

buku ajar

OpenCV Pengolahan Citra Digital Menggunakan Python

 Image Processing using Phyton and Open CV for international student

Cahya Rahmad

Rosa Andrie asmara

Muhammad Hasyim Ratsanjani

Farid …..

pOLITEKNIK NEGERI MALANG

JURUSAN TEKNOLOGI INFORMASI

January 1, 2020

# Kata Pengantar

Puji syukur selalu terlantun kehadirat Allah SWT. Serta sholawat serta salam semoga senantiasa tercurah kepada nabi Muhammad SAW.

Pengolahan citra digital merupakan bidang ilmu pengetahuan rumpun teknologi informasi yang cukup fenomenal beberapa tahun terakhir. Seiring berkembangnya teknologi informasi, berbagai bentuk kreatifitas tercipta untuk menunjang kehidupan sehari-hari manusia. Banyak sekali inovasi teknologi informasi tercipta menggunakan teknik-teknik pengolahan citra digital, yang kemudian bermanfaat untuk kehidupan yang semakin mudah dan menyenangkan. Hal ini menjadikan ilmu tentang pengolahan citra digital semakin penting untuk dimiliki.

Dalam buku ajar ini, teori-teori pengolahan citra digital disampaikan beserta contoh dalam lingkungan python. Dengan begitu, diharapkan para pembaca dapat secara cepat memahami serta menguasai teori dan implementasi teknik-teknik serta konsep pengolahan citra digital.

Dalam kesempatan ini, penulis menyampaikan penghargaan dan ucapan terima kasih yang sebesar-besarnya kepada :

1. Drs. Awan Setyawan, MM, sebagai Direktur Politeknik Negeri Malang,
2. Dr. Abdul Muqit,M.Pd. Sebagai Pimpinan Polinema Press,
3. Rekan-rekan sesama pengajar yang telah membantu memberikan berbagai informasi.

Semoga buku ajar ini bermanfaat untuk mahasiswa serta semua pihak yang membutuhkannya.

|  |  |
| --- | --- |
|  | Malang, 31 Desember 2019  Tim Penyusun |

# Daftar Isi

[Kata Pengantar i](#_Toc28935447)

[Daftar Isi ii](#_Toc28935448)

[Glossarium v](#_Toc28935449)

[Pendahuluan Citra Digital dan Pengolahannya 1](#_Toc28935450)

[1.1 Definisi Citra 2](#_Toc28935451)

[1.2 Pengolahan Citra Digital 3](#_Toc28935452)

[1.3 Macam-macam Operasi Pengolahan Citra 3](#_Toc28935453)

[1.4 Pengolahan Citra dengan Bidang Ilmu Komputer Yang Lain 5](#_Toc28935454)

[1.5 Dasar-dasar Pengolahan Citra Digital 5](#_Toc28935455)

[1.5.1 Macam-Macam Cara Pengambilan Citra Digital 7](#_Toc28935456)

[1.5.2 Format dan Model Citra Digital 11](#_Toc28935457)

[1.5.3 Contoh Aplikasi Pengolahan Citra Digital 17](#_Toc28935458)

[1.5.4 Tingkatan Pengolahan Citra 19](#_Toc28935459)

[1.6 Latihan Bab I 20](#_Toc28935460)

[Pengolahan Citra Dasar 21](#_Toc28935461)

[2.1 Membaca dan Menampilkan Citra 21](#_Toc28935462)

[2.2 Mendapatkan Informasi Atribut Citra 23](#_Toc28935463)

[2.3 Mendapatkan Nilai Piksel 24](#_Toc28935464)

[2.4 Merubah Nilai Piksel 26](#_Toc28935465)

[2.5 Grayscale 27](#_Toc28935466)

[2.6 Latihan Bab II 29](#_Toc28935467)

[Operasi Titik 30](#_Toc28935468)

[3.1 Pengertian Operasi titik 30](#_Toc28935469)

[3.2 Histogram 30](#_Toc28935470)

[3.2.1 Pembuatan Histogram 31](#_Toc28935471)

[3.2.2 Histogram Tingkat Keabuan 33](#_Toc28935472)

[3.2.3 Kegunaan Histogram pada pengolahan citra 33](#_Toc28935473)

[3.3 Fungsi transformasi skala keabuan 34](#_Toc28935474)

[3.4 Modifikasi kecemerlangan citra 35](#_Toc28935475)

[3.5 Peningkatan kontras 38](#_Toc28935476)

[3.6 Negasi atau Inversi Citra 41](#_Toc28935477)

[3.7 Operasi citra True Color Ke Keabuan 43](#_Toc28935478)

[3.8 Pengambangan (thresholding) 46](#_Toc28935479)

[3.9 Konversi Ke Citra Biner 48](#_Toc28935480)

[3.10 Latihan Bab III 50](#_Toc28935481)

[Operasi Global 51](#_Toc28935482)

[4.1. Pengertian Operasi Global 51](#_Toc28935483)

[4.2. Ekualisasi Histogram 51](#_Toc28935484)

[4.3 Latihan Bab IV 54](#_Toc28935485)

[Operasi Berbasis Bingkai 55](#_Toc28935486)

[5.1. Pengertian Operasi Berbasis Bingkai 55](#_Toc28935487)

[5.2. Operasi Aritmatik 55](#_Toc28935488)

[5.3. Penggabungan citra (Image blending) 56](#_Toc28935489)

[5.4. Deteksi gerakan 59](#_Toc28935490)

[5.5. Operasi Logika 60](#_Toc28935491)

[5.6 Latihan Bab V 63](#_Toc28935492)

[Operasi Geometri 64](#_Toc28935493)

[6.1. Pengertian Operasi Geometri 64](#_Toc28935494)

[6.2. Transformasi Spasial 65](#_Toc28935495)

[6.3. Transformasi maju 65](#_Toc28935496)

[6.4. Transformasi balik 66](#_Toc28935497)

[6.5. Interpolasi nilai keabuan 67](#_Toc28935498)

[6.6. Interpolasi tetangga terdekat 68](#_Toc28935499)

[6.7. Interpolasi bilinier 68](#_Toc28935500)

[6.8. Perubahan Ukuran Citra 69](#_Toc28935501)

[6.9. Aplikasi Operasi Geometri 69](#_Toc28935502)

[6.9.1. Operasi pencerminan 69](#_Toc28935503)

[6.9.2. Operasi *cropping* 73](#_Toc28935504)

[6.9.3. Operasi penskalaan (*scaling*) 75](#_Toc28935505)

[6.9.4. Operasi rotasi bebas 76](#_Toc28935506)

[6.9.5. Operasi *skew* 77](#_Toc28935507)

[6.10 Latihan Bab VI 78](#_Toc28935508)

[Aplikasi Operasi Bertetangga 79](#_Toc28935509)

[7.1. Penghalusan Citra 79](#_Toc28935510)

[7.2. Pendeteksian Tepi 79](#_Toc28935511)

[7.2.1. Pendeteksian tepi menggunakan operator gradien 79](#_Toc28935512)

[7.2.2. Penajaman Citra (*Sharping*) 80](#_Toc28935513)

[7.3 Latihan Bab VII 80](#_Toc28935514)

[Image Filtering 81](#_Toc28935515)

[8.1 Blur Kernel 83](#_Toc28935516)

[8.2 Bottom Sobel Kernel 84](#_Toc28935517)

[8.3 Right Sobel Kernel 84](#_Toc28935518)

[8.4 Left Sobel 85](#_Toc28935519)

[8.5 Emboss Kernel 86](#_Toc28935520)

[8.6 Outline Kernel 87](#_Toc28935521)

[8.7 Prewitt Kernel 88](#_Toc28935522)

[8.8 Sobel Kernel 89](#_Toc28935523)

[8.9 Latihan Bab VIII 90](#_Toc28935524)

# Glossarium

|  |  |  |
| --- | --- | --- |
| *algoritma* | : | Prosedur langkah-demi-langkah untuk penghitungan |
| *biner* | : | Sistem bilangan basis dua. Sebuah sistem penulisan angka dengan menggunakan dua simbol yaitu 0 dan 1 |
| *biometric* | : | Karakteristik biologi yang terukur. Ciri-ciri unik yang dimiliki secara fisik/tubuh setiap orang. |
| *bluring* | : | Pengaburan citra. Mengurangi kesan kontras dalam nilai piksel yang berdekatan |
| *brigtness* | : | Kecerahan citra. Jika intensitas pixel dikurangi dengan nilai tertentu maka citra akan menjadi lebih gelap, dan sebaliknya jika intensitas pixelnya ditambah dengan nilai tertentu maka akan lebih terang |
| *citra* | : | Gambar dua dimensi, seperti foto ataupun lukisan |
| *contrast* | : | Perbedaan gradasi,kecerahan, atau nada (warna) antara bidang gelap (shadow) dengan bidang terang, atau warna putih yang mencolok sekali pada objek |
| *cropping* | : | Operas pemotongan satu bagian dari citra sehinggga diperoleh citra yang berukuran lebih kecil |
| *ekstraksi ciri* | : | Proses pengambilan informasi/ciri yang dimiliki oleh suatu objek dalam citra |
| *embossing* | : | Emboss memberikan ilusi kedalaman dengan menekankan perbedaan piksel dalam arah yang diberikan |
| *enkripsi* | : | Proses mengamankan suatu informasi dengan membuat informasi tersebut tidak dapat dibaca tanpa bantuan pengetahuan khusus |
| *GIS* | : | Sistem Informasi Geografis. Adalah sistem informasi khusus yang mengelola data yang memiliki informasi spasial (bereferensi keruangan) |
| *grayscale* | : | Warna-warna piksel yang berada dalam rentang gradasi warna hitam dan putih |
| *histogram* | : | Distribusi data dari nilai derajat keabuan |
| *image clustering* | : | Mengelompokkan gambar menjadi kelompok-kelompok dimana gambar dalam satu kelompok akan memiliki karakteristik yang sama |
| *image kernel* | : | Matriks ukuran kecil berisi angka-angka tertentu yang digunakan untuk menciptakan suatu efek dalam pengolahan citra. |
| *intensitas* | : | Nilai kedalaman warna untuk suatu piksel |
| *interpolasi* | : | Cara menentukan nilai yang berada di antara dua nilai diketahui berasarkan suatu fungsi persamaan |
| *konvolusi* | : | Cara untuk mengkombinasikan dua buah deret angka yang menghasilkan deret angka yang ketiga |
| *koordinat spasial* | : | Sistem penunjukan suatu titik lokasi pada bidang dua dimensi diwakili x dan y |
| *kuantisasi* | : | Besarnya nilai tingkat kecerahan yang dinyatakan dalam nilai tingkat keabuan (gray scale) kuantisasi ini pada citra menyatakan jumlah warna pada masing-masing pixel |
| *library* | : | Koleksi dari rutin-rutin program yang digunakan untuk membangun dan mengembangkan perangkat lunak. |
| *matriks* | : | Sekumpulan bilangan yang disusun secara baris dan kolom |
| *piksel* | : | Unsur gambar yang merupakan representasi titik terkecil dalam sebuah gambar grafis yang dihitung per inci. |
| *resolusi* | : | Jumlah titik atau jumlah pixel dalam sebuah gambar |
| *sampling* | : | Besarnya kotak-kotak yang disusun dalam baris dan kolom atau dengan kata lain sampling pada citra menyatakan besar kecilnya ukuran pixel (titik) pada citra |
| *segmentasi citra* | : | Proses pemisahan objek yang satu dengan objek yang yang lain dalam suatu gambar (citra) menjadi objek- objek berdasarkan karakteristik tertentu |
| *sharpening* | : | Penajaman citra. Meningkatkan kesan kontras dalam nilai piksel yang berdekatan |
| *threshold* | : | Ambang batas |
| *transformasi* | : | Perubahan rupa |

# Pendahuluan Citra Digital dan Pengolahannya

Bab I

Capaian Mahasiswa:

* Mampu memahami definisi citra digital serta pengolahan citra digital.
* Mampu memahami dasar-dasar operasi pengolahan citra digital.
* Mengetahui bagaimana penerapan pengolahan citra digital dalam kehidupan sehari-hari.

Dalam menjalani kehidupan sehari-hari, manusia tidak bisa dipisahkan dari sebuah citra, foto ataupun gambar. Di dalamnya sering merekam suatu keindahan, peristiwa, atau bahkan kenangan berharga yang menjadikan citra tersebut sebagai media informasi yang cukup penting. Secara alami manusia lebih tertarik untuk melihat gambar dan menafsirkan sendiri suatu kejadian yang ada di dalamnya dari pada membaca deskripsi tertulis tentang suatu kejadian itu sendiri. Hal ini menjadikan suatu citra sebagai sebuah alat komunikasi yang sangat efektif dan menarik bagi semua orang.

Sementara itu, pesatnya perkembangan teknologi informasi saat ini, menjadikan setiap orang mampu mengakses media fotografi sehingga cukup mudah untuk menciptakan suatu citra digital. Ditambah lagi dengan pesatnya perkembangan media komunikasi, menjadikan suatu citra dengan sangat cepat menyebar ke seluruh pelosok negeri. Sehingga data citra berada dimana-mana dengan jumlah yang sangat banyak. Hal ini menjadikan ilmu pengolahan citra semakin populer, terutama jika ingin mendapat manfaat lebih lanjut dari perkembangan teknologi pencitraan.

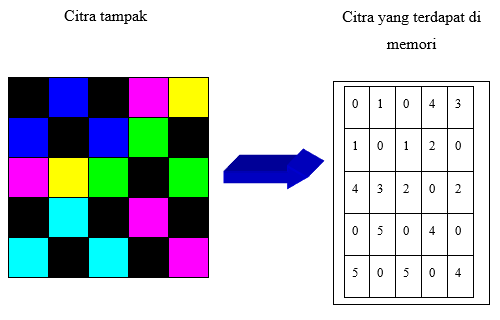
Pengolahan citra adalah bidang yang sifatnya multidisiplin ilmu, terdiri dari berbagai aspek, diantaranya: fisika, elektronika, matematika seni, fotografi dan teknologi komputer. Pengolahan citra berhubungan erat dengan disiplin ilmu yang lain. Jika sebuah disiplin ilmu dinyatakan dalam bentuk proses input menjadi output, maka pengolahan citra memiliki input berupa citra dan output berupa citra juga.

## Definisi Citra

Menurut Kamus Webster, “citra merupakan suatu representasi, kemiripan, atau imitasi dari suatu obyek atau benda”. Citra dapat dikelompokkkan menjadi citra tampak (misalnya,: Foto diri atau lukisan dinding) dan citra tidak tampak (misalnya; data gambar dalam file (citra digital) dan citra yang dipresentasikan berupa fungsi matematis.

Diantara jenis citra tersebut hanya citra digital yang dapat diolah menggunakan komputer. Citra yang lain jika ingin diolah menggunakan komputer harus diubah terlebih dahulu menjadi citra digital.

Citra merupakan kumpulan dari titik-titik dengan gradasi warna tertentu yang membentuk pola tertentu, di komputer titik-titik tersebut berbentuk kotak bujur sangkar dan disebut pixel *(picture element*) dan disebut juga dot. Titik-titik tersebut terletak pada bidang dua dimensi yang dapat dinyatakan berupa *f(x,y)* dengan *x* dan *y* terletak pada sistem koordinat spasial sedangkan nilai *f* pada koordinat *x* dan *y* tersebut sebanding dengan tingkat kecerahan. Ilustrasi sebuah citra terdapat pada gambar 1.1.



Gambar 1.1 Ilustrasi citra dan data yang disimpan di memori

Pada gambar 1.1 warna hitam diilustrasikan dengan angka 0, biru angka 1 dan kuning angka 3.

## Pengolahan Citra Digital

Pada awalnya pengolahan citra dilakukan untuk memperbaiki kualitas citra, namun seiring dengan berkembangnya ilmu komputer yang memungkinkan manusia dapat mengambil informasi dari suatu citra maka *image processing* tidak dapat dilepaskan dengan bidang *computer vision*.

Pengolahan citra mempunyai dua tujuan utama, yaitu:

1. Memperbaiki kualitas citra agar menghasilkan citra baru yang menampilkan informasi secara jelas, dimana manusia dapat melihat informasi yang diinginkan dengan menginterprestasikan citra yang ada.
2. Mengambil informasi ciri yang penting (Mengekstraksi ciri) pada suatu citra dan hasilnya dapat berupa informasi ciri citra berupa sederetan angka (Secara Numerik) dan data numerik tersebut memiliki besaran-besaran yang dapat dibedakan sehingga dapat diterjemahkan oleh sebuah komputer untuk melakukan sebuah keputusan yang Sangat penting.

