APLIKASI PEMBESAR CITRA DIGITAL BERWARNA MENGGUNAKAN MODIFIKASI ALGORITMA *NEW EDGE DIRECTED INTERPOLATION*

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

Oleh:
FRANSISCUS PRIHARSONO
0310960031-96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
2008

APLIKASI PEMBESAR CITRA DIGITAL BERWARNA MENGGUNAKAN MODIFIKASI ALGORITMA *NEW EDGE DIRECTED INTERPOLATION*

ABSTRAK

Algoritma interpolasi telah digunakan sebagai salah satu metode untuk pembesaran citra digital. Pada saat citra digital mengalami pembesaran, citra tersebut mengalami penambahan jumlah piksel. Piksel-piksel baru yang muncul saat pembesaran citra digital belum memiliki nilai. Penggunaan algoritma interpolasi dalam pembesaran citra digital dikarenakan melalui algoritma ini dapat memperkirakan suatu nilai piksel yang belum diketahui nilainya berdasarkan nilai-nilai piksel yang sudah diketahui.

Telah banyak algoritma interpolasi yang telah ditemukan dan digunakan. Salah satunya adalah algoritma interpolasi *New Edge Directed Intepolation* (NEDI) yang ditemukan oleh Xin Li dan Michael Orchard pada tahun 2001. Penelitian yang telah dilakukan oleh penemunya masih diterapkan pada citra *grayscale*.

Oleh karena itu, dalam skripsi ini akan dibuat sebuah aplikasi pembesar citra digital menggunakan algoritma interpolasi NEDI yang telah dimodifikasi yang diaplikasikan pada citra berwarna (model warna RGB). Pemodifikasian dilakukan untuk mengatasi masalah ketidakmampuan algoritma interpolasi NEDI menghasilkan sebuah nilai interpolasi saat terbentuknya matriks singular. Matriks singular akan membuat invers matriks yang merupakan bagian dalam perhitungan tidak dapat dilakukan. Pemodifikasian dilakukan dengan menggunakan algoritma interpolasi bilinear saat interpolasi NEDI tidak bisa menghasilkan sebuah nilai interpolasi.

Selain itu aplikasi yang akan dibuat juga digunakan untuk mengevaluasi citra digital yang dihasilkan. Sehingga melalui penelitian ini dapat diketahui efektivitas algoritma interpolasi modifikasi NEDI dalam memperbesar citra digital berwarna.

Melalui penelitian ini didapatkan bahwa pembesaran citra digital pada citra berwarna dapat dilakukan dengan menerapkan algoritma interpolasi modifikasi NEDI ini pada setiap elemen warnanya (R,G, dan B). Kesimpulan lain yang didapat melalui penelitian ini adalah citra hasil pembesaran menggunakan interpolasi NEDI dipengaruhi oleh jumlah tetangga yang digunakan. Semakin

banyak jumlah tetangga yang digunakan maka semakin baik citra hasil pembesarannya. Selain itu, citra hasil pembesaran menggunakan interpolasi modifikasi NEDI dipengaruhi oleh banyaknya piksel yang mempunyai nilai jauh berbeda dengan piksel–piksel tetangganya. Semakin banyak sebuah citra asli memiliki piksel yang mempunyai nilai jauh berbeda dengan piksel-piksel tetangganya, semakin jelek citra hasil pembesarannya

DIGITAL COLOR IMAGE MAGNIFYING APPLICATION USING MODIFICATION *NEW EDGE DIRECTED INTERPOLATION* ALGORITHM

ABSTRACT

Interpolation has used as a method for digital image magnifying. When digital image magnified, that image gets increasing in its total pixel. The new pixels which appear when digital image magnified don't have any value. A method which can used to give a value for those new pixel is interpolation. Interpolation can estimate a value for those new pixel based on pixels around them which have value.

Many interpolation algorithm have founded and used. One of them is New Edge Directed Interpolation which has founded by Xin Li and Michael Orchard in 2001. This algorithm more populer known as NEDI. The former experiment with NEDI is done by its founder in magnifying grayscale image with NEDI.

