**PRAKTIKUM PENGOLAHAN CITRA DIGITAL**



**Oleh:**

**KELOMPOK V**

* 1. Novar Setiawan (1504505023)
  2. I Made Yoga Mardia Wara H. (1504505039)
  3. Made Prastha Nugraha (1605551043)
  4. Ni Putu Ayu Widiari (1605551068)
  5. Tita Lattifia (1605552010)

**PROGRAM STUDI TEKNOLOGI INFORMASI**

**FAKULTAS TEKNIK**

**UNIVERSITAS UDAYANA**

**2018**

# KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa atas segala rahmat-Nya karena berkat beliaulah laporan akhir Praktikum Pengolahan Citra Digital ini dapat diselesaikan tepat pada waktunya. Tidak lupa juga penulis mengucapkan banyak terimakasih kepada.

1. Ibu Ni Kadek Ayu Wirdiani,S.T.,M.T selaku dosen pengampu mata kuliah Praktikum Pengolahan Citra Digital
2. Asisten dosen yang telah membimbing dalam melakukan praktikum dan penyusunan laporan Praktikum Pengolahan Citra Digital
3. Teman-teman dan semua pihak yang telah membantu memberikan ide atas terselesaikannya laporan Praktikum Pengolahan Citra Digital

Harapan penulis semoga laporan ini dapat menambah pengetahuan dan pengalaman bagi para pembaca, dan untuk ke depannya dapat memperbaiki bentuk maupun menambah isi dari laporan ini agar menjadi lebih baik lagi. Keterbatasan penulis dalam pengetahuan maupun pengalaman, penulis yakin masih banyak kekurangan dalam laporan ini, oleh karena itu penulis sangat mengharapkan saran dan kritik yang membangun dari para pembaca demi kesempurnaan laporan ini.

Jimbaran, 18 Mei 2018

Penulis

# ABSTRAK

Pengolahan citra digital adalah pemrosesan citra dengan menggunakan komputer digital untuk menghasilkan sebuah citra baru yang sesuai dengan keinginan pengguna. bidang ilmu yang membahas mengenai citra digital yang dikenal sebagai Pengolahan Citra (*Image* *Processing*). Secara umum, operasi-operasi dalam pengolahan citra dapat diklasifikasikan menjadi perbaikan citra, peningkatan kualitas citra, regristrasi citra, pemampatan data citra dan segmentasi citra. Salah satu implementasi dari pengolahan citra digital yang dibahas dalam praktikum pengolahan citr digital ini yaitu implementasi dasar pengolahan citra, perbaikan kualitas citra dengan menggunakan *Low Pass Filter*, *High Pass Filter* dan *Filter Median*, transformasi citra dengan menggunakan *discrete fourier transform*, *discrete cosine transform*, dan *fast fourier transform*, serta segmentasi dan morfologi pada citra.

Kata Kunci: Pengolahan Citra Digital, *image processing*.

# DAFTAR ISI

# DAFTAR GAMBAR

# DAFTAR TABEL

# DAFTAR KODE PROGRAM

# ****BAB I**** ****PENDAHULUAN****

BAB I membahas latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah dan sistematika yang digunakan dalam penulisan laporan akhir Praktikum Pengolahan Citra Digital.

## Latar Belakang

Pengolahan citra merupakan proses pengolahan dan analisis citra yang banyak melibatkan persepsi visual. Proses pengolahan citra ini memiliki ciri data masukan dan informasi keluaran yang berbentuk citra. Citra yang dihasilkan dari proses pengolahan citra ini memiliki kualitas yang lebih baik dibandingkan dengan citra pada aslinya. Istilah pengolahan citra digital pada umumnya, dapat didefinisikan sebagai pemrosesan citra dua dimensi dengan komputer [Renaldi, 2004]. Perkembangan pengolahan citra dalam perbaikan citra yang digunakan untuk mendeteksi objek sangat bervariasi, tergantung dari objek yang ingin diteliti. Perbaikan citra digital dengan menggunakan teknik komputer diawali pada tahun 1964, yaitu sebuah citra bulan yang berasal dari *Jet* *Propulsion Lab*, yang ditransmisikan oleh *Range*r-7 mulai tahun 1964, sampai sekarang.

Secara harfiah, citra merupakan gambar pada bidang dua dimensi, sedangkan ditinjau dari sudut pandang matematis, citra merupakan fungsi kontinu dari intensitas cahaya pada bidang dua dimensi. Citra dibedakan menjadi dua jenis, yaitu citra kontinu dan citra diskrit. Citra kontinu dihasilkan dari sistem optik yang menerima sinyal analog, misalnya mata manusia dan kamera analog. Citra diskrit dihasilkan melalui proses digitalisasi terhadap citra kontinu. Citra diskrit disebut juga citra digital. Citra digital merupakan suatu citra dengan kondisi ketika *x, y* dan nilai intensitas dari *f* adalah terbatas. Citra digital terdiri dari sejumlah elemen dan setiap elemen mempunyai lokasi dan nilai tertentu. Elemen-elemen ini disebut *picture elements, image elements, pels* atau *pixels.* Istilah untuk menyatakan elemen citra digital yang paling banyak digunakan saat ini adalah piksel.

Meskipun kaya akan informasi, seringkali citra digital mengalami penurunan mutu sehingga citra tersebut sulit diinterpretasikan dan informasi yang disampaikan menjadi berkurang, sehingga lahirlah suatu bidang ilmu yang membahas mengenai citra digital yang dikenal sebagai pengolahan citra (*Image* *Processing*). Pengolahan citra adalah pemrosesan citra, khususnya dengan menggunakan komputer, menjadi citra yang kualitasnya lebih baik. Operasi dalam pengolahan citra dapat diklasifikasikan menjadi perbaikan citra, peningkatan kualitas citra, regristrasi citra, pemampatan data citra dan segmentasi citra.

## Rumusan Masalah

Berdasarkan latar belakang tersebut, beberapa rumusan masalah yang dapat diambil antara lain sebagai berikut.

1. Bagaimana mengimplementasikan dasar pengolahan citra seperti membaca sebuah file citra (*image*) dan mengubah citra berwarna (RGB) ke ruang warna R, G, B, C, M dan Y menggunakan bahasa pemrograman VB dan Matlab pada modul 1 dasar pengolahan citra.
2. Bagaimana mengimplementasikan perbaikan kualitas citra seperti melakukan proses *filtering* dengan menggunakan *Low Pass Filter*, *High Pass Filter* dan *Filter Median* dan memahami perbandingan hasil antara bahasa pemrograman yang digunakan dengan Matlabpada modul 2 perbaikan kualitas citra.
3. Bagaimana mengimplementasikan transformasi citra seperti menganalisa perubahan citra 2 Dimensi ke ruang frekuensi dengan menggunakan DFT, DCT dan FFT serta perubahan citra 2 Dimensi ke ruang skala dan waktu menggunakan Transformasi *Wavelet* dengan bahasa pemrograman yang digunakan yaitu Matlab pada modul 3 transformasi citra.
4. Bagaimana mengimplementasikan pratikum pengolahan citra seperti mengolah struktur objek dalam citra sehingga bisa diperoleh struktur objek yang diinginkan, dapat melakukan proses dilasi, erosi, *opening* dan *closing* serta dapat menganalisa hasil yang diperoleh dengan menggunakan jenis struktur elemen yang berbeda untuk setiap proses morfologi dengan bahasa pemrograman yang digunakan yaitu Matlab pada modul 4 morfologi dan segmentasi citra.

## Tujuan Penelitian

Berdasarkan rumusan masalah yang telah dipaparkan sebelumnya, tujuan dari penulisan laporan Praktikum Pengolahan Citra Digital ini diantaranya adalah sebagai berikut.

1. Peserta dapat mengimplementasikan pratikum pengolahan citra seperti membaca sebuah file citra (*image*) menggunakan bahasa pemrograman VB, dan Matlab serta mengubah citra berwarna (RGB) ke ruang warna R, G, B, C, M dan Y.
2. Peserta dapat mengimplementasikan pratikum pengolahan citra dalam melakukan proses *filtering* dengan menggunakan *Low Pass Filter*, *High Pass Filter* dan *Filter Median* dan memahami perbandingan hasil antara bahasa pemrograman yang digunakan dengan Matlab pada modul 2 perbaikan kualitas citra.
3. Peserta mengerti cara mengimplementasikan pratikum pengolahan citra seperti menganalisa perubahan citra 2 Dimensi ke ruang frekuensi dengan menggunakan DFT, DCT dan FFT citra 2 Dimensi ke ruang skala dan waktu menggunakan Transformasi *Wavelet* dan bahasa pemrograman yang digunakan dengan Matlab pada modul 3 transformasi citra.
4. Peserta mengerti cara mengimplementasikan pratikum pengolahan citra seperti mengolah struktur objek dalam citra sehingga bisa diperoleh struktur objek yang diinginkan, dapat melakukan proses dilasi, erosi, *opening* dan *closing* serta dapat menganalisa hasil yang diperoleh dengan menggunakan jenis struktur elemen yang berbeda untuk setiap proses morfologi dengan bahasa pemrograman yang digunakan yaitu Matlab pada modul 4 morfologi dan segmentasi citra.

## Manfaat Penelitian

Manfaat yang didapat melalui Praktikum Pengolahan Cita Digital diantaranya adalah sebagai berikut.

1. Mengetahui dan memahami penerapan dasar pengolahan citra digital.
2. Mengetahui dan memahami perbaikan kualitas pada citra.
3. Mengetahui dan memahami penerapan transformasi pada citra
4. Mengetahui dan memahami penerapan morfologi dan segmentasi pada citra.

## Batasan Masalah

Mengingat pembahasan mengenai pengolahan citra digital sangatlah luas maka pada laporan ini akan dibatasi dalam lingkup pembuatan program menggunakan Visual Basic dan Matlab, dasar pengolahan citra, perbaikan kualitas citra, transformasi citra dan morfologi serta segmentasi citra.

## Sistematika Penulisan

Sistematika penulisan yang digunakan pada laporan Praktikum Pengolahan Citra Digital ini terdiri dari pembahasan umum secara teori kemudian dilengkapi dengan pemaparan contoh dari setiap penjelasan tersebut.

**Tabel 1. 1** Sistematika Penulisan

|  |  |  |
| --- | --- | --- |
| NO | BAB | Keterangan |
| 1 | BAB I PENDAHULUAN | Bab I membahas mengenai latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah, dan sistematika penulisan. |
| 2 | BAB II  TINJAUAN PUSTAKA | Bab II membahas mengenai landasan teori secara umum yang menjadikan dasar untuk mendukung pengerjaan laporan praktikum. |
| 3 | BAB III METODOLOGI PENELITIAN | Bab III membahas mengenai tempat dan waktu penelitian, sumber data, kode program praktikum pengolahan citra digital. |
| 4 | BAB IV PEMBAHASAN | Bab IV membahas mengenai hasil dan pembahasan dari pengerjaan praktikum dari empat modul yang ada pada mata kuliah praktikum pengolahan citra digital. |
| 5 | BAB V  PENUTUP | Bab V membahas mengenai kesimpulan dari keseluruhan modul praktikum serta kritik dan saran untuk penulis. |

Tabel 1.1 merupakan tabel yang menjelaskan isi dari setiap bab yang dibuat dalam laporan Praktikum Pengolahan Citra Digital.

# BAB II TINJAUAN PUSTAKA

BAB II berisikan bagian dari tinjauan pustaka yaitu pembahasan atau landasan teori mengenai dasar pengolahan citra digital dan perbaikan citra.

## Dasar Pengolahan Citra Digital

Pengolahan citra adalah pemrosesan citra, khususnya dengan menggunakan komputer, menjadi citra yang kualitasnya lebih baik. Teknik-teknik pengolahan citra mentransformasikan citra menjadi citra lain. Inputan pada proses ini adalah citra dan keluarannya juga berupa citra dengan kualitas lebih baik daripada citra inputan sebelumnya. Citra kaya informasi tidak menutup kemungkinan bahwa citra tersebut mengalami pernah penurunan mutu (degradasi), misalnya mengandung cacat atau derau (*noise*), warnanya terlalu kontras, kurang tajam, atau kabur (*blurring*). Citra ini menjadi lebih sulit diinterpretasi karena informasi yang disampaikan oleh citra tersebut menjadi berkurang. Agar citra yang mengalami gangguan mudah diinterpretasi (baik oleh manusia maupun mesin), maka citra tersebut perlu dimanipulasi menjadi citra lain yang kualitasnya lebih baik. Citra digital merupakan sebuah larik (*array*) yang direpresentasikan dalam bentuk deretan bit tertentu[1].

Pembentukan citra digital melalui beberapa tahapan yaitu akuisisi citra, *sampling* dan kuantisasi. Pada modul 1 bertujuan supaya praktikan mampu mempelajari sekaligus mengimplementasikan membaca dan mengopi *file* gambar kemudian mengubah ke ruang warna R, G, B, C, M dan Y.

### Akuisisi Citra

Proses akuisisi citra adalah pemetaan suatu pandangan (*scene*) menjadi citra kontinu dengan menggunakan sensor. Terdapat beberapa jenis sensor yang digunakan antara lain sensor aktif dan sensor pasif.

### Sampling

Digitasi atau *sampling* adalah proses membagi gambar secara horizontal dan vertikal menjadi bagian bagian yang kecil atau proses digitasi pada bidang spasial x dan y. (Munir, 2004, hal 19-21).

|  |
| --- |
|  |

**Gambar 2.1** Proses digitasi atau sampling

Gambar 2.1 menjelaskan bahwa bagian-bagian yang kecil atau elemen *array* disebut dengan *pixel*. Pembagian suatu citra menjadi sejumlah *pixel* dengan ukuran tertentu akan menentukan resolusi spasial yang diperoleh. Resolusi gambar akan semakin tinggi dan gambar akan semakin halus atau terang apabila ukuran *pixel* pada gambar semakin besar (makin banyak jumlah *pixel*), karena informasi yang hilang akibat pengelompokan tingkat keabuan atau warna ketika proses pembuatan kisi-kisi akan semakin kecil.

### Kuantisasi

Proses yang diperlukan selanjutnya yaitu proses kuantisasi. Tingkat keabuan setiap *pixel* dinyatakan dengan suatu bilangan bulat (*integer*) dalam proses kuantisasi. Batas-batas harga *integer* atau besarnya daerah tingkat keabuan yang digunakan untuk menyatakan suatu tingkat keabuan *pixel* akan menentukan resolusi kecerahan dari gambar yang akan diperoleh. Penggunaan tiga bit untuk menyimpan harga *integer* tersebut akan memberikan delapan tingkat keabuan. Gambar yang akan dihasilkan akan semakin baik apabila tingkat keabuan yang digunakan semakin baik, karena kontinuitas dari tingkat keabuan akan semakin tinggi sehingga mendekati citra aslinya.

|  |
| --- |
|  |

**Gambar 2.2** Citra Kontinu (Analog) Di-sampling Oleh Array Sensor Kuantisasi

Gambar 2.2 menunjukkan contoh suatu gambar yang didigitalisasi dengan ukuran *sampling* yang masih cukup besar, sehingga tepian gambar akan berbentuk kasar (kotak-kotak).

### Format Ruang Warna

Nilai intensitas piksel berupa bilangan bulat pada interval tertentu bergantung pada tingkat kuantisasinya. Proses transformasi dari gambar analog ke gambar digital disebut dengan digitisasi dimana energi warna pada gambar analog dikuantifikasikan dengan nilai berupa angka pada interval tertentu. Elemen nilai warna terdiri dari tiga nilai yang menunjukkan gabungan antara nilai merah (*red*), hijau (*green*) dan biru (*blue*) atau dikenal dengan nilai RGB. Proses pengolahan citra difokuskan pada proses manipulasi nilai-nilai angka tersebut untuk menghasilkan nilai baru yang tampilan visualnya sesuai dengan yang diinginkan. Citra berdasarkan komposisi pikselnya dapat dikelompokkan menjadi tiga yaitu citra warna, citra *graylevel* dan citra biner. Citra warna dibangun menggunakan tiga *channel* warna yaitu *Red*, *Green* dan *Blue* dan secara umum tiap *channel* warna memiliki nilai interval sebesar 8 bit yaitu antara 0 sampai dengan 255 (256 nilai intensitas). Citra *graylevel* dapat dianggap sebagai citra dengan satu *channel* warna dengan interval yang sama yaitu 0 sampai dengan 255, sedangkan citra biner adalah citra dengan dua nilai intensitas, misalnya 0 dan 255.

### Memuat dan Menampilkan Gambar

Program dapat bekerja secara fleksibel apabila gambar yang diinginkan dapat diambil (*loading*) dari tempat penyimpanan dengan menggunakan elemen *OpenFile Dialog*.

### Mengakses *Pixel* Citra Digital Secara Global

Representasi internal citra digital semata-mata berupa kumpulan angka-angka dalam tiga kelompok (*layer*, *channel*) yang secara matematis dapat disebut sebagai matriks citra dua dimensi. Cara pengaksesan secara pemrograman menggunakan fasilitas *looping* yaitu *for* yang bergerak sepanjang baris dan kolom gambar. Elemen yang digunakan untuk mengambil nilai warna adalah dengan *GetPixel* dan memberikan nilai warna dengan *SetPixel*.

## Perbaikan Kualitas Citra

Citra direpresentasikan dalam bentuk matrik. Citra disimpan menggunakan format berbeda seperti bmp, png, gif, jpg atau format lainnya. Citra yang diperoleh menggunakan kamera atau *device* lain tidak selalu menghasilkan citra berkualitas karena kualitas citra disebabkan oleh faktor pencahayaan, kondisi objek maupun kamera yang digunakan, sehingga perlu perbaikan kualitas citra menggunakan beberapa metode diantaranya sebagai berikut.

### Operasi Titik

Operasi Titik dalam *image enhancement* dilakukan dengan memodifikasi histogram citra masukan agar sesuai dengan karakteristik yang diharapkan. Histogram dari suatu citra adalah grafik yang menunjukkan distribusi frekuensi dari nilai intensitas piksel dalam citra tersebut.

* + - 1. Mengubah Citra Berwarna ke *Grayscale*

Peningkatan kualitas citra dapat dilakukan melalui transformasi intensitas citra, yaitu besar intensitas setiap piksel pada citra diubah, tetapi dengan posisi piksel yang tetap. Transformasi ini dapat dilakukan dengan menggunakan fungsi transformasi skala keabuan atau *Gray-Scale Transformation Function*.

|  |
| --- |
|  |

**Gambar 2.3** Formula transformasi skala keabuan

Gambar 2.3 merupakan formula transformasi skala keabuan yang akan digunakan untuk meningkatkan kualitas pada citra.

* + - 1. *Thresholding*

*Thresholding* merupakan proses pemisahan piksel-piksel berdasarkan derajat keabuan yang dimilikinya. Piksel yang memiliki derajat keabuan lebih kecil dari nilai batas yang ditentukan akan diberikan nilai 0, sementara piksel yang memiliki derajat keabuan yang lebih besar dari batas akan diubah menjadi bernilai 1.