## Macam-macam Operasi Pengolahan Citra

Pada dasarnya pengolahan citra dilakukan dengan cara mengubah-ubah nilai setiap titik (pixel) pada citra tersebut, macam-macam cara mengubah nilai pixel dapat dikelompokkan sebagi berikut:

1. Operasi titik, Pada operasi ini setiap titik (pixel) di ubah-ubah tanpa melibatkan titik yang lain pada citra tersebut.
2. Operasi global, Pada operasi ini terlebih dahulu diambil karakteristik global dari suatu citra kemudian dengan menggunakan karakter global sebuah citra maka proses pengubahan titik yang lain dilakukan.
3. Operasi berbasis bingkai, Pada operasi ini proses pengubahan nilai citra memerlukan citra yang lain, dengan kata lain citra 1 dioperasikan dengan citra 2 menghasilkan citra 3.
4. Operasi geometri, Pada operasi ini orientasi citra ( bentuk, ukuran, kemirigan ) diubah seara geometris.
5. Opersi banyak titik bertetangga, Pada operasi ini sebuah pixel/titik diubah nilainya dengan melibatkan nilai tetangganya.
6. Operasi morfologi, Pada operasi ini sebuah pixel/titik diubah dengan melihat bagian dalam yang dimiliki sebuah citra yang mendapat perhatian khusus.

Pengolahan citra dari tahun ke tahun berikutnya telah mengalami berbagai perkembangan dan dalam banyak penerapan pengolahan citra ini digunaka sebagai pengganti mata yang diletakkan pada komputer. Untuk mengambil sebuah citra komputer memerlukan sebuah peralatan yang disebut image capture.

Peralatan *image capture* diantaranya adalah kamera dan scanner. Data citra yang telah didapat dari camera dan scanner selanjutnya diproses oleh komputer dengan menggunakan langkah-langkah tertentu (algoritma) untuk menghasilkan sebuah citra yang lain atau sebuah keputusan.

Dengan adanya pemrosesan terhadap suatu citra dalam perkembangan selanjutnya telah memunculkan beberapa cabang dalam dunia komputasi seperti pengenalan pola (*pattern recognition*), pengenalan identifikasi manusia berdasarkan ciri-ciri biologis yang tampak pada badan manusia (*biometric*), mendapatkan kembali citra atau video dengan informasi tertentu (*content based image and video retrieval*) dan lain-lain.

Biometric merupakan Salah satu bidang pengolahan citra yang saat ini banyak dikembangkan orang yaitu bidang yang mempelajari ciri-ciri unik yang dimiliki secara fisik/tubuh setiap orang adalah misalnya goresan mata, bentuk wajah, warna wajah, ekspresi wajah, pola struktur DNA pada darah,sidik jari dan lain-lain.

Sebagai sebuah contoh adalah identifikasi sidik jari, setiap manusia memiliki ciri yang spesifik yang terdapat pada sidik jarinya, sidik jari ini cenderung berbeda antara satu orang dengan orang yang lain (kemungkinan sama adalah satu banding beberapa milyar) sehingga dapat digunakan untuk membedakan antara satu orang dengan yang lainnya.

Untuk melakukan identifikasi sidik jari diperlukan pengolahan citra untuk melakukan *capture* (penangkapan citra sidik jari), sampai pada ekstraksi ciri yaitu mengekstrak besaran-besaran numerik yang dapat dijadikan suatu ciri sidik jari seperti core (pusat sidik jari) dan minusi (percabangan yang ada pada sidik jari), yang pada akhirnya dilakukan proses pembelajaran agar komputer dapat secara tepat mengidentifikasikan sidik jari.

## Pengolahan Citra dengan Bidang Ilmu Komputer Yang Lain

Pengolahan citra memiliki hubungan dengan bidang komputer yang lain, hubungan ini berdasarkan jenis inputnya dan jenis outputnya hubungan ini dapat dijelaskan dengan tabel 1.1 berikut ini.

Tabel 1.1 Hubungan Pengolahan citra dengan Bidang komputer lain

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Output | |
| Image | Deskripsi |
| Input | Image | Image Processing | Pattern Recognition, Computer Vision |
| Deskripsi | Computer Graphics | Data Processing lainnya |

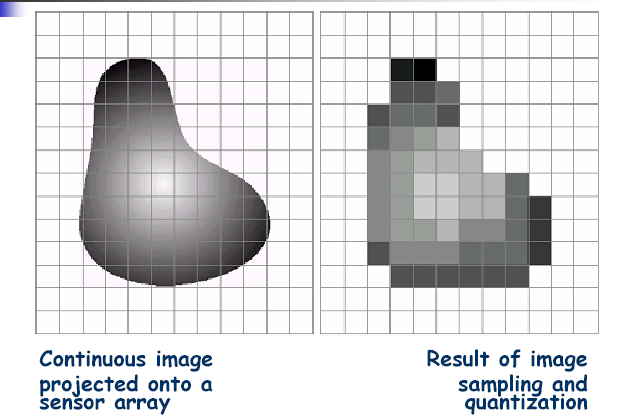
Pada tabel 1.1. pengolahan citra (*image processing*) merupakan suatu bidang pengetahuan dengan input berupa citra dan output juga berupa citra sedangkan disiplin ilmu yang lain adalah *Computer Graphics*.

## Dasar-dasar Pengolahan Citra Digital

Hal penting pada Pengolahan citra digital antara lain:

1. Cara-cara pengambilan citra
2. Format dan Model citra digital
3. Sampling dan kuantisasi
4. Threshold
5. Histogram
6. Proses filtering, perbaikan citra sampai pada pengolahan citra digital yang lebih lanjut seperti segmentasi, image clustering dan ekstraksi ciri.

Bebeapa perbedaan mendasar antara citra digital dan citra analog, terlihat gambar 1.2. yaitu:



Gambar 1.2. Contoh Perbedaan Citra Analog dan Citra Digital

Citra digital adalah representatif dari citra yang diambil oleh mesin dengan bentuk pendekatan berdasarkan sampling dan kuantisasi.

Sampling merupakan besarnya kotak-kotak yang disusun dalam baris dan kolom atau dengan kata lain sampling pada citra menyatakan besar kecilnya ukuran pixel (titik) pada citra.

kuantisasi adalah besarnya nilai tingkat kecerahan yang dinyatakan dalam nilai tingkat keabuan (gray scale) kuantisasi ini pada citra menyatakan jumlah warna pada masing-masing pixel,.

Gambar 1.3 adalah diagram pemrosesan citra digital, pada blok pengambilan bisa dilakukan dengan camera, webcam atau scanner, sedangkan pada Pengolahan citra gambar yang didapat diproses dengan teknik-teknik tertentu dan pada hasil akan diperoleh citra yang lain yang memiliki kualitas lain.

Pengambilan citra

Pengolahan Citra

Hasil Pengolahan Citra

Gambar 1.3. Diagram pengolahan citra

### Macam-Macam Cara Pengambilan Citra Digital

Beberapa Cara pengambilan digital dilakukan antara lain dengan menggunakan kamera digital (bisa juga web-cam) atau menggunakan scanner (bisa scanner umum, finger-print scanner atau bar-code scanner. Hasil dari citra yang diambil menggunakan kamera atau scanner berupa citra raster (atau citra dengan model matrik).

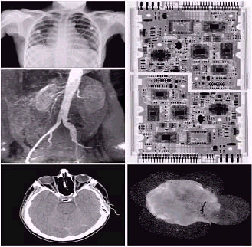


Gambar 1.4. Kamera

Cara yang lain untuk memperoleh citra yaitu dengan menggunakan cara grafika komputer yang memiliki hasil berupa citra raster atau bisa berupa citra vector caranya adalah dengan membentuk obyek citra komputer sesuai dengan cara mengambil citra hal ini banyak digunakan untuk keperluan entertainment seperti pembuatan animasi atau untuk keperluan GIS (*Geographics Information System*).

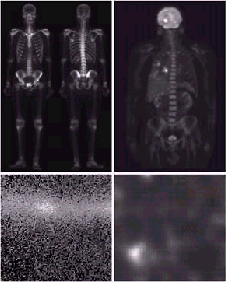
Cara pengambilan gambar akan membedakan proses citra yang akan digunakan di dalamnya. Misalkan scanner dan camera akan menghasilkan citra dalam format gambar tunggal, camera video dan webcam akan menghasilkan citra dalam format video. Demikian pula dengan resolusi dan format warna yang juga akan akan berbeda. Berikut ini beberapa contoh citra selain gambar atau photo yang sering dilihat baik dengan scanner atau camera, yang dibedakan berdasarkan cara pengambilan sinyalnya.

1. *X-Ray Imaging*, citra ini banyak ditemui untuk keperluan medis, rangkaian elektronik dan astronomi.



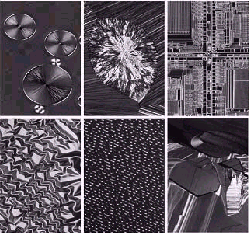
Gambar 1.5. X-Ray Imaging

1. *Gamma Ray Imaging*, citra ini biasanya digunakan untuk citra-citra nuklir dan proses nuklir, juga digunakan untuk keperluan astronomi.



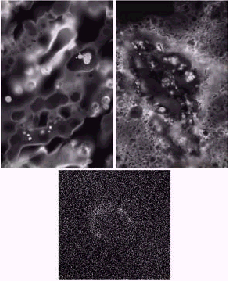
Gambar 1.6. Gamma Ray Imaging

1. *Image Infrared*, citra ini banyak ditemui untuk keperluan farmasi, mikrobiotik, dan mikro material.



Gambar 1.7. Image Infrared

1. *Image Ultraviolet*, citra ini banyak ditemui untuk keperluan citra biologi, microscopy, lithograpy, laser dan astronomi.



Gambar 1.8. Image Ultraviolet

### Format dan Model Citra Digital

Citra digital memiliki beberapa karakteristik,diantaranya ukuran citra, resolusi, dan format lainnya. Ukuran citra dapat dinyatakan dalam satuan panjang, yang mana harus ada hubungan antara ukuran titik penyusun citra dengan satuan panjang. Satuan yang biasanya digunakan adalah dpi (dot per inch).

Citra digital adalah citra yang diambil berdasarkan sampling dan kuantisasi tertentu sehingga citra digital ini terbentuk dari piksel-piksel yang besarnya tergantung pada besar kecilnya sampling dan nilainya (besarnya derajat keabuan) tergantung pada kuantisasi.

Format citra digital ada berbagai macam., karena sebenarnya citra merepresentasikan informasi tertentu, sedangkan informasi tersebut dapat dinyatakan secara bervariasi, maka citra yang mewakilinya dapat muncul dalam berbagai format.

Citra yang mempresentasikan informasi yang bersifat biner untuk membedakan 2 keadaan tentu tidak sama dengan citra berinformasi yang lebih kompleks sehingga memerlukan lebih banyak keadaan yang mewakilinya. Pada citra digital semua informasi tadi disimpan dalam bentuk angka, sedangkan penampilan angka tersebut biasanya dikaitkan dengan warna.

Citra digital tersusun atas titik-titik yang biasanya berbentuk persegi panjang atau bujursangkar (pada beberapa sistem pencitraan, piksel-piksel penyusun citra ada pula yang berbentuk segi enam) yang secara beraturan membentuk baris-baris dan kolom-kolom. Setiap titik memiliki koordinat sesuai dengan posisinya dalam citra. Koordinat ini biasanya dinyatakan dalam bilangan bulat positif yang dimulai dari 0 atau 1 tergantung pada system yang digunakan.

#### 1.5.2.1 Model Citra Digital Berupa Matrik

Berdasarkan pengertian tersebut model citra digital dinyatakan dalam bentuk matrik yang nilainya berupa nilai derajat keabuan seperti terlihat pada citra 1.8. berikut. Model ini menyatakan model dari citra gray-scale yaitu citra yang terdiri dari derajat keabuan tertentu. Dengan kata lain dapat dinyatakan bahwa citra digital adalah citra yang didefinisikan sebagai fungsi *f(x,y)* dimana *x* menyatakan nomor baris, *y* menyatakan nilai kolom, dan f menyatakan nilai derajat keabuan dari citra. Sehingga *(x,y)* adalah posisi dari piksel dan *f* adalah nilai derajat keabuan pada titik *(x,y)* seperti terlihat pada gambar 1.9.



Gambar 1.9 Citra Sebagai Matrik



Gambar 1.10 Model Citra Pada Koordinat f(x,y)

Representasi model data di atas adalah representasi dari citra gray-scale (menggunakan derajat keabuan). Untuk citra berwarna, maka digunakan model RGB (Red-Green-Blue) dimana satu citra berwarna dinyatakan sebagai 3 buah matrik gray-scale yang berupa matrik untuk Red (R-layer), matrik untuk Green (G-layer) dan matrik untuk Blue (B-layer).

Citra digital harus mempunyai format tertentu yang sesuai sehingga dapat mempresentasikan obyek pencitraan dalam bentuk kombinasi data biner. Format citra digital berhubungan erat dengan warna, dimana yang banyak dipakai adalah citra biner, warna , warna berindeks dan skala keabuan.

#### 1.5.2.2 Resolusi

Resolusi adalah jumlah titik atau jumlah pixel dalam sebuah gambar, pada gambar 1.1. citra memiliki jumlah baris 5 dan jumlah kolom 5 sehingga berjumlah 25 pixel, citra ini disebut memiliki resolusi 25 pixel atau berupa pasangan angka 5x5 pixel.

#### 1.5.2.3 Kedalaman

Kedaaman warna*, Bit Depth* , *pixel depth* atau *color depth* dinyatakan dalam bit. contoh untuk jumlah warna 16 disebut memiliki kedalaman 4 bit atau 24 = 16, untuk gambar dengan 256 warna disebut memiliki kedalaman 8 bit atau 28= 256.

#### 1.5.2.4 Format File Citra

Format file citra adalah bagaimana citra itu direpresentasikan dalam sebuah file kondisinya bisa dalam keadaan terkompres atau pada posisi terenkripsi. Antara satu format dengan format yang lain memiliki kelebihan dan kelemahan. Setiap format file dapat di bedakan dari ekstensi filenya yaitu diakhiri titik dengan tiga huruf tambahan. Macam-macamnya :

* Format file Bitmap/bmp adalah standar file bitmap / raster biasanya mempunyai ukuran file yang relatif besar tidak terkompres dan tidak mendukung gambar transparansi maupun animasi.
* Format file GIF (*Graphics Interchange Format*) menggunakan maksimal 8 bit warna (28 = 256 warna) gambar ini menggunakan kompresi dengan LZW *compression* yang merupakan kompresi loseless yang berarti tidak ada data yang dibuang. Format ini juga mendukung gambar transparansi dan animasi.
* Format file JPEG/JPG (*Joint Photographic Experts Group*) menggunakan 24 bit warna (224 = 16 juta warna) dan melakukan kompresi dengan cara membuang data yang dianggap tidak penting pada gambar (bersifat *lossy compression*). Semakin kecil file yang diinginkan semakin banyak data yang akan dibuang sehingga kualitasnya akan semakin menurun. Format ini tidak mendukung transparansi dan animasi.
* Format file PNG-8 terdiri 8 bit warna, mempunyai hasil kompresi yang lebih kecil dari format GIF.
* Format file PNG-24 menggunakan 24 bit warna, mempunyai ukuran yang lebih besar dan warna yang lebih banyak dari pada PNG-8.
* Format file PDF (Portable Document Format) digunakan untuk keperluan dokumen lintas sistem dan lintas program aplikasi dan mempunya ukuran file relatif kecil.

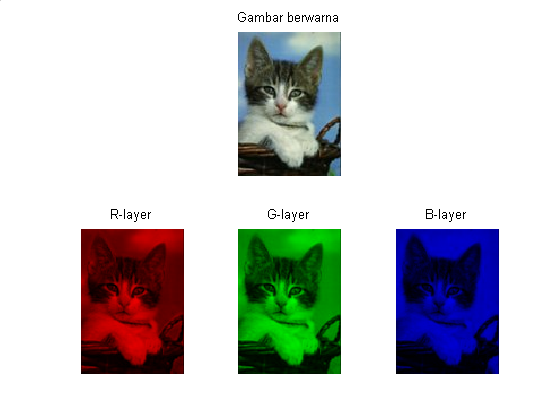
#### 1.5.2.5 Citra Biner

Pada citra ini, setiap titik bernilai 0 atau 1, yang masing-masing mempresentasikan warna tertentu. Setiap titik pada citra hanya membutuhkan 1 bit, sehingga setiap byte dapat menampung informasi 8 titik.

#### 1.5.2.6 Citra warna (True Color)

Pada citra ini setiap titik memiliki warna yang spesifik yang merupakan kombinasi dari tiga warna dasar yaitu merah, hijau dan biru, yang mana sering disebut sebagai citra RGB (red-green-blue). Untuk citra 32 bit setiap warna dasar mempunyai intensitas sendiri dengan intensitas maksimum 255 (8bit) yaitu RGB total 24 bit ditambah alfa 8 bit.

Jumlah kombinasi warna yang mungkin untuk format ini adalah lebih dari 16 juta warna, sehingga dinamakan citra true color, contoh citra berwarna terdapat pada gambar berikut.