In this thesis, a modification of NEDI used for digital color image magnifiying. Modification is done to solve the problem lack of ability of NEDI to produce a interpolation value when singular matrix formed. When singular matrix formed, invers matrix which is a part from calculation NEDI, can not be done. Modification is done by using bilinear interpolation to produce a interpolation value.

In addition, the application will be made used for evaluate digital image which resulted by this application. Through this experiment, can be known the effectiveness modfication of NEDI in magnify digital color image.

Results in this experiment are digital color image magnifiying using modification NEDI can be done by apply this algorithm for each color channel (Red, Green, and Blue), the resulted image depends on amount neighbor which used. The more total neighbor used the better result can be obtained. The resulted image also depends on the number of pixel which have big differences value with the other neighbor pixel. More the number of pixel which have big differences value with the other neighbor pixel which had by a digital image more worst the resulted image can be obtained.

KATA PENGANTAR

Puji dan syukur kepada Allah, karena dengan berkat dan rahmat-Nya, juga bimbingan dan kekuatan-Nya, maka Skripsi yang berjudul "Aplikasi Pembesar Citra Digital Berwarna dengan Menggunakan Modifikasi Algoritma *New Edge Directed Interpolation* " ini dapat diselesaikan. Skripsi ini disusun dan diajukan sebagai syarat untuk memperoleh gelar sarjana pada Program Studi Ilmu Komputer, Jurusan Matematika, Fakultas MIPA, Universitas Brawijaya.

Adanya keinginan untuk menjaga kualitas citra pada saat dilakukan pembesaran citra, telah menarik minat penulis dalam melakukan penelitian. Penelitian yang dilakukan pada Skripsi ini ditujukan untuk meneliti penerapan algoritma interpolasi NEDI yang dimodifikasi untuk pembesaran citra.

Dalam menyelesaikan Skripsi ini, banyak berbagai bentuk dukungan dan bantuan yang penulis telah dapatkan, sehingga penulis ingin menyampaikan penghargaan dan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Drs. Marji, MT., selaku pembimbing Tugas Akhir dan pembimbing Akademik atas waktu, dorongan semangat, kritik dan saran, serta bimbingan yang telah diberikan.
2. Nurul Hidayat, Spd., M.Sc., selaku pembimbing Tugas Akhir atas waktu, dorongan semangat, kritik dan saran, serta bimbingan yang telah diberikan.
3. Wayan F. Mahmudy, S.Si., MT., selaku Ketua Program Studi Ilmu Komputer Universitas Brawijaya Malang.
4. Dr. Agus Suryanto, M.Sc., selaku Ketua Jurusan Matematika Universitas Brawijaya Malang.
5. Bapak, ibu dan adik (E. Priyo Setiono, Stefana Tri Raharsi dan Maria Puji N.) yang senantiasa berdoa dan memberi dukungan dan semangat.
6. Seluruh anggota keluarga besar penulis yang selalu memberikan dukungan, nasehat dan bantuan.
7. Maria Kristantina W., terimakasih atas cinta, semangat, doa dan bantuannya.
8. Sahabat-sahabat *WeDoes*, terimakasih atas semangat dan bantuannya.
9. Semua teman-teman Ilmu Komputer angkatan 2003. Terima kasih atas semangat dan doanya.

10. Pihak lain yang tidak bisa penulis sebutkan satu-persatu.

Semoga penulisan laporan Skripsi ini bermanfaat bagi pembaca sekalian. Penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan sehingga penulis mengharapkan kritik dan saran yang membangun dari pembaca.