* + - 1. Operasi Titik Penambahan dan Pengurangan intensitas

Operasi titik dilakukan dengan penambahan dan pengurangan intensitas dilakukan dengan melakukan manipulasi nilai *gray* citra pada setiap titik. Tingkat kecerahan suatu citra dapat dilihat dari histogramnya dimana semakin tinggi tingkat kecerahan suatu citra maka konsentrasi nilai piksel pada histogram akan bergeser ke sisi kanan, sebaliknya semakin rendah tingkat kecerahan suatu citra maka konsentrasi nilai piksel pada histogram akan bergeser ke sisi kiri. Tingkat kecerahan nilai warna pada citra dapat dikontrol sesuai dengan yang diinginkan, dengan menggunakan fungsi sebagai berikut.

|  |
| --- |
| U’= U+c |

**Gambar 2.4** Formula Intensitas

Gambar 2.4 merupakan formula intensitas. *U’* adalah citra setelah operasi, *U* adalah citra sebelum operasi dan *c* adalah konstanta penyesuaian.

### Operasi Negasi

Operasi negasi digunakan untuk mendapatkan citra negatif. Negasi adalah proses pemetaan nilai piksel suatu citra. Pada citra biner, piksel hitam dijadikan putih dan piksel putih dijadikan hitam sedangkan pada citra *grayscale*, nilai maksimum piksel dikurangi dengan nilai piksel yang sedang diproses yang secara matematis dapat ditulis sebagai berikut.

|  |
| --- |
| U’= 2l – U |

**Gambar 2.5** Formula Negasi

Gambar 2.5 merupakan formula negasi pada operasi citra.*U’* adalah citra setelah operasi, *U* adalah citra sebelum operasi, *l* adalah nilai bit dari *gray* *level*.

### Operasi Spasial

Operasi spasial dalam pengolahan citra digital dilakukan melalui penggunaan suatu kernel konvolusi 2-dimensi. Teknik *enhancement* berdasarkan operasi titik dibagi tiga, yaitu sebagai berikut.

#### *Low Pass Filter*

*Low-pass ﬁltering* merupakan metode penghalusan sebuah sinyal atau citra. *Smoothing/blurring*/penghalusan dicapai dalam domain frekuensi dengan pelemahan frekuensi tinggi. *Smoothing* dapat membantu menghilangkan *noise*, karena *noise/interference*disebabkan oleh frekuensi tinggi.

#### *High Pass Filter*

*High-pass ﬁltering* merupakan kebalikan dari *low-pass filtering*, yaitu metode yang membuat sebuah sinyal atau citra menjadi kurang halus. Metode yang digunakan adalah melakukan pelemahan dalam domain frekuensi yang memiliki frekuensi rendah. *highpass filtering* biasa digunakan untuk *Unsharp Masking, Deconvolution, Edge Detection*, mengurangi *blur*, atau menambah *noise*. (Morse, 2010).

#### *Median* Filter

Filter median sangat bermanfaat untuk menghilangkan *outliers*, yaitu nilai-nilai piksel yang ekstrim. Filter median menggunakan *sliding neighborhood* untuk memproses suatu citra, yaitu suatu operasi dimana filter ini akan menentukan nilai masing-masing piksel keluaran dengan memeriksa tetangga m×n di sekitar piksel masukan yang bersangkutan. Filter median mengatur nilai-nilai piksel dalam satu tetangga dan memilih nilai tengah atau median sebagai hasil. Metode ini biasanya digunakan untuk menghilangkan *noise* pada citra.

## Transformasi Citra

Secara harfiah, transformasi citra dapat diartikan sebagai perubahan bentuk suatu citra. Perubahan bentuk tersebut dapat berupa perubahan geometri *pixel* seperti perputaran (rotasi), pergeseran (translasi), perskalaan, atau dapat juga berupa perubahan ruang (domain) citra ke domain lainnya. Metode yang digunakan untuk melakukan tranformasi citra, dua diantaranya adalah Transformasi *Fourier* dan Transformasi *Wavelet*.

### Transformasi Fourier

Transformasi *fourier* adalah sebuah transformasi integral yang menyatakan kembali sebuah fungsi dalam fungsi basis sinusoidal yakni sebuah fungsi sinusoidal penjumlahan atau integral dikalikan oleh beberapa koefisien (amplitudo). Hal ini dikemukakan oleh Joseph Fourier. Terdapat variasi yang berhubungan dengan transformasi yang tergantung pada jenis fungsi yang ditransformasikan. Transformasi *fourier* bekerja dengan mendekomposisi sinyal ke bentuk fungsi eksponensial dari frekuensi yang berbeda-beda dengan cara definisikan kedalam dua persamaan berikut [2].

|  |
| --- |
|  |

**Gambar 2.6** Formula Transformasi Fourier

Gambar 2.6 formula transformasi *fourier*, t adalah waktu dan f adalah frekuensi. x merupakan notasi sinyal dalam ruang waktu dan X adalah notasi untuk sinyal dalam domain frekuensi. Persamaan (1) disebut Transformasi *Fourier* dari x(t) sedangkan persamaan (2) disebut *Inverse* Transformasi *Fourier* dari X(f), yakni x(t).

|  |
| --- |
|  |

**Gambar 2.7** Noise yang muncul pada spektrum Fourier

Gambar 2.7 merupakan *noise* yang muncul pada spektrum *fourier*. Transformasi *fourier* dapat menangkap informasi apakah suatu sinyal memiliki frekuensi tertentu atau tidak, tapi tidak dapat menangkap dimana frekuensi itu terjadi.

* + - 1. Discrete *Fourier* Transform

*Discrete Fourier Transform* (DFT) adalah prosedur yang digunakan dalam pemrosesan sinyal digital dan filterisasi digital.  DFT menungkinkan untuk analisa, manipulasi dan sintesis sinyal yang tidak mungkin dapat dilakukan dalam pemrosesan sinyal analog. (Richards G. Lyons, 2004)

* + - 1. *Discrete Cosine Transform*

*Discrete Cosine Transform* (DCT) biasa digunakan untuk mengubah sebuah sinyal menjadi komponen frekuensi dasarnya. DCT pertama kali diperkenalkan oleh Ahmed, Natarajan dan Rao pada tahun 1974 dalam makalahnya yang berjudul “*On image processing and a discrete cosine transform*” (Watson, 1994).

* + - 1. Fast *Fourier* Transform

*Fast fourier transform* merupakan algoritma penghitungan yang mengurangi kompleksitas FT biasa dari N menjadi N log N saja. Pada implementasinya, FFT merupakan cara yang umum digunakan untuk menghitung FT diskret. *Inverse* FT juga dapat dihitung dengan kompleksitas N log2 N (IFFT). Di Matlab : fft(x) atau fft2(X) untuk FT dan ifft(x) atau ifft2(X) untuk *inverse* FT.

### Transformasi Wavelet

*Wavelet* merupakan klas dari suatu fungsi yang digunakan untuk melokalisasi suatu fungsi dalam ruang dan skala. *Wavelet* dapat dibentuk dari satu fungsi Ψ(x) , dikenal sebagai *mother* *wavelet* dalam suatu interval berhingga. *Daughter* *Wavelet* dengan fungsi Ψa,b (x) dibentuk oleh translasi (b) dan kontraksi (a). *Wavelet* sangat berguna untuk kompresi data *image*, karena transformasi *wavelet* mempunyai property yang ada dalam beberapa cara terhadap transformasi *fourier* konvensional.

*Discrete Wavelet Transform* (DWT) adalah salah satu transformasi *Wavelet* yang merepresentasikan sinyal dalam domain waktu dan frekuensi. DWT memiliki keunggulan diantaranya mudah diimplementasikan dan efisien dalam hal waktu komputasi.

|  |
| --- |
|  |

**Gambar 2.8** (a) Komponen aproksimasi (Skala tinggi, komponen sinyal ber-frekuensi

rendah (LPF)), komponen Detail (Skala rendah, frekuensi tinggi (HPF). (b) Proses

filterisasi dan down sampling.

Gambar 2.8(a) Komponen aproksimasi (Skala tinggi, komponen sinyal ber-frekuensi rendah (LPF)), komponen Detail (Skala rendah, frekuensi tinggi (HPF). Gambar 2.8(b) Proses filterisasi dan *down sampling*. Analisis sinyal dengan DWT dilakukan pada frekuensi yang berbeda dengan resolusi yang berbeda pula dengan mendekomposisi sinyal menjadi komponen detail dan komponen aproksimasi. Pada transformasi ini terjadi filterisasi dan *down* *sampling*, yaitu pengurangan koefesien pada fungsi genap.

Prinsip dasar dari DWT adalah bagaimana cara mendapatkan representasi waktu dan skala dari sebuah sinyal menggunakan teknik pemfilteran digital dan operasi sub-*sampling* (Reza, 2013). DWT biasanya digunakan untuk menghitung koefisien *Wavelet* di segala skala yang memungkinkan. DWT ini juga menghasilkan jumlah data yang sangat besar. Untuk menaksirkan sinyal hasil transformasi ini dibentuk filter *Low Pass* dan *High Pass*.

|  |
| --- |
|  |

**Gambar 2.9** Proses Multi-level decomposition

Gambar 2.9 merupakan proses *Multi-level decomposition*. Pada tahap pertama, sinyal (S) dilewatkan pada rangkaian *high pass filter* dan *low pass filter*, kemudian setengah dari masing-masing keluaran diambil sebagai sampel melalui operasi sub-*sampling*. Proses ini disebut sebagai proses dekomposisi satu tingkat. Keluaran dari *low pass filter* digunakan sebagai masukan diproses dekomposisi tingkat berikutnya. Proses ini diulang sampai tingkat proses dekomposisi yang diinginkan. Gabungan dari keluaran-keluaran *high pass filter* dan *low pass* yang terakhir, disebut sebagai koefisien *wavelet*, yang berisi informasi sinyal hasil transformasi yang telah terkompresi.

Pada transformasi DWT terdapat proses pengembalian kembali komponen-komponen yang telah kita gunakan. *Invers Discrete Wavelet Transform* (IDWT) merupakan kebalikan dari *Discrete Wavelet Transform* (DWT). Pada transformasi ini dilakukan proses rekonstruksi sinyal, yaitu mengembalikan komponen frekuensi menjadi komponen sinyal semula. Transformasi dilakukan dengan proses *up* sampling dan pemfilteran dengan koefisien filter *invers*. Sehingga dalam satu sistem transformasi w*avelet* menggunakan empat macam filter, yaitu *low pass filter* dan *high pass filter* dekomposisi, dan *low pass filter* dan *high pass filter* rekonstruksi.

## Morfologi dan Segmentasi Citra

Morfologi merupakan teknik pengolahan citra berdasarkan bentuk segmen citra yang bertujuan untuk memperbaiki hasil segmentasi. Teknik morfologi biasanya digunakan pada citra biner atau untuk beberapa kasus juga bisa diterapkan pada citra keabuan (*grayscale*).

Segmentasi adalah proses mempartisi citra digital menjadi beberapa segmen (*set* piksel, juga dikenal sebagai *superpixels*). Tujuan dari segmentasi adalah untuk menyederhanakan dan atau mengubah penyajian gambar ke sesuatu yang lebih bermakna dan lebih mudah untuk menganalisis[4]. Berikut merupakan penjelasan mengenai morfologi dan segmentasi citra.

### Morfologi

Morfologi merupakan teknik pengolahan citra digital dengan menggunakan bentuk (*shape*) sebagai pedoman dalam pengolahan. Nilai dari setiap *pixel* dalam citra digital hasil diperoleh melalui proses perbandingan antara *pixel* yang bersesuaian pada citra digital masukan dengan *pixel* tetangganya. Operasi morfologi bergantung pada urutan kemunculan dari *pixel*, tidak memperhatikan nilai *numeric* dari *pixel* sehingga teknik morfologi sesuai apabila digunakan untuk melakukan pengolahan *binary image* dan *grayscale image*.

|  |
| --- |
|  |

**Gambar 2.10** Formula translasi citra

Gambar 2.10 merupakan formula translasi citra. Diasumsikan A adalah himpunan *pixel* pada citra biner dan w = (x,y) adalah koordinat suatu titik.Kemudian Aw adalah himpunan A ditranslasi / digeser pada arah (x,y). Himpunan A digeser pada arah x dan y sesuai dengan nilai w yang diberikan.

|  |
| --- |
|  |

**Gambar 2.11** Formula translasi citra

Gambar 2.11 merupakan formula refleksi citra. A merupakan himpunan *pixel* dan kemudian dilakukan refleksi atau pencerminan dengan notasi *À* seperti gambar X.

1. Dilasi

Dilasi adalah operasi morfologi yang akan menambahkan *pixel* pada batas antar objek dalam suatu citra digital. Dilasi merupakan suatu proses menambahkan piksel pada batasan dari objek dalam suatu *image* sehingga nantinya apabila dilakukan operasi ini maka *image* hasilnya lebih besar ukurannya dibandingkan dengan *image* aslinya.

|  |
| --- |
|  |

**Gambar 2.12** Hasil image yang telah didilasi

Gambar 2.12 merupakan hasil *image* setelah dilakukan proses dilasi dari *Original Image* dengan *Stucturing Element* yang telah ditentukan. Warna hijau merupakan *pixel* tambahan setelah dilakukan Dilasi.

1. Erosi

   Erosi merupakan kebalikkan dari Dilasi. Proses ini akan membuat ukuran sebuah citra menjadi lebih kecil. Berbeda dengan dilatasi, apabila erosi dilakukan maka yang dikerjakan adalah memindahkan piksel pada batasan-batasan objek yang akan di erosi. Jumlah dari pikselyang ditambah atau dihilangkan bergantung pada ukuran dan bentuk dari *structuring element* yang digunakan untuk memproses *image* tersebut.

|  |
| --- |
|  |

**Gambar 2.13** Hasil image yang telah di erosi

Gambar 2.13 merupakan hasil *image* setelah dilakukan proses Erosi dari *Original Image* dengan *Stucturing Element* yang telah ditentukan. Warna hijau merupakan *pixel* dihilangkan setelah dilakukan proses Erosi, sehingga *pixel* hasilnya hanya yang berwarna hitam.

1. *Opening*

*Opening* merupakan kombinasi proses dimana suatu citra digital dikenai operasi erosi dilanjutkan dengan dilasi. Operasi *opening* pada citra mempunyai efek memperhalus batas-batas objek, memisahkan objek-objek yang sebelumnya bergandengan, dan menghilangkan objek-objek yang lebih kecil daripada ukuran *structuring*.

|  |
| --- |
|  |

**Gambar 2.14** hasil image setelah dilakukan proses Opening

Gambar 2.14 merupakan hasil *image* setelah dilakukan proses *Opening* dari *Original Image* dengan *Stucturing Element* yang telah ditentukan. Proses yang dilakukan untuk mendapatkan *opening* adalah dengan cara melakukan operasi erosi, lalu hasil erosi tersebut di dilasi lagi.

1. *Closing*

*Closing* merupakan kombinasi dimana suatu citra dikenai operasi dilasi dilanjutkan dengan erosi. Operasi *closing* juga cenderung akan memperhalus objek pada citra, namun dengan cara menyambung pecahan-pecahan (*fuses narrow breaks and thin gulf*) dan menghilangkan lubang-lubang kecil pada objek.

|  |
| --- |
|  |

**Gambar 2.15** Hasil Image Setelah Dilakukan Proses Closing

Gambar 2.15 merupakan hasil *image* setelah dilakukan proses *closing* dari *Original Image* dengan *Stucturing Element* yang telah ditentukan. Proses yang dilakukan untuk mendapatkan *Closing* adalah dengan cara melakukan operasi dilasi, lalu hasil dilasi tersebut dilakukan proses erosi lagi.

1. *Thinning*

*Thinning* merupakan suatu proses penting sebelum melakukan proses-proses atau operasi-operasi pengolahan citra, seperti dalam proses pengenalan karakter optic, pengenalan sidik jari, pemrosesan teks, dan lain sebagainya. Tujuannya adalah mengurangi bagian yang tidak perlu (*redundant*) sehingga dihasilkan informasi yang esensial saja[4] .

### Segmentasi

Segmentasi adalah proses pembagian sebuah citra ke dalam sejumlah bagian atau objek (Gonzalez dan Wintz, 1987). Segmentasi citra mempunyai arti membagi suatu citra menjadi wilayah-wilayah yang homogen berdasarkan kriteria keserupaan yang tertentu antara tingkat keabuan suatu piksel dengan tingkat keabuan piksel-piksel tetangganya, kemudian hasil dari proses segmentasi ini akan digunakan untuk proses tingkat tinggi lebih lanjut yang dapat dilakukan terhadap suatu citra, misalnya proses klasifikasi citra dan proses identifikasi objek

Segmentasi citra memiliki dua pendekatan utama, yaitu didasarkan pada tepi (*edge-based*) dan didasarkan pada wilayah (*region-based*). Segmentasi didasarkan pada tepi membagi citra berdasarkan diskontinuitas di antara sub-wilayah (*sub-region*), sedangkan segmentasi yang didasarkan pada wilayah bekerjanya berdasarkan keseragaman yang ada pada sub-wilayah tersebut. Dalam *computer vision*, segmentasi mengacu pada proses pembagian citra digital ke dalam *multiple region* (himpunan *pixel*). Tujuan akhir dari segmentasi adalah menyederhanakan dan atau merubah representasi suatu citra ke dalam gambaran yang lebih mempunyai arti dan lebih mudah untuk di analisa. Segmentasi citra secara khusus digunakan untuk melokalisasi objek atau batas dalam citra. Hasil dari segmentasi citra adalah sekumpulan wilayah yang melingkupi citra tersebut, atau sekumpulan kontur yang diekstrak dari citra (pada deteksi tepi).

|  |
| --- |
|  |

**Gambar 2.16** Contoh segmentasi

Gambar 2.16 merupakan contoh segmentasi pada citra. Tiap piksel dalam suatu wilayah mempunyai kesamaan karakteristik atau propeti yang dapat dihitung (*computed property*), seperti warna (*color*), intensitas (*intensity*),dan tekstur (*texture*).

Segmentasi wilayah merupakan pendekatan lanjutan dari deteksi tepi. Dalam deteksi tepi segmentasi citra dilakukan melalui identifikasi batas-batas objek (*boundaries of object*). Batas merupakan lokasi dimana terjadi perubahan intensitas. Dalam pendekatan didasarkan pada wilayah, maka identifikasi dilakukan melalui wilayah yang terdapat dalam objek tersebut. Adapun beberapa teknik atau algoritma yang termasuk dalam segmentasi citra yaitu, metode pengelompokkan (*Clustering Methods*), *Histogram-Based Methods*, *Edge Detection Methods*, *Region Growing* *Methods*, *Level Set Methods*, *Graph Partitioning Methods*, *Watershed Transformation*, *Model based segmentation*, *Multi-scale segmentation* dan *Semi-automatic segmentation*.

1. Region

Metodologi berbasis transformasi *region* (wilayah) dibagi menjadi *Region Growing*, *Region Splitting*, *Region Merging*, segmentasi berbasis *Region Growing*. Contoh segmentasi *region growing* yaitu transformasi *watershed*.