Gambar 1.11. Tiga layer layer RGB pada citra berwarna

R-layer adalah matrik yang menyatakan derajat kecerahan untuk warna merah (misalkan untuk skala keabuan 0-255, maka nilai 0 menyatakan gelap (hitam) dan 255 menyatakan warna merah. G-layer adalah matrik yang menyatakan derajat kecerahan untuk warna hijau, dan B-layer adalah matrik yang menyatakan derajat kecerahan untuk warna biru.

Citra pada gambar menyatakan model RGB dari citra berwarna yang dipisah menjadi R-layer, G-layer dan B-layer. Model matrik pada citra digital memungkinkan dilakukannya operasional-operasional matrik, seperti aritmatika matrik (seperti penjumlahan, pengurangan dan perkalian) atau proses seperti invers dan tranpose. Hal inilah yang menjadi dasar dari pengolahan citra. Pada bab-bab berikutnya proses pengolahan matrik ini menjadi acuan dalam mengolah citra.

#### 1.5.2.7 Citra warna berindeks

Format citra warna berindeks disedikan untuk kasus dimana jumlah warna yang ada dalam suatu citra terkadangsangat terbatas (dibawah 16 juta kemungkinan warna yang ada), karena banyaknya warna dalam sebuah citra tidak mungkin melebihi banyaknya titik dalam citra itu sendiri. Informasi setiap titik merupakan indeks suatu tabel yang berisi informasi warna yang tersedia yang disebut palet warna.

Palet warna merupakan bagian dari citra warna berindeks, sehingga saat menyimpannya dalam file, informasi palet ini harus disertakan. Jumlah bit yang dibutuhkan oleh setiap titik pada citra bergantung pada jumlah warna yang ada pada palet warna.

Dengan menggunakan palet warna kita dapat memanipulasi warna dengan cepat tanpa mengubah informasi pada setiap titik dalam citra. Selain itu besarnya data yang dibutuhkan untuk menyimpan citra ini lebih kecil dibanding citra warna true color.

#### 1.5.2.8 Citra Skala Keabuan

Format ini disebut skala keabuan karena warna yang umum dipakai adalah hitam sebagai warna minimal dan putih sebagai warna maksimal, sehingga warna antaranya adalah abu-abu. Namun dalam prakteknya warna yang dipakai tidak terbatas pada abu-abu, tergantung intensitas warnanya, sehingga beberapa buku menyebut format citra ini sebagai citra intensitas.

### Contoh Aplikasi Pengolahan Citra Digital

Gambaran mengenai cara-cara pengolahan citra digital dan macam-macamnya antara lain:

#### 1.5.3.1 Segmentation

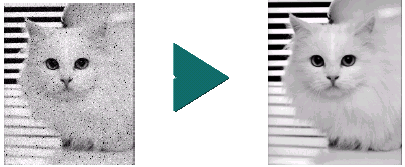
Adalah proses untuk membedakan atau memisah-kan obyek-obyek yang ada dalam suatu citra, seprti memisahkan obyek dengan latar belakangnya.



Gambar 1.12. Contoh Image Segmentation

#### 1.5.3.2 Image Enhancement

Berupa proses perbaikan citra dengan meningkatkan kualitas citra baik secara kontras maupun secara kecerahan.



Gambar 1.13. Contoh Image Enhancement

#### 1.5.3.3 Color Image Processing

Adalah suatu proses yang melibatkan citra berwarna, baik berupa image enhancement, image restoration atau yang lainnya.

#### 1.5.3.4 Image Restoration

Proses memperbaiki model citra, biasanya berhubungan dengan bentuk citra yang sesuai.



Gambar 1.14. Contoh Image Restoration

#### 1.5.3.5 Wavelet dan Multiresolution Processing,

Merupakan suatu proses yang menyatakan citra dalam beberapa resolusi.

#### 1.5.3.6 Object Recognition

Suatu proses yang dilakukan untuk mengenali obyek-obyek apa saja yang ada dalam sutau citra.

#### 1.5.3.7 Image Compression

Merupakan proses yang digunakan untuk mengubah ukuran data pada citra. Proses ini saat ini terus dikembangkan. JPEG-4 merupakan salah satu bentuk image compression yang baik dan saat ini banyak digunakan.

#### 1.5.3.7 Morphological Processing

Proses untuk memperoleh informasi yang menyatakan deskripsi dari suatu bentuk pada citra.

### Tingkatan Pengolahan Citra

Secara umum cara pengolahan citra digital dibagi menjadi 3 tingkat pengolahan yaitu:

1. **Low-Level Processing** (pengolahan tingkat rendah), pengolahan ini operasional-operasional dasar dalam pengolahan citra, seperti pengurangan noise (*noise reduction*), perbaikan citra (*image enhancement*) dan restorasi citra (*image restoration*).
2. **Mid-Level Processing** (pengolahan tingkat menengah) yang meliputi segmentasi pada citra, deskripsi obyek dan klasifikasi obyek secara terpisah.
3. **High-Level Processing** (pengolahan tingkat tinggi), yang meliputi analisa citra.

## Latihan Bab I

1. Deskripsikan tentang bagaimana manusia menggunakan citra dalam kehidupan sehari-hari?
2. Jelaskan ciri(*feature)* utama yang terkandung dalam suatucitra?
3. Sebutkan beberapa contoh aplikasi pengolahan citra?

# Pengolahan Citra Dasar

Bab II

Capaian Mahasiswa:

* Mampu menguasai teknik dasar berinteraksi dengan citra digital dalam lingkungan python.

Dalam bab ini akan dibahas teknik pengolahan citra dasar seperti; membaca citra kedalam program kemudian menampilkannya, mendapatkan informasi atribut citra, mendapatkan nilai piksel, merubah nilai piksel, serta menyimpan citra yang telah dirubah menjadi citra baru. Penjelasan praktikal disampaikan dalam lingkungan bahasa pemrograman python dengan memanfaatkan beberapa library yang biasa digunakan untuk melakukan operasi-operasi pengolahan citra, salah satunya adalah OpenCV(*Open Source Computer Vision*). Di samping library tersebut juga diperlukan beberapa library dasar yang biasa digunakan untuk kegiatan scientifik, seperti NumPy untuk menangani masalah array.

Untuk mendapatkan library OpenCV bagi komputer yang sudah terinstall python dapat digunakan perintah command line:

pip install opencv-python

Perintah tersebut akan secara otomatis mengunduh source OpenCV kemudian menginstallnya ke dalam sistem. Setelah proses instalasi komplit, maka OpenCV siap digunakan.

## 2.1 Membaca dan Menampilkan Citra

Untuk dapat melakukan operasi pengolahan citra yang harus dilakukan adalah membaca citra digital dan menampungnya dalam variabel yang tersimpan di memory. Pertama-tama buat sebuah file dengan nama “readDisplay.py”. Pastikan file yang dibuat diakhiri dengan ekstensi “py”, yang menandakan bahwa file tersebut berisi kode program python. Buka file tersebut dan ketikkan baris program di bawah.

|  |
| --- |
| #1. Import library OpenCV  import cv2  #2. Membaca citra  img = cv2.imread('mangga.jpg')  #3. Menampilkan citra  cv2.imshow('image', img)  cv2.waitKey(0)  cv2.destroyAllWindows() |

Penjelasan kode di atas adalah sebagai berikut:

#1. Untuk menggunakan library OpenCV, pertama-tama yang harus dilakukan adalah melakukan import librari OpenCV ke dalam program, dimana dilakukan dengan sintaks import cv2. Alasan mengapa tertulis import cv2 dibandingkan import opencv-python agar Lebih mudah bagi pengguna untuk memahami opencv-python daripada cv2 dan membuatnya lebih mudah untuk menemukan paket dengan mesin pencari. cv2 (antarmuka lama di versi OpenCV lama dinamai cv) adalah nama yang dipilih pengembang OpenCV saat mereka membuat generator pengikat. Ini disimpan sebagai nama impor agar konsisten dengan berbagai jenis tutorial di internet. Mengubah nama atau perilaku impor juga akan membingungkan pengguna berpengalaman yang terbiasa dengan impor cv2.

#2. Membaca citra menggunak fitur IO OpenCV, dilakukan dengan sintaks cv2.imread(‘mangga.jpg') kemudian menampungnya di dalam variable img. Dimana Citra digital yang dibaca memiliki nama file “mangga.jpg”, yang terletak di dalam folder yang sama dimana program dijalankan. Bila lokasi gambar terletak pada folder yang berbeda maka yang dimasukan kedalam parameter tersebut adalah path dari file gambar (contoh : cv2.imread(‘D:\PVCK\mangga.jpg’)) . Tetapi bila masih didalam satu folder project yang sama dengan sub folder yang berbeda maka tinggal memasukan nama sub folder slash nama file gambar (contoh : cv2.imread(‘subfolder\mangga.jpg’)). Selain itu cv2.imread() memiliki parameter kedua yang sifatnya opsional untuk di isi yaitu bernama flag, parameter ini berfungsi untuk memodifikasi warna pada saat gambar tersebut dibaca. Contoh :

cv2.imread(‘mangga.jpg', cv2.IMREAD\_GRAYSCALE)

perintah diatas akan membaca gambar manga dan mengubahnya menjadi gambar grayscale saat ditampilkan.

#3. menampilkan citra menggunakan fitur OpenCV.

cv2.imshow('image', img)

perintah tersebut berfungsi untuk menampilkan gambar menjadi sebuah window seperti pada Gambar 2.1, terdapat dua parameter yang digunakan untuk menggunakan fitur tersebut, yang pertama adalah nama window saat ditampilkan (‘image’) dan yang kedua adalah variabel yang telah menampung gambar (img). Dikarenakan kode program diatas berjalan lebih cepat dari 0,01 detik bila dijalankan maka hanya akan muncul window sekilas karena berjalan terlalu cepat sehingga mata manusia tidak bisa mengikutinya. Untuk mengatasi hal tersebut digunakanlah kedua kode program berikut

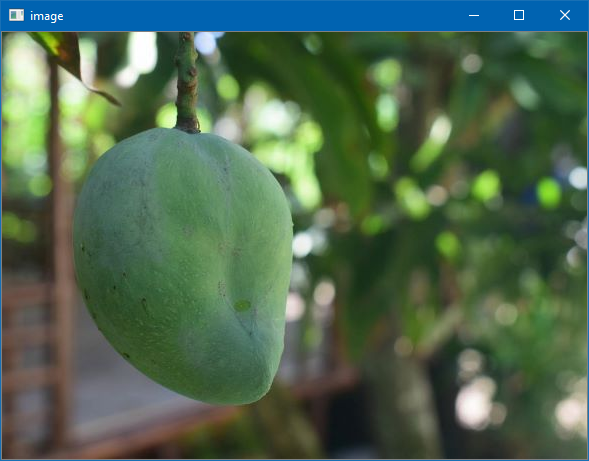
cv2.waitKey(0)

cv2.destroyAllWindows()

perintah cv2.waitKey(0) berfungsi menunggu pengguna untuk menekan tombol apa saja dan ketika ada tombol yang tertekan maka akan melanjutkan kode program selanjutnya (ini diperlukan untuk menghindari bentuk kernel Python crash). Dan selanjutnya cv2.destroyAllWindows() yang berfungsi untuk menutup semua window

Untuk menjalankan kode program di atas, dapat dilakukan dengan command line:

python readDisplay.py



Gambar 2.1 Membaca dan Menampilkan Gambar

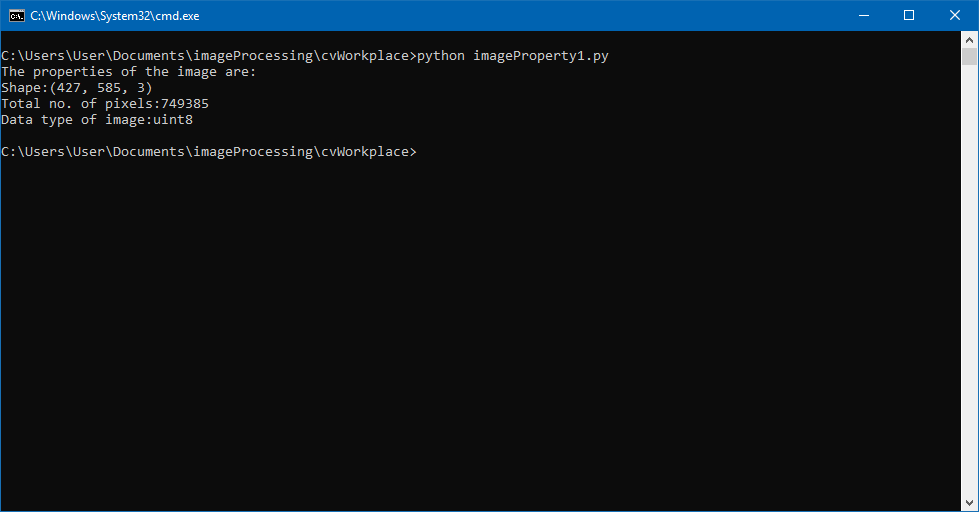
## 2.2 Mendapatkan Informasi Atribut Citra

Untuk mendapatkan informasi atribut suatu citra, dapat digunakan fungsi shape, size, serta dtype dalam OpenCV. Kode program di bawah adalah contoh menggunakannya.

|  |
| --- |
| #1. Import library OpenCV  import cv2  #2. Membaca citra  img = cv2.imread('mangga.jpg')  print('The properties of the image are:')  #3. Menampilkan atribut citra  print('Shape:' + str(img.shape))  print('Total no. of pixels:' + str(img.size))  print('Data type of image:' + str(img.dtype)) |

Simpan kode program di atas dengan nama ‘imageProperty.py’, kemudian jalankan dengan command :

python imageProperty.py



Gambar 2.2 Menampilkan Informasi Atribut Citra

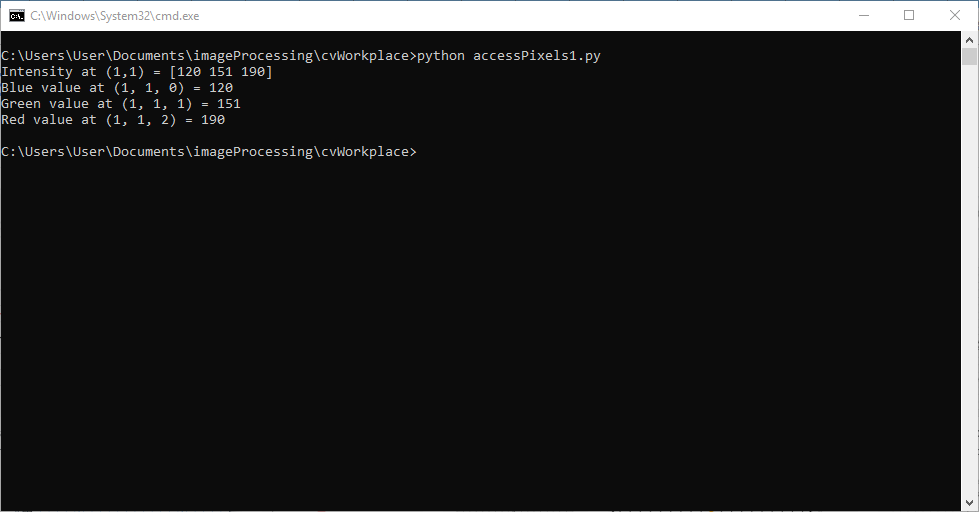
Variabel img yang telah diinialisasi sebelumnya sudah menjadi sebuah tipe data atau objek gambar dengan library opencv-python atau cv2 dengan begitu bisa menggunakan fitur yang disediakan oleh library untuk mengambil data tertentu pada suatu gambar tersebut. Fungsi shape digunakan untuk menampilkan resolusi citra dan jumlah channel. Pada contoh di atas, citra yang dibaca memiliki resolusi lebar 585 piksel dan tinggi 427 piksel serta memiliki 3 channel intensitas yakni red, green, dan blue.

## 2.3 Mendapatkan Nilai Piksel

Simpan kode program di bawah dengan nama ‘accessPixel.py’.

|  |
| --- |
| import cv2  import numpy as np  m = cv2.imread('taj.jpg')  height,width,dept = np.shape(m)  y = 1  x = 1  #1 Mengakses nilai intensitas pada koordinat y=1, x=1  print('Intensity at (1,1) = ' + str(m[y][x]))  #2 Mengakses nilai intensitas blue pada koordinat y=1, x=1  print('Blue value at (1, 1, 0) = ' + str(m[y][x][0]))  #3 Mengakses nilai intensitas green pada koordinat y=1, x=1  print('Green value at (1, 1, 1) = ' + str(m[y][x][1]))  #4 Mengakses nilai intensitas red pada koordinat y=1, x=1  print('Red value at (1, 1, 2) = ' + str(m[y][x][2])) |

Hasil ketika dijalankan adalah sebagai berikut:



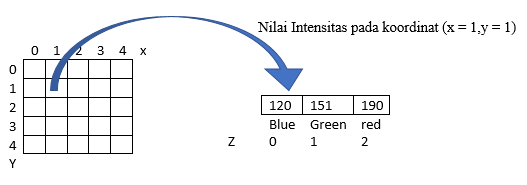
Gambar 2.3 Mengaksis Nilai Intensitas

Kode program di atas menampilkan nilai intensitas gambar yakni, blue = 120, green = 151, red = 190 yang dibaca pada koordinat (x = 1,y = 1) .

height,width,dept = np.shape(m)

np.shape(m) merupakan sebuah array pada object gambar pada library cv2 yang berisi Panjang, lebar dan kedalaman suatu gambar. Sehingga bila ingin memisahkannya dari gambar pada variabel m kedalam sebuah variabel secara terpisah dapat menggunakan cara tersebut dan harus berurutan seperti index ke - 0 untuk tinggi, index ke – 1 untuk lebar dan index ke – 2 untuk channel gambar (red / green / blue).