Malang, 18 Juni 2008

Penulis

DAFTAR ISI

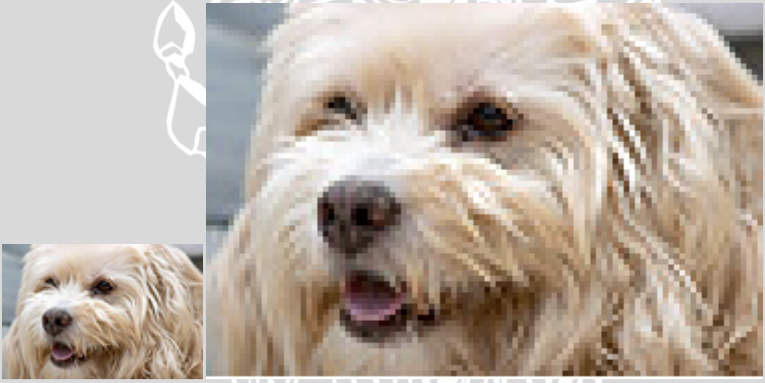
DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Fasilitas pembesaran citra digital telah disediakan oleh *software* pengolah citra digital. Fasilitas ini disediakan untuk mengatasi permasalahan yang timbul saat sebuah citra diperbesar ukurannya. Masalah yang akan timbul saat citra digital diperbesar adalah hasil pembesarannya akan terlihat seperti "pecah-pecah" atau terlihat "kotak-kotak". Citra terlihat "pecah-pecah" atau terlihat "kotak-kotak" dapat terjadi karena citra digital akan mengalami penambahan jumlah piksel saat diperbesar dan piksel-piksel tesebut mempunyai nilai yang sama dengan tetangga terdekatnya. GAMBAR 1.1 mengilustrasikan perbesaran citra berukuran kecil dan masalah yang timbul pada hasil citra setelah diperbesar. Pada GAMBAR 1.1(a) merupakan citra asli berukuran kecil. GAMBAR 1.1 (b) merupakan citra asli yang telah diperbesar dan hasilnya terlihat "kotak-kotak".



(a) (b)

Gambar 1.1 Perbesaran citra digital berukuran kecil: (a) citra digital asli; (b) citra digital yang telah dibesarkan.

Oleh karena itu, untuk mengatasi masalah yang timbul pada waktu pembesaran citra digital**,** diperlukan adanya sebuah metode yang mampu menemukan atau memberi perkiraan nilai yang tepat kepada piksel-piksel baru yang muncul saat proses pembesaran. Metode ini diharapkan mampu menghasilkan sebuah citra digital

yang berukuran besar dengan kualitas yang hampir sama dengan citra digital aslinya yang berukuran kecil.

Salah satu metode yang dapat digunakan dalam perbesaran citra digital adalah metode interpolasi. Interpolasi adalah sebuah teknik perkiraan dimana diberikan sebuah set data untuk mencari data lainnya berdasarkan data yang sudah ada (Chi woo,2007). Dari pengertian tersebut maka dapat dijelaskan bahwa interpolasi citra digital adalah sebuah metode yang digunakan untuk mendapatkan perkiraan nilai untuk sebuah nilai piksel yang belum diketahui melalui piksel yang sudah diketahui nilainya.

Beberapa algoritma interpolasi citra digital yang telah ditemukan antara lain interpolasi *Nearest Neighbourhood*, interpolasi *bilinear*, interpolasi *bicubic* yang ditemukan oleh A.N. Netravali dan B.G. Hasskell pada tahun1995, interpolasi *fractal*, E*dge Directed Interpolation* atau sering disebut EDI yang ditemukan oleh Jan Allebach dan Ping Wong pada tahun 1996, *New Edge Directed Intepolation* atau dapat disingkat NEDI ditemukan oleh Xin Li and Michael Orchard pada tahun 2001 yang merupakan pengembangan dari algoritma EDI, *Data-Dependent Triangulation Interpolation* yang ditemukan oleh Su dan Willis pada tahun 2004, dan masih banyak yang lain.

Setiap algoritma interpolasi yang telah disebutkan akan menghasilkan citra hasil interpolasi yang mempunyai hasil dan kualitas yang berbeda jika diaplikasikan pada citra digital yang sama. Xin Li menegaskan bahwa kualitas sebuah citra hasil interpolasi sangat bergantung pada kemulusan sepanjang tepi dan ketajaman di semua tepi. Untuk mendapatkan kualitas citra hasil interpolasi yang baik, Xin Li melalui penelitiannya menemukan sebuah algoritma baru yang diberi nama NEDI yang diharapkan mampu menghasilkan kemulusan sepanjang tepi dan ketajaman untuk semua tepi.