1. Deteksi Tepi

Deteksi tepi (*edge detection*) adalah suatu proses yang menghasilkan tepi-tepi dari obyek-obyek citra, tujuannya adalah untuk memperbaiki detail dari citra yang kabur, yang terjadi karena *error* atau adanya efek dari proses akuisisi citra. Suatu titik (x,y) dikatakan sebagai tepi (*edge*) dari suatu citra bila titik tersebut mempunyai perbedaan yang tinggi dengan tetangganya (Ramdhani dan Murinto, 2013). Deteksi tepi merupakan salah satu operasi dasar dari pemrosesan citra. Tepi merupakan batas dari suatu objek. Pada proses klasifikasi citra, deteksi tepi sangat diperlukan sebelum pemrosesan segmentasi citra. Batas objek suatu citra dapat dideteksi dari perbedaan tingkat keabuannya (Purnomo dan Muntasa, 2010). Tepi atau sisi dari sebuah obyek adalah daerah di mana terdapat perubahan intensitas warna yang cukup tinggi. Proses deteksi tepi (*edge detection*) akan melakukan konversi terhadap daerah ini menjadi dua macam nilai yaitu intensitas warna rendah atau tinggi, contoh bernilai nol atau satu. Deteksi tepi akan menghasilkan nilai tinggi apabila ditemukan tepi dan nilai rendah jika sebaliknya (Lusiana, 2013). Deteksi tepi banyak dipakai untuk mengidentifikasi suatu objek dalam sebuah gambar. Tujuan dari deteksi tepi adalah untuk menandai bagian yang menjadi detail citra dan memperbaiki detail citra yang kabur karena adanya kerusakan atau efek akuisisi data. Dalam citra, sebagian besar informasi terletak pada batas antara dua daerah yang berbeda (Yulianto dkk,2009). Pelacakan tepi merupakan operasi untuk menemukan perubahan intensitas lokal yang berbeda dalam sebuah citra. Gradien adalah hasil pengukuran perubahan dalam sebuah fungsi intensitas, dan sebuah citra dapat dipandang sebagai kumpulan beberapa fungsi intensitas kontinu sebuah citra. Perubahan mendadak pada nilai intensitas dalam suatu citra dapat dilacak menggunakan perkiraan diskrit pada gradien. Gradien disini adalah kesamaan dua dimensi dari turunan pertama dan didefinisikan sebagai vektor (Lusiana, 2013). Oleh karena itu teknik deteksi tepi sering digunakan sebagai dasar teknik segmentasi untuk proses segmentasi yang lain.

Beberapa metode yang terkenal dan banyak digunakan untuk pendektesian tepi di dalam citra, yaitu operator *Robert*, operator *Prewitt* dan operator *Sobel*. Metode *Sobel* paling banyak digunakan sebagai pelacak tepi karena kesederhananaan dan keampuhannya (Munir, 2004). Kelebihan dari metode ini adalah kemampuan untuk mengurangi *noise* sebelum melakukan perhitungan deteksi tepi. Masing-masing metode deteksi memiliki sub metode yang cukup banyak, tetapi metode deteksi citra yang baik adalah metode yang dapat mengeliminasi derau (*noise*) yang semaksimal mungkin (Ballard dkk, 1982). Deteksi tepi operator *Sobel* diperkenalkan oleh Irwin Sobel pada tahun 1970. Operator ini identik dengan bentuk matriks 3x3 atau jendela ukuran 3x3 piksel (Lusiana, 2013). Operator *Sobel* melakukan perhitungan secara 2D terhadap suatu ruang di dalam sebuah citra. Operator ini biasanya digunakan untuk mencari gradien dari masing-masing piksel citra input yang telah dikonversi ke *grayscale* sebelumnya Operator *Sobel* terdiri dari matriks 3x3 masing-masing adalah Gx dan Gy. Matriks *mask* tersebut dirancang untuk memberikan respon secara maksimal terhadap tepi objek baik horizontal maupun vertikal. *Mask* dapat diaplikasikan secara terpisah terhadap input citra. Operator *Sobel* menggunakan kernel operator gradien 3 x 3, dengan koefisien yang telah ditentukan dan dapat dinyatakan sebagai berikut (Munir, 2004).

|  |
| --- |
|  |

**Gambar 2.17** Operator Sobel menggunakan kernel operator gradien 3 x 3

Gambar 2.17 merupakan Operator *Sobel* menggunakan kernel operator *gradien* 3x3. Kernel di atas dirancang untuk menyelesaikan permasalahan deteksi tepi baik secara vertikal maupun horizontal. Penggunaan kernel-kernel ini dapat digunakan bersamaan ataupun secara tepisah (Purnomo dan Muntasa, 2010).

|  |
| --- |
|  |

**Gambar 2.18** Nilai magnitude

Gambar 2.18 merupakan nilai *magnitude*. Nilai maksimum dari operator *Sobel* didapatkan dengan menghitung kekuatan tepi citra terhadap warna kecerahannya dengan cara mencari nilai magnitude yang dapat dihitung dengan persamaan pada gambar 2.18 (Munir 2004).

1. *Laplacian of Gaussian*

Operator *Laplacian of Gaussian* (LOG) merupakan deteksi tepi turunan kedua. Operator *Laplace* mendeteksi lokasi tepi khususnya pada citra tepi yang curam, berbentuk *Omny-directional* (tidak horisontal-tidak vertikal) menangkap tepian di semua arah dan tajam.

|  |
| --- |
|  |

**Gambar 2.19** Kernel konvolusi

Gambar 2.19 merupakan gambar kernel konvolusi, dengan berbagai macam pembobotan, kernel *Laplacian* tersebut dapat dimodifikasi menjadi beberapa kernel konvolusi seperti gambar 2.19.

1. Operator *Canny*

Operator *Canny* merupakan salah satu algoritma deteksi tepi modern adalah deteksi tepi dengan menggunakan metode *Canny* yang dapat mendeteksi tepian sebenarnya dengan tingkat kesalahan minimum.

Algoritma dari operator *canny* adalah sebagai berikut. Implementasi filter *Gaussian* untuk menghilangkan *noise*, melakukan salah satu operator deteksi tepi, yang melakukan pencarian secara horisontal Gx dan secara vertikal Gy, menentukan arah tepian dengan rumus Ѳ= arctg(Gy/Gx), membagi ke dalam 4 warna sehingga garis dengan arah berbeda dan memiliki warna yang berbeda, memperkecil garis tepi yang muncul dengan menerapkan *nonmaximum* *suppression* sehingga menghasilkan garis tepian yang lebih ramping dan langkah terakhir adalah binerisasi dengan menerapkan dua buah *tresholding*.

# BAB III METODE DAN PERANCANGAN SISTEM

Bab III merupakan bagian dari metode dan perancangan sistem, yang berisikan tempat dan waktu penelitian, gambaran umum sistem, dan alur aplikasi yang telah dibuat.

## 3.1 Tempat dan Waktu Penelitian

Penelitian dilakukan di sekitar lingkungan Fakultas Teknik Universitas Udayana, khususnya jurusan Teknologi Informasi, di mana penelitian ini dilakukan selama tiga bulan, yaitu dari bulan Maret hingga Mei.

**Tabel 3.1** Jadwal Praktikum Setiap Modul

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **No** | **Modul Praktikum** | | **Waktu Pelaksanaan Praktikum** | | | | | | | | | | | | | **Maret** | | | | **April** | | | | **Mei** | | | | | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** | | 1 | Dasar Pengolahan Citra | |  |  |  |  |  |  |  |  |  |  |  |  | | 2 | Perbaikan Kualitas Citra | |  |  |  |  |  |  |  |  |  |  |  |  | | 3 | Transformasi Citra | |  |  |  |  |  |  |  |  |  |  |  |  | | 4 | Morfologi dan Segmentasi Citra | |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |

Tabel 3.1 merupakan tabel jadwal rentangan waktu praktikum setiap modul praktikum pengolahan citra digital.

## 3.2 Gambaran Umum Sistem

Gambaran umum sistem laporan Praktikum Pengolahan Citra Digitalmerupakan gambaran umum sistem modul 1 hingga modul 4 yang menjelaskan jalannya sistem secara umum.

### Dasar Pengolahan Citra

*Use* *case* diagram mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan aplikasi yang akan dibuat. Berikut merupakan *use case* diagram modul 1 dasar pengolahan citra pada visual basic dan matlab.

|  |
| --- |
|  |

**Gambar 3.1** Use Case Diagram Modul 1

Gambar 3.1 merupakan *use case* *diagram* modul 1 dasar pengolahan citra. *User* dapat melakukan buka *file* dan *exit. User* dapat merubah citra kedalam warna *red*, *green*, *blue*, *cyan*, *magenta*, dan *yellow* dengan membuka *file* dan *copy* gambar terlebih dahulu maka citra akan dirubah ke warna sesuai dengan yang diinginkan oleh *user*.

### Perbaikan Kualitas Citra

*Use* *case* *diagram* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan aplikasi yang akan dibuat. Berikut merupakan *use case* *diagram* modul 2 perbaikan kualitas citra pada visual basic dan matlab.

|  |
| --- |
|  |

**Gambar 3.2** Use Case Diagram Modul 2 Visual Basic

Gambar 3.2 merupakan *use case* diagram modul 2visual basic. *User* dapat melakukan *open* *file* dan *exit*, jika user ingin merubah gambar asli ke *grayscale* dan *binary* maka *user* terlebih dahulu harus membuka *file* gambar yang ingin dimasukkan. *User* dapat mengubah gambar ke *binary* setelah gambar sebelumnya dirubah ke *grayscale*.

|  |
| --- |
|  |

**Gambar 3.3** Use Case Diagram Modul 2 Matlab

Gambar 3.3 merupakan *use case diagram* modul 2 pada Matlab. *User* dapat melakukan buka *file* dan *exit*, jika *user* ingin merubah gambar asli ke *grayscale* dan *binary* maka *user* terlebih dahulu harus membuka file gambar yang ingin dimasukkan. *User* dapat menambahkan *brightness* pada file gambar yang telah dimasukkan. *User* dapat mengubah gambar ke *binary* setelah gambar sebelumnya dirubah ke *grayscale*, dan *user* dapat menambahkan *noise* pada citra, setelah gambar ditambahkan *noise* maka *user* dapat merubah citra ke dalam LPF, HPF, dan median dengan kernel 3x3, kernel 5x5, kernel 7x7, dan kernel 9x9.

### Transformasi Citra

*Use* *case* *diagram* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan aplikasi yang akan dibuat. Berikut merupakan *use case* *diagram* modul 3 Transformasi Citra pada Matlab.

|  |
| --- |
|  |

**Gambar 3.4** Use Case Diagram Modul 3

Gambar 3.4 merupakan *use case diagram* modul 3 pada Matlab. *User* dapat melakukan *browse picture* dan *exit*. *User* dapat mengubah gambar ke DCT, FFT, dan *Wavelet* setelah *user* memasukkan *file* gambar terlebih dahulu, jika *user* ingin melakukan *inverse* DCT maka *user* terlebih dahulu harus melakukan DCT. *User* dapat melakukan *inverse* DFT setelah melakukan DFT. *User* dapat melakukan *inverse* FFT setelah melakukan FFT dan *inverse Wavelet* dapat dilakukan setelah *user* melakukan *Wavelet* terlebih dahulu.

### Morfologi dan Segmentasi Citra

*Use* *case* *diagram* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan aplikasi yang akan dibuat. Berikut merupakan *use case* *diagram* modul 4 morfologi dan segmentasi citra pada Matlab.

|  |
| --- |
|  |

**Gambar 3.5** Use Case Diagram Modul 4

Gambar 3.5 merupakan *use case diagram* modul 4 pada Matlab. Gambar 3.5 merupakan *use case* *diagram* modul 4 pada Matlab. *User* dapat melakukan *browse picture* dan *exit*. *User* dapat melakukan perubahan gambar dilasi, erosi, *opening, closing, thinning*, *sobel filter*, *robert filter*, *prewitt filter*, *canny*, *log*, *region growing* dan *watershed* dengan terlebih dahulu memasukkan *file* gambar. *Opening* dapat dilakukan jika *user* terlebih dahulu melakukan proses erosi dan dilasi. *Closing* dapat dilakukan jika *user* terlebih dahulu melakukan proses dilasi dan erosi, lalu selanjutnya *user* dapat melakukan proses *thinning*. Gambar akan dirubah ke dalam *sobel horizontal*, *vertical* dan *sobel gradient magnitude* jika *user* melakukan *sobel filter.* Gambar akan dirubah kedalam *robert horizontal*, *robert vertical*, dan *robert gradient magnitude* ketika *user* melakukan *robert filter*. Gambar otomatis dirubah ke dalam *prewitt* *vertical*, *horizontal* dan *prewitt gradient magnitude* jika *user* ingin merubah gambar kedalam *prewitt filter*. *User* juga dapat merubah gambar ke *cany*, *laplacian of gaussian*, *region* *growing* dan *watershed*.

# BAB IV HASIL DAN PEMBAHASAN

Bab IV berisikan hasil pembahasan praktikum Pengolahan Citra Digital yang terdiri dari Dasar Pengolahan Citra, Perbaikan Kualitas Citra, Transformasi Citra, morfologi dan segmentasi citra.

## Dasar Pengolahan Citra

Modul I merupakan modul yang berisikan pembahasan mengenai dasar pengolahan citra untuk mengubah citra warna ke ruang warna *Red*, *Green*, *Blue* dan *Cyan*, *Magenta*, *Yellow* serta melakukan analisis perbandingan citra menggunakan program Visual Basic dan Matlab.

### *Source Code* Modul 1

Berikut merupakan kode program untuk menyalin citra (*image*) dan kode program untuk mengubah citra berwarna (RGB) ke dalam ruang warna R, G, B, C, M, Y menggunakan bahasa pemrograman Visual Basic.

|  |
| --- |
| Public Class Form1  Private Property Tot\_Pix As Object  Private Sub Open*Image*(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles open\_btn.Click  ofd.InitialDirectory = "c:\"  ofd.Filter = "*Image* files (\*.bmp, \*.jpg,\*.png)|\*.bmp;\*.jpg;\*.png"  If ofd.ShowDialog() = DialogResult.OK Then    PictureBox1.*Image* = New Bitmap(ofd.FileName)    PictureBox1.SizeMode = PictureBoxSizeMode.Stretch*Image*  PictureBox2.*Image* = Nothing  PictureBox3.*Image* = Nothing  End If  End Sub |

**Kode Program 4.1**  Sintak memasukan gambar pada Picture Box 1

Kode program 4.1 merupakan sintaks untuk memasukkan gambar pada *picture box* 1. Citra akan di *load* ke dalam *picture box* 1 jika f*ile* yang telah dimasukkan *valid* dan user menekan tombol ok. Mode untuk menampilkan ukuran citra pada *picture* *box* adalah *stretch*, dimana ukuran tidak sesuai dengan citra asli, hanya berupa tampilan sehingga saat citra diambil dari *picture* *box*, *user* tetap mendapatkan citra dengan ukuran yang asli.

|  |
| --- |
| Private Sub Copy*Image*(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles copy\_btn.Click  If PictureBox1.*Image* Is Nothing Then  MessageBox.Show("Please open the *image* first!", "Warning",  MessageBoxButtons.OK, MessageBoxIcon.Error)  Else  Dim source As New Bitmap(PictureBox1.*Image*)  Dim red, green, blue As Integer  For y As Integer = 0 To source.Height - 1  For x As Integer = 0 To source.Width - 1  red = source.Get*Pixel*(x, y).R 'mengambil nilai x dan y berwarna merah  green = source.Get*Pixel*(x, y).G 'mengambil nilai x dan y berwarna hijau  blue = source.Get*Pixel*(x, y).B 'mengambil nilai x dan y berwarna biru  source.Set*Pixel*(x, y, Color.FromArgb(red, green, blue)) 'set *pixel* x dan y berdasarkan warna rgb pada gambar yang dibuka  Next  Next  PictureBox2.*Image* = source 'picturebox3 diisi hasil variabel source  PictureBox2.SizeMode = PictureBoxSizeMode.Stretch*Image* 'untuk menyesuaikan gambar dengan ukuran picturebox  End If  End S  Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load  End Sub |

**Kode Program 4.2** Set Pixel X dan Y Berdasarkan Warna RGB

Kode Program 4.2 merupakan sintaks untuk membuat variabel dengan tipe data *bitmap* pada *picturebox* 1 dan membuat variabel *red*, *green*, *blue* dengan tipe data *integer*, menghitung nilai y (tinggi) serta menghitung nilai x (lebar) dan mengambil nilai x dan y berwarna merah, hijau, biru lalu set *pixel* x dan y berdasarkan warna rgb pada gambar yang dibuka dan menyesuaikan gambar yang telah dibuka dengan ukuran *picture* *box*.

|  |
| --- |
| private Sub ChangedToRed(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles red\_rb.CheckedChanged  If PictureBox2.*Image* Is Nothing Then  MessageBox.Show("Please copy the *image* first!", "Warning",  MessageBoxButtons.OK, MessageBoxIcon.Error)  Else  Dim source As New Bitmap(PictureBox1.*Image*)  Dim red, green, blue As Integer  For y As Integer = 0 To source.Height - 1  For x As Integer = 0 To source.Width - 1  red = source.Get*Pixel*(x, y).R  green = source.Get*Pixel*(x, y).G  blue = source.Get*Pixel*(x, y).B  source.Set*Pixel*(x, y, Color.FromArgb(red, 0, 0))  Next  Next  PictureBox3.*Image* = source  PictureBox3.SizeMode = PictureBoxSizeMode.Stretch*Image*  End If  End Sub |

**Kode Program 4.3** Sintaks Set Pixel X dan Y Berdasarkan Warna Merah

Kode Program 4.3 merupakan sintaks untuk *set* nilai warna *pixel* (x,y) dengan hanya nilai warna merah atau *red* pada gambar yang dibuka. Setpixel (red, 0, 0) nilai *green* dan *blue* bernilai 0.

|  |
| --- |
| Private Sub ChangeToGreen(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles green\_rb.CheckedChanged  If PictureBox2.*Image* Is Nothing Then  MessageBox.Show("Please copy the *image* first!", "Warning",  MessageBoxButtons.OK, MessageBoxIcon.Error)  Else  Dim source As New Bitmap(PictureBox1.*Image*)  Dim red, green, blue As Integer  For y As Integer = 0 To source.Height - 1  For x As Integer = 0 To source.Width - 1  red = source.Get*Pixel*(x, y).R  green = source.Get*Pixel*(x, y).G  blue = source.Get*Pixel*(x, y).B  source.Set*Pixel*(x, y, Color.FromArgb(0, green, 0))  Next  Next  PictureBox3.*Image* = source  PictureBox3.SizeMode = PictureBoxSizeMode.Stretch*Image*  End If  End Sub |

**Kode Program 4.4** Sintaks Set Pixel X dan Y Berdasarkan Warna Hijau

Kode Program 4.4 merupakan sintaks untuk *set* nilai warna *pixel* (x,y) dengan hanya nilai warna hijau atau *green*  pada gambar yang dibuka. *Setpixel* (0, *green*, 0) nilai *red* dan *blue* bernilai 0.