Citra digital didefinisikan sebagai suatu bidang dua dimensi yang masing titik-titik pikselnya memiliki nilai intensitas dalam tiga channel, yakni red, green, dan blue serta satu channel alpha. Sehingga dalam python, suatu citra yang dibaca kemudian disimpan dalam sebuah array 3 dimensi (y, x, z). Y mewakili posisi vertikal serta X mewakili posisi horizontal. Sedangkan z adalah lokasi intensitas blue, green, dan red.

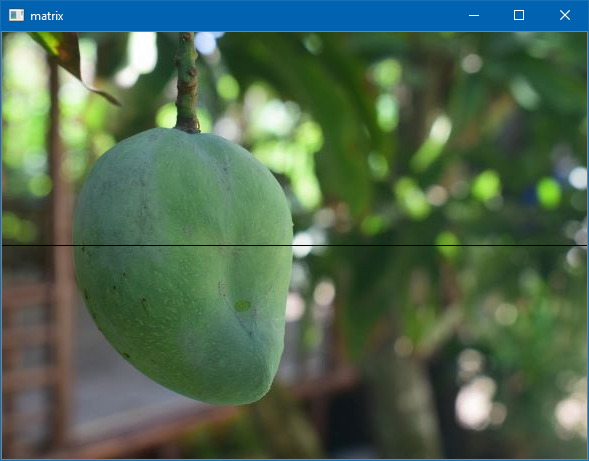
 Gambar 2.4 Mengakses Nilai Piksel di Suatu Koordinat

Maka, satu titik koordinat piksel akan berisikan tiga nilai intensitas yakni blue, green, dan red. Namun dalam python satu hal yang berbeda dari umumnya adalah posisi index urutan channel intensitas yang terdapat dalam suatu koordinat pixel. Bila pada umumnya urutan channel intensitas adalah red, green, kemudian blue, maka dalam python indek ke-0 menampung nilai intensitas green, indeks ke-1 menampung nilai intensitas red, dan indeks ke-2 menampung nilai intensitas red.

## 2.4 Merubah Nilai Piksel

Kode program di bawah merubah nilai intensitas piksel pada baris 213 menjadi red = 0, green = 0, serta blue = 0, sehingga akan menampilkan warna hitam. Selanjutnya untuk menyimpan data citra yang telah dirubah tersebut ke dalam citra yang baru, digunakan fungsi imwrite.

|  |
| --- |
| import cv2  import numpy as np  m = cv2.imread('mangga.jpg')  height, width, depth = np.shape(m)  #1 merubah nilai intensitas pada baris 213  for px in range(0, width) :  m[213] [px][0] = 0  m[213] [px][1] = 0  m[213] [px][2] = 0  cv2.imshow('matrix', m)  #2 menyimpan ke dalam file  cv2.imwrite('output2.png', m)  cv2.waitKey(0)  cv2.destroyAllWindows() |



Gambar 2.5 Mengubah Nilai Intensitas Menjadi Hitam

#1 merubah nilai intensitas pada baris 213

Setelah mendapatkan tinggi, lebar dan kedalaman gambar menggunakan np.shape(m) dan disimpan ke variabel heigh, width, dan depth selanjutnya dilakukan perulangan untuk merubah setiap warna pixel pada koordinat yang ditentukan

for px in range(0, width) :

m[213] [px][0] = 0

m[213] [px][1] = 0

m[213] [px][2] = 0

Jika px bernilai diantara 0 sampai dengan nilai variabel width, maka akan merubah kode warna gambar m pada pixel (koordinat) ketinggian pixel ke 213 dan lebar pixel ke px dengan channel gambar 0 sebagai blue, 1 sebagai green dan 2 sebagai red. Sehingga bila kode program tersebut dijalankan maka akan merubah semua baris pixel pada ketinggian 213 menjadi 0 (warna hitam).

#2 menyimpan ke dalam file

Setelah pixel gambar berubah seletah menjalankan perintah pertama (#1) masih tersimpan kedalam variabel (m) dan untuk menyimpannya bisa menggunakan perintah

cv2.imwrite('output2.png', m)

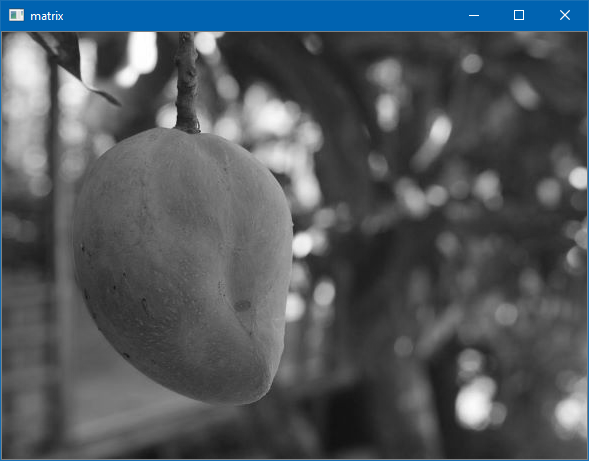
mirip dengan cv2.imwrite() parameter pertama adalah path gambar (‘output2.png’), dan parameter kedua adalah variabel yang bertipe data gambar.

## 2.5 Grayscale

Citra *grayscale* adalah citra yang hanya mengandung warna dalam rentang gradasi warna keabuan putih sempurna atau 0 hingga hitam sempurna atau 255. Untuk mendapatkan citra grayscale dapat dilakukan dengan cara menghitung rata-rata dari ketiga channel RGB.

Setelah nilai rata-rata didapatkan, maka nilai tersebut digunakan untuk mengisi ketiga channel RGB di suatu piksel.

|  |
| --- |
| import cv2  import numpy as np  m = cv2.imread('mangga.jpg')  height, width, depth = np.shape(m)  for py in range(0, height) :  for px in range(0, width) :  rgb = (int(m[py][px][0]) + int(m[py][px][1]) + int(m[py][px][2])) / 3  m[py][px][0] = int(rgb)  m[py][px][1] = int(rgb)  m[py][px][2] = int(rgb)  cv2.imshow('matrix', m)  cv2.imwrite('output2.png', m)  cv2.waitKey(0)  cv2.destroyAllWindows() |



Gambar 2.6 Citra *Grayscale*

Nested loop (perulangan didalam perulangan) pada kode program diatas berfungsi untuk merubah setiap pixel pada gambar menjadi grayscale dengan menggunakan nested loop bisa mengakses pixel satu persatu setelah itu tiap tiap pixel akan dirubah nilainya dengan menghitung rata rata warna setiap channel dan hasil dari rata rata tersebut akan disimpan di masing masing channel (blue, green, red) lalu ketika ke tiga channel dikombinasikan dengan angka yang sama maka akan menghasilkan warna grayscale karena warna merah, hijau dan biru tidak ada warna yang mendominasi (sama semua).

## 2.6 Latihan Bab II

1. Bagaimana cara membaca image berformat jpg?

2. Berapa dimensi array yang digunakan untuk menyimpan image berformat png?

3. Apa perbedaan image jpg dan png dari segi channel intensitasnya?

# Operasi Titik

bab III

Capaian Mahasiswa:

* Mahasiswa mampu memahami pengertian operasi titik dan macam-macam operasi titik.
* Mahasiswa mampu melakukan tranformasi skala keabuan, modifikasi kecemerlangan, modifikasi kontras, serta operasi negasi.

## 3.1 Pengertian Operasi titik

Operasi titik adalah operasi terhadap citra dimana setiap titik diolah secara tidak gayut(tergantung) dengan titik-titik yang lain.

Tiap titik memiliki 2 buah karakteristik, yaitu :

* Koordinat, yang menunjukkkan lokasi dari titik tersebut didalam citra
* Nilai dari titik itu sendiri (tingkat keabuan atau warna)

Operasi titik tidak gayut terhadap koordinat titik itu sendiri maupun titik-titik yang lain. Hanya nilai dari titik itu sendiri yang dimodifikasi. Sehingga, pada titik yang memiliki nilai keabuan atau warna yang sama akan diubah menjadi keabuan atau warna yang sama pula.

Beberapa operasi pengolahan citra yang termasuk dalam kelompok operasi titik adalah, operasi modifikasi kecemerlangan*(Brigtness modification*), peningkatan kontras (*contrast enhacement*), negasi (*negation*), dan operasi pengambangan (*tresholding*).

## 3.2 Histogram

Sebelum mengkaji lebih jauh macam-macam operasi titik yang disebut sebelumnya terlebih dulu akan dibahas mengenai Histogram yang merupakan sebuah fungsi yang menjelaskan berapa kali sebuah nilai keabuan muncul dalam sebuah gambar. Histogram ini digunakan Untuk menyatakan distribusi data dari nilai derajat keabuan.

Misal terdapat gambar dengan code warna sebagai berikut :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 4 | 3 |
| 1 | 0 | 1 | 2 | 0 |
| 4 | 3 | 2 | 0 | 2 |
| 0 | 5 | 0 | 4 | 0 |
| 5 | 0 | 5 | 0 | 4 |

Maka Histogramnya adalah : X = [10,3,3,2,4,3]

Artinya angka 0 muncul 10 kali, angka 1 muncul 3 kali, angka 2 muncul 3 kali, angka 3 muncul 2 kali, angka 4 muncul 4 kali dan angka 5 muncul 2 kali, total kemunculan adalah 25 kali atau sebanyak jumlah pixel.

### 3.2.1 Pembuatan Histogram

Langkah untuk menentukan Histogram skala keabuan:

* Pertama siapkan variabel vektor untuk menyimpan cacah piksel untuk masing-masing keabuan.
* Untuk semua piksel dalam citra, periksa nilai piksel tersebutdan naikkan 1 nilai variabel yang menyimpan cacah piksel dengan nilai keabuan tersebut.

Algoritma untuk menghitung histogram warna dilakukan dengan cara yang sama terhadap masin-masing elemen warna (merah(R), hijau (G) dan Biru(B)). Histogram hanya dihitung bila sudah ada citra yang diambil dari sebuah file, karena itu periksa apakah telah ada citra atau belum

Di bawah ini merupakan contoh psoudocode untuk mengihitung histogram secara manual dalam bahasa pemrograman python.

|  |
| --- |
| import cv2  import numpy as np  import matplotlib.pyplot as plt  m = cv2.imread('mangga.jpg')  height, width, depth = np.shape(m)  names = np.arange(256)  blue = [0] \* 256  green = [0] \* 256  red = [0] \* 256  for y in range(0, height) :  for x in range(0, width) :  blue[m[y][x][0]] += 1  green[m[y][x][1]] += 1  red[m[y][x][2]] += 1  plt.bar(names, blue)  plt.show() |

Variabel names merupakan sebuah array dimana setiap nilai dan juga Panjang arraynya ditentukan oleh np.arange(x) yang merupakan salah satu fitur library dari numpy dimana panjang arraynya sama denga nilai x dan juga nilai dari setiap index akan sama seperti x. Variabel blue, green, dan red juga merupakan sebuah array dengan panjang 256 dengan semua nilai disetiap indexnya adalah 0. Kedua perulangan pada kode program diatas digunakan untuk megakses setiap pixel sesuai dengan tinggi dan panjang gambar.

blue[m[y][x][0]] += 1

kode program diatas berarti menambakan nilai array blue pada index m[y][x][0] (tingkat warna biru dari gambar m dengan sumbu pixel ketinggian y dan lebar x) sebanyak 1. Sehingga blue, green dan red akan menyimpan jumlah warna dengan angka yang sama

Selain cara manual di atas, dalam OpenCv tersedia fungsi perhitungan histogram yang siap digunakan dengan mudah. Fungsi perhitungan histogram tersebut seperti ditunjukkan dalam pseudocode berikut:

|  |
| --- |
| img = cv.imread('home.jpg',0)  hist = cv.calcHist([img],[0],None,[256],[0,256]) |

cv.calcHist *(images, channels, mask, histSize, ranges[hist[accumulate]])* merupakan salah satu fitur yang disediakan oleh library opencv-python yang berfungsi untuk melakukan perhitungan histogram dan terdapat beberapa parameter didalamnya

images : variabel bertipe gambar. Harus diberikan dalam tanda kurung siku, contoh, "[img]".

channels : channel atau warna dari gambar yang ingin di hitung histrogramnya. Misalnya, jika input berupa citra grayscale, nilainya [0]. Untuk citra berwarna dapat menggunakan [0], [1] atau [2] untuk menghitung histogram saluran biru, hijau, atau merah.

mask : mask image. Untuk menemukan histogram dari gambar penuh, itu diberikan sebagai "None". Tetapi jika ingin mencari histogram dari wilayah citra tertentu, harus membuat mask image untuk itu dan memberikannya sebagai mask

histSize : ini mewakili jumlah BIN. Perlu diberikan dalam tanda kurung siku. Untuk skala penuh menggunakan [256].

ranges : ini adalah area. normalnya [0,256].

### 3.2.2 Histogram Tingkat Keabuan

Merupakan suatu fungsi yang menunjukkan jumlah titik yang ada dalam suatu citra untuk tingkat keabuan. Histogram Tingkat keabuan ini merupakan alat bantu yang paling sederhana dan sangat berguna dalam pengolahan citra digital.

Absis (*x*) nya adalah tingkat keabuan, dan ordinat (*y*) adalah frekuensi kemunculan atau banyaknya titik dengan nilai keabuan tertentu.

### 3.2.3 Kegunaan Histogram pada pengolahan citra

Kegunaan histogram dalam pengolahan citra antara lain:

* Penentuan parameter digitisasi

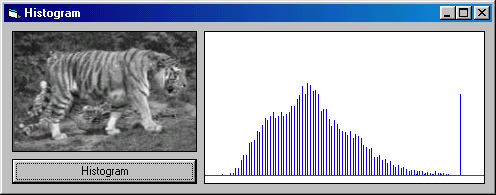
Digunakan sebagai indikasi visual untuk menentukan apakah suatu citra telah berada dalam area yang tepat pada skala keabuan.

* Pemilihan batas ambang

Digunakan untuk menonjolkan citra suatu obyek dari latar belakangnya.

Banyak sekali proses pengolahan citra yang melibatkan distribusi data, seperti pada contoh konversi biner, *image enhancement* (perbaikan citra), distribusi dari nilai derajat keabuan pada citra menjadi suatu acuan dasar. Untuk menyatakan distribusi data dari nilai derajat keabuan ini dapat digunakan nilai histogram.

Contoh hasil histogram pada sebuah gambar sebagai berikut:



Gambar 3.1. Contoh hasil histogram

## 3.3 Fungsi transformasi skala keabuan

Operasi titik dilakukan dengan memodifikasi nilai skala keabuan dari titik yang ditinjau tersebut berdasarkan fungsi tertentu, yang disebut sebagai fungsi transformasi skala keabuan (*gray-scale transformation*). Fungsi ini memetakan tingkat keabuan input (*Ki*) ke citra keabuan output(*K0*).

Secara Umum fungsi tersebut dapat dirumuskan sebagai:

*K0 = f(K)*

Bentuk fungsi *Gray-Scale Transformation (f)*, bisa berupa fungsi linier atau non linier tergantung aplikasi pengolahan citra yang kita gunakan. Pada umumnya bentuk fungsi *Gray-Scale Transformation (GST),* bersifat monotonically increasing, dimana dengan bertambahnya nilai keabuan masukan (*input*), keabuan keluaran (*output*)nya tidak pernah menurun.

Dalam citra True Color, Fungsi GST(*f*) diterapkan pada ketiga elemen warna yang ada (merah,hijau dan biru). Ketiga fungsi GST bisa memiliki bentuk sama atau berbeda. Artinya perlakuan pada ketiga elemen warna tersebut tidak harus sama.

Secara umum, Operasi titik pada citra warna dapat dirumuskan sebagai :

*R0 = fR(Ri)*

*G0 = fG(Gi)*

*B0 = fB(Bi)*

Dimana,

*Ri, Gi, Bi*, merupakan komponen warna citra input untuk elemen merah, hijau dan biru.

*R0, G0, B0*, merupakan komponen warna citra output, dan

*fR, fG, fB*, merupakan fungsi GST untuk elemen merah, hijau dan biru.

