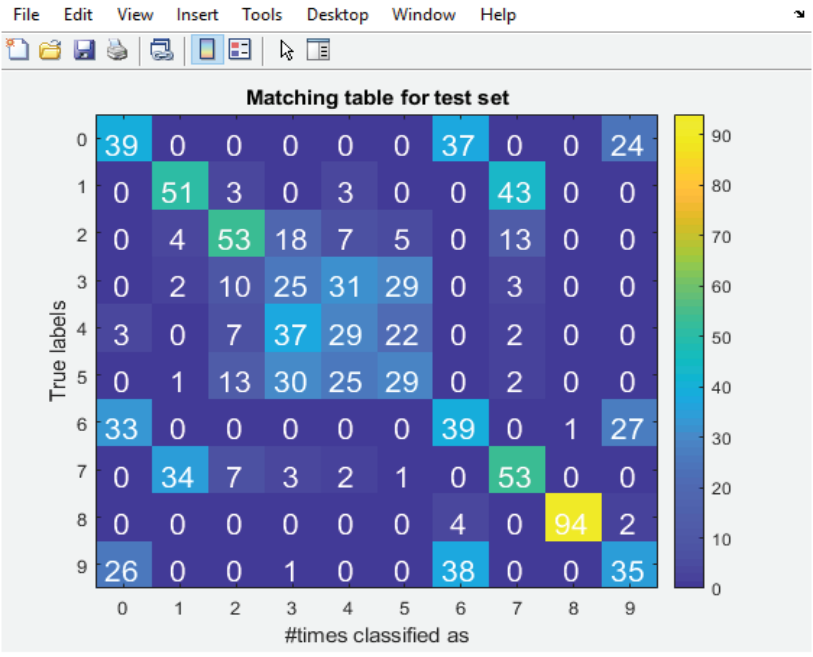
```
tst_data_binary.X = fv_test;
tst_data_binary.y = tst_data.y;
```

Langkah 15. Jalankan klasifikasi KNN untuk menetapkan label pada gambar uji.

```
labels = knnclass(tst_data_binary.X,model);
display_kNN_results;
```

Hasil menjalankan klasifikasi kNN pada dataset uji akan menunjukkan yang berikut:

```
classification_result on test set data: 553 out of 1000 missclassified
class “0” missclassified   61 times
class “1” missclassified   49 times
class “2” missclassified   47 times
class “3” missclassified   75 times
class “4” missclassified   71 times
class “5” missclassified   71 times
class “6” missclassified   61 times
class “7” missclassified   47 times
class “8” missclassified    6 times
class “9” missclassified   65 times
```



Gambar 6. 11 Matriks kebingungan dengan hasil klasifikasi KNN untuk fitur yang dipilih

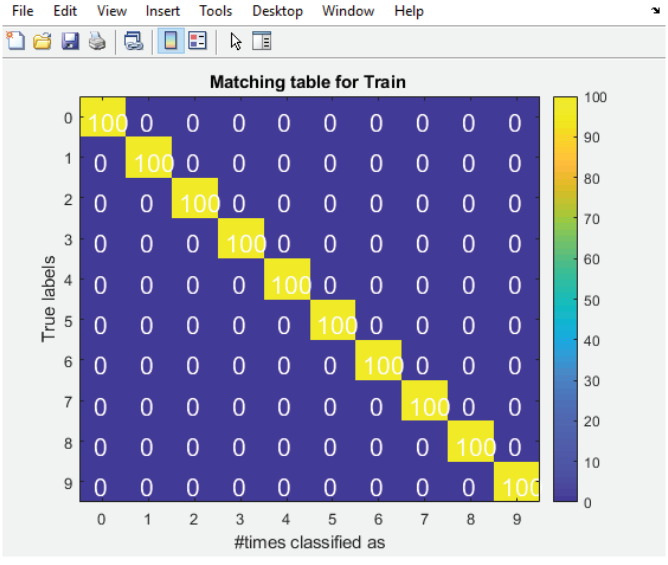
Pertanyaan 5 : Mengapa hasilnya sangat buruk (tingkat keberhasilan kurang dari 42%) secara keseluruhan?