|  |
| --- |
| Private Sub ChangedToBlue(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles blue\_rb.CheckedChanged  If PictureBox2.*Image* Is Nothing Then  MessageBox.Show("Please copy the *image* first!", "Warning",  MessageBoxButtons.OK, MessageBoxIcon.Error)  Else  Dim source As New Bitmap(PictureBox1.*Image*)  Dim red, green, blue As Integer  For y As Integer = 0 To source.Height - 1  For x As Integer = 0 To source.Width - 1  red = source.Get*Pixel*(x, y).R  green = source.Get*Pixel*(x, y).G  blue = source.Get*Pixel*(x, y).B  source.Set*Pixel*(x, y, Color.FromArgb(50, 0, 0, blue)) 'set *pixel* x dan y berdasarkan warna biru pada gambar yang dibuka  Next  Next  PictureBox3.*Image* = source  PictureBox3.SizeMode = PictureBoxSizeMode.Stretch*Image*  End If  End Sub |

**Kode Program 4.5** Sintaks Set Pixel X dan Y Berdasarkan Warna Biru

Kode Program 4.5 merupakan sintaks untuk *set* nilai warna *pixel* (x,y) dengan hanya nilai warna biru atau *blue* pada gambar yang dibuka. *Setpixel* (0, 0, *blue*) nilai *red* dan *green* bernilai 0.

|  |
| --- |
| Private Sub ChangedToCyan(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cyan\_rb.CheckedChanged  If PictureBox2.*Image* Is Nothing Then  MessageBox.Show("Please copy the *image* first!", "Warning",  MessageBoxButtons.OK, MessageBoxIcon.Error)  Else  Dim source As New Bitmap(PictureBox1.*Image*)  Dim red, green, blue As Integer  For y As Integer = 0 To source.Height - 1  For x As Integer = 0 To source.Width - 1  red = source.Get*Pixel*(x, y).R  green = source.Get*Pixel*(x, y).G  blue = source.Get*Pixel*(x, y).B  source.Set*Pixel*(x, y, Color.FromArgb(0, green, blue)) 'set *pixel* x dan y berdasarkan warna hijau dan biru pada gambar yang dibuka  Next  Next  PictureBox3.*Image* = source  PictureBox3.SizeMode = PictureBoxSizeMode.Stretch*Image*  End If  End Sub |

**Kode Program 4.6** Sintaks Set Pixel X dan Y berdasaran Warna Hijau dan Biru

Kode Program 4.6 merupakan sintaks untuk *set* nilai warna *pixel* (x,y) dengan hanya nilai warna biru atau *blue* dan warna hijaupada gambar yang dibuka. Dimana setpixel (0, green, blue) nilai *red* bernilai 0.

|  |
| --- |
| Private Sub ChangedToMagenta(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles magenta\_rb.CheckedChanged  If PictureBox2.*Image* Is Nothing Then  MessageBox.Show("Please copy the *image* first!", "Warning",  MessageBoxButtons.OK, MessageBoxIcon.Error)  Else  Dim source As New Bitmap(PictureBox1.*Image*)  Dim red, green, blue As Integer  For y As Integer = 0 To source.Height - 1  For x As Integer = 0 To source.Width - 1  red = source.Get*Pixel*(x, y).R  green = source.Get*Pixel*(x, y).G  blue = source.Get*Pixel*(x, y).B  source.Set*Pixel*(x, y, Color.FromArgb(red, 0, blue))  Next  Next  PictureBox3.*Image* = source  PictureBox3.SizeMode = PictureBoxSizeMode.Stretch*Image*  End If  End Sub |

**Kode Program 4.7** Sintaks Set Pixel X dan Y Berdasarkan Warna Merah dan Biru

Kode Program 4.7 merupakan sintaks untuk *set* nilai warna *pixel* (x,y) berdasarkan warna magenta (campuran merah dan biru) pada gambar yang dibuka, sehinga nilai *green* di-set 0 (set*pixel* (red, 0, blue)).

|  |
| --- |
| Private Sub ChangedToCyan(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cyan\_rb.CheckedChanged  If PictureBox2.*Image* Is Nothing Then  MessageBox.Show("Please copy the *image* first!", "Warning",  MessageBoxButtons.OK, MessageBoxIcon.Error)  Else  Dim source As New Bitmap(PictureBox1.*Image*)  Dim red, green, blue As Integer  For y As Integer = 0 To source.Height - 1  For x As Integer = 0 To source.Width - 1  red = source.Get*Pixel*(x, y).R  green = source.Get*Pixel*(x, y).G  blue = source.Get*Pixel*(x, y).B  source.Set*Pixel*(x, y, Color.FromArgb(0, green, blue)) 'set *pixel* x dan y berdasarkan warna hijau dan biru pada gambar yang dibuka    Next  Next  PictureBox3.*Image* = source  PictureBox3.SizeMode = PictureBoxSizeMode.Stretch*Image*    End If  End Sub |

**Kode Program 4.8** Sintaks Set Pixel X dan Y Berdasarkan Warna Biru dan Hijau

Kode Program 4.8 merupakan sintaks untuk *set* nilai warna *pixel* (x,y) berdasarkan warna hijau dan biru pada gambar yang dibuka, sehingga nilai *red* di-*set* 0 (set*pixel* (0, hijau, biru)).

|  |
| --- |
| Private Sub ChangedToYellow(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles yellow\_rb.CheckedChanged  If PictureBox2.*Image* Is Nothing Then  MessageBox.Show("Please copy the *image* first!", "Warning",  MessageBoxButtons.OK, MessageBoxIcon.Error)  Else  Dim source As New Bitmap(PictureBox1.*Image*)  Dim red, green, blue As Integer  For y As Integer = 0 To source.Height - 1  For x As Integer = 0 To source.Width - 1  red = source.Get*Pixel*(x, y).R  green = source.Get*Pixel*(x, y).G  blue = source.Get*Pixel*(x, y).B  source.Set*Pixel*(x, y, Color.FromArgb(red, green, 0)) 'set *pixel* x dan y berdasarkan warna merah dan hijau pada gambar yang dibuka  Next  Next  PictureBox3.*Image* = source  PictureBox3.SizeMode = PictureBoxSizeMode.Stretch*Image*  End If  End Sub  Private Sub Label1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Label1.Click  End Sub  Private Sub ExitProgram(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles reset\_btn.Click  Application.Exit()  End Sub  Private Sub Button4\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)  End Sub  Private Sub PictureBox1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles PictureBox1.Click  End Sub  Private Sub PictureBox3\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles PictureBox3.Click  End Sub  Private Sub Label3\_Click(sender As Object, e As EventArgs) Handles Label3.Click  End Sub  End Class |

**Kode Program 4.9** Sintaks Set Pixel X dan Y Berdasarkan Warna Merah dan Hijau

Kode Program 4.9 merupakan sintaks untuk *set* *pixel* x dan y berdasarkan warna hijau dan merah pada gambar yang dibuka, sehingga nilai *blue* di-*set* 0 (*setpixel* (merah, hijau, 0)).

### Langkah-Langkah Percobaan

Langkah-langkah percobaan praktikum untuk program mengubah gambar ke ruang warna R, G, B serta C, M, Y yang ditampilkan dalam *PictureBox* dengan menggunakan aplikasi *Visual Basic* adalah sebagai berikut.

**Gambar 4.1** Validasi program Visual Basic

|  |
| --- |
|  |

Gambar 4.1 merupakan validasi program Visual Basic. Validasi ini bertujuan agar *user* menekan *button* Buka *File* terlebih dahulu baru dilanjutkan dengan pemrosesan data berupa penyalinan gambar. Validasi ini menghindari adanya kesalahan dalam melakukan pemrosesan data. Validasi berupa peringatan untuk membuka gambar terlebih dahulu di *button* Buka File.

|  |
| --- |
|  |

**Gambar 4.2** Validasi penyalinan gambar pada Visual Basic

Gambar 4.2 merupakan validasi penyalinan gambar pada Visual Basic. Validasi ini bertujuan agar gambar yang telah diinputkan pada *pictureBox* 1 disalin terlebih dahulu pada *PictureBox* 2. Validasi ini menghindari adanya kesalahan dalam melakukan pemrosesan data. Apabila *user* tidak menyalin gambar terlebih dahulu, maka program akan memunculkan validasi berupa peringatan untuk mengkopi gambar terlebih dahulu di *button* *Copy* Gambar.

|  |
| --- |
|  |

**Gambar 4.3** Validasi pada Program Matlab

Gambar 4.3 merupakan validasi penyalinan gambar pada Matlab. Validasi ini bertujuan agar pemrosesan gambar pertama diinputkan dari *PictureBox* 1 yang berfungsi untuk membuka *file* dengan menekan *button* Buka *File*. Apabila *user* tidak menginputkan gambar melalui *pictureBox* 1, maka program akan memunculkan validasi berupa peringatan untuk membuka gambar terlebih dahulu di *button* Buka *File*.

|  |
| --- |
| C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\vb1.jpg |

**Gambar 4.4** Tampilan gambar hasil copy per pixel

Gambar 4.4 merupakan gambar hasil *copy* per *pixel* dari gambar asli. Gambar hasil *copy* per *pixel* ini didapatkan setelah mengklik *button Copy* Gambar dan ditampilkan pada *Picturebox*2.

|  |
| --- |
| vb2 |

**Gambar 4.5** Perubahan citra asli ke ruang warna R (Red)

Gambar 4.5 merupakan perubahan citra asli ke ruang warna R (*Red*) yang ditampilkan pada *picturebox* 3. Program menampilkan *radiobutton* yang dipilih adalah *Red*, maka program mengubah citra ke ruang warna *Red*.

|  |
| --- |
| vb3 |

**Gambar 4.6** Perubahan citra asli ke ruang warna G (Green)

Gambar 4.6 merupakan perubahan citra asli ke ruang warna G (*Green*) yang ditampilkan pada *picturebox* 3. Program menampilkan *radiobutton* yang dipilih adalah *Green*, maka program mengubah citra ke ruang warna *Green*.

|  |
| --- |
| vb4 |

**Gambar 4.7** Perubahan citra asli ke ruang warna B (Blue)

Gambar 4.7 merupakan perubahan citra asli ke ruang warna B (*Blue*) yang ditampilkan pada *picturebox* 3. Program menampilkan *radiobutton* yang dipilih adalah *Blue*, maka program mengubah citra ke ruang warna *Blue*.

|  |
| --- |
| vb5 |

**Gambar 4.8** Perubahan citra asli ke ruang warna C (Cyan)

Gambar 4.8 merupakan perubahan citra asli ke ruang warna C (Cyan) yang ditampilkan pada picturebox 3. Program menampilkan radiobutton yang dipilih adalah Cyan, maka program mengubah citra ke ruang warna Cyan. Cyan merupakan hasil perpaduan warna antara Green dan Blue.

|  |
| --- |
| vb6 |

**Gambar 4.9** Perubahan citra asli ke ruang warna M (Magenta)

Gambar 4.9 merupakan perubahan citra asli ke ruang warna M (Magenta) yang ditampilkan pada picturebox 3. Program menampilkan radiobutton yang dipilih adalah Magenta, maka program mengubah citra ke ruang warna Magenta. Magenta merupakan hasil perpaduan warna antara Red dan Blue.

|  |
| --- |
| vb7 |

**Gambar 4.10** Perubahan citra asli ke ruang warna Y (Yellow)

Gambar 4.10 merupakan perubahan citra asli ke ruang warna Y (Yellow) yang ditampilkan pada picturebox 3. Program menampilkan radiobutton yang dipilih adalah Yellow, maka program mengubah citra ke ruang warna Yellow. Yellow merupakan hasil perpaduan warna antara Red dan Green.

|  |
| --- |
| matlab1 |

**Gambar 4.11** Tampilan gambar hasil copy per pixel

Gambar 4.11 merupakan gambar hasil *copy* per *pixel* dari gambar asli. Gambar hasil *copy* per *pixel* ini didapatkan setelah mengklik *button Copy*dan ditampilkan pada *Picturebox*2.

|  |
| --- |
| C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\vb2.jpg |

**Gambar 4.12** Perubahan citra asli ke ruang warna R (Red) pada Matlab

Gambar 4.12 merupakan perubahan citra asli ke ruang warna R (*Red*) yang ditampilkan pada *picturebox* 3. Program menampilkan *radiobutton* yang dipilih adalah *Red*, maka program mengubah citra ke ruang warna *Red* pada Matlab.

|  |
| --- |
| matlab3 |

**Gambar 4.13** Perubahan citra asli ke ruang warna G (Green) pada Matlab

Gambar 4.13 merupakan perubahan citra asli ke ruang warna G (*Green*) yang ditampilkan pada *picturebox* 3. Program menampilkan *radiobutton* yang dipilih adalah *Green*, maka program mengubah citra ke ruang warna *Green* pada Matlab.

|  |
| --- |
| matlab4 |

**Gambar 4.14** Perubahan citra asli ke ruang warna B (Blue) pada Matlab

Gambar 4.14 merupakan perubahan citra asli ke ruang warna B (*Blue*) yang ditampilkan pada *picturebox* 3. Program menampilkan *radiobutton* yang dipilih adalah *Blue*, maka program mengubah citra ke ruang warna *Blue* pada Matlab.

|  |
| --- |
| matlab5 |

**Gambar 4.15** Perubahan citra asli ke ruang warna C (Cyan) pada Matlab

Gambar 4.15 merupakan perubahan citra asli ke ruang warna C (Cyan) yang ditampilkan pada picturebox 3. Program menampilkan radiobutton yang dipilih adalah Cyan, maka program mengubah citra ke ruang warna Cyan. Cyan merupakan hasil perpaduan warna antara Green dan Blue pada Matlab.

|  |
| --- |
| matlab6 |

**Gambar 4.16** Perubahan citra asli ke ruang warna M (Magenta) pada Matlab

Gambar 4.16 merupakan perubahan citra asli ke ruang warna M (Magenta) yang ditampilkan pada picturebox 3. Program menampilkan radiobutton yang dipilih adalah Magenta, maka program mengubah citra ke ruang warna Magenta. Magenta merupakan hasil perpaduan warna antara Red dan Blue pada Matlab.

|  |
| --- |
| matlab7 |

**Gambar 4.17** Perubahan citra asli ke ruang warna Y (Yellow) pada Matlab

Gambar 4.17 merupakan perubahan citra asli ke ruang warna Y (Yellow) yang ditampilkan pada picturebox 3. Program menampilkan radiobutton yang dipilih adalah Yellow, maka program mengubah citra ke ruang warna Yellow. Yellow merupakan hasil perpaduan warna antara Red dan Green pada Matlab.

### Analisis Perbandingan Citra

Berikut merupakan analisis perbandingan citra dari program Visual Basic dan Matlab. Perbandingan yang digunakan adalah perbandingan format citra berbeda, perbandingan resolusi citra, perbandingan ukuran citra, dan perbandingan kontras citra.

|  |
| --- |
|  |

**Gambar 4.18** Perbandingan format citra (JPEG) di Visual Basic

Gambar 4.18 merupakan gambar wanita dengan memakai baju bergaris-garis putih yang berlatar hitam dengan format JPEG. Citra yang dibuka pada program Visual Basic mengalami kompresi citra, namun kompresi citra yang dihasilkan tidak menurunkan kualitas citra secara signifikan.

|  |
| --- |
|  |

**Gambar 4.19** Perbandingan format citra (JPEG) di Matlab

Gambar 4.19 merupakan gambar wanita dengan memakai baju bergaris-garis putih yang berlatar hitam dengan format JPEG. Citra yang dibuka pada program Matlab akan mengalami kompresi citra dan mengalami penurunan kualitas citra. Gambar baju dengan motif garis putih terlihat pecah-pecah dan kotak-kotak.

|  |
| --- |
| vb jpeg kontras modul 1 |

**Gambar 4.20** Perbandingan Kontras Citra di Visual Basic

Gambar 4.20 merupakan gambar panorama laut untuk perbandingan kontras citra pada Visual Basic. Citra yang dihasilkan pada program Visual Basic mengalami peningkatan kontras, namun peningkatan kontras yang dialami oleh citra pada program Visual Basic tidak setajam citra pada program Matlab. Peningkatan kontras citra pada Visual Basic tidak terlalu signifikan.

|  |
| --- |
| matlab jpeg kontras modul 1 |

**Gambar 4.21** Perbandingan Kontras Citra di Matlab

Gambar 4.21 merupakan gambar panorama laut untuk perbandingan kontras citra pada Matlab. Citra yang dihasilkan pada program Matlab mengalami peningkatan kontras sehingga gambar menjadi lebih terang.

|  |
| --- |
| C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\VB_JPG_Resolusi buruk.jpg |

**Gambar 4.22** Perbandingan Resolusi Citra di Visual Basic

Gambar 4.22 merupakan gambar Soekarno yang memiliki citra resolusi rendah. Gambar Soekarno yang tampil dalam program Visual Basic memiliki resolusi citra dengan ukuran 27 x 30 *pixel*. Kerapatan titik yang ditampilkan pada gambar Soekarno adalah rendah karena gambar Soekarno tidak jelas dan tidak tajam.

|  |
| --- |
| C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Mathlab_JPG_Resolusi buruk.jpg |

**Gambar 4.23** Perbandingan Resolusi Citra di Matlab

Gambar 4.23 merupakan gambar Soekarno yang memiliki citra resolusi rendah. Gambar Soekarno yang tampil dalam program Matlab memiliki resolusi citra dengan ukuran 27 x 30 pixel. Program Matlab akan mengkompresi citra menjadi ukuran yang lebih kecil, sehingga gambar Soekarno yang memiliki resolusi rendah akan memiliki resolusi yang lebih rendah dari gambar sebelumnya. Gambar 4.23 menunjukkan resolusi citra yang ditampilkan pada Matlab lebih buruk dibandingkan dengan resolusi citra pada gambar 4.23 di Visual Basic.

|  |
| --- |
| vb tif modul 1 |

**Gambar 4.24** Perbandingan Ukuran (size) Citra di Visual Basic

Gambar 4.24 merupakan gambar peta dengan *format* citra TIFF. *Format* TIFF merupakan format dengan ukuran besar karena dalam *file* ini gambar tidak dikompresi. Gambar peta yang ditunjukkan pada gambar 4.24 menggunakan program Visual Basic mengalami kompresi karena ukuran gambar tidak menyerupai ukuran aslinya. Artinya, dalam program Visual Basic citra TIFF akan dikompresi untuk menyesuaikan ukuran *pictureBox* yang telah diatur sebelumnya dalam program.

|  |
| --- |
| matlab tif modul 1 |

**Gambar 4.25** Perbandingan Ukuran (size) Citra di Matlab

Gambar 4.25 merupakan gambar peta dengan *format* citra TIFF. Gambar peta yang ditunjukkan pada gambar 4.25 menggunakan program Matlab mengalami kompresi karena ukuran gambar tidak menyerupai ukuran aslinya. Ukuran gambar diperkecil dan secara otomatis kualitas gambar menurun. Dibandingkan dengan gambar pada program Visual Basic, gambar pada program Matlab mengalami kompresi ukuran yang lebih besar sehingga gambar yang dihasilkan lebih kecil dan kualitas gambar yang dihasilkan juga menurun.

|  |
| --- |
|  |

**Gambar 4.26** Perbandingan citra format PNG di Visual Basic

Gambar 4.26 merupakan gambar logo Universitas Udayana dengan format PNG. *Background* pada citra saat dibuka dengan Visual Basic menjadi transparan karena *background* PNG tidak memiliki nilai pada *pixel* tersebut. Saat melakukan penyalinan gambar per *pixel*, *pixel* yang tidak ada nilai tersebut *default* di *set* 0 yang akan berwarna hitam.

|  |
| --- |
|  |

**Gambar 4.27** Perbandingan citra format PNG di Matlab

Gambar 4.27 merupakan gambar logo Universitas Udayana dengan format PNG. *Background* pada citra saat dibuka dengan Matlab berwarna hitam karena matlab berbasis matriks sehingga pada saat membuka *file*, maka matlab akan langsung membaca matriks pada suatu citra. *Background* citra transparan atau berformat PNG tidak memiliki nilai maka akan dianggap sebagai 0 atau berwarna hitam.

|  |
| --- |
|  |

**Gambar 4.28** Percobaan Alfa

Gambar 4.28 merupakan gambar logo Universitas Udayana yang ditambahkan nilai Alfa sebesar 50 (jika tidak diisi alfa maka akan *default* 255. Alfa berguna untuk mengatur tingkat *opacity* pada gambar. *Setting* alfa akan berwarna keputih-putihan jika alfa citra tersebut dibawah 255.