Penelitian yang telah dilakukan oleh Xin Li and Michael Orchard masih diterapkan pada citra berwarna keabuan (*grayscale*). Oleh karena itu, dalam tugas akhir ini akan dibuat sebuah aplikasi pembesar citra digital menggunakan algoritma interpolasi NEDI yang diaplikasikan pada citra berwarna (model warna RGB).

Dalam perhitungan algoritma interpolasi NEDI terdapat perhitungan invers matriks. Invers matriks tidak dapat dilakukan jika matriks yang akan diinvers merupakan matriks singular dan mengakibatkan nilai interpolasi NEDI juga tidak bisa didapatkan.

Untuk mengatasi permasalahan apabila ditemukan matriks singular pada perhitungan algoritma interpolasi NEDI maka dilakukan modifikasi yaitu dengan menggunakan algoritma interpolasi bilinear untuk mencari nilai interpolasi untuk sebuah piksel yang belum bernilai. Hasil dari perhitungan algoritma interpolasi bilinearlah yang akan digunakan untuk memberi nilai pada piksel yang belum bernilai jika algoritma interpolasi NEDI tidak menghasilkan sebuah nilai.

Selain itu aplikasi yang akan dibuat juga digunakan untuk mengevaluasi citra digital yang dihasilkan. Sehingga melalui penelitian ini dapat diketahui efektivitas pemakaian algoritma interpolasi NEDI yang dimodifikasi dalam memperbesar citra digital berwarna.

Berdasarkan latar belakang yang telah dipaparkan, maka penulis mengambil judul pada skripsi ini "**Aplikasi Pembesar Citra Digital Berwarna Menggunakan Modifikasi Algoritma *New Edge Directed Interpolation***".

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah, permasalahan skripsi ini dapat dirumuskan sebagai berikut:

1. Bagaimana mengimplementasikan modifikasi algoritma interpolasi NEDI untuk pembesaran citra digital berwarna sehingga hasil pembesaran tidak jauh berbeda dengan citra aslinya yang berukuran kecil.
2. Bagaimana kualitas citra hasil pembesaran dengan modifikasi algoritma NEDI yang diukur dengan metode *Mean Square Error* (MSE) dan sebuah metode yang menggunakan *Discrete Wavelet Transform* (DWT) yaitu M-DWT.
3. Bagaimana perbandingan hasil pembesaran citra digital menggunakan modifikasi algoritma NEDI dengan algoritma bilinear

1.3 Batasan Masalah

Untuk menghindari melebarnya masalah yang akan dibahas, diberikan batasan masalah sebagai berikut:

1. Penelitian hanya berfokus pada penggunaan modifikasi algoritma NEDI untuk pembesaran citra digital berwarna dengan skala pembesaran tertentu.
2. Menggunakan citra berekstensi *Bitmap (.bmp)* karena format citra *Bitmap* belum mengalami proses kompresi citra.
3. Citra yang diproses adalah citra dengan model warna RGB dan mempunyai kedalaman warna 24 *bit*.
4. Citra yang dapat diproses mempunyai tinggi dan lebar masing-masing minimal 50 piksel dan maksimal 1024 piksel.
5. Skala pembesaran yang digunakan untuk pengujian bernilai 2, 4, dan 8.
6. Pengujian citra hasil pembesaran mengabaikan pengaruh dari proses pengecilan citra uji.
7. Pembesaran dengan algoritma interpolasi bilinear dan pengujian citra hasil pembesarannya dilakukan di luar aplikasi yang akan dibuat.
8. Pemrosesan pembesaran citra berwarna dilakukan pada domain spasial.