## 3.4 Modifikasi kecemerlangan citra

Dalam pengolahan citra penambahan nilai warna putih dilakukan dengan cara meningkatkan skala keabuan dari seluruh bagian (setiap titik) dalam citra , untuk meningkatkan kecemerlangannya (Brightness). Sebaliknya jika citra terlalu cemerlang atau tampak pucat, maka tingkat keabuan dari setiap titik dalam citra itu perlu diturunkan.

Proses memetakan skala keabuan dapat dirumuskan dalam fungsi linier berikut:

*K0 =  Ki +  C*

Dimana,

*C* merupakan konstanta yang bernilai positif bila hendak meningkatkan kecemerlangan citra , dan sebaliknya bernilai negatif bila hendak menurunkan kecemerlangan citra. Pada operasi ini Terjadi pergeseran histogram kearah kanan pada operasi peningkatan kecemerlangan dan kekiri pada operasi peredupan (penurunan kecemerlangan).

Pada citra true color, modifikasi kecemerlangan diterapkan pada setiap elemen warna yang masing-masing nilainya berbeda dengan lainnya. Contohnya, jika warna citra akan dibuat lebih hijau, maka nilai hijau pada setiap titik ditambah dengan nilai tertentu, sebaliknya jika warna merah pada citra dikurangi, maka elemen merah setiap titik dikurangi.

Secara umum formula tersebut dapat dinyatakan dengan:

R0 = Ri + CR

G0 = Gi + CG

B0 = Bi + CB

Dimana,

CR, CGdan  *CB*, merupakan komponen warna citra input untuk elemen merah, hijau dan biru.

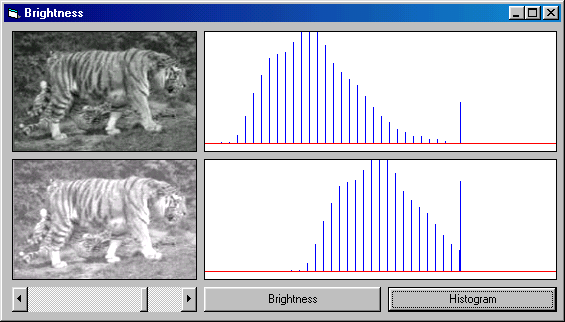
*R0, G0, B0*, merupakan komponen warna citra output, dan

*fR, fG, fB*, merupakan fungsi GST untuk elemen merah, hijau dan biru.

Contoh pseudocode modifikasi kecemerlangan secara manual adalah sebagai berikut:

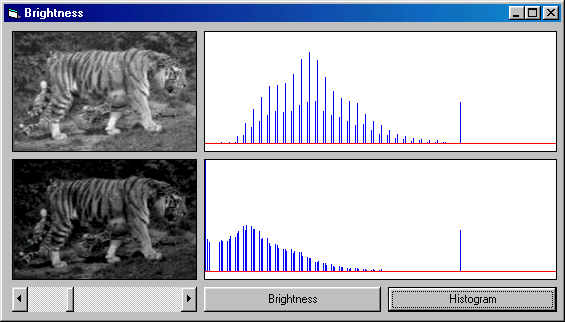
|  |
| --- |
| import cv2  import numpy as np  import matplotlib.pyplot as plt  m = cv2.imread('mangga.jpg')  height, width, depth = np.shape(m)  cR = 5  cG = -5  cB = -5  for y in range(0, height) :  for x in range(0, width) :  m[y][x][0] = m[y][x][0] + cB  if (m[y][x][0] < 0):  m[y][x][0] = 0  if (m[y][x][0] > 255):  m[y][x][0] = 255  m[y][x][1] = m[y][x][1] + cG  if (m[y][x][1] < 0):  m[y][x][1] = 0  if (m[y][x][1] > 255):  m[y][x][1] = 255  m[y][x][2] = m[y][x][2] + cR  if (m[y][x][2] < 0):  m[y][x][2] = 0  if (m[y][x][2] > 255):  m[y][x][2] = 255  cv2.imshow('image', m)  cv2.waitKey(0)  cv2.destroyAllWindows() |

Contoh Gambar modifikasi kecemerlangan adalah:



Gambar 3.2.Form untuk proses brightness positif

Hasil dari proses brightness dengan nilai penambah negatif dengan menggeser scroll ke kiri adalah sebagai berikut:



Gambar 3.3. Form untuk proses brightness negative

## 3.5 Peningkatan kontras

Sebuah citra akan kelihatan kurang kontras bila dengan kurva histogram citra tersebut memiliki nilai keabuan yang tidak terlalu berbeda untuk semua titik. Ini dikarenakan kurva tersebut mempunyai kurva histogram yang sempit, dengan tepi kiri dan kanan yang berdekatan, sehingga titik tergelap dalam citra tersebut tidak mencapai hitam pekat dan titik paling terang dalam citra tersebut tidak berwarna putih cemerlang. Ada berbagai formula peningkatan kontras yang dapat digunakan, diantara fungsi GST yang bisa digunakan adalah:

*K0 = G(Ki – P) + P*

Dimana,

*G* merupakan koefisien penguatan kontras, dan

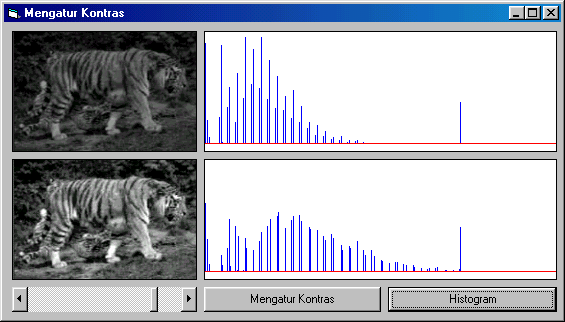
*P* merupakan nilai skala keabuan yang dipakai sebagai pusat pengontrasan.

Pada peningkatan kontras dasarnya adalah membuat titik yang cenderung gelap menjadi lebih gelap dan yang cenderung terang menjadi lebih cemerlang. Efek dari peningkatan kontras terhadap histogram mengakibatkan terjadinya pelebaran histogram dengan pusat di P kearah kanan dan kiri. Apabila koefisien penguatan kontras G kurang dari 1, maka pada gambar akan terjadi pengurangan kontras.

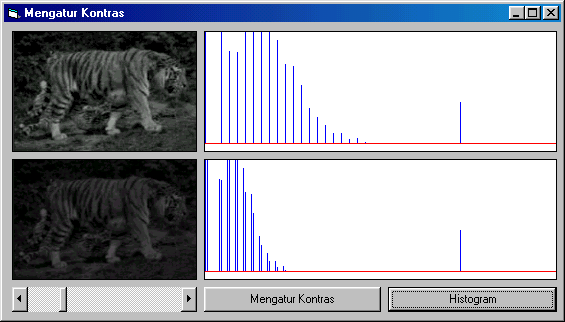
Contoh pseudocode peningkatan kontras adalah sebagai berikut:

|  |
| --- |
| import cv2  import numpy as np  import matplotlib.pyplot as plt  m = cv2.imread('mangga.jpg')  height, width, depth = np.shape(m)  G = 2  P = 3  for y in range(0, height) :  for x in range(0, width) :  m[y][x][0] = (G \* (m[y][x][0] - P)) + P  if (m[y][x][0] < 0):  m[y][x][0] = 0  if (m[y][x][0] > 255):  m[y][x][0] = 255  m[y][x][1] = (G \* (m[y][x][1] - P)) + P  if (m[y][x][1] < 0):  m[y][x][1] = 0  if (m[y][x][1] > 255):  m[y][x][1] = 255  m[y][x][2] = (G \* (m[y][x][2] - P)) + P  if (m[y][x][2] < 0):  m[y][x][2] = 0  if (m[y][x][2] > 255):  m[y][x][2] = 255  cv2.imshow('image', m)  cv2.waitKey(0)  cv2.destroyAllWindows() |

Contoh Peningkatan kontras :



Gambar 3.4. Form untuk proses peningkatan kontras



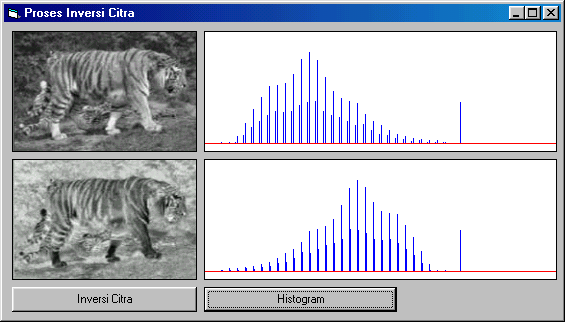
Gambar 3.5.Form untuk proses pengurangan kontras

## 3.6 Negasi atau Inversi Citra

Negasi citra adalah proses negatif pada photo, dimana setiap nilai citra dibalik dengan acuan threshold yang diberikan. Proses ini banyak digunakan pada citra-citra medis seperti USG dan X-Ray. Untuk citra dengan derajat keabuan 256, proses Negasi citra didefinisikan dengan:

*xn = 255 - x*

Contoh Hasil dari proses negasi adalah sebagai berikut:



Gambar 3.6. Contoh hasil proses Negasi

Perhatikan histogram dari kedua gambar. Histogram Negasi adalah kebalikan secara horisontal (horisontal flip) dari histogram dari citra. Proses Negasi ini merupakan salah satu proses dasar dalam image enhancement dengan menggunakan histogram.

Jika kita bandingkan foto hasil cetakan dengan hasil negatifnya, maka titik yang berwarna putih pada citra mempunyai warna hitam pada film negatifnya, demikian pula sebaliknya. Dengan demikian operasi negasi dilakukan dengan mengubah nilai keabuan titik dalam citra dengan nilai ”negatif” nya.

Sehingga Fungsi GST yang bisa diformulasikan adalah :

*K0 =K max – Ki*

Dimana,

*K max*, merupakan nilai skala keabuan yang tertinggi, untuk skala keabuan 8 bit, maka Kmax = 255.

Pengaruh operasi negasi terhadap histogram dapat dilihat pada gambar dibawah ini, dimana histogram citra asli dicerminkan kearah horisontal.

Contoh psuodocode negasi adalah sebagai berikut:

|  |
| --- |
| import cv2  import numpy as np  import matplotlib.pyplot as plt  m = cv2.imread('mangga.jpg')  height, width, depth = np.shape(m)  for y in range(0, height) :  for x in range(0, width) :  m[y][x][0] = 255 - m[y][x][0]  m[y][x][1] = 255 - m[y][x][1]  m[y][x][2] = 255 - m[y][x][2]  cv2.imshow('image', m)  cv2.waitKey(0)  cv2.destroyAllWindows() |

## 3.7 Operasi citra True Color Ke Keabuan

Citra True Color bisa dikonversi menjadi citra keabuan dengan operasi titik Secara sederhana, intensitas dapat didefinisikan sebagai nilai rerata dari ketiga nilai elemen warna.

Untuk menyederhanakan model citra langkah awal yang sering dilakukan dalam *image processing* adalah mengubah citra berwarna menjadi citra gray-scale. Seperti diketahui citra berwarna terdiri dari 3 layer matrik yaitu R-layer, G-layer dan B-layer. Sehingga untuk melakukan proses-proses selanjutnya tetap diperhatikan tiga layer di atas.

Bila setiap proses perhitungan dilakukan menggunakan tiga layer, berarti dilakukan tiga perhitungan yang sama. Sehingga konsep itu diubah dengan mengubah 3 layer di atas menjadi 1 layer matrik gray-scale dan hasilnya adalah citra gray-scale. Dalam citra ini tidak ada lagi warna, yang ada adalah derajat keabuan.

Untuk mengubah citra berwarna yang mempunyai nilai matrik masing-masing r, g dan b menjadi citra gray scale dengan nilai s, maka konversi dapat dilakukan dengan mengambil rata-rata dari nilai r, g dan b sehingga dapat dituliskan menjadi:



Pada penjelasan di atas pengubahan citra berwarna menjadi citra gray-scale dilakukan dengan menggunakan rata-rata nilai gray-scale dari setiap layer R,G dan B. Hal ini bukanlah suatu keharusan. Meskipun hasilnya sudah cukup bagus, pemakaian nilai rata-rata masih belum optimal untuk menunjukkan citra gray-scale sehingga terkadang harus dilakukan pengubahan komposisi, misalkan dengan:

*s = 0.42 r + 0.32 g + 0.28 b*

Jika rumus tersebut dicoba saat dilihat sepintas mungkin hasilnya tidak berbeda dengan penggunaan nilai rata-rata, tetapi bila dicermati akan tampak perbedaan sesungguhnya.

Contoh hasil pemakaian nilai rata-rata dimana *s=(r+g+b)/3* dan pemakaian nilai gray scale optimal *s = 0.42r + 0.32g + 0.28b* adalah sebagai berikut:



Gambar 3.7. Perbandingan gray-scale rata-rata dan optimal

Contoh pseudocode merubah citra true color ke citra grayscale optimal secara manual adalah sebagai berikut:

|  |
| --- |
| import cv2  import numpy as np  m = cv2.imread('mangga.jpg')  height, width, depth = np.shape(m)  for py in range(0, height) :  for px in range(0, width) :  rgb = ((0.42 \* (int(m[py][px][0]))) + (0.32 \* (int(m[py][px][1]))) + (0.28 \* (int(m[py][px][2]))))  m[py][px][0] = int(rgb)  m[py][px][1] = int(rgb)  m[py][px][2] = int(rgb)  cv2.imshow('matrix', m)  cv2.imwrite('output2.png', m)  cv2.waitKey(0)  cv2.destroyAllWindows() |

## 3.8 Pengambangan (thresholding)

Thresholding digunakan untuk mengatur jumlah derajat keabuan yang ada pada citra. Dengan menggunakan thresholding maka derajat keabuan bisa diubah sesuai keinginan, misalkan diinginkan menggunakan derajat keabuan 16, maka tinggal membagi nilai derajat keabuan dengan 16. Proses thresholding ini pada dasarnya adalah proses pengubahan kuantisasi pada citra, sehingga untuk melakukan thresholding dengan derajat keabuan a dapat digunakan rumus:



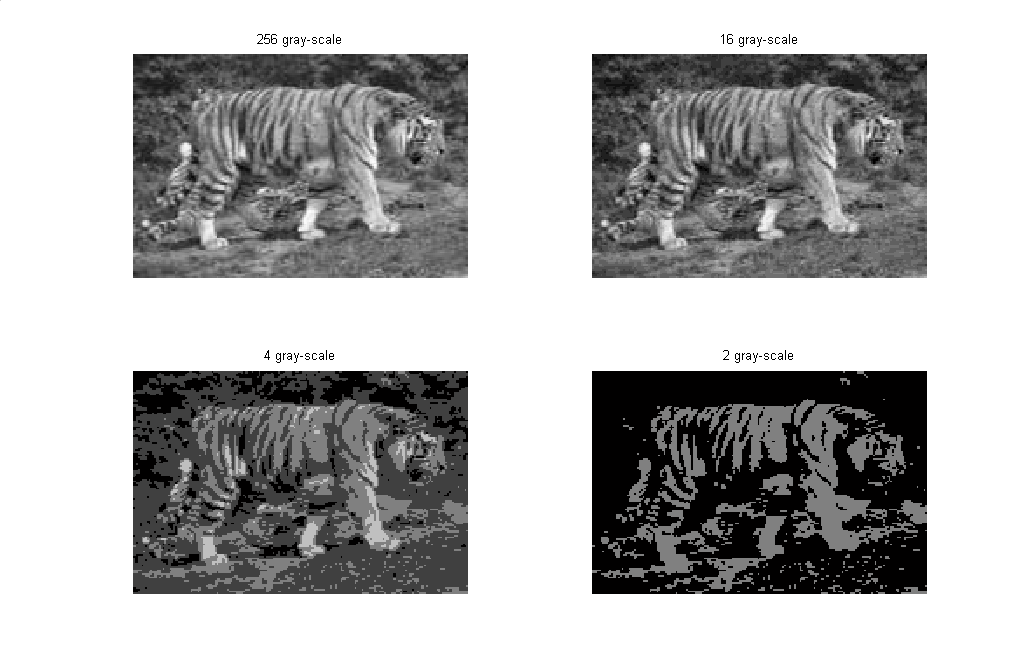
dimana :

w adalah nilai derajat keabuan sebelum thresholding

x adalah nilai derajat keabuan setelah thresholding



Berikut ini contoh thresholding mulai di 256, 16, 4 dan 2.



Gambar 3.8. Contoh thresholding

Untuk gambar berwarna terlebih dahulu diubah menjadi gray-scale dengan menggunakan cara rata-rata x=(r+g+b)/3 baru kemudian dithreshold. Hanya proses RGBtoGray tidak ditampilkan.

Pada threshold yang tinggi, hampir tidak tampak perbedaan karena keterbatasan mata, tetapi pada nilai threshold yang kecil seperti 2, 4, 8 dan 16 tampak sekali perbedaannya. Untuk threshold 2, ini sama artinya dengan mengubah citra menjadi citra biner.