## Perbaikan Kualitas Citra

Modul II merupakan modul yang berisikan pembahasan mengenai perbaikan kualitas citra pengolahan citra berbasis titik seperti mengubah dari citra berwarna ke *grayscale*, mengubah citra berwarna ke biner, mengatur kecerahan (*brightness*), mengubah ke citra negasi.

### *Source Code* Modul 2

Berikut merupakan kode program modul 2 perbaikan kualitas citra yang dibuat dengan menggunakan bahasa pemrograman Visual Basic dan MATLAB.

|  |
| --- |
| Public Class Form1  Dim r, g, b As Integer 'membuat variabel a,r,g,b dengan tipe integer  Dim brightness As Integer  Dim valid As Integer  Private Sub Open\_File(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click  Dim ofd As New OpenFileDialog  ofd.InitialDirectory = "c:\"  ofd.Filter = "*Image* files (\*.bmp, \*.jpg)|\*.bmp;\*.jpg;\*.png"  If ofd.ShowDialog() = DialogResult.OK Then  PictureBox1.*Image* = New Bitmap(ofd.FileName)  PictureBox1.SizeMode = PictureBoxSizeMode.Stretch*Image*  PictureBox2.*Image* = Nothing  PictureBox3.*Image* = Nothing  PictureBox4.*Image* = New Bitmap(ofd.FileName)  PictureBox4.SizeMode = PictureBoxSizeMode.Stretch*Image*  'rightValue1.Value = 0  End If  End Sub |

**Kode Program 4.10** Sintaks Memasukkan Gambar Picturebox 1 Pada Modul 2

Kode program 4.10 merupakan sintaks untuk memasukkan gambar pada *picture box 1*. Mode untuk menampilkan ukuran citra pada *picture* *box* dengan *stretch* dimana ukuran tidak sesuai dengan citra asli, hanya berupa tampilan sehingga saat citra diambil dari *picture* *box*, *user* tetap mendapatkan citra dengan ukuran yang asli.

|  |
| --- |
| Private Sub Grayscale(sender As Object, e As EventArgs) Handles Button2.Click  If PictureBox1.*Image* Is Nothing Then  MessageBox.Show("Please open the *image* first!", "Warning",  MessageBoxButtons.OK, MessageBoxIcon.Error)  Else  Dim source As New Bitmap(PictureBox1.*Image*)  Dim red, green, blue, gray As Integer  For y As Integer = 0 To source.Height - 1  For x As Integer = 0 To source.Width - 1  red = source.Get*Pixel*(x, y).R  green = source.Get*Pixel*(x, y).G  blue = source.Get*Pixel*(x, y).B  gray = (red + green + blue) / 3  source.Set*Pixel*(x, y, Color.FromArgb(gray, gray, gray))  Next  Next  PictureBox2.*Image* = source    PictureBox2.SizeMode = PictureBoxSizeMode.Stretch*Image*  End If  End Sub |

**Kode Program 4.11** Sintaks Grayscale

Kode Program 4.11 merupakan sintaks untuk mengubah citra ke *grayscale*. Program membuat variabel *red*, *green*, *blue*, *gray* dengan tipe data *integer*, warna merah didapat dengan mengambil *pixel* (x,y).R lalu warna hijau didapat dengan mengambil *pixel* (x,y).G, warna biru didapat dengan mengambil *pixel* (x,y).B dan warna *grayscale* didapat dengan menjumlahkan nilai warna merah, hijau dan biru kemudian dibagi tiga.

|  |
| --- |
| Private Sub Binary(sender As Object, e As EventArgs) Handles Button3.Click  If PictureBox2.*Image* Is Nothing Then  MessageBox.Show("Please grayscale the *image* first!", "Warning",  MessageBoxButtons.OK, MessageBoxIcon.Error)  Else  Dim source As New Bitmap(PictureBox2.*Image*)  Dim red, green, blue, rgb As Integer  For y As Integer = 0 To source.Height - 1  For x As Integer = 0 To source.Width - 1  red = source.Get*Pixel*(x, y).R  green = source.Get*Pixel*(x, y).G  blue = source.Get*Pixel*(x, y).B  rgb = red + green + blue  If rgb < 255 Then  source.Set*Pixel*(x, y, Color.FromArgb(0, 0, 0))  Else  source.Set*Pixel*(x, y, Color.FromArgb(255, 255, 255))  End If  Next  Next  PictureBox3.*Image* = source  PictureBox3.SizeMode = PictureBoxSizeMode.Stretch*Image*  End If  End Sub  Private Sub Exit\_Program(sender As Object, e As EventArgs) Handles Button4.Click  Application.Exit()  End Sub |

**Kode Program 4.12** Sintaks Binary

Kode Program 4.12 merupakan sintaks untuk merubah citra *grayscale* kedalam *binary*. Program membuat variabel *red*, *green*, *blue*, *rgb* dengan tipe data *integer*, warna merah didapat dengan mengambil *pixel* (x,y).R, lalu warna hijau didapat dengan mengambil *pixel* (x,y).G, warna biru didapat dengan mengambil *pixel* (x,y).B dan rgb didapatkan dengan menjumlahkan warna merah, hijau, dan biru. *Set* *pixel* (x,y) warna argb bernilai (0,0,0) jika nilai warna rgb kurang dari 255. *Set* *pixel* (x,y) warna argb bernilai (255, 255, 255) jika nilai warna rgb lebih dari atau sama dengan 255. Gambar yang telah di *binary* akan di *stretch* dan masuk kedalam *picture box* 3.

|  |
| --- |
| Private Sub ValueBright\_Scroll(sender As Object, e As EventArgs) Handles ValueBright.Scroll  Dim source As New Bitmap(PictureBox1.*Image*)  PictureBox1.*Image* = source  Dim source1 As New Bitmap(PictureBox4.*Image*)  PictureBox4.*Image* = source1  Dim merah As Integer  Dim hijau As Integer  Dim biru As Integer  If PictureBox1.*Image* Is Nothing Then  MessageBox.Show("Buka Gambar Terlebih Dahulu!", "Gagal Menggelapkan", MessageBoxButtons.OK, MessageBoxIcon.Warning) 'mengeluarkan pesan dan memberi tombol OK serta memunculkan ikon warning  Else  For y = 0 To source.Height - 1  For x = 0 To source.Width - 1  merah = source1.Get*Pixel*(x, y).R  hijau = source1.Get*Pixel*(x, y).G  biru = source1.Get*Pixel*(x, y).B  Dim merah1 As Integer = merah  Dim hijau1 As Integer = hijau  Dim biru1 As Integer = biru  Dim merah2 As Integer  Dim hijau2 As Integer  Dim biru2 As Integer  merah2 = merah1 + ValueBright.Value  hijau2 = hijau1 + ValueBright.Value  biru2 = biru1 + ValueBright.Value  If merah2 < 0 Then  merah2 = 0  ElseIf merah2 > 255 Then  merah2 = 255  End If  If hijau2 < 0 Then  hijau2 = 0  ElseIf hijau2 > 255 Then  hijau2 = 255  End If  If biru2 < 0 Then  biru2 = 0  ElseIf biru2 > 255 Then  biru2 = 255  End If  source.Set*Pixel*(x, y, Color.FromArgb(merah2, hijau2, biru2))  Next  Next  End If  Label1.Text = ValueBright.Value  PictureBox1.SizeMode = PictureBoxSizeMode.Stretch*Image*  End Sub  End Class |

**Kode Program 4.13** Sintaks Brightness

Kode Program 4.13 merupakan sintaks *Brightness* pada program visual basic. Gambar pada *picturebox* 1 diatur dengan format bitmap ketika dibuka. *Picturebox* 1 tidak dimasukkan citranya, maka akan memunculkan dialog pesan “Buka Gambar terlebih dahulu”. Nilai warna pada sintaks *brightness* adalahdengannilai minimum adalah 0 dan nilai maksimum 255, apabila nilai *brightness* kurang dari 0 secara otomatis akan diatur 0 dan apabila nilai *brightness* lebih dari 255 akan secara otomatis diatur 255.

|  |
| --- |
| empty = isempty(get(handles.axes4, 'Children'));  if (empty == 1)  errordlg('Please add *noise* on grayscaled *image* first...','Error');  else    f = ones(3,3)/9;  disp(f)  f*image* = imfilter(*noise*, f);  axes(handles.axes5);  imshow(uint8(f*image*));  end |

**Kode Program 4.14** LPF 3x3 dengan Matlab

Kode Program 4.14 f=ones(3,3)/9; merupakan sintaks pada Matlab untuk menginisialisasi variabel f sebagai filter *gaussian* dengan menggunakan kernel matriks 3x3 kemudian ditampilkan pada *pixel* sebanyak 9.

|  |
| --- |
| empty = isempty(get(handles.axes4, 'Children'));  if (empty == 1)  errordlg('Please add *noise* on grayscaled *image* first...','Error');  else  f = ones(5,5)/25;  disp(f)  f*image* = imfilter(*noise*, f);  axes(handles.axes5);  imshow(uint8(f*image*));  end |

**Kode Program 4.15** LPF 5x5 dengan Matlab

Kode Program 4.15 f=ones(5,5)/25; merupakan sintaks pada Matlab untuk menginisialisasi variable f sebagai filter *gaussian* dengan menggunakan kernel matriks 5x5 kemudian ditampilkan pada *pixel* sebanyak 25.

|  |
| --- |
| empty = isempty(get(handles.axes4, 'Children'));  if (empty == 1)  errordlg('Please add *noise* on grayscaled *image* first...','Error');  else  f = ones(7,7)/49;  disp(f)  f*image* = imfilter(*noise*, f);  axes(handles.axes5);  imshow(uint8(f*image*));  end |

**Kode Program 4.16** LPF 7x7 dengan Matlab

Kode Program 4.16 f=ones(7,7)/49; merupakan sintaks pada Matlab untuk menginisialisasi variable f sebagai filter *gaussian* dengan menggunakan kernel matriks 7x7 kemudian ditampilkan pada *pixel* sebanyak 49.

|  |
| --- |
| empty = isempty(get(handles.axes4, 'Children'));  if (empty == 1)  errordlg('Please add *noise* on grayscaled *image* first...','Error');  else  f = ones(9,9)/81;  disp(f)  f*image* = imfilter(*noise*, f);  axes(handles.axes5);  imshow(uint8(f*image*));  end |

**Kode Program 4.17** LPF 9x9 dengan Matlab

Kode Program 4.17 f=ones(9,9)/81; merupakan sintaks pada Matlab untuk menginisialisasi variabel f sebagai filter *gaussian* dengan menggunakan kernel matriks 9x9 kemudian ditampilkan pada *pixel* sebanyak 81.

|  |
| --- |
| empty = isempty(get(handles.axes4, 'Children'));  if (empty == 1)  errordlg('Please add *noise* on grayscaled *image* first...','Error');  else  f = -1 \* ones(3,3);  f(2,2) = 8;  disp(f)  f*image* = imfilter(*noise*, f);  axes(handles.axes5);  imshow(uint8(f*image*));  end |

**Kode Program 4.18** HPF 3x3 dengan Matlab

Kode Program 4.18 f= -1\*ones(3,3); merupakan sintaks pada Matlab untuk *high pass filter* dengan menggunakan ukuran kernel matriks 3x3. Kode f(2,2) = 8; adalah titik f pada koordinat (2,2) diberikan nilai 8.

|  |
| --- |
| empty = isempty(get(handles.axes4, 'Children'));  if (empty == 1)  errordlg('Please add *noise* on grayscaled *image* first...','Error');  else  f = -1 \* ones(5,5);  f(3,3) = 24;  disp(f)  f*image* = imfilter(*noise*, f);  axes(handles.axes5);  imshow(uint8(f*image*));  end |

**Kode Program 4.19** HPF 5x5 dengan Matlab

Kode Program 4.19 f= -1\*ones(5,5); merupakan sintaks pada Matlab untuk *high pass filter* dengan menggunakan ukuran kernel matriks 5x5. Kode f(3,3) = 24; adalah titik f pada koordinat (3,3) diberikan nilai 24.

|  |
| --- |
| empty = isempty(get(handles.axes4, 'Children'));  if (empty == 1)  errordlg('Please add *noise* on grayscaled *image* first...','Error');  else  f = -1 \* ones(7,7);  f(4,4) = 48;  disp(f)  f*image* = imfilter(*noise*, f);  axes(handles.axes5);  imshow(uint8(f*image*));  end |

**Kode Program 4.20** HPF 7x7 dengan Matlab

Kode Program 4.20 f =-1\*ones(7,7); merupakan sintaks pada Matlab untuk *high pass filter* dengan menggunakan ukuran kernel matriks 7x7. Kode f(4,4) = 48; adalah titik f pada koordinat (4,4) diberikan nilai 48.

|  |
| --- |
| empty = isempty(get(handles.axes4, 'Children'));  if (empty == 1)  errordlg('Please add *noise* on grayscaled *image* first...','Error');  else  f = -1 \* ones(9,9);  f(5,5) = 80;  disp(f)  f*image* = imfilter(*noise*, f);  axes(handles.axes5);  imshow(uint8(f*image*));  end |

**Kode Program 4.21** HPF 9x9 dengan Matlab

Kode Program 4.21 f= -1\*ones(9,9); merupakan sintaks pada Matlab untuk *high pass filter* dengan menggunakan ukuran kernel matriks 9x9. Kode f(5,5) = 80; adalah titik f pada koordinat (5,5) diberikan nilai 80.

|  |
| --- |
| empty = isempty(get(handles.axes4, 'Children'));  if (empty == 1)  errordlg('Please add *noise* on grayscaled *image* first...','Error');  else  f*image* = medfilt2(*noise*, [3 3]);  axes(handles.axes5);  imshow(uint8(f*image*));  end |

**Kode Program 4.22** Median 3x3 dengan Matlab

Kode Program 4.22 f*image* = medfilt2(*noise*, [3 3]); merupakan sintaks untuk menginisialisasi variabel f dengan rumus *median* filter dengan kernel matriks 3x3.

|  |
| --- |
| empty = isempty(get(handles.axes4, 'Children'));  if (empty == 1)  errordlg('Please add *noise* on grayscaled *image* first...','Error');  else  f*image* = medfilt2(*noise*, [5 5]);  axes(handles.axes5);  imshow(uint8(f*image*));  end |

**Kode Program 4.23** Median 5x5 dengan Matlab

Kode Program 4.23 f*image* = medfilt2(*noise*, [5 5]); merupakan sintaks untuk menginisialisasi variabel f dengan rumus *median* filter dengan kernel matriks 5x5.

|  |
| --- |
| empty = isempty(get(handles.axes4, 'Children'));  if (empty == 1)  errordlg('Please add *noise* on grayscaled *image* first...','Error');  else  f*image* = medfilt2(*noise*, [7 7]);  axes(handles.axes5);  imshow(uint8(f*image*));  end |

**Kode Program 4.24** Median 7x7 dengan Matlab

Kode Program 4.24 f*image* = medfilt2(*noise*, [7 7]); merupakan sintaks untuk menginisialisasi variabel f dengan rumus *median* filter dengan kernel matriks 7x7.

|  |
| --- |
| empty = isempty(get(handles.axes4, 'Children'));  if (empty == 1)  errordlg('Please add *noise* on grayscaled *image* first...','Error');  else  f*image* = medfilt2(*noise*, [9 9]);  axes(handles.axes5);  imshow(uint8(f*image*));  end |

**Kode Program 4.25** Median 9x9 dengan Matlab

Kode Program 4.25 f*image* = medfilt2(*noise*, [9 9]); merupakan sintaks untuk menginisialisasi variabel f dengan rumus *median* filter dengan kernel matriks 9x9.

|  |
| --- |
| global a;  global value;    empty = isempty(get(handles.axes1, 'Children'));    if (empty == 1)  errordlg('Gambar Belum Dimasukan! Silahkan Klik Browse...','Terjadi kesalahan');  else  brighten = a;  brighten=brighten+50;  value = +50;  if (value >= 250)  errordlg('Gambar Dah Putih Bro!','Terjadi kesalahan');  else  axes(handles.axes1);  imshow(brighten);    brighten\_val = sprintf('%d', value);  set(handles.brightVal, 'String', brighten\_val);  end  end |

**Kode Program 4.26** Slidebar Brightness

Kode Program 4.26 merupakan sintaks untuk membuat *slidebar brightness* pada citra, dimana batas nilai *brightness* dapat diatur hingga 250. Kode global a; global value; berfungsi untuk mengatur variabel a dan value secara global. Batas nilai *brightness* diatur hingga 250, jika nilai *brightness* lebih dari 250 maka program akan memunculkan pesan error seperti pada kode program if (value >= 250) errordlg('Gambar Dah Putih Bro!','Terjadi kesalahan');.