1.4 Tujuan Penelitian

Tujuan penelitian yang ingin dicapai adalah:

1. Mengimplementasikan modifikasi algoritma interpolasi NEDI untuk pembesaran citra digital berwarna sehingga hasil pembesaran tidak jauh berbeda dengan citra aslinya yang berukuran kecil.
2. Mengevaluasi kualitas citra hasil pembesaran dengan modifikasi algoritma NEDI yang diukur dengan metode *Mean Square Error* (MSE) dan sebuah metode yang menggunakan *Discrete Wavelet Transform* (DWT) yaitu M-DWT.
3. Membandingkan citra digital hasil pembesaran menggunakan modifikasi algoritma NEDI dengan citra digital hasil pembesaran menggunakan algoritma bilinear

1.5 Manfaat Penelitian

Manfaat dari penulisan skripsi ini adalah menyediakan *software* (aplikasi) yang dapat menerapkan modifikasi algoritma interpolasi NEDI sebagai metode pembesaran citra digital berwarna.

1.6 Metodologi Pemecahan Masalah

Untuk mencapai tujuan yang dirumuskan sebelumnya, maka metodologi yang digunakan dalam penulisan tugas akhir ini adalah:

1. Studi Literatur

   Mempelajari teori-teori yang berhubungan dengan algoritma interpolasi NEDI, algoritma interpolasi bilinear dan metode pengukuran kualitas citra hasil pembesaran menggunakan metode MSE dan M-DWT dari berbagai referensi.
2. Pendefinisian dan analisis masalah

   Mendefinisikan dan menganalisis masalah untuk mencari solusi yang tepat.
3. Perancangan dan implementasi sistem

   Membuat perancangan perangkat lunak dan mengimplementasikan hasil rancangan tersebut yaitu membuat perangkat lunak yang menggunakan modifikasi interpolasi NEDI untuk pembesaran citra digital berwarna.
4. Uji coba dan analisa hasil implementasi

   Menguji perangkat lunak, dan menganalisa hasil dari implementasi tersebut apakah sudah sesuai dengan tujuan yang dirumuskan sebelumnya, untuk kemudian dievaluasi.

1.7 Sistematika Penulisan

Skripsi ini disusun berdasarkan sistematika penulisan sebagai berikut:

1. BAB I PENDAHULUAN

   Berisi latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi pemecahan masalah, dan sistematika penulisan.
2. BAB II TINJAUAN PUSTAKA

   Menguraikan teori-teori yang berhubungan dengan konsep citra digital, interpolasi citra digital, algoritma interpolasi NEDI, algoritma interpolasi bilinear, MSE, dan M-DWT.
3. BAB III METODE DAN PERANCANGAN

   Pada bab ini akan dijelaskan mengenai penggunaan teori-teori dalam perancangan perangkat lunak

menggunakan modifikasi interpolasi NEDI untuk pembesaran citra digital berwarna. Serta membahas bagaimana perangkat lunak ini direncanakan, diimplementasikan dan dievaluasi.

4. BAB IV IMPLEMENTASI DAN UJI COBA SISTEM

   Pada bab ini akan dilakukan implementasi sistem, pengujian dan analisa sistem perangkat lunak yang dibangun, yaitu mengevaluasi kualitas hasil citra digital yang dihasilkan, akan dilakukan dengan cara membandingkan sebuah citra asli (citra berukuran besar) dengan sebuah citra hasil pembesaran dari sebuah citra asli yang dikecilkan ukurannya. Perbandingan akan dilakukan menggunakan MSE dan juga metode M-DWT. Perbandingan juga akan dilakukan terhadap citra digital hasil pembesaran menggunakan modifikasi algoritma NEDI dengan citra digital hasil pembesaran menggunakan algoritma bilinear

5. BAB V KESIMPULAN DAN SARAN

   Bab ini berisi kesimpulan dari seluruh rangkaian penelitian serta saran kemungkinan pengembangannya.