Contoh pseoudocode pengambangan dalam bahasa python secara manual adalah sebagai berikut.

|  |
| --- |
| import cv2  import numpy as np  import matplotlib.pyplot as plt  m = cv2.imread('mangga.jpg', cv2.IMREAD\_GRAYSCALE)  height, width = np.shape(m)  a = 16  for y in range(0, height) :  for x in range(0, width) :  b = int(256/a)  n = b \* int(m[y][x]/b)  m[y][x] = n  cv2.imshow('image', m)  cv2.waitKey(0)  cv2.destroyAllWindows() |

## 3.9 Konversi Ke Citra Biner

Citra biner (hitam-putih) merupakan citra yang banyak dimanfaatkan untuk keperluan *pattern recognition* yang sederhana seperti pengenalan angka atau pengenalan huruf. Untuk mengubah suatu citra gray scale menjadi citra biner, sebetulnya prosesnya sama dengan threshold yaitu mengubah kuantisasi citra. Untuk citra dengan derajat keabuan 256, maka nilai tengahnya adalah 128 sehingga untuk mengubah menjadi citra biner dapat dituliskan:

*Jika x<128 maka x=0, jika tidak maka x=255*



Gambar 3.9. Hasil program citra biner

Hasil biner di atas tampak kurang sempurna, hal ini disebabkan distribusi nilai derajat keabuan tidak simetri antara yang dibawah 128 dan yang di atas 128. Untuk itu diperlukan jaminan simetri untuk mendapatkan hasil nilai biner yang baik. Sebelumnya dihitung dulu nilai rata-rata derajat keabuan dan kemudian ditentukan thresholdingnya.



*Jika x<xr maka x=0, jika tidak x=255*



Gambar 3.10. Hasil program citra biner dengan rata-rata

Terlihat hasilnya lebih banyak obyek yang dihasilkan, hal ini disebabakan distribusinya dibuat simetri dengan pemakaian rata-rata. Pada gambar 3.10. ini jumlah titik hitam dan jumlah titik putih relatif sama.

Contoh pseudocode citra biner dalam bahasa python secara manual adalah sebagai berikut.

|  |
| --- |
| import cv2  import numpy as np  import matplotlib.pyplot as plt  m = cv2.imread('mangga.jpg', cv2.IMREAD\_GRAYSCALE)  height, width = np.shape(m)  r = 0  for y in range(0, height) :  for x in range(0, width) :  r += m[y][x]  r = r/(height\*width)  for y in range(0, height) :  for x in range(0, width) :  m[y][x] = 0 if m[y][x] < r else 255  cv2.imshow('image', m)  cv2.waitKey(0)  cv2.destroyAllWindows() |

## 3.10 Latihan Bab III

1. Jelaskan macam-macam operasi titik dan kegunaan operasi titik tersebut?
2. Ada berapa macam teknik merubah citra *True Color* ke *grayscale*?
3. Jelaskan kasus tertentu yang membutuhkan penggunaan citra *grayscale*?

# Operasi Global

bab iv

Capain Mahasiswa:

* Memahami macam-macam operasi global pada citra.
* Mampu mengerti kegunaan operasi ekualisasi histogram pada citra serta menerapkannya.

## 4.1. Pengertian Operasi Global

Pada operasi global, proses yang dilakukan bergantung pada karakteristik global dari citra yang hendak dimodifikasi. Karakteristik global tersebut biasanya berupa sifat statistik dari citra itu sendiri, yang direpresentasikan dengan histogram tingkat keabuan.

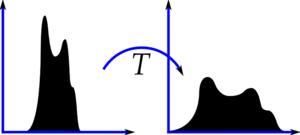
Sebagian pakar mengelompokkan operasi pengambangan (*thresholding*) sebagai operasi global, karena pemilihan batas ambang dilakukan dengan menginspeksi histogram tingkat keabuan yang diturunkan secara global (mempertimbangkan keseluruhan titik pada citra tersebut). Namun operasi yang sebenarnya dilakukan adalah secara titik per titik. Sebaliknya, ada yang memasukkan operasi global ini kedalam kelompok operasi titk, karena penerapannya juga dilakukan secara lokal atau titik per titik.

Dalam buku ini, definisi operasi global adalah operasi yang tak bisa tidak, atau harus, bergantung pada sifat global citra. Salah satu operasi global adalah ekualisasi histogram.

## 4.2. Ekualisasi Histogram

Pada bab sebelumnya, citra yang kurang kontras ditandai dengan sempitnya daerah yang dipakai oleh kurva histogram tingkat keabuan. Dengan operasi peningkatan kontras yang optimal, kurva histogram akan memiliki rentang yang maksimum, dari batas kiri ke batas kanan histogram. Cara lain yang dapat dilakukan untuk mendapatkan kontras yang optimal adalah dengan mendistribusikan kembali nilai-nilai skala keabuan citra untuk memperoleh kurva histogram yang datar atau seragam.

Untuk citra skala keabuan *k* bit yang berukuran tinggi *h* dan lebar *w*, maka jumlah titk untuk setiap tingkat keabuan adalah seragam sebesar (*w.h/2*). Untuk memperoleh hasil seperti itu, distribusi titk dalam citra asli harus disebarkan secara lebih merata ke seluruh nilai keabuan.

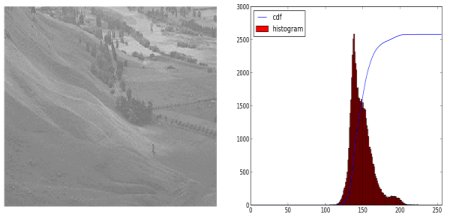


Gambar 4.1 Ilustrasi Ekualisasi histogram

Berikut ini adalah contoh pseudocode implementasi ekualisasi histogram secara manual.

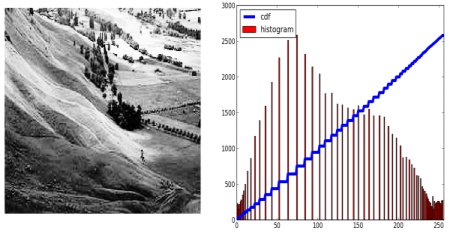
|  |
| --- |
| import cv2  import numpy as np  from matplotlib import pyplot as plt  img = cv2.imread('wiki2.jpg',0)  hist,bins = np.histogram(img.flatten(),256,[0,256])  cdf = hist.cumsum()  cdf\_normalized = cdf \* hist.max()/ cdf.max()  cdf\_m = np.ma.masked\_equal(cdf,0)  cdf\_m = (cdf\_m - cdf\_m.min())\*255/(cdf\_m.max()-cdf\_m.min())  cdf = np.ma.filled(cdf\_m,0).astype('uint8')  img2 = cdf[img]  cv2.imwrite('output2.jpg', img2)  plt.plot(cdf\_normalized, color = 'b')  plt.hist(img2.flatten(),256,[0,256], color = 'r')  plt.xlim([0,256])  plt.legend(('cdf','histogram'), loc = 'upper left')  plt.show() |

Sebelum dilakukan operasi ekualisasi histogram, terlihat distribusi nilai piksel hanya berada mengumpul di nilai tertentu saja. Hal ini mengakibatkan penampakan citra yang tidak terlalu jelas atau kurang kontras, seperti terlihat pada gambar 4.2



Gambar 4.2 Sebelum Dilakukan Operasi Ekualisasi Histogram

Setelah dilakukan operasi ekualisasi histogram, terlihat distribusi nilai piksel merata dan melebar. Sehingga mengakibatkan gambar terlihat lebih jelas atau memiliki kontras yang cukup.



Gambar 4.2 Setelah Dilakukan Operasi Ekualisasi Histogram

Selain menggunakan teknik manual di atas dalam OpenCv juga disediakan fungsi untuk melakukan operasi ekualisasi histogram. Untuk menggunakan fugsi tersebut dicontohkan dalam pseudocode berikut.

|  |
| --- |
| img = cv2.imread('wiki.jpg',0)  equ = cv2.equalizeHist(img)  res = np.hstack((img,equ)) #stacking images side-by-side  cv2.imwrite('res.png',res) |

## 4.3 Latihan Bab IV

1. Terangkan mengapa ketika dilakukan operasi equalisasi histogram, citra akan terlihat lebih kontras?
2. Sebutkan operasi global yang lain, selain operasi ekualisasi histogram!

# Operasi Berbasis Bingkai

bab v

Capain Mahasiswa:

* Mampu memahami macam-macam jenis operasi berbasis bingkai
* Mampu mengimplementasikan macam-macam operasi berbasis bingkai pada kode program.

## 5.1. Pengertian Operasi Berbasis Bingkai

Operasi berbasis bingkai (*frame*) adalah operasi yang melibatkan 2 buah citra atau lebih dan menghasilkan sebuah citra keluaran yang merupakan hasil operasi metematis. Operasi ini dilakukan titk per titik dengan lokasi yang bersesuaian pada citra-citra masukan tersebut. Untuk operasi berbasis bingkai antara 2 buah citra, misalnya antara citra *A* dan citra *B* yang menghasilkan citra *C*, secara umum dapat dituliskan sebagai :

*C(x,y) = A(x,y) op B(x,y)*

dimana *op* adalah operator yang akan diterapkan terhadap kedua citra tersebut. Untuk operasi yang melibatkan *N* buah citra *A*1 sampai dengan *AN*, maka rumus diatas menjadi :

C(x,y) = A1(x,y) op A2(x,y) op A3(x,y) ... op AN(x,y)

Berdasarkan operator yang dipakai, operasi berbasis bingkai meliputi operasi aritmatik dan operasi logika.

## 5.2. Operasi Aritmatik

Operasi aritmatik biasanya diterapkan terhadap citra keabuan dan warna. Berdasarkan operator aritmatik yang umum digunakan, terdapat 4 operasi aritmatik yang dapat dirumuskan sebagai berikut (untuk 2 citra)

*C(x,y) = A(x,y) + B(x,y)*

*C(x,y) = A(x,y) - B(x,y)*

*C(x,y) = A(x,y) \* B(x,y)*

*C(x,y) = A(x,y) / B(x,y)*

Operasi yang banyak digunakan pada berbagai aplikasi ádalah penjumlahan dan pengurangan, sedangkan operasi perkalian dan pembagian hanya digunakan untuk kasus-kasus tertentu.

Contoh aplikasi operasi aritmatik ádalah untuk pengurangan derau (*noise*), penggabungan citra (*image blending*), perbaikan ketidak linieran pencahayaan, deteksi gerakan, pembuatan citra diferensi untuk deteksi tepi, serta *digital angiography*. Operasi pembagian menghasilkan citra rasio yang berguna pada analisis citra warna/multispektral. Operasi perkalian dengan sebuah citra *mask* menghilangkan bagian tertentu dari citra dan menampakkan hanya obyek yang diinginkan.

## 5.3. Penggabungan citra (Image blending)

Penggabungan citra dilakukan dengan cara menimpakan (*superimpose*) sebuah citra pada citra yang lain. Operasi yang sesunguhnya adalah dengan melakukan operasi penjumlahan terhadap citra yang ada dengan bobot masing-masing. Rumus untuk penggabungan 2 buah citra adalah:

*C(x,y)* =  *wA\*A(x,y) + wB\*B(x,y)*

dengan *wA* dan *wB* adalah bobot untuk citra *A* dan *B*. Biasanya nilai total dari bobot untuk sebuah lokasi pixel adalah 1 sehingga

*wB = 1 - wA*

Contoh pseudocode penggabungan citra secara manual adalah sebagai berikut:

|  |
| --- |
| import cv2  import numpy as np  m1 = cv2.imread('mangga.jpg')  m2 = cv2.imread('apel.jpg')  height, width, depth = np.shape(m1)  height2, width2, depth2 = np.shape(m2)  wA = 0.7  wB = 1 - wA  for py in range(0, height if height < height2 else height2) :  for px in range(0, width if width < width2 else width2) :  m1[py][px][0] = (wA \* int(m1[py][px][0])) + (wB \* int(m2[py][px][0]) if m2[py][px][0] != None else 0)  m1[py][px][1] = (wA \* int(m1[py][px][1])) + (wB \* int(m2[py][px][1]) if m2[py][px][1] != None else 0)  m1[py][px][2] = (wA \* int(m1[py][px][2])) + (wB \* int(m2[py][px][2]) if m2[py][px][2] != None else 0)  if (m1[py][px][0] > 255):  m1[py][px][0] = 255  if (m1[py][px][1] > 255):  m1[py][px][1] = 255  if (m1[py][px][2] > 255):  m1[py][px][2] = 255  cv2.imshow('matrix', m1)  cv2.imwrite('output2.png', m1)  cv2.waitKey(0)  cv2.destroyAllWindows() |



Citra B

Citra A



Citra A + Citra B

Gambar 5.1 Operasi Penggabungan Citra

OpenCV juga menyediakan fungsi untuk melakukan penggabungan citra. Untuk menggunakan fugsi tersebut dicontohkan dalam pseudocode berikut.

|  |
| --- |
| img1 = cv2.imread('ml.png')  img2 = cv2.imread('opencv\_logo.jpg')  dst = cv2.addWeighted(img1,0.7,img2,0.3,0)  cv2.imshow('dst',dst)  cv2.waitKey(0)  cv2.destroyAllWindows() |

## 5.4. Deteksi gerakan

Deteksi gerakan (*motion detection*) secara sederhana dapat dilakukan dengan mencari beda antara 2 buah citra yang berurutan pada hasil pencitraan menggunakan kamera video *digital*, Operator yang digunakan adalah pengurangan. Operasi pengurangan pada bagian yang tidak bergerak dalam citra akan menghasilkan nilai nol, sedangkan bagian yang bergerak akan memberikan nilai yang tidak nol. Dengan mengevaluasi nilai selisih tersebut, dapat diketahui apakah pada citra terdapat obyek yang bergerak.

Contoh pseudocode deteksi gerakan adalah sebagai berikut:

|  |
| --- |
| import cv2  import numpy as np  m1 = cv2.imread('mangga.jpg', cv2.IMREAD\_GRAYSCALE)  m2 = cv2.imread('mangga2.jpg', cv2.IMREAD\_GRAYSCALE)  height, width = np.shape(m1)  for py in range(0, height) :  for px in range(0, width) :  b = abs(m1[py][px] - m2[py][px])  m1[py][px] = 0 if b < 0 else 255 if b > 255 else b  cv2.imshow('matrix', m1)  cv2.imwrite('output2.png', m1)  cv2.waitKey(0)  cv2.destroyAllWindows() |





Gambar 5.2 Contoh Deteksi Gerak

## 5.5. Operasi Logika

Operasi Logika juga dapat dilakukan terhadap 2 atau lebih citra. Beberapa operator logika yang sering dipergunakan adalah:

*C(x,y) = A(x,y)*AND *B(x,y)*

*C(x,y) = A(x,y)*OR *B(x,y)*

*C(x,y) = A(x,y)*XOR *B(x,y)*

*C(x,y) = A(x,y)*SUB *B(x,y)*

*C(x,y) =* NOT *A(x,y)*

Dengan SUB adalah operasi pengurangan biasa, tetapi jika hasil pengurangan adalah negatif, maka hasilnya diganti dengan 0.

Contoh operasi logika dalam bahasa python adalah sebagai berikut:

|  |
| --- |
| **import** cv2 **import** numpy **as** np  m1 = cv2.imread(**'mangga.jpg'**) m2 = cv2.imread(**'apel.jpg'**)  m = m1.copy()  height, width, depth = np.shape(m1) height2, width2, depth2 = np.shape(m2)   **for** py **in** range(0, height **if** height < height2 **else** height2) :  **for** px **in** range(0, width **if** width < width2 **else** width2) :  m[py][px][0] = m1[py][px][0] | m2[py][px][0]  m[py][px][1] = m1[py][px][1] | m2[py][px][1]  m[py][px][2] = m1[py][px][2] | m2[py][px][2]  m[py][px][0] = 0 **if** m[py][px][0] < 0 **else** 255 **if** m[py][px][0] > 255 **else** m[py][px][0]  m[py][px][1] = 0 **if** m[py][px][1] < 0 **else** 255 **if** m[py][px][1] > 255 **else** m[py][px][1]  m[py][px][2] = 0 **if** m[py][px][2] < 0 **else** 255 **if** m[py][px][2] > 255 **else** m[py][px][2]    cv2.imshow(**'matrix'**, m) cv2.imwrite(**'output2.png'**, m) cv2.waitKey(0) cv2.destroyAllWindows() |







Gambar 5.3 Contoh Operasi Logika Or

## 5.6 Latihan Bab V

1. Apa fungsi operasi aritmatik pengurangan pada citra?

2. Ada berapa macam operasi berbasis bingkai pada pengolahan citra?

# Operasi Geometri

Bab VI

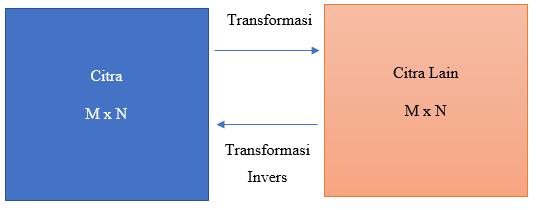
Capaian Mahasiswa:

* Memahami macam-macam transformasi yang melibatkan operasi geometri pada pengolahan citra.
* Melakukan berbagai macam transformasi geometri pengolahan citra dalam kode program.