### Langkah-Langkah Percobaan

Langkah-langkah percobaan praktikum untuk program mengubah citra berwarna ke *grayscale*, mengubah citra berwarna ke biner, mengatur keserahan dan mengubah ke citra negasi adalah sebagai berikut.

|  |
| --- |
|  |

**Gambar 4.29** Validasi program Visual Basic

Gambar 4.29 merupakan validasi program Visual Basic. Validasi ini bertujuan agar pemrosesan gambar pertama diinputkan dari PictureBox 1 yang berfungsi untuk membuka *file* dengan menekan *button* *Open* *Image*. Apabila *user* tidak menginputkan gambar melalui PictureBox 1, maka program akan memunculkan validasi berupa peringatan untuk membuka gambar terlebih dahulu di *button* Open *Image*.

|  |
| --- |
|  |

**Gambar 4.30** Perubahan Citra Asli Ke Grayscale

Gambar 4.30 merupakan gambar hasil perubahan citra asli ke dalam *grayscale*. Gambar hasil citra *grayscale* ini didapatkan setelah mengklik *button grayscale* dan ditampilkan pada *Picturebox*2.

|  |
| --- |
|  |

**Gambar 4.31** Perubahan Citra Asli Ke Binary

Gambar 4.31 merupakan gambar hasil perubahan citra asli dengan ekstensi .jpg ke dalam *Binary*. Gambar hasil citra *binary* ini didapatkan setelah mengklik *button binary* dan ditampilkan pada *picturebox*3.

|  |
| --- |
|  |

**Gambar 4.32** Perubahan Citra Asli Ke Binary

Gambar 4.32 merupakan gambar dengan ekstensi .jpg hasil perubahan citra asli ke dalam *Binary*. Gambar hasil citra *binary* ini didapatkan setelah mengklik *button binary* dan ditampilkan pada *Picturebox*3.

|  |
| --- |
|  |

**Gambar 4.33** Perubahan Citra Asli Ke Grayscale dengan Matlab

Gambar 4.33 merupakan gambar hasil perubahan citra asli ke dalam *grayscale* dengan menggunakan Matlab. Gambar hasil citra *grayscale* ini didapatkan setelah mengklik *button grayscale* dan ditampilkan pada *Picturebox*2.

|  |
| --- |
|  |

**Gambar 4.34** Perubahan Citra Asli Ke Binary dengan Matlab

Gambar 4.34 merupakan gambar hasil perubahan citra asli ke dalam *binary*. Gambar hasil citra *binary* ini didapatkan setelah mengklik *button binary* dan ditampilkan pada *Picturebox*3.

|  |
| --- |
|  |

**Gambar 4.35** Perubahan Citra Asli Ke Noise Image dengan Matlab

Gambar 4.35 merupakan gambar hasil perubahan citra asli ke dalam *noise image*. Gambar hasil citra *noise image* ini didapatkan setelah mengklik *button noise image* dan ditampilkan pada *Picturebox*4.

|  |
| --- |
|  |

**Gambar 4.36** Proses Filtering LPF dengan kernel 3x3

Gambar 4.36 merupakan proses *filtering* dengan menggunakan *low pass filter* dengan menggunakan kernel ukuran 3x3. Gambar hasil *filtering* ini didapatkan setelah mengklik radio *button* LPF 3x3 dan ditampilkan pada *Picturebox*5.

|  |
| --- |
|  |

**Gambar 4.37** Proses Filtering LPF dengan kernel 5x5

Gambar 4.37 merupakan proses *filtering* dengan menggunakan *low pass filter* dengan menggunakan kernel ukuran 5x5. Gambar hasil *filtering* ini didapatkan setelah mengklik radio *button* LPF 5x5 dan ditampilkan pada *Picturebox*5.

|  |
| --- |
|  |

**Gambar 4.38** Proses Filtering LPF dengan kernel 9x9

Gambar 4.38 merupakan proses *filtering* dengan menggunakan *low pass filter* dengan menggunakan kernel ukuran 9x9. Gambar hasil *filtering* ini didapatkan setelah mengklik radio *button* LPF 9x9 dan ditampilkan pada *Picturebox*5.

|  |
| --- |
|  |

**Gambar 4.39** Proses Filtering HPF dengan kernel 3x3

Gambar 4.39 merupakan proses *filtering* dengan menggunakan *high pass filter* dengan menggunakan kernel ukuran 3x3. Gambar hasil *filtering* ini didapatkan setelah mengklik radio *button* HPF 3x3 dan ditampilkan pada *Picturebox*5.

|  |
| --- |
|  |

**Gambar 4.40** Proses Filtering HPF dengan kernel 5X5

Gambar 4.40 merupakan proses *filtering* dengan menggunakan *high pass filter* dengan menggunakan kernel ukuran 5x5. Gambar hasil *filtering* ini didapatkan setelah mengklik radio *button* HPF 5x5 dan ditampilkan pada *Picturebox*5.

|  |
| --- |
|  |

**Gambar 4.41** Proses Filtering HPF dengan kernel 7x7

Gambar 4.41 merupakan proses *filtering* menggunakan *high pass filter* dengan menggunakan kernel ukuran 7x7. Gambar hasil *filtering* ini didapatkan setelah mengklik radio *button* HPF 7x7 dan ditampilkan pada *Picturebox*5.

|  |
| --- |
|  |

**Gambar 4.42** Proses Median Filter dengan kernel 3x3

Gambar 4.42 merupakan proses *median filter* dengan menggunakan kernel ukuran 3x3. Gambar hasil *filtering* ini didapatkan setelah mengklik radio *button median* 3x3 dan ditampilkan pada *Picturebox*5.

|  |
| --- |
|  |

**Gambar 4.43** Proses Median Filter dengan kernel 5x5

Gambar 4.43 merupakan proses *median filter* dengan menggunakan kernel ukuran 5x5. Gambar hasil *filtering* ini didapatkan setelah mengklik radio *button median* 5x5 dan ditampilkan pada *Picturebox*5.

|  |
| --- |
|  |

**Gambar 4.44** Proses Median Filter dengan kernel 7x7

Gambar 4.44 merupakan proses *median filter* dengan menggunakan kernel ukuran 7x7. Gambar hasil *filtering* ini didapatkan setelah mengklik radio *button median* 7x7 dan ditampilkan pada *Picturebox*5.

|  |
| --- |
|  |

**Gambar 4.45** Proses Median Filter dengan kernel 9x9

Gambar 4.45 merupakan proses *median filter* dengan menggunakan kernel ukuran 9x9. Gambar hasil *filtering* ini didapatkan setelah mengklik radio *button median* 9x9 dan ditampilkan pada *Picturebox*5.

### Analisis Perbandingan Citra

Berikut merupakan analisis perbandingan perbaikan kualitas citra dari program Visual Basic dan Matlab. Perbaikan kualitas citra pada Visual Basic adalah penggunaan *brightness*, sedangkan perbaikan kualitas citra pada Matlab adalah penggunaan kernel *Low Pass Filter*, *High Pass Filter*, dan *Median Filter*.

|  |
| --- |
|  |

**Gambar 4.46** Pengurangan Brightness

Gambar 4.46 merupakan percobaan pengurangan brightness pada program Visual Basic dari citra awal yang diatur dari picturebox 1. Pengurangan brightness menyebabakan citra menjadi lebih gelap dari sebelumnya.

|  |
| --- |
|  |

**Gambar 4.47** Penambahan Brightness

Gambar 4.47 merupakan percobaan penambahan brightness pada program Visual Basic dari citra awal yang diatur dari picturebox 1. Penambahan brightness menyebabakan citra menjadi lebih terang dari sebelumnya.

|  |
| --- |
|  |

**Gambar 4.48** Percobaan LPF 3x3

Gambar 4.48 merupakan LPF (*Low Pass Filter*) dengan kernel 3x3. Hasil dari proses LPF (*Low Pass Filter*) kernel 3x3 akan mengurangi *noise* pada citra, namun tidak semua *noise* berhasil dihilangkan.

|  |
| --- |
|  |

**Gambar 4.49** Percobaan LPF 5x5

Gambar 4.49 merupakan LPF (*Low Pass Filter*) dengan kernel 5x5. Hasil dari proses LPF kernel 5x5 akan mengurangi *noise* pada citra lebih dari LPF kernel 3x3, karena semakin banyak matriks kernel yang digunakan maka *noise* akan semakin berkurang.

|  |
| --- |
|  |

**Gambar 4.50** Percobaan LPF 7x7

Gambar 4.50 merupakan LPF (*Low Pass Filter*) dengan kernel 7x7. Hasil dari proses LPF kernel 7x7 akan mengurangi *noise* pada citra lebih dari LPF kernel 3x3, 5x5 karena semakin banyak matriks kernel yang digunakan maka *noise* akan semakin berkurang.

|  |
| --- |
|  |

**Gambar 4.51** Percobaan LPF 9x9

Gambar 4.51 merupakan LPF (*Low Pass Filter*) dengan kernel 9x9. Hasil dari proses LPF kernel 9x9 akan mengurangi *noise* pada citra lebih dari LPF kernel 3x3, 5x5, dan 7x7 karena semakin banyak matriks kernel yang digunakan maka *noise* akan semakin berkurang.

|  |
| --- |
|  |

**Gambar 4.52** Percobaan HPF 3x3

Gambar 4.52 merupakan HPF (*High Pass Filter*) dengan kernel 3x3. Hasil dari proses HPF kernel 3x3 akan menambah *noise* pada citra, dimana citra menjadi lebih gelap.

|  |
| --- |
|  |

**Gambar 4.53** Percobaan HPF 5x5

Gambar 4.53 merupakan HPF (*High Pass Filter*) dengan kernel 5x5. Hasil dari proses HPF kernel 5x5 akan menambah *noise* pada citra, dimana bercak *noise* pada citra berwarna keputih-putihan.

|  |
| --- |
|  |

**Gambar 4.54** Percobaan HPF 7x7

Gambar 4.54 merupakan HPF (*High Pass Filter*) dengan kernel 7x7. Hasil dari proses HPF kernel 7x7 akan menambah *noise* pada citra, dimana bercak *noise* pada citra berwarna keputih-putihan dari matriks kernel 3x3 dan 5x5.

|  |
| --- |
|  |

**Gambar 4.55** Percobaan HPF 9x9

Gambar 4.55 merupakan HPF (*High Pass Filter*) dengan kernel 5x5. Hasil dari proses HPF kernel 5x5 akan menambah *noise* pada citra, dimana bercak *noise* pada citra berwarna keputih-putihan.

|  |
| --- |
|  |

**Gambar 4.56** Percobaan Median 3x3

Gambar 4.56 merupakan Median dengan kernel 3x3. Hasil dari proses Median kernel 3x3 adalah citra tidak terdapat *noise* dan menjadi *blur*.

|  |
| --- |
|  |

**Gambar 4.57** Percobaan Median 5x5

Gambar 4.57 merupakan Median dengan kernel 5x5. Hasil dari proses Median kernel 5x5 adalah citra menjadi lebih halus, tidak terdapat *noise*, dan menjadi *blur* dari citra dengan kernel 3x3.

|  |
| --- |
|  |

**Gambar 4.58** Percobaan Median 7x7

Gambar 4.58 merupakan Median dengan kernel 7x7. Hasil dari proses Median kernel 7x7 adalah citra menjadi lebih halus, tidak terdapat *noise*, dan menjadi *blur* dari citra dengan kernel 3x3 dan 5x5.

|  |
| --- |
|  |

**Gambar 4.59** Percobaan Median 9x9

Gambar 4.59 merupakan Median dengan kernel 9x9. Hasil dari proses Median kernel 9x9 adalah citra menjadi lebih halus, tidak terdapat *noise*, dan menjadi *blur* dari citra dengan kernel 3x3, 5x5, dan 7x7.

## Transfomasi Citra

Modul III merupakan modul yang berisikan pembahasan mengenai transformasi citra 2 Dimensi ke ruang frekuensi dengan menggunakan DFT (*Discrete Fourier Transform*), DCT (*Discrete Cosine Transform*) dan (FFT) *Fast Fourier Transfrom* serta perubahan citra 2 Dimensi ke ruang skala dan waktu menggunakan Transformasi *Wavelet*.

### *Source Code* Modul 3

Berikut merupakan kode program modul 3 Transformasi citra yang dibuat dengan menggunakan bahasa pemrograman MATLAB.

|  |
| --- |
| global img  global dft  empty = isempty(get(handles.pb1, 'Children'));    if (empty == 1)  errordlg('Gambar Belum Dimasukan! Silahkan Klik Pilih Gambar...','Terjadi kesalahan');  else  gray = rgb2gray(img);  imdft = fft2(gray);  dft = fftshift(imdft);  guidata(hObject,handles);  axes(handles.pb2);  imshow(log(abs(dft)),[]);  end |

**Kode Program 4.27** Transformasi DFT (Discreate Fourier Transform)

Kode Program 4.27 merupakan kode program transformasi DFT (*Discreate Fourier Transform*). Kode Program 4.27 transformasi *fourier* merupakan representasi dari sebuah citra sebagai penjumlahan eksponensial kompleks dari *magnitude*, frekuensi, dan fasa. Kode Program gray=rgb2gray(img); digunakan untuk mengubah citra asli ke citra *grayscale*. Kode imdft=fft2(gray); digunakan untuk mengimplementasikan transformasi *fourier* pada MATLAB, sedangkan kode dft=fftshift(imdft); menunjukkan transformasi fourier yang dirubah ke bentuk *Discreate Fourier Transform*.

|  |
| --- |
| global img  global dct  empty = isempty(get(handles.pb1, 'Children'));    if (empty == 1)  errordlg('Gambar Belum Dimasukan! Silahkan Klik Pilih Gambar...','Terjadi kesalahan');  else  gray = rgb2gray(img);  dct = dct2(gray);  guidata(hObject,handles);  axes(handles.pb2);  imshow(log(abs(dct)),[]);  end |

**Kode Program 4.28** Transformasi DCT (Discreate Cosinus Transform)

Kode Program 4.28 merupakan kode program transformasi DFT (*Discreate Fourier Transform*). Kode Program 4.28 *Discrete Cosine Transform* merepresentasikan sebuah citra dari penjumlahan sinusoida dari *magnitude* dan frekuensi yang berubah -ubah. Sifat dari DCT adalah mengubah informasi citra yang signifkan dikonsentrasikan hanya pada beberapa koefisien DCT. Kode Program gray=rgb2gray(img); digunakan untuk mengubah citra asli ke citra *grayscale*. Kode dct=dct2(gray); menunjukkan transformasi *fourier* yang dirubah ke bentuk *Discreate Cosinus Transform*.

|  |
| --- |
| global img  global fft;  empty = isempty(get(handles.pb1, 'Children'));    if (empty == 1)  errordlg('Gambar Belum Dimasukan! Silahkan Klik Pilih Gambar...','Terjadi kesalahan');  else  gray = rgb2gray(img);  imfft = fft2(gray);  %ab = abs(imdft);  fft = fftshift(imfft);  guidata(hObject,handles);  axes(handles.pb2);  imshow(log(1+abs(fft)),[]);  end |

**Kode Program 4.29** Transformasi FFT (Fast Fourier Transform)

Kode Program 4.29 merupakan transformasi FFT (*Fast Fourier Transform*) yang digunakan untuk menghitung Transformasi *Fourier* diskret. Kode Program gray=rgb2gray(img); digunakan untuk mengubah citra asli ke citra *grayscale*. Kode imfft=fft2(gray); digunakan untuk mengimplementasikan transformasi *fourier* pada MATLAB, sedangkan kode fft=fftshift(imfft); menunjukkan transformasi *fourier* yang dirubah ke bentuk *Fast Cosinus Transform*.

|  |
| --- |
| global img  global dwt;  empty = isempty(get(handles.pb1, 'Children'));    if (empty == 1)  errordlg('Gambar Belum Dimasukan! Silahkan Klik Pilih Gambar...','Terjadi kesalahan');  else  gray = rgb2gray(img);  dwt = dwt2(gray,'haar');  guidata(hObject,handles);  axes(handles.pb2);  imshow(dwt);  end |

**Kode Program 4.30** Transformasi Wavelet

Kode Program 4.30 merupakan kode program transformasi wavelet DWT (*Discrete Wavelet Transform*). Kode Program gray=rgb2gray(img); digunakan untuk mengubah citra asli ke citra *grayscale*. Kode Program dwt=dwt2(gray,'haar'); digunakan untuk mengimplementasikan transformasi *wavelet* ke bentuk *Discrete Wavelet Transform*.

|  |
| --- |
| global dft;    empty = isempty(get(handles.pb2, 'Children'));    if (empty == 1)  errordlg('Gambar Belum Dimasukan! Silahkan Klik Pilih Gambar...','Terjadi kesalahan');  else  F = ifftshift(dft);  invdft = ifft2(F);  guidata(hObject,handles);  axes(handles.pb3);  imshow(invdft,[]);  end |

**Kode Program 4.31** Transformasi inverse DFT

Kode Program 4.31 merupakan transformasi *inverse* DFT (*Discrete Fourier Transform*). Kode Program F=ifftshift(dft); merupakan inisialisasi variabel citra F ke citra DFT (*Discrete Fourier Transform*), sedangkan kode invdft=ifft2(F); berfungsi untuk mengubah citra F ke bentuk *inverse* DFT (*Discrete Fourier Transform*).

|  |
| --- |
| global dct;    empty = isempty(get(handles.pb2, 'Children'));    if (empty == 1)  errordlg('Gambar Belum Dimasukan! Silahkan Klik Pilih Gambar...','Terjadi kesalahan');  else  invdct = idct2(dct);  guidata(hObject,handles);  axes(handles.pb3);  imshow(invdct,[]);  end |

**Kode Program 4.32** Transformasi inverse DCT

Kode Program 4.32 merupakan transformasi *inverse* DCT (*Discrete Cosinus Transform*). Kode Program F=ifftshift(dft); merupakan inisialisasi variabel citra F ke citra DFT (*Discrete Fourier Transform*), sedangkan kode invdft=ifft2(F); berfungsi untuk mengubah citra F ke bentuk *inverse* DFT (*Discrete Fourier Transform*).

|  |
| --- |
| global fft;    empty = isempty(get(handles.pb2, 'Children'));    if (empty == 1)  errordlg('Gambar Belum Dimasukan! Silahkan Klik Pilih Gambar...','Terjadi kesalahan');  else  F = ifftshift(fft);  invfft = ifft2(F);  guidata(hObject,handles);  axes(handles.pb3);  imshow(invfft,[]);  end |

**Kode Program 4.33** Transformasi inverse FFT

Kode Program 4.33 merupakan transformasi *inverse* FFT (*Fast Fourier Transform*). Kode Program F=ifftshift(fft); merupakan inisialisasi variabel citra F ke citra FFT (*Fast Fourier Transform*), sedangkan kode invfft=ifft2(F); berfungsi untuk mengubah citra F ke bentuk *inverse* FFT (*Fast Fourier Transform*).

|  |
| --- |
| global dwt;    empty = isempty(get(handles.pb2, 'Children'));    if (empty == 1)  errordlg('Gambar Belum Dimasukan! Silahkan Klik Pilih Gambar...','Terjadi kesalahan');  else  %sx = size(dwt);  idwt = idwt2(dwt,'haar');  guidata(hObject,handles);  axes(handles.pb3);  imshow(idwt);  end |

**Kode Program 4.34** Transformasi inverse Wavelet

Kode program 4.34 merupakan kode program *inverse* *wavelet* DWT (*Discrete Wavelet Transform*). Kode program idwt = idwt2(dwt,'haar'); berfungsi untuk mengubah bentuk citra DWT (*Discrete Wavelet Transform*) ke bentuk *inverse* DWT (*Discrete Wavelet Transform*).

|  |
| --- |
| global img;  [nama\_file, nama\_path] = uigetfile('\*.png;\*.jpg;\*.bmp;\*.gif','Select *Image*');  if ~isequal (nama\_file,0)  img = imread(fullfile(nama\_path,nama\_file));  guidata(hObject,handles);  axes(handles.pb1);  imshow(img);  else  return;  end |

**Kode Program 4.35** Sintaks Open Image

Kode Program 4.35 merupakan kode program *Open Image* pada MATLAB. Kode global img; berfungsi untuk menginialisasi variabel img sebagai variabel global. Kode [nama\_file,nama\_path] = uigetfile ('\*.png; \*.jpg; \*.bmp;\*.gif', 'Select *Image*'); merupakan kode program untuk mengambil gambar/citra dengan format png, jpg, bmp, dan gif. Kode program img= imread(fullfile(nama\_path,nama\_file)); berfungsi untuk menginialisasi variabel img berdasarkan nama *path folder* gambar dan nama *file* gambar/citra yang diinputkan.