BAB II
TINJAUAN PUSTAKA

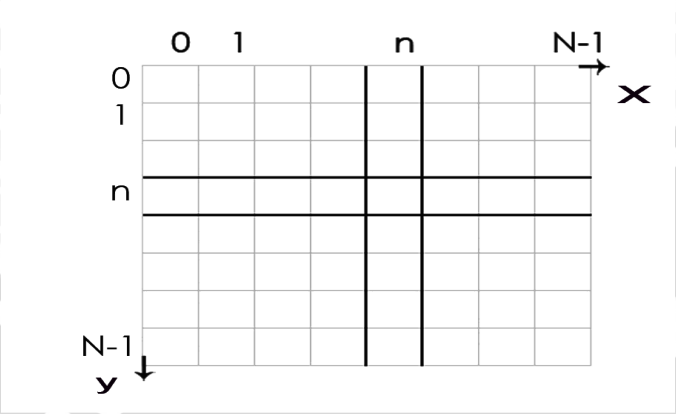
2.1 Konsep Dasar Citra Digital

Menurut Balza Ahmad dan Kartika Firdausy (2005), citra dapat dikelompokkan menjadi citra tampak dan tidak tampak. Contoh dari citra tampak adalah foto, lukisan, dsb. Sedangkan citra tak tampak misalnya citra digital, fungsi matematis ataupun citra distribusi panas di tubuh manusia. Agar dapat dilihat oleh mata manusia maka citra tak tampak harus ditampilkan dalam monitor atau dicetak di atas kertas

Diantara contoh citra tampak dan tak tampak, hanya citra digital yang dapat diolah menggunakan komputer. Apabila ingin melakukan perubahan terhadap citra analog ke citra digital maka citra analog dapat didigitalkan dengan menggunakan sebuah alat yang dapat mengubahnya ke dalam bentuk digital melalui proses pemindaian (*scanning*) (Balza dan Kartika, 2005).

Citra digital adalah representasi citra dalam bentuk diskrit, baik pada koordinat spasial maupun intensitas cahayanya (Gonzalez,1992). Sebuah citra digital dapat direpresentasikan ke dalam sebuah matriks berukuran *M* x *N* yang merupakan koordinat sebuah titik pada citra. Elemen dari matriks ini mengandung sebuah nilai yang menyatakan intensitas cahaya dari citra tersebut. Suatu titik pada sebuah citra digital sering disebut sebagai *image element, picture element,* atau piksel.

Representasi spasial dari sebuah citra adalah piksel, yang berupa koordinat titik dari tiap citra. Di dalam tiap piksel mengandung informasi tentang tingkat *irradiance* atau tingkat kecerahan suatu citra. GAMBAR 2.1 menunjukkan bahwa susunan piksel sama seperti susunan matriks, hanya saja, awal koordinat dimulai dari atas ke kanan (kolom) untuk *x* dan ke bawah (baris) untuk *y*. Sama seperti array, indeks piksel dimulai dari 0, untuk piksel yang berada pada koordinat X bergerak dari 0 hingga *N-1* dimana N adalah bilangan bulat positif, begitu juga halnya dengan piksel pada koordinat Y bergerak dari 0 hingga *N-1* dimana N adalah bilangan bulat positif (Jähne,2002).



Gambar 2.1 Representasi spasial citra – piksel.

GAMBAR 2.2 menunjukkan pengaruh banyaknya piksel yang dipunyai oleh sebuah citra. GAMBAR 2.2 (a) adalah sebuah citra yang berukuran 4 x 3. GAMBAR 2.2 (b) adalah sebuah citra yang berukuran 24 x 18. GAMBAR 2.2 (c) adalah sebuah citra yang berukuran 64 x 48. GAMBAR 2.2 (d) adalah sebuah citra yang berukuran 256 x 192. Contoh citra pada GAMBAR 2.2 menunjukkan bahwa semakin banyak piksel yang mewakili suatu citra maka citra terlihat lebih halus. Tidak ada ukuran yang khusus mengenai ukuran ideal suatu citra, namun apabila citra sudah dalam bentuk digital dan mengalami pengurangan ukuran piksel maka informasi akan berkurang sedikit demi sedikit.