## 6.1. Pengertian Operasi Geometri

Seperti telah diketahui bahwa karakteristik yang dimiliki oleh setiap pixel dalam statu cerita adalah koordinat dan nilai (keabuan atau warna) dari píxel tersebut. Pada bab-bab sebelumnya, operasi yang dilakukan (operasi titik, global, dan berbasis bingkai) dilakukan terhadap nilai keabuan atau warna piksel yang ada di dalam citra. Secara umum, operasi geometrik dilakukan terutama dengan memodifikasi koordinat piksel dalam statu citra dengan kemungkinan mengubah nilai skala keabuan dari titik tersebut dengan pendekatan tertentu.

Operasi geometrik berhubungan dengan perubahan bentuk geometri citra, yaitu baik ukuran ataupun orientasinya. Operasi geometri di antaranya meliputi pencerminan *(flipping)*, rotasi/pemutaran *(rotating)*, penskalaan *(scaling/zooming)*, dan pembengkokan *(warping).*



Gambar 6.1 Ilustrasi Transformasi Geometri

## 6.2. Transformasi Spasial

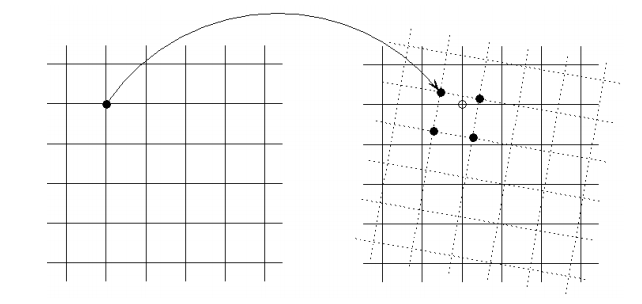
Operasi geometrik dilakukan dengan cara memetakan koordinat titik-titik citra asal ke koordinat titik-titik di citra hasil. Pemetaan dilakukan dengan menerapkan fungsi transformasi spasial. Secara umum operasi geometrik dapat dirumuskan dengan:

*x = fx(x,y)*

*y = fy(x,y)*

*Ko(x,y) = Ki(x,y)*

di mana *Ki(x,y)* adalah keabuan píxel pada koordinat asal dan *Ko(x’,y’)* adalah keabuan hasil pada koordinat tujuan.

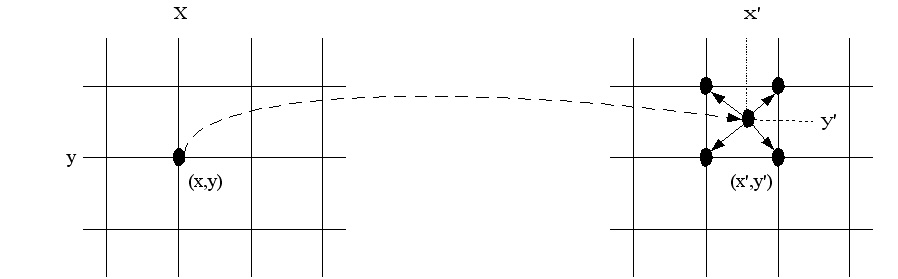


Gambar 6.2 Ilustrasi Transformasi Spasial

## 6.3. Transformasi maju

Citra hasil diperoleh dengan cara menghitung koordinat titik hasil berdasarkan koordinat titik asal menggunakan ketiga persamaan tadi. Metode ini disebut transformasi maju *(forward transformation).* Perlu diingat bahwa koordinat titik pada citra harus berupa bilangan bulat. Pada waktu dilakukan pemetaan dengan transformasi maju, koordinat titik asal (*x,y*) yang merupakan bilangan bulat tidak selalu menghasilkan koordinat titik hasil (*x’,y’*) yang berupa bilangan bulat pula.

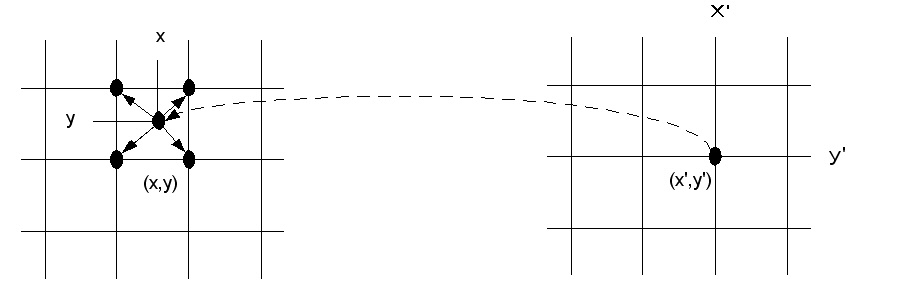
Seringkali diperoleh koordinat hasil berupa bilangan pecahan, yaitu bila koordinat hasil pemetaan tersebut jatuh diantara 4 buah titk hasil. Apabila hal itu terjadi, maka nilai keabuan titik asal didistribusikan ke keempat titik pada citra hasil. Pendistribusian nilai keabuan tadi dilakukan dengan aturan interpolasi tertentu. Cara ini sering disebut *píxel carry-over*. Kelebihan dari metode ini adalah biasanya fungsi transformasi yang digunakan untuk suatu operasi geometrik sudah diketahui, jadi penyelesaiannya menjadi lebih mudah.



Gambar 6.3 Ilustrasi Transformasi Maju

## 6.4. Transformasi balik

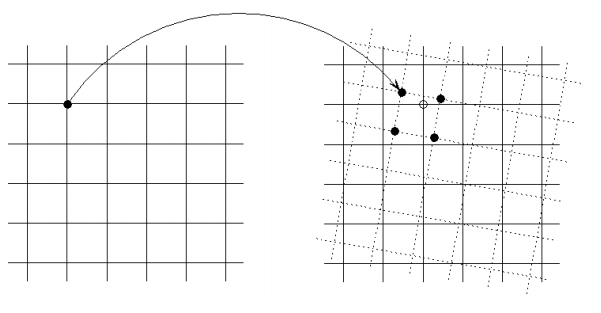
Pada transformasi ini tiap koordinat titik hasil dipetakan balik untuk mencari titik asalnya. Selanjutnya keabuan titik hasil diisi berdasarkan keabuan titik asalnya. Jika hasil pemetaan balik jatuh diantara 4 titik (nilai x dan y hasil transformasi balik tidak bulat), maka keabuan titik hasil dihitung berdasarkan nilai keabuan keempat titik tadi berdasarkan aturan interpoladi tertentu. Cara ini sering disebut pxel filling. Kelebihan transformasi mundur adalah nilai keabuan titik hasil dapat ditentukan hanya dengan melibatkan sedikit titik asal, sehingga nilainya dapat dihitung tanpa harus memetakan semua titik asal. Sayangnya, fungsi invers diatas pada kasus tertentu tidak mudah untuk diperoleh atau ditentukan dari fungsi transformasi maju yang sudah diketahui.



Gambar 6.4 Ilustrasi Transformasi Balik

## 6.5. Interpolasi nilai keabuan

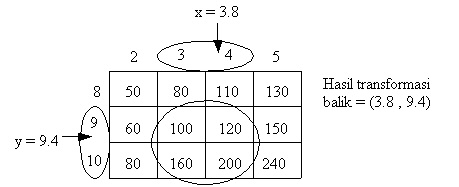
Pada kedua transformasi diatas seringkali terjadi situasi dimana dari penerapan transformasi spasial diperoleh koordinat titik hasil (pada transformasi maju) maupun titik asal (pada transformasi balik), yang bernilai tidak bulat, padahal koordinat titik harus selalu bernilai bulat Jika demikian maka diperlukan interpolasi untuk menentukan nilai keabuan pada koordinat titik asal. Interpolasi yang sering digunakan adalah intrpolasi tetengga terdekat, bilinier, dn interpolasi dengan orde lebih tinggi.



Gambar 6.5 Ilustrasi Interpolasi Nilai Keabuan

## 6.6. Interpolasi tetangga terdekat

Pada interpolasi tetanggga terdekat, nilai keabuan titik hasil diambil dari nilai keabuan pada titik asal yang paling dekat dengan koordinat hasil perhitungan dari transformasi spasial. Untuk citra 2 dimensi, tetangga terdekat dipilih antara 4 titik asal yang saling berbatasan satu sama lain.

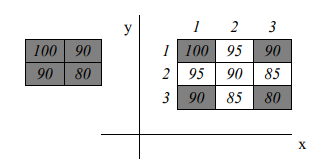


Gambar 6.6 Ilustrasi Interpolasi Tetangga Terdekat

Kelebihan dari interpolasi tetangga terdekat adalah kemudahan dan kecepatan eksekusinya. Namun pada banyak kasus, penggunaan interpolasi tetangga terdekat menghasilkan citra yang kurang memuaskan karena timbulnya aliasing pada bagian tepi obyek.

## 6.7. Interpolasi bilinier

Pada interpolasi bilinier, nilai keabuan dari keempat titik yang bertetangga tadi menberi sumbangan terhadap nilai keabuan hasil, dengan bobot masing-masing yang linier dengan jaraknya terhadap koordinat yang dimaksud. Makin dekat titik tetangga tersebut, makin besar bobotnya, dan sebaliknya makin jauh akan makin kecil bobotnya.



Gambar 6.7 Ilustrasi Interpolasi Bilinier

## 6.8. Perubahan Ukuran Citra

Pada operasi geometri, akibat transformasi spasial yang dilakukan, ukuran citra sangat mungkin berubah. Oleh karena itu ukuran citra harus ditentukan sebelum perhitungan keabuan output dilakukan. Apabila ukuran tidk berubah, maka ada kemungkinan sebagian titik asal dipetakan keluar dari batas citra sehingga tidak nampak pada citra hasil.

## 6.9. Aplikasi Operasi Geometri

### 6.9.1. Operasi pencerminan

Operasi pencerminan merupakan salah satu operasi geometri yang paling sederhana, karena tidak terjadi perubahan ukuran citra dan hasil pemetaan selalu menghasilkan koordinat titik berupa bilangan bulat, sehingga tidak memerlukan interpolasi nilai keabuan.

Efek operasi pencerminan adalah berubahnya orientasi citra baik secara horisontal, vertikal, maupun gabungan keduanya. Pada pencerminan secara horisontal, koordinat *y* tetap, sedangkan koordinat *x* dicerminkan. Rumus yang digunakan untuk mencerminkan koordinat *x* terhadap sumbu *y* adalah :

*x’ = -x*

Karena koordinat hasil *(x’*) bernilai nol atau positif, maka koordinat asal (*x*) yang diperoleh dari rumus tersebut akan bernilai negatif. Oleh karena itu rumus tersebut dimodifikasi agar x selalu positif. Hal ini dilakukan dengan melakukan pencerminan terhadap garis vertikal di tengah citra (pada koordinat *xc*).

Berikut ini adalah contoh pencerminan terhadap sumbu x dalam bahasa python.

|  |
| --- |
| import cv2  import numpy as np  m = cv2.imread('mangga.jpg')  m2 = m.copy()  height, width, depth = np.shape(m)  for y in range(0, height):  for x in range(0, width):  m2[y][x][0] = m[height - y -1][x][0]  m2[y][x][1] = m[height - y - 1][x][1]  m2[y][x][2] = m[height - y - 1][x][2]  print(str(y)+' - '+str(height - 1- y))  cv2.imshow('image', m2)  cv2.waitKey(0)  cv2.destroyAllWindows() |



(Citra Asli)



(Citra Hasil Pencerminan)

6.8 Contoh Operasi Pencerminan Terhadap Sumbu X

Contoh pseudocode operasi pencerminan terhadap sumbu y adalah sebagai berikut:

|  |
| --- |
| **import** cv2 **import** numpy **as** np  m = cv2.imread(**'mangga.jpg'**) m2 = m.copy() height, width, depth = np.shape(m)  **for** y **in** range(0, height):  **for** x **in** range(0, width):  m2[y][x][0] = m[y][width -1 - x][0]  m2[y][x][1] = m[y][width -1 - x][1]  m2[y][x][2] = m[y][width -1 - x][2]  cv2.imshow(**'image'**, m2) cv2.waitKey(0) cv2.destroyAllWindows() |

Pseudocde di atas menghasilkan citra



6.9 Contoh Operasi Pencerminan Terhadap Sumbu Y

Sedangkan contoh operasi pencerminan terhadap sumbu x dan y adalah sebagai berikut:

|  |
| --- |
| **import** cv2 **import** numpy **as** np  m = cv2.imread(**'mangga.jpg'**) m2 = m.copy() height, width, depth = np.shape(m)  **for** y **in** range(0, height):  **for** x **in** range(0, width):  m2[y][x][0] = m[height -1 - y][width -1 - x][0]  m2[y][x][1] = m[height -1 - y][width -1 - x][1]  m2[y][x][2] = m[height -1 - y][width -1 - x][2]  cv2.imshow(**'image'**, m2) cv2.waitKey(0) cv2.destroyAllWindows() |

Pseudocode di atas menghasilkan citra:



6.10 Contoh Operasi Pencerminan Terhadap Sumbu X dan Y

### 6.9.2. Operasi *cropping*

*Cropping* adalah memotong satu bagian dari citra sehinggga diperoleh citra yang berukuran lebih kecil. Operasi ini pada dasarnya adalah operasi translasi, yaitu menggeser koordinat titik citra. Rumus yang digunakan untuk operasi ini adalah :

*x’ = x – xL untuk x = xL sampai xR*

*y’ = y – yL untuk y = yT sampai yB*

*(xL,yT)* dan *(xR,yB)*, masing-masing adalah koordinat titik pojok kiri atas dan pojok kanan bawah bagian citra yang hendak di-*crop.*

Ukuran citra berubah menjadi

*h’ = yB - yT*

dan transformasi baliknya adalah ;

*x = x’ + xL untuk x’ = 0 sampai w’-1*

y *= y’ + xT untuk y = 0 sampai h’-1*

Contoh implementasi image cropping adalah sebagai berikut:

|  |
| --- |
| import cv2  import numpy as np  from PIL import Image  m = cv2.imread('mangga.jpg')  height, width, depth = np.shape(m)  xl = 50  xr = 300  yt = 50  yb = 100  m2 = np.zeros((yb-yt, xr-xl, 3), dtype=np.uint8)  for y in range(yt, yb):  for x in range(xl, xr):  m2[y-yt][x-xl][0] = m[y][x][0]  m2[y-yt][x-xl][1] = m[y][x][1]  m2[y-yt][x-xl][2] = m[y][x][2]  img = Image.fromarray(m2)  img.save('my.png')  img.show()  cv2.waitKey(0)  cv2.destroyAllWindows() |





6.11 Contoh Operasi Cropping

### 6.9.3. Operasi penskalaan (*scaling*)

Operasi penskalaan *(scaling)* dimaksudkan untuk memperbesar atau memperkecil citra. Hal ini dapat dilakukan dengan mengintroduksi parameter skala, baik kearah horisontal (Sh) maupun vertikal (Sv). Skala yang bernilai lebih dari 1 akan memperkecil citra. Apabila *aspect ratio* (perbandingan antara tinggi dan lebar citra) hendak dipertahankan, maka dipilih Sh = Sv. Transformasi spasial yang dipakai adalah:

*x’ = Shx*

*y’ = Svy*

Untuk melakukan operasi penskalaan dalam OpenCV seperti dicontohkan dalam pseudocode berikut:

|  |
| --- |
| import numpy as np  import cv2 as cv  img = cv.imread('messi5.jpg')  height, width = img.shape[:2]  res = cv.resize(img,(2\*width, 2\*height), interpolation = cv.INTER\_CUBIC) |

### 6.9.4. Operasi rotasi bebas

Operasi rotasi dengan sudut putar tidak tepat pada kelipatan 90o tidak sama dengan operasi ¼ dan ½ putaran yang sudah dibahas pada sub bab sebelumnya. Jika pada operasi rotasi sebelumnya, titik asal yang diperoleh dari transformasi balik selalu berupa bilangan bulat, pada operasi dengan sudut bebas, kemungkinan besar titik asal tidak bulat. Oleh karena itu perlu dilakukan interpolasi. Pada dasarnya, operasi ini sama dengan operasi penskalaan. Perbedaannya adalah pada pesamaan transformasi spasial yang digunakan.