### Langkah-Langkah Percobaan Modul 3

Langkah-langkah percobaan praktikum untuk program transformasi citra ke ruang frekuensi dengan menggunakan DFT (*Discrete Fourier Transform*), DCT (*Discrete Cosine Transform*) dan (FFT) *Fast Fourier Transfrom* serta perubahan citra 2 dimensi ke ruang skala dan waktu menggunakan Transformasi *Wavelet* adalah sebagai berikut.

|  |
| --- |
|  |

**Gambar 4.60** Validasi program Matlab

Gambar 4.60 merupakan validasi program Matlab. Validasi ini bertujuan agar pemrosesan gambar pertama yang diinputkan dari PictureBox 1 yang berfungsi untuk membuka file dengan menekan *button* *Open* *Image*. *User* yang tidak menginputkan gambar melalui *PictureBox* 1, maka program akan memunculkan validasi berupa peringatan untuk membuka gambar terlebih dahulu di *button* Open *Image*.

|  |
| --- |
|  |

**Gambar 4.61** Transformasi DFT

Gambar 4.61 merupakan gambar hasil transformasi perubahan citra asli ke dalam *Discrate Fourier Transform*. Gambar hasil citraini didapatkan setelah mengklik *button* DFTdan ditampilkan pada *Picturebox*2.

|  |
| --- |
|  |

**Gambar 4.62** Transformasi DCT (Discrate Cosine Transform)

Gambar 4.62 merupakan gambar hasil transformasi perubahan citra asli ke dalam *Discrate Cosine Transform*. Gambar hasil citraini didapatkan setelah mengklik *button* DCTdan ditampilkan pada *Picturebox*2.

|  |
| --- |
|  |

**Gambar 4.63** Transformasi FFT

Gambar 4.63 merupakan gambar hasil transformasi perubahan citra asli ke dalam *Fast Fourier Transform*. Gambar hasil citraini didapatkan setelah mengklik *button* FFTdan ditampilkan pada *Picturebox*2.

|  |
| --- |
| C:\Users\Asus X450J\AppData\Local\Microsoft\Windows\INetCache\Content.Word\messageImage_1526689281291.jpg |

**Gambar 4.64** Transformasi Wavelet

Gambar 4.64 merupakan gambar hasil transformasi perubahan citra asli ke dalam *Wavelet*. Gambar hasil citraini didapatkan setelah mengklik *button* *Wavelet* dan ditampilkan pada *Picturebox*2.

|  |
| --- |
| C:\Users\Asus X450J\AppData\Local\Microsoft\Windows\INetCache\Content.Word\messageImage_1526689432225.jpg |

**Gambar 4.65** Inverse DFT

Gambar 4.65 merupakan gambar hasil *inverse* dari *Discrate Fourier Transform*. Gambar hasil *inverse* citraini didapatkan setelah mengklik *button* *inverse* DFTdan ditampilkan pada *Picturebox*3.

|  |
| --- |
| C:\Users\Asus X450J\AppData\Local\Microsoft\Windows\INetCache\Content.Word\messageImage_1526689473321.jpg |

**Gambar 4.66** Inverse DCT

Gambar 4.66 merupakan gambar hasil *inverse* dari *Discrate Cosine Transform*. Gambar hasil *inverse* citraini didapatkan setelah mengklik *button* *inverse* DCTdan ditampilkan pada *Picturebox*3.

|  |
| --- |
| C:\Users\Asus X450J\AppData\Local\Microsoft\Windows\INetCache\Content.Word\messageImage_1526689498429.jpg |

**Gambar 4.67** Inverse FFT

Gambar 4.67 merupakan gambar hasil *inverse* dari *Fast Fourier Transform*. Gambar hasil *inverse* citraini didapatkan setelah mengklik *button* *inverse* FFTdan ditampilkan pada *Picturebox*3.

|  |
| --- |
| C:\Users\Asus X450J\AppData\Local\Microsoft\Windows\INetCache\Content.Word\messageImage_1526689522624.jpg |

**Gambar 4.68** Inverse Wavelet

Gambar 4.68 merupakan gambar hasil *inverse* dari transformasi *Wavelet*. Gambar hasil *inverse* citraini didapatkan setelah mengklik *button* *inverse Wavelet* dan ditampilkan pada *Picturebox*3.

### Analisis Perbandingan Citra

Berikut merupakan analisis perbandingan perbaikan kualitas citra dari program Matlab. Perbaikan kualitas citra pada Matlab berupa perbandingan citra pada hasil transformasi DFT (*Discreate Fourier Transform*), DCT (*Discrete Cosinus Transform*), FFT (*Fast Fourier Transform*), *Wavelet*, *inverse* DFT, *inverse* DCT, *inverse* FFT, dan *inverse* *Wavelet*.

|  |
| --- |
|  |

**Gambar 4.69** Hasil transformasi citra DFT dan inverse DFT

Gambar 4.69 merupakan hasil transformasi citra DFT dan hasil *inverse* DFT. Gambar 4.69 pada transformasi DFT (*Discrete Fourier Transform*) menunjukkan frekuensi-frekuensi titik berpusat di tengah citra dengan berwarna putih, sedangkan hasil citra *inverse* DFT(*Discrete Fourier Transform*) mengembalikan nilai citra asli ke citra *grayscale*.

|  |
| --- |
|  |

**Gambar 4.70** Hasil transformasi citra DCT dan inverse DCT

Gambar 4.70 merupakan hasil transformasi citra DCT (*Discrete Cosine Transform*) dan hasil *inverse* DCT (*Discrete Cosine Transform*). Gambar 4.70 pada transformasi DCT (*Discrete Cosine Transform*) menunjukkan frekuensi-frekuensi titik menyebar di keseluruhan gambar/citra, sedangkan hasil citra *inverse* DCT (*Discrete Cosine Transform*) mengembalikan nilai citra asli ke citra *grayscale*.

|  |
| --- |
|  |

**Gambar 4.71** Hasil transformasi citra FFT dan inverse FFT

Gambar 4.71 merupakan hasil transformasi citra FFT (*Fast Fourier Transform*) dan hasil *inverse* FFT. Gambar 4.71 pada transformasi FFT (*Fast Fourier Transform*) menunjukkan frekuensi-frekuensi titik berpusat di tengah citra dengan berwarna putih, sedangkan hasil citra *inverse* FFT (*Fast Fourier Transform*) mengembalikan nilai citra asli ke citra *grayscale*.

|  |
| --- |
|  |

**Gambar 4.72** Hasil transformasi citra Wavelet dan inverse Wavelet

Gambar 4.72 merupakan hasil transformasi citra *Wavelet* dan hasil *inverse* *Wavelet*. Gambar 4.72 pada transformasi *wavelet* menunjukkan frekuensi-frekuensi titik total berwarna putih pada keseluruhan area citra, sedangkan hasil citra *wavelet* mengembalikan nilai citra asli ke citra *grayscale*.

## Morfologi dan Segmentasi Citra

Modul IV merupakan modul yang berisikan pembahasan mengenai morfologi citra dilasi, *erosi*, *closing*, *opening*, dan *thinning* serta segmentasi citra *edge detection* dengan menggunakan *sobel filter*, *robert filter*, *prewitt filter*, *canny*, dan LOG, serta segmentasi citra berbasis area dengan menggunakan *region growing* dan *watershed*.

### *Source Code* Modul 4

Berikut merupakan kode program modul 4 Morfologi dan Segmentasi Citra yang dibuat dengan menggunakan bahasa pemrograman MATLAB.

|  |
| --- |
| bin = im2bw(img,graythresh(img) );  se = ones(3, 3);  dil = imdilate(bin, se);  guidata(hObject,handles);  axes(handles.pb2);  imshow(dil);    val = sprintf('Dilation');  set(handles.st2, 'String', val); |

**Kode Program 4.36** Metode Morfologi Dilasi

Kode Program 4.36 merupakan kode program untuk mengubah citra ke dilasi. Sintaks yang pertama dimasukkan yaitu merubah RGB atau *grayscale* *image* menjadi *Binary lmage*, selanjutnya *Structuring Element* (SE) yang digunakan sebagai operator untuk membentuk objek dalam citra yaitu (3,3), selanjutnya memasukkan sintaks fungsi dilasi, dan menampilkan objek sebelum dan sesudah proses dilasi dengan menggunakan sintaksyang terdapat pada kode program 4.36.

|  |
| --- |
| bin = im2bw(img,graythresh(img) );  se = ones(3, 3);  ero = imerode(bin, se);  guidata(hObject,handles);  axes(handles.pb3);  imshow(ero);    val = sprintf('Erosion');  set(handles.st3, 'String', val); |

**Kode Program 4.37** Metode Morfologi Erosi

Kode Program 4.37 merupakan kode program untuk mengubah citra ke erosi. Sintaks yang pertama dimasukkan yaitu merubah RGB atau *grayscale* *image* menjadi *Binary lmage*, selanjutnya *Structuring Element* (SE) yang digunakan sebagai operator untuk membentuk objek dalam citra yaitu (3,3), selanjutnya memasukkan sintaks fungsi erosi, dan menampilkan objek sebelum dan sesudah proses erosi dengan menggunakan sintaksyang terdapat pada kode program 4.37.

|  |
| --- |
| bin = im2bw(img,graythresh(img) );  se = ones(3,3);  op = imopen(bin, se);  guidata(hObject,handles);  axes(handles.pb2);  imshow(op);  val = sprintf('*Opening*');  set(handles.st2, 'String', val); |

**Kode Program 4.38** Metode Morfologi Opening

Kode program 4.38 merupakan kode program untuk *Opening*. *Opening* merupakan kombinasi erosi-dilasi dengan structuring element yang sama dan structuring element yang digunakan yaitu (3,3). Kode bin= im2bw(img,graythresh(img)); berfungsi mengubah RGB atau *grayscale* *image* menjadi *Binary lmage*. Kode se=ones(3,3); adalah kode *Structuring Element* (SE) yang digunakan sebagai operator untuk membentuk objek dalam citra yaitu (3,3). Kode op = imopen(bin, se); berfungsi untuk memasukkan sintaks fungsi *opening*, dan kode imshow(op); berfungsi untuk menampilkan menampilkan objek sebelum dan sesudah proses *opening*.

|  |
| --- |
| global img  global cl  bin = im2bw(img,graythresh(img));  se = ones(3, 3);  cl = imclose(bin, se);  guidata(hObject,handles);  axes(handles.pb3);  imshow(cl);  val = sprintf('*Closing*');  set(handles.st3, 'String', val); |

**Kode Program 4.39** Metode Morfologi Closing

Kode program 4.39 merupakan kode program untuk *Closing*. *Closing* merupakan kombinasi dilasi-erosi dengan structuring element yang sama yaitu (3,3). Kode bin= im2bw(img,graythresh(img)); berfungsi mengubah RGB atau *grayscale* *image* menjadi *Binary lmage*. Kode se=ones(3,3); adalah kode *Structuring Element* (SE) yang digunakan sebagai operator untuk membentuk objek dalam citra yaitu (3,3). Kode cl = imclose(bin, se); berfungsi untuk memasukkan sintaks fungsi closing, dan kode imshow(cl); berfungsi untuk menampilkan menampilkan objek sebelum dan sesudah proses *closing*.

|  |
| --- |
| global img  global thin  bin = im2bw(img,graythresh(img));  thin = bwmorph(bin,'thin',Inf);  guidata(hObject,handles);  axes(handles.pb2);  imshow(thin);    val = sprintf('*Thinning*');  set(handles.st2, 'String', val); |

**Kode Program 4.40** Metode Morfologi Thinning

Kode program 4.40 merupakan kode program untuk *Thinning*. Kode bin= im2bw(img,graythresh(img)); berfungsi mengubah RGB atau *grayscale* *image* menjadi *Binary lmage*. Kode thin=bwmorph(bin,'thin',Inf);adalah kode yang berfungsi untuk memasukkan sintaks fungsi *thinning*, dan kode imshow(thin); berfungsi untuk menampilkan menampilkan objek sebelum dan sesudah proses *thinning*.

|  |
| --- |
| global img    bin = im2bw(img,graythresh(img));  sobHor = edge(bin,'*sobel*','horizontal');  sobVer = edge(bin,'*sobel*','vertical');  sobGmag = imgradient(bin,'*sobel*');    guidata(hObject,handles);  axes(handles.pb2);  imshow(sobHor);  val = sprintf('*Sobel* Horizontal');  set(handles.st2, 'String', val);    guidata(hObject,handles);  axes(handles.pb3);  imshow(sobVer);  val = sprintf('*Sobel* Vertical');  set(handles.st3, 'String', val);    guidata(hObject,handles);  axes(handles.pb4);  imshow(sobGmag);  val = sprintf('*Sobel* Gradient Magnitude');  set(handles.st4, 'String', val); |

**Kode Program 4.41** Metode Segmentasi Sobel Filter

Kode program 4.41 merupakan kode program Segmentasi Sobel *Filter*. Kode bin = im2bw(img,graythresh(img)); berfungsi mengubah RGB atau *grayscale* *image* menjadi *Binary lmage*. Kode sobHor=edge (bin, *sobel*', 'horizontal'); merupakan kode sobel *filter* untuk horizontal dan kode sobVer=edge (bin, '*sobel*','vertical'); untuk kode *sobel* *filter* untuk vertikal. Kode sobGmag= imgradient(bin,'*sobel*'); berfungsi untuk memasukkan sintaks fungsi *sobel*. Kode imshow(sobHor); berfungsi menunjukkan *sobel* *filter* horizontal dan kode imshow(sobVer); berfungsi menunjukkan *sobel* *filter* vertikal.

|  |
| --- |
| global img    bin = im2bw(img,graythresh(img));  rbHor = edge(bin,'roberts','horizontal');  rbVer = edge(bin,'roberts','vertical');  rbGmag = imgradient(bin,'roberts');    guidata(hObject,handles);  axes(handles.pb2);  imshow(rbHor);  val = sprintf('Roberts Horizontal');  set(handles.st2, 'String', val);    guidata(hObject,handles);  axes(handles.pb3);  imshow(rbVer);  val = sprintf('Roberts Vertical');  set(handles.st3, 'String', val);    guidata(hObject,handles);  axes(handles.pb4);  imshow(rbGmag);  val = sprintf('Roberts Gradient Magnitude');  set(handles.st4, 'String', val); |

**Kode Program 4.42** Metode Segmentasi Roberts Filter

Kode program 4.42 merupakan kode program Segmentasi *Roberts* *Filter*. Kode bin = im2bw(img,graythresh(img)); berfungsi mengubah RGB atau *grayscale* *image* menjadi *Binary lmage*. Kode rbHor= edge(bin, roberts', 'horizontal'); merupakan kode *roberts* *filter* untuk horizontal dan kode rbVer=edge (bin, '*sobel*','vertical'); untuk kode *roberts* *filter* untuk vertikal. Kode rbGmag= imgradient(bin,'roberts'); berfungsi untuk memasukkan sintaks fungsi *roberts*. Kode imshow(rbHor); berfungsi menunjukkan *roberts* *filter* horizontal dan kode imshow(rbVer); berfungsi menunjukkan *roberts* *filter* vertikal.

|  |
| --- |
| global img    bin = im2bw(img,graythresh(img));  pwHor = edge(bin,'prewitt','horizontal');  pwVer = edge(bin,'prewitt','vertical');  pwGmag = imgradient(bin,'prewitt');    guidata(hObject,handles);  axes(handles.pb2);  imshow(pwHor);  val = sprintf('Prewitt Horizontal');  set(handles.st2, 'String', val);    guidata(hObject,handles);  axes(handles.pb3);  imshow(pwVer);  val = sprintf('Prewitt Vertical');  set(handles.st3, 'String', val);    guidata(hObject,handles);  axes(handles.pb4);  imshow(pwGmag);  val = sprintf('Prewitt Gradient Magnitude');  set(handles.st4, 'String', val); |

**Kode Program 4.43** Metode Segmentasi Prewitt Filter

Kode program 4.43 merupakan kode program Segmentasi *Prewitt* *Filter*. Kode bin = im2bw(img,graythresh(img)); berfungsi mengubah RGB atau *grayscale* *image* menjadi *Binary lmage*. Kode pwHor= edge(bin, 'prewitt', 'horizontal'); merupakan kode *prewit* *filter* untuk horizontal dan kode pwVer=edge (bin,'prewwit','vertical'); untuk kode *prewit* *filter* untuk vertikal. Kode pwGmag= imgradient(bin,'prewitt'); berfungsi untuk memasukkan sintaks fungsi *prewit*. Kode imshow(pwHor); berfungsi menunjukkan *prewit* *filter* horizontal dan kode imshow(pwVer); berfungsi menunjukkan *prewit* *filter* vertikal.

|  |
| --- |
| global img    bin = im2bw(img,graythresh(img));  can = edge(bin,'canny');    guidata(hObject,handles);  axes(handles.pb2);  imshow(can);  val = sprintf('Canny');  % null = sprintf(' ');  set(handles.st2, 'String', val);  % set(handles.st3, 'String', null);  % set(handles.st4, 'String', null);  % cla(handles.pb3);  % cla(handles.pb4); |

**Kode Program 4.44** Metode Segmentasi Canny

Kode program 4.44 merupakan metode segmentasi *canny*. Kode bin = im2bw(img,graythresh(img)); berfungsi mengubah RGB atau *grayscale* *image* menjadi *Binary lmage*. Kode can=edge(bin,'canny'); berfungsi untuk memasukkan sintaks fungsi *canny*. Kode imshow(can); berfungsi menunjukkan *canny* *filter*.

|  |
| --- |
| global img    bin = im2bw(img,graythresh(img));  log = edge(bin,'log');    axes(handles.pb3);  imshow(log);    val = sprintf('Laplacian of Gaussian (LOG)');  % null = sprintf(' ');  set(handles.st3, 'String', val);  % set(handles.st3, 'String', null);  % set(handles.st4, 'String', null);  % cla(handles.pb3);  % cla(handles.pb4); |

**Kode Program 4.45** Metode Segmentasi Laplacian of Gaussian

Kode program 4.45 merupakan metode segmentasi *Laplacian of Gaussian.* Kode bin = im2bw(img,graythresh(img)); berfungsi mengubah RGB atau *grayscale* *image* menjadi *Binary lmage*. Kode log=edge(bin,'log'); berfungsi untuk memasukkan sintaks fungsi log. Kode imshow(log);berfungsi menunjukkan LOG *filter*.

|  |
| --- |
| x = 127; %koordinat awal/seed x  y = 127; %koordinat awal/seed y  threshold = 65; %threshold  Phi = false(size(img,1),size(img,2));  ref = true(size(img,1),size(img,2));  PhiOld = Phi;  Phi(uint8(x),uint8(y)) = 1;  while(sum(Phi(:)) ~= sum(PhiOld(:)))  PhiOld = Phi;  segm\_val = img(Phi);  meanSeg = mean(segm\_val);  posVoisinsPhi = imdilate(Phi,strel('disk',1,0)) - Phi;  voisins = find(posVoisinsPhi);  valeursVoisins = img(voisins);  Phi(voisins(valeursVoisins > meanSeg - threshold & valeursVoisins < meanSeg + threshold)) = 1;  end  axes(handles.pb2), imshow(Phi);  guidata(hObject,handles);  val = sprintf('Region Growing');  set(handles.st2, 'String', val); = 127; %koordinat awal/seed x  y = 127; %koordinat awal/seed y  threshold = 65; %threshold  Phi = false(size(img,1),size(img,2));  ref = true(size(img,1),size(img,2));  PhiOld = Phi;  Phi(uint8(x),uint8(y)) = 1;  while(sum(Phi(:)) ~= sum(PhiOld(:)))  PhiOld = Phi;  segm\_val = img(Phi);  meanSeg = mean(segm\_val);  posVoisinsPhi = imdilate(Phi,strel('disk',1,0)) - Phi;  voisins = find(posVoisinsPhi);  valeursVoisins = img(voisins);  Phi(voisins(valeursVoisins > meanSeg - threshold & valeursVoisins < meanSeg + threshold)) = 1;  end  axes(handles.pb2), imshow(Phi);  guidata(hObject,handles);  val = sprintf('Region Growing');  set(handles.st2, 'String', val);% cla(handles.pb4); |

**Kode Program 4.46** Metode Segmentasi Region Growing

Kode Program 4.46 merupakan kode program fungsi segmentasi citra *region growing*. Piksel yang dekat pada daerah yang sama memiliki fitur visual yang sama seperti level keabuan, nilai warna, atau teksturnya. Kumpulan piksel (*seed*) yang telah ditentukan akan dikembangkan areanya dengan penambahan setiap *seed* dimana piksel tetangga memiliki kemiripan karakteristik dengan *seed*. Sehingga, *seed* akan mencoba menemukan segmentasi citra yang akurat ke dalam daerah dengan *property* dimana setiap komponen daerah yang saling berhubungan bertemu dengan salah satu *seed*-nya.

|  |
| --- |
| global img;  bin = im2bw(img,graythresh(img));  dist = bwdist(bin);  ws = watershed(dist);  ws = label2rgb(ws, 'spring', 'c', 'shuffle');  axes(handles.pb3);  imshow(ws);  val = sprintf('Watershed');  set(handles.st3, 'String', val); |

**Kode Program 4.47** Metode Segmentasi Watershed

Kode Program 4.47 merupakan kode program fungsi segementasi citra yang digunakan untuk merepresentasikan citra dalam bentuk tiga dimensi yaitu posisi x dan y dengan masing-masing tingkatan warna yang dimilikinya. Posisi x dan y merupakan bidang dasar dan tingkat warna pixel, yang dalam hal ini adalah citra asli dengan format warna RGB telah diubah menjadi citra abu (*graylevel*) merupakan ketinggian (n) dengan anggapan bahwa nilai yang makin mendekati warna putih mempunyai ketinggian yang semakin tinggi.