Pertanyaan 6 : Mengapa kinerja untuk satu kelas (angka 8) jauh lebih baik daripada kelas lainnya?

6.5.1 Refleksi

Klasifikasi tidak memberikan kinerja sebaik yang diharapkan. Inilah saatnya untuk melihat kembali setiap langkah dari prosedur yang diperlihatkan dan pertimbangkan apa yang perlu diubah atau ditingkatkan. Pada hal ini, semua bukti menunjuk ke arah fitur yang lebih baik, yaitu, lebih deskriptif dan diskriminatif.

Jika Anda bertanya-tanya tentang kualitas klasifikasi, Gambar 6.12 menunjukkan matriks confusion yang dihasilkan jika kita menggunakan gambar yang sama persis dan menguji, tetapi vektor fitur yang berbeda. Dalam hal ini, nilai abu-abu aktual dari gambar asli digunakan sebagai "fitur" yaitu, setiap gambar diwakili menggunakan vektor fitur 169 × 1 dari ukuran asli 13 ×13, tanpa menimbulkan tahap preprocessing dan ekstraksi fitur apa pun yang digunakan. Hasilnya jauh lebih baik (tingkat keberhasilan keseluruhan 98,2%) daripada yang diperoleh dengan fitur-fitur pilihan ini (Gambar 6.11). Komentar lain yang menarik: kelas berkinerja terburuk (nomor 8) untuk desain yang dimodifikasi adalah yang berkinerja terbaik dalam metode asli yang digunakan.



Gambar 6. 12 Matriks Confusion dengan hasil klasifikasi KNN untuk kasus di mana nilai abu-abu gambar digunakan sebagai "fitur."

Pertanyaan 7 : Apakah metode alternatif berskala baik untuk ukuran gambar yang lebih besar (dan lebih realistis)? Menjelaskan.

Pertanyaan 8 : Perubahan apa yang akan Anda buat pada desain ini (dengan asumsi dataset dan klasifikasi yang sama) dan dalam urutan apa Anda akan bereksperimen dengannya?.

6.6. LATIHAN PERMASALAHAN

1 Amati Matriks confusion pada Gambar 6.13, gunakan MATLAB untuk menghitung precision dan recall per kategori.

2 Rancang dan terapkanlah (dalam MATLAB) machine vision solution untuk menghitung jumlah uang receh, nikel, uang receh, dan empat bagian dalam gambar dalam mengandung koin.

	ω_1	ω_2	ω_3	ω_4
ω_1	97	0	2	1
ω_2	0	94	5	1
ω_3	0	0	100	0
ω_4	3	0	5	92

Gambar 6. 13 Kasus Matriks Confusion

REFERENSI

[1]. Attaway, S. (2017). Matlab A Practical Introduction to Programming and Problem Solving (4th ed.).

[2]. Gilat, A. (2017). MATLAB An Introduction with Applications. Wiley (6th ed.). Wiley.

[3]. Gong, S., Liu, C., Ji, Y., Zhong, B., Li, Y., & Dong, H. (2019). Advanced Image and Video Processing Using MATLAB. Springer (Vol. 12). https://doi.org/10.1007/978-3-319-77223-3

[4]. Gonzalez, R. C., & Woods, R. E. (2008). Digital Image Processing. Prentice Hall.

[5]. Hanselman, D., & Littlefield, B. (2012). Mastering MATLAB. Pearson. Pearson.

PENGENALAN BERBASIS CITRA DUA DIMENSI MENGGUNAKAN MATLAB

Buku ini sebagai pengantar praktis untuk topik paling penting dalam pemrosesan citra, menggunakan media pemprograman MATLAB sebagai alat untuk menunjukkan teknik dan algoritma yang relevan. Diharapkan dapat memungkinkan pembaca untuk mengembangkan proyek-proyek praktis, yaitu, prototipe kerja, menggunakan pengetahuan yang diperoleh dari buku.

Dicetak dan didistribusikan oleh:

Distributor buku, Penerbit & Percetakan
THE BEST SOLUTION

ISBN 978-623-7313-30-4