(a)

(b)

(c)

(d)

Gambar 2.2 Pengaruh ukuran piksel terhadap kualitas citra: (a) citra yang berukuran 4 x 3; (b) citra yang berukuran 24 x 18; (c) citra yang berukuran 64 x 48; (d) citra yang berukuran 256 x 192.

2.2 RGB

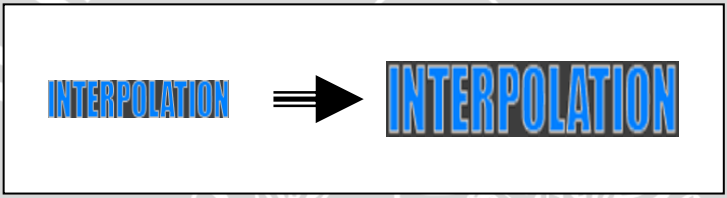
RGB (*red*, *green*, and *blue*) adalah sebuah sistem untuk mewakili warna yang digunakan pada sebuah tampilan komputer. RGB dapat dikombinasikan dalam berbagai macam ukuran untuk mendapatkan banyak warna dalam spektrum yang terlihat. Nilai dari R, G, dan B mempunyai range antara 0 sampai 100 persen dari intensitas penuh. Setiap nilai diwakili oleh kisaran angka dari 0 sampai 255 (256 untuk setiap warna), yang setara dengan kisaran angka biner dari 00000000 sampai 11111111, atau heksadesimal 00 sampai FF. Jumlah total warna yang tersedia adalah 256 x 256 x 256 atau 16,777,216 warna (Anonymous, 1999).

2.3 Interpolasi Citra Digital

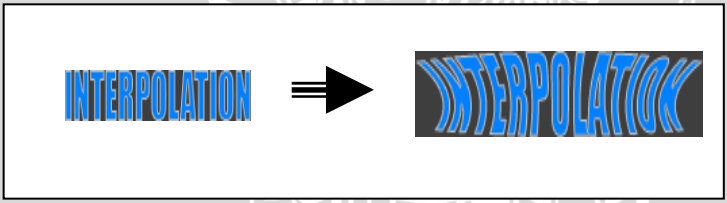
Interpolasi adalah sebuah teknik perkiraan dimana diberikan sebuah set data untuk mencari data lainnya berdasarkan data yang sudah ada (Chi woo,2007). Dari pengertian tersebut maka dapat dijelaskan bahwa interpolasi citra digital adalah sebuah metode yang digunakan untuk mendapatkan perkiraan nilai yang belum diketahui pada sebuah piksel citra digital melalui piksel yang sudah diketahui nilainya. Proses interpolasi citra digital mencoba mendapatkan sebuah pendekatan terbaik untuk nilai piksel berdasarkan nilai piksel sekelilingnya. Interpolasi citra digital hanya sebuah metode

pendekatan, karena itu sebuah citra akan selalu kehilangan kualitasnya setiap kali interpolasi dilakukan.

Interpolasi citra digital terjadi setiap kali dilakukan *rezise* (perubahan ukuran resolusi citra) atau *remap* (pemetaan kembali piksel) dari sebuah susunan piksel tertentu ke susunan piksel yang lain. *Rezise* diperlukan saat menambah atau mengurangi jumlah piksel, sebaliknya *remapping* dapat terjadi dalam kasus yang lebih luas seperti distorsi citra digital, perubahan perspektif, dan rotasi citra digital. GAMBAR 2.3 menunjukkan contoh proses interpolasi pada citra digital. Pada GAMBAR 2.3 (a) ditunjukkan proses interpolasi yang terjadi saat dilakukan *resize* pada citra digital. Pada GAMBAR 2.3 (b) ditunjukkan proses interpolasi yang terjadi saat dilakukan distorsi pada citra digital.