Untuk melakukan operasi rotasi dalam OpenCV seperti dicontohkan dalam pseudocode berikut:

|  |
| --- |
| import cv2 as cv  import numpy as np  import matplotlib.pyplot as plt  img = cv.imread('messi5.jpg',0)  rows,cols = img.shape  # cols-1 and rows-1 are the coordinate limits.  M = cv.getRotationMatrix2D(((cols-1)/2.0,(rows-1)/2.0),90,1)  dst = cv.warpAffine(img,M,(cols,rows)) |



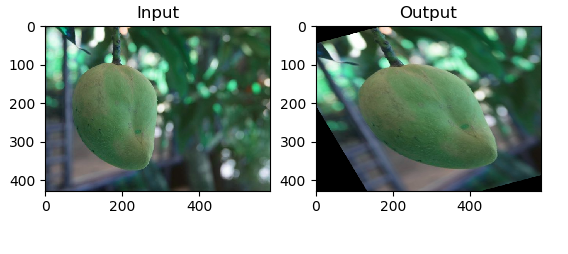
6.12 Contoh Operasi Rotasi Bebas

### 6.9.5. Operasi *skew*

Operasi *skew* (pendoyongan atau *affine*) dipergunakan untuk memperbaiki kemiringan citra. Operasi skew kearah horisontal sebenarnya adalah penggeseran baris-baris pada citra denga besar pergeseran yang berubah secara gradual. Pergeseran maksimum terjadi pada bagian bawah citra, sedangkan pergeseran pada baris tengah adalah proporsional terhadap posisi vertikalnya. Operasi ini hanya mengubah ukuran lebar citra, sedangkan tinggi citra tetap. Lebar cita bertambah sebesar pergeseran maksimum yang dilakukan.

Untuk melakukan operasi *skew* dalam OpenCV seperti dicontohkan dalam pseudocode berikut:

|  |
| --- |
| **import** cv2 **as** cv **import** numpy **as** np **import** matplotlib.pyplot **as** plt  img = cv.imread(**'mangga.jpg'**) rows,cols,ch = img.shape pts1 = np.float32([[50,50],[200,50],[50,200]]) pts2 = np.float32([[10,100],[200,50],[100,250]]) M = cv.getAffineTransform(pts1,pts2) dst = cv.warpAffine(img,M,(cols,rows)) plt.subplot(121),plt.imshow(img),plt.title(**'Input'**) plt.subplot(122),plt.imshow(dst),plt.title(**'Output'**) plt.show() |



6.13 Contoh Operasi *Skew*

## 6.10 Latihan Bab VI

1. Sebutkan macam-macam operasi geometri?

2. Mengapa diperlukan interpolasi nilai keabuan?

3. Jelaskan mengapa nilai keabuan tetangga terdekat bisa dijadikan acuan untuk proses interpolasi?

# Aplikasi Operasi Bertetangga

Bab VII

Capaian mahasiswa:

* Mengetahui macam-macam operasi ketetanggan dalam pengolahan citra
* Mengetahui penggunaan macam-macam operasi bertetangga

## 7.1. Penghalusan Citra

Dapat dilakukan dengan memberi nilai yang sama pada semua bobot *mask* yang digunakan. Operasi ini dapat pula digunakan untuk menghilangkan derau yang timbul saat proses pencitraan.

## 7.2. Pendeteksian Tepi

Digunakan untuk menentukan lokasi titik-titk yang merupakan tepi obyek. Dimana tepi obyek dinyatakan sebagai titik yang nilai keabuannya berbeda cukup besar dengan titik yang ada disebelahnya. Bentuk *mask* yang bisa digunakan dalam operasi ini antara lain :

Operator Robert, Prewitt, Sobel, Canny, Laplacian, dan Laplacian of Gaussian. Konvolusi dengan mask tersebut dipakai untuk menghitung nilai keabuan atau warna hasil.

### 7.2.1. Pendeteksian tepi menggunakan operator gradien

Pendeteksian tepi dapat dilakukan dengan menghitung selisih atau diferensi antara 2 buah titik yang bertetangga sehingga diperoleh nilai gradien (turunan pertama) citra.

Diferensi diambil secara diagonal untuk mendapatkan *mask* yang berbentuk bujur sangkar, dan diperoleh 2 *mask* mengakomodasi kedua arah diagonal. Hasil konvolasi citra asal dengan kedua *mask* tersebut kemudian digabungkan dengan cara tertentu sehingga menghasilkan nilai yang dapat digunakan untuk menentukan apakah sebuah titik merupakan tepi obyek dalam citra atau bukan.

### 7.2.2. Penajaman Citra (*Sharping*)

Karena *Laplacian* merupakan operator derivatif, penggunaannya akan menonjolkan ketidak sinambungan dalam citra yang terjadi pada tepi obyek. Oleh karena itu, operasi ini dapat dimanfaatkan untuk meningkatkan tepi obyek sehingga citra berkesan menjadi lebih tajam.

Operasi penajaman citra pada dasarnya adalah penjumlahan atas citra tepi (hasil dari deteksi tepi diatas) dengan citra aslinya. Dengan demikian bagian tepi obyek akan terlihat lebih berbeda dengan latarnya, sehingga citra terkesan lebih tajam. Disini banyaknya penambahan komponen citra tepi diatur dengan suatu nilai derajat penajaman. Sehingga dengan mengatur nilai derajat penajaman, tingkat ketajaman citra dapat disesuaiakan dengan keinginan kita.

## 7.3 Latihan Bab VII

1. Apa yang dimaksud operasi bertetangga?

2. Bagaimana melakukan operasi bertetangga dengan menggunakan operator kernel?

# Image Filtering

Bab VIII

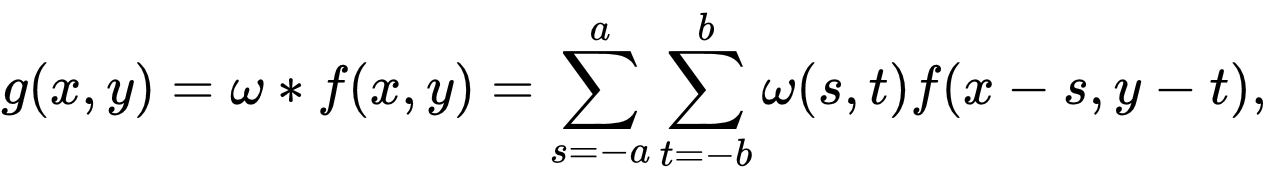
Capaian Mahasiswa:

* Mahasiswa mampu memahami konsep image filtering dan macama-macam image filtering.
* Mahasiswa mampu mengimplementasikan konsep image filtering ke dalam kode program.

Penyaringan citra(*Image Filtering)* digunakan untuk memodifikasi citra sehingga tercipta berbagai macam efek pada citra seperti pengaburan (*bluring),* penajaman (*sharpening),* efek denah (*outlining)* serta efek timbul (*embossing). Image filtering* juga biasa digunakan dalam proses ekstraksi fitur citra untuk mencari fitur tertentu atau menghilangkan fitur tertentu agar fokus pada fitur yang diinginkan. *Image filtering* merupakan operasi ketetanggaan dimana output nilai pada suatu piksel ditentukan dengan suatu algoritma yang mengikutkan piksel tetangga pada operasi perhitungannya.

Teknik *Image filtering* menggunakan suatu matriks dua dimensi sebagai filter yang biasa disebut *image kernel* atau biasa juga disebut dengan istilah jendela atau *mask.* *Image Kernel* tersebut diaplikasikan ke dalam citra dua dimensi menggunakan teknik konvolusi. Yakni dilakukan dengan cara menjumlahkan perkalian setiap piksel beserta tetangganya dengan nilai yang ada pada matrik filter yang bersesuaian. Dimana nilai yang berada pada bagian tengah filter matriks dikalikan dengan piksel yang saat ini diproses, sedengkan nilai lain dari filter matriks dikalikan dengan piksel tetangga yang sesuai.

*Image Filtering* secara umum diformulasikan:



Dimana *g(x,y)* adalah output citra yang telah diaplikasi *filtering,* *f(x,y)* adalah citra asli serta *w* adalah filter matriks.

Implementasi formula di atas dalam bahasa python adalah sebagai berikut:

|  |
| --- |
| **import** cv2 **import** numpy **as** np  m = cv2.imread(**'mangga.jpg'**) m2 = m.copy()  filter = [[-1, -1, -1],  [-1, 9, -1],  [-1, -1, -1]]   l = int(len(filter)/2) height, width, depth = np.shape(m)  **for** y **in** range(1, height - 1):  **for** x **in** range(1, width - 1):  b, g, r = 0, 0, 0   **for** filterY **in** range(0, len(filter)):  **for** filterx **in** range(0, len(filter[0])):  b += filter[filterY][filterx] \* m[y + filterY - l][x + filterx - l][0]  g += filter[filterY][filterx] \* m[y + filterY - l][x + filterx - l][1]  r += filter[filterY][filterx] \* m[y + filterY - l][x + filterx - l][2]   b = 0 **if** b < 0 **else** 255 **if** b > 255 **else** b  g = 0 **if** g < 0 **else** 255 **if** g > 255 **else** g  r = 0 **if** r < 0 **else** 255 **if** r > 255 **else** r  m2[y][x][0] = b  m2[y][x][1] = g  m2[y][x][2] = r  cv2.imshow(**'image'**, m2) cv2.waitKey(0) cv2.destroyAllWindows() |

Pada kode program di atas dilakukan penjumlahan hasil perkalian setiap piksel berikut tetangga terdekat dengan nilai yang ada pada matrik filter yang bersesuaian, dimana dilakukan untuk setiap channel warna. Selanjutnya dilakukan normalisasi nilai intensitas jika didapati nilai intensitas tersebut tidak diantara 0 dan 255.

Terdapat berbagai macam kernel yang umum digunakan dalam pemrosesan citra digital, antara lain sebagai berikut:

## 8.1 Blur Kernel

Blur kernel akan relatif mengurangi kesan kontras dalam nilai piksel yang berdekatan. Kernel yang biasa dipakai untuk blur kernel adalah sebagai berikut:

|  |  |  |
| --- | --- | --- |
| 0,0625 | 0,125 | 0,0625 |
| 0,125 | 0,25 | 0,125 |
| 0,0625 | 0,125 | 0,0625 |

Citra yang dihasilkan menggunakan blur kernel akan terkesan halus.



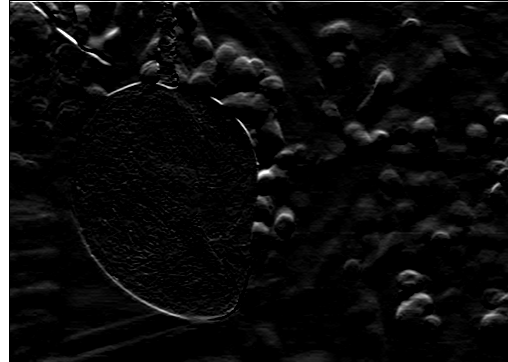
Gambar 8.1 Hasil blur kernel

## 8.2 Bottom Sobel Kernel

Kernel sobel digunakan untuk menunjukkan perbedaan nilai piksel yang berdekatan dalam arah bawah.

|  |  |  |
| --- | --- | --- |
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

Kernel di atas menghasilkan citra sebagai berikut:



Gambar 8.2 Hasil Bottom Sobel Kernel

## 8.3 Right Sobel Kernel

Kernel sobel digunakan untuk menunjukkan perbedaan nilai piksel yang berdekatan dalam arah kanan.

|  |  |  |
| --- | --- | --- |
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

Kernel di atas menghasilkan citra sebagai berikut:



Gambar 8.3 Hasil Right Sobel Kernel

## 8.4 Left Sobel

Kernel sobel digunakan untuk menunjukkan perbedaan nilai piksel yang berdekatan dalam arah kiri.

|  |  |  |
| --- | --- | --- |
| 1 | 0 | -1 |
| 2 | 0 | -2 |
| 1 | 0 | -1 |

Filter tersebut menghasilkan citra sebagai berikut:



Gambar 8.4 Hasil Left Sobel Kernel

## 8.5 Emboss Kernel

Kernel emboss memberikan ilusi kedalaman dengan menekankan perbedaan piksel dalam arah yang diberikan. Atau dikenel dengan istilah timbul.

|  |  |  |
| --- | --- | --- |
| -2 | -1 | 0 |
| -1 | 1 | 1 |
| 0 | 1 | 2 |

Contoh hasil filter emboss adalah sebagai berikut:



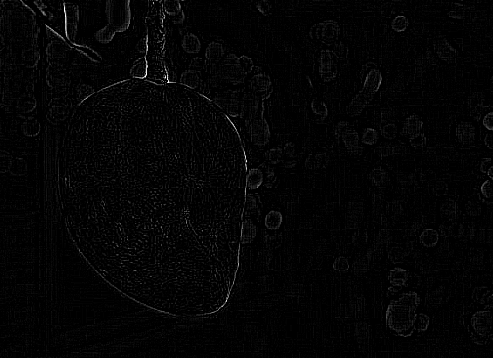
Gambar 8.5 Hasil Emboss Kernel

## 8.6 Outline Kernel

Outline Kernel (disebut juga kernel tepi) digunakan untuk menampakkan perbedaan nilai piksel yang cukup besar antar piksel. Sebuah piksel dengan piksel tetangganya yang memiliki nilai intensitas yang tidak terlalu jauh akan tampak hitam ,sementara piksel tetangga yang sangat berbeda akan tampak putih.

|  |  |  |
| --- | --- | --- |
| -1 | -1 | -1 |
| -1 | 8 | -1 |
| -1 | -1 | -1 |

Contoh hasil filter outline adalah sebagai berikut:



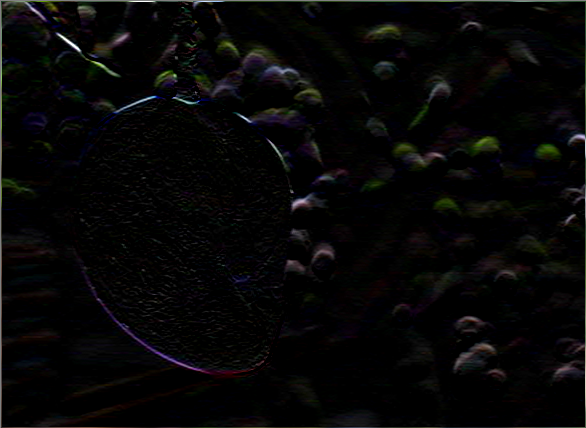
Gambar 8.6 Hasil Outline Kernel

## 8.7 Prewitt Kernel

Prewitt Kernel menggunakan matriks filter sebagai berikut:

|  |  |  |
| --- | --- | --- |
| -1 | -1 | -1 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

Menggunakan filter di atas menghasilkan gambar sebagai berikut:



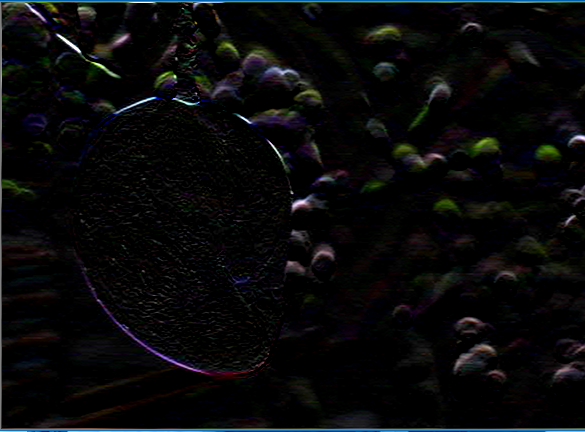
Gambar 8.7 Hasil Prewitt Kernel

## 8.8 Sobel Kernel

Sobel Kernel menggunakan matriks filter sebagai berikut:

|  |  |  |
| --- | --- | --- |
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

Menggunakan filter di atas menghasilkan gambar sebagai berikut:



Gambar 8.8 Hasil Sobel Kernel

## 8.9 Latihan Bab VIII

1. Mengapa konvolusi diperlukan dalam preprocessing pada pengolahan citra?

2. Aplikasikan image filtering untuk mendeteksi tepi obyek tertentu dalam sebuah citra.

# Daftar Pustaka

Rafael C. Gonzalez dan Richard E. Woods, 2007. *Digital Image Processing,* *third edition*, Prentice Hall.

Nick Efford, 2000. *Digital Image Processing: A Practical Introduction Using Java.* Pearson Education Limited

# Biografi Penulis

|  |  |
| --- | --- |
|  | DR. Eng. Cahya Rahmad, S.T., M.Kom., Lahir di Sumenep 2 Februari 1972, Mendapat gelar Sarjana Teknik dari Universitas Brawijaya 1998, gelar Magister Komputer dari ITS tahun 2005 dan gelar DR.Eng dari Saga University jepang tahun 2013. Menjadi dosen di Politeknik Negeri Malang dari tahun 2005 sampai sekarang. |
|  | M. Hasyim Ratsanjani, S.Kom., M.Kom. lahir di gresik 5 Maret 1990. Menyelesaikan Sarjana komputer di Universitas Islam Negeri Maulana Malik Ibrahim Malang pada tahun 2013 dan Magister Ilmu Komputer di Universitas Indonesia pada tahun 2016. Menjadi dosen di Politeknik Negeri Malang pada tahun 2019 hingga kini. |