### Langkah-Langkah Percobaan Modul 4

Langkah-langkah percobaan praktikum untuk program morfologi citra dilasi, *erosi*, *closing*, *opening*, dan *thinning* serta segmentasi citra *edge detection* dengan menggunakan *sobel filter*, *robert filter*, *prewit filter*, *canny*, dan LOG, serta segmentasi citra berbasis area dengan menggunakan *region growing* dan *watershed*.

|  |
| --- |
|  |

**Gambar 4.73** Dilasi dan Erosi

Gambar 4.73 merupakan gambar hasil perubahan citra asli ke dalam dilasi dan erosi. Gambar hasil citraini didapatkan setelah mengklik *button* *dilation* dan ditampilkan pada *Picturebox*2, serta *button erosion* dan ditampilkan pada *Picturebox*3.

|  |
| --- |
|  |

**Gambar 4.74** Opening dan Closing

Gambar 4.74 merupakan gambar hasil perubahan citra asli ke dalam *Opening* dan *Closing*. Gambar hasil citraini didapatkan setelah mengklik *button* *opening* dan ditampilkan pada *Picturebox*2, serta *button closing* dan ditampilkan pada *Picturebox*3.

|  |
| --- |
|  |

**Gambar 4.75** Thinning

Gambar 4.75 merupakan gambar hasil perubahan citra asli ke dalam *thinning*. Gambar hasil citraini didapatkan setelah mengklik *button* *thinning* dan ditampilkan pada *Picturebox*2.

|  |
| --- |
|  |

**Gambar 4.76** Sobel filter

Gambar 4.76 merupakan gambar hasil perubahan citra asli ke dalam proses segmentasi dengan *sobel filter.* Ketika *button* *sobel filter* ditekan maka akan dilakukan proses deteksi tepi menggunakan *sobel filter* dalam arah horisontal yang ditampilkan pada *picturebox* 2, *sobel filter* dalam arah vertikal ditampilkan pada *picturebox* 3 dan *sobel filter* *magnitudo gradien* ditampilkan pada *picturebox* 4.

|  |
| --- |
|  |

**Gambar 4.77** Robert Filter

Gambar 4.77 merupakan gambar hasil perubahan citra asli ke dalam proses segmentasi dengan *Robert filter*, ketika *button* *Robert filter* ditekan maka akan dilakukan proses deteksi tepi menggunakan *Robert filter* dalam arah horisontal yang ditampilkan pada *picturebox* 2, *Robert filter* dalam arah vertikal ditampilkan pada *picturebox* 3 dan *Robert filter* *magnitudo* gradien ditampilkan pada *picturebox* 4.

|  |
| --- |
|  |

**Gambar 4.78** Prewit filter

Gambar 4.78 merupakan gambar hasil perubahan citra asli ke dalam proses segmentasi dengan *Prewit filter,* ketika *button Prewit filter* ditekan maka akan dilakukan proses deteksi tepi menggunakan *Prewit filter* dalam arah horisontal yang ditampilkan pada *picturebox* 2, *Prewit filter* dalam arah vertikal ditampilkan pada *picturebox* 3 dan *Prewit filter* *magnitudo gradien* ditampilkan pada *picturebox* 4.

|  |
| --- |
|  |

**Gambar 4.79** Canny dan LOG

Gambar 4.79 merupakan gambar hasil perubahan citra asli ke dalam proses segmentasi dengan *edge detection* dengan menggunakan *filter canny* yang ditampilkan pada *picturebox* 2 dan menggunakan *filter* *laplacian of gaussian* (log)yang ditampilkan pada *picturebox* 3.

|  |
| --- |
|  |

**Gambar 4.80** Region growing dan Watershed

Gambar 4.80 merupakan gambar hasil perubahan citra asli ke dalam proses segmentasi berbasis area dengan metode *region growing* yang ditampilkan pada *picturebox* 2 dan *Watershed* yang ditampilkan pada *picturebox* 3.

### Analisis Perbandingan Citra

Berikut merupakan analisis perbandingan dari program morfologi citra dilasi, *erosi*, *closing*, *opening*, dan *Thinning* serta segmentasi citra *edge detection* dengan menggunakan *sobel filter*, *robert filter*, *prewit filter*, *canny*, dan LOG, serta segmentasi citra berbasis area dengan menggunakan *region growing* dan *watershed*.

|  |
| --- |
|  |

**Gambar 4.82** Perbandingan dilasi dan erosi dengan SE 3x3

Gambar 4.82 merupakan perbandingan tiga gambar berbeda yang dilakukan proses dilasi dan erosi dengan menggunakan SE (Struktur Elemen) 3x3, dimana terdapat perbedaan dari kedua hasil morfologi tersebut. Citra hasil morfologi dilasi lebih banyak bintik putih karena proses dilasi memperbesar objek yang memiliki pixel dengan nilai 1, sedangkan citra hasil morfologi erosi adalah kebalikan dari dilasi. Hasil dari proses erosi adalah mempersempit atau mengurangi objek yang memiliki pixel bernilai 1.

|  |
| --- |
|  |

**Gambar 4.83** Perbandingan dilasi dan erosi dengan SE 5x5

Gambar 4.83 merupakan perbandingan tiga gambar berbeda yang dilakukan proses dilasi dan erosi dengan menggunakan SE (Struktur Elemen) 5x5, dimana terdapat perbedaan dari kedua hasil morfologi tersebut. Citra hasil morfologi dilasi lebih banyak bintik putih karena proses dilasi memperbesar objek yang memiliki pixel dengan nilai 1, sedangkan citra hasil morfologi erosi adalah kebalikan dari dilasi. Hasil dari proses erosi adalah mempersempit atau mengurangi objek yang memiliki pixel bernilai 1.

|  |
| --- |
|  |

**Gambar 4.84** Perbandingan dilasi dan erosi dengan SE 7x7

Gambar 4.84 merupakan perbandingan tiga gambar berbeda yang dilakukan proses dilasi dan erosi dengan menggunakan SE (Struktur Elemen) 7x7, dimana terdapat perbedaan dari kedua hasil morfologi tersebut. Citra hasil morfologi dilasi dan erosi dengan kernel 7x7 mengasilkan banyak bintik putih yang lebih besar pada proses dilasi, sedangkan pada proses erosi tidak terdapat bintik putih dan objek dengan pixel bernilai 1 menjadi lebih mengecil.

|  |
| --- |
|  |

**Gambar 4.85** Perbandingan opening dan closing dengan SE 3x3

Gambar 4.85 merupakan percobaan perbandingan operasi opening dan closing dengan menggunakan SE (Struktur Elemen) 3x3. Perbedaan yang terlihat jelas pada operasi ini adalah hasil dari operasi morfologi closing lebih banyak noise atau bintik-bintik putih dibandingkan dengan hasil dari operasi opening.

|  |
| --- |
|  |

**Gambar 4.86** Perbandingan opening dan closing dengan SE 5x5

Gambar 4.86 merupakan percobaan perbandingan operasi opening dan closing dengan menggunakan SE (Struktur Elemen) 5x5. Perbedaan yang terlihat jelas pada operasi ini adalah hasil dari operasi morfologi closing dan opening dengan menggunakan kernel 5x5 membuat noise pada citra opening menjadi berkurang, sedangkan pada citra hasil closing noise menjadi lebih banyak dan terlihat jelas.

|  |
| --- |
|  |

**Gambar 4.87** Perbandingan opening dan closing dengan SE 7x7

Gambar 4.87 merupakan percobaan perbandingan operasi opening dan closing dengan menggunakan SE (Struktur Elemen) 7x7. Perbedaan yang terlihat jelas pada operasi ini adalah hasil dari operasi morfologi closing noise yang dihasilkan menjadi lebih banyak dan besar dibandingkan dengan hasil dari operasi opening yang bersih dari noise.

|  |
| --- |
|  |

**Gambar 4.88** Perbandingan thining

Gambar 4.88 merupakan hasil dari operasi morfologi *thining*, dimana pada operasi ini digunakan pada 3 citra yang berbeda. Hasil oprasi *thining* pada 3 citra yang berbeda berhasil menghasilkan struktur serat atau tulang dari objek, namun pada citra yang ke-2 objek lingkaran menjadi hilang.

|  |
| --- |
|  |

**Gambar 4.89** Perbandingan Sobel Filter

Gambar 4.89 merupakan hasil dari filter sobel. Proses sobel yang dilakukan menggunakan 3 citra yang berbeda, yaitu citra berwarna, citra biner, dan citra *grayscale*. Hasil uji coba yang dilakukan menunjukan hasil yang baik mulai dari yang horizontal, vertikal, maupun magnitude gradien pada setiap citra yang di uji namun pada sobel magnitude gradien menunjukan hasil citra yang lebih baik dan detail.

|  |
| --- |
|  |

**Gambar 4.90** Perbandingan Robert Filter

Gambar 4.90 merupakan hasil dari filter Robert. Proses robert yang dilakukan menggunakan 3 citra yang berbeda, yaitu citra berwarna, citra biner, dan citra *grayscale*. Hasil uji coba yang dilakukan menunjukan hasil yang sama seperti filter sobel baik mulai dari yang horizontal, vertikal, maupun magnitude gradien pada setiap citra yang di uji namun pada robert magnitude gradien menunjukan hasil citra yang lebih terang dan detail.

|  |
| --- |
|  |

**Gambar 4.91** Perbandingan Prewit Filter

Gambar 4.91 merupakan hasil dari filter prewitt. Proses prewitt yang dilakukan menggunakan 3 citra yang berbeda, yaitu citra berwarna, citra biner, dan citra *grayscale*. Hasil uji coba yang dilakukan menunjukan hasil yang sama seperti filter sobel dan robert, baik mulai dari yang horizontal, vertikal, maupun magnitude gradien pada setiap citra yang di uji namun pada prewitt magnitude gradien menunjukan hasil citra yang lebih terang dan detail.

|  |
| --- |
|  |

**Gambar 4.92** Perbandingan Canny dan LOG

Gambar 4.92 merupakan perbandingan hasil dari proses *canny* dan LOG. Proses *canny* dan LOG yang dilakukan menggunakan 3 citra yang berbeda, yaitu citra berwarna, citra biner, dan citra *grayscale*. Hasil dari proses *canny* menghasilkan bentuk tepian objek yang jelas dan terang, sedangan hasil dari LOG tepian dari objek menjadi kurang detail dan terdapat noise.

|  |
| --- |
|  |

**Gambar 4.93** Region Growing dan Watershed

Gambar 4.93 merupakan hasil dari proses perbandingan *region growing* dan *watershed*. Proses *region growing* dan *watershed* yang dilakukan menggunakan 3 citra yang berbeda, yaitu citra berwarna, citra biner, dan citra *grayscale*. Hasil dari proses segmentasi *region growing* pada citra biner dan *grayscale* berhasil mensegmentasi objek pada citra.Hasil dari proses segmentasi *watershed* yang dilakukan menghasilkan citra dengan menampilkan wilayah-wilayah dari citra yang di deteksi dan diberi warna atau label yang berbeda-beda sesuai dengan wilayah yang di deteksi.

# BAB V PENUTUP

## Kesimpulan

Kesimpulan yang dapat diambil setelah melakukan pengimplementasian dasar pengolahan citra, perbaikan kualitas citra, transformasi citra dan morfologi serta segmentasi citra adalah sebagai berikut.

1. Dasar pengolahan citra digital sebuah citra (*image*) yaitu mengubah citra berwarna (RGB) ke ruang warna R, G, B, C, M dan Y menggunakan bahasa pemrograman VB dan Matlab dapat diimplementasikan dengan menggunakan fungsi yang sudah terdapat pada Visual Basic dan Matlab. Kualitas hasil pengolahan citra ke ruang warna R, G, B, C, M, dan Y dan waktu pemrosesan pengolahan citra lebih cepat menggunakan Matlab dan lebih baik dari pengolahan citra menggunakan Visual Basic dikarenakan fungsi sudah disediakan di Matlab.
2. Perbaikan kualitas citra dengan melakukan proses *filtering* menggunakan *Low Pass Filter*, *High Pass Filter* dan *Filter Median* dapat diimplementasikan dalam Matlab dengan menggunakan fungsi yang sudah disediakan. Hasil dari proses *filtering* memberikan perbedaan antara hasil dengan menggunakan Visual Basic dengan Matlab. Kualitas hasil perbandingan citra dan waktu pemrosesan pengolahan citra dengan menggunakan Matlab lebih baik dan lebih cepat dibandingkan dengan menggunakan Visual Basic.
3. Transformasi citra dengan menganalisa perubahan citra 2 Dimensi ke ruang frekuensi menggunakan DFT, DCT dan FFT serta perubahan citra 2 Dimensi ke ruang skala dan waktu menggunakan Transformasi *Wavelet* dapat diimplementasikan dengan menggunakan fungsi yang sudah terdapat pada Matlab.
4. Proses Morfologi citra terdiri dari dilasi, erosi, *opening*, dan *closing* baik digunakan dalam melakukan perbaikan citra berbeda dengan proses *thining* yang baik digunakan untuk menentukan struktur dari objek tersebut semisal untuk mencari dari betuk struktur sidik jari. Proses Segmentasi dari proses *filtering* menggunakan Sobel *Filter*, Robert *Filter*, Canny *Filter*, Prewitt *Filter*, dan Laplacian of Gaussian baik digunakan dalam mengsegmentasi bentuk objek karena filter ini menghasilkan pola bentuk dari objek tersebut. Segmentasi menggunakan proses *Region Growing* dapat menghasilkan objek yang dinginkan dengan baik dan detail. Proses segmentasi menggunakan *Watershed* baik digunakan untuk mengsegmentasi objek dengan memberi label dengan warna yang berbeda.

## Saran

Menyadari bahwa penulisan laporan Praktikum Pengolahan Citra Digital masih jauh dari kata sempurna, kedepannya penulisan akan lebih fokus dan detail dalam menjelaskan tentang modul-modul praktikum pengolahan citra dengan sumber–sumber yang lebih banyak yang tentunya dapat dipertanggungjawabkan. Untuk saran bisa berisi kritik atau saran terhadap penulisan juga bisa untuk menanggapi terhadap kesimpulan dari bahasan tiap-tiap modul yang telah dijelaskan. Saran yang dapat diberikan untuk Praktikum Pengolahan Citra Digital adalah sebagai berikut.

1. Metode yang dilakukan dalam praktikum pengolahan citra sudah bagus, tapi mungkin bisa ditingkatkan dari jadwal pelaksanaan agar dapat lebih maksimal melakukan asistensi.
2. Penyediaan materi dari mata kuliah pengolahan citra digital sebaiknya lebih dipercepat agar bisa segera diterapkan pada praktikum, karena dalam beberapa waktu selama pelaksaan praktikum, bahasan materi dari mata kuliah pengolahan citra digital sendiri masih belum tuntas sedangkan modul dari praktikum membutuhkan materi tersebut.

# DAFTAR PUSTAKA

[1] Bernade. 2010. “Dasar Pengolahan Citra Digital”. [online]. https://bernadetastmik.wordpress.com/pengolahan-citra/. [diakses 9 Maret 2018]

[2] Amutiara. 2016. “Fast *Fourier* Transform”. [online]. http://amutiara.staff.gunadarma.ac.id/Downloads/files/39977/Bab5\_Konvolusi+dan+Transformasi+Fourier.pdf. [diakses 10 April 2018].

[3] Hidayat. 2012. “Deteksi Tepi pada Citra”. [online]. https://www.scribd.com/doc/97487310/segmentasi-citra. [diakses 10 April 2018].

[4] Abhishek & Lakshmesha K.N. 2016. “Thinning Approach In Digital Image Processing“. [online]. https://www.ijltet.org/journal/151065732874.pdf. [diakses 19 April 2018].

[5] Dyahapriliani, Murinto. 2013. “Segmentasi Citra Digital”. [online]. https://media.neliti.com/media/publications/210936-none.pdf [diakses 19 April 2018]

[6] M. Rafiqul Islam. 2011. “Wavelets, its Application and Technique in signal

and image processing”. [online]. https://pdfs.semanticscholar.org/48ce/448ffd01a04dff0287c8325575f3cda9f76c.pdf [diakses 20 April 2018]

[7] S.S.Pandey, Manu Pratap Singh, and Vikas Pandey. 2015. Image Transformation and Compression using Fourier Transformation. http://inpressco.com/wp-content/uploads/2015/04/Paper1061178-1182.pdf [diakses 20 April 2018]