(a)

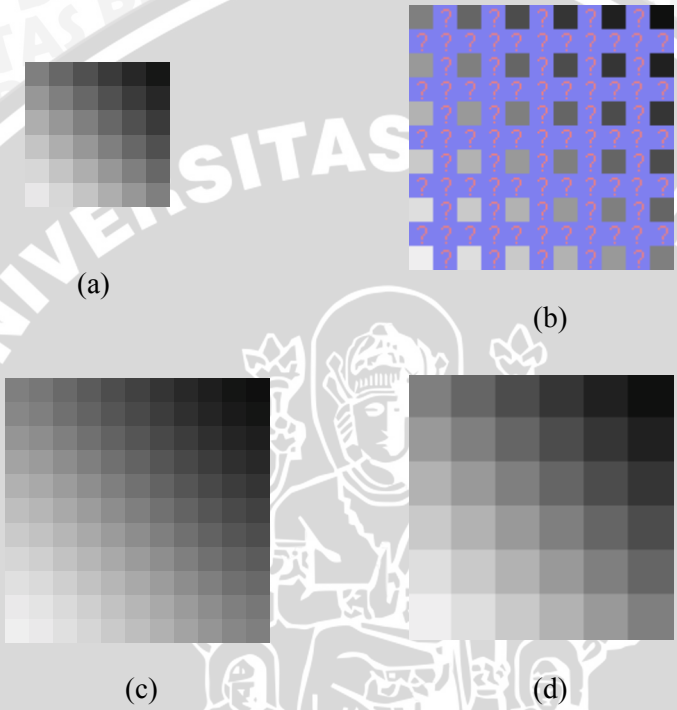


(b)

Gambar 2.3 Contoh proses interpolasi pada citra digital: (a) contoh interpolasi pada *rezise* citra; (b) contoh interpolasi pasa distorsi citra. (Anonymous, 2007[a])

Penelitan yang akan dilakukan adalah menggunakan interpolasi untuk pembesaran citra digital. GAMBAR 2.4 menunjukkan contoh interpolasi yang terjadi pada proses pembesaran citra digital. GAMBAR 2.4 (a) adalah sebuah citra asli. GAMBAR 2.4 (b) menunjukkan citra asli yang dipebesar sebelum dilakukan interpolasi sehingga tampak beberapa piksel yang belum diketahui nilainya.

GAMBAR 2.4 (c) menunjukkan citra pada GAMBAR 2.4 (b) setelah dilakukan interpolasi. GAMBAR 2.4 (d) menunjukkan citra pada GAMBAR 2.4 (b) jika tidak dilakukan interpolasi.



Gambar 2.4 Proses interpolasi pada pembesaran citra digital: (a) citra asli; (b) citra setelah dilakukan pembesaran; (c) citra yang mengalami proses interpolasi; (d) citra yang tidak mengalami proses interpolasi. (Anonymous, 2007[a])

Untuk citra digital yang sama, citra hasil pembesaran dapat bervariasi bergantung pada algoritma interpolasi yang digunakan pada saat dilakukan pembesaran citra digital.

Beberapa contoh algoritma interpolasi citra digital yang telah ditemukan adalah: *bilinear*, *bicubic*, *spline*, *sicn*, *lanczos*, NEDI dan masih banyak lainnya. Algoritma-algoritma ini menggunakan dari 2 sampai 256 (atau lebih) piksel tetangga saat proses interpolasi. Semakin banyak piksel tetangga yang digunakan, semakin akurat

algoritma interpolasi itu, tetapi waktu proses yang dibutuhkan menjadi semakin lama.

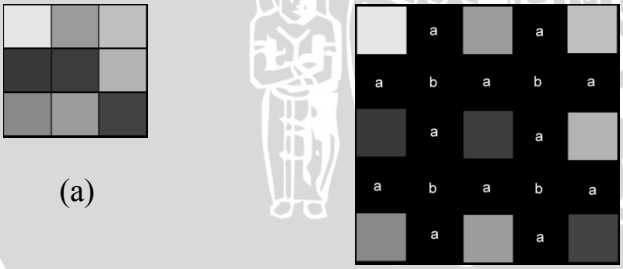
2.3.1 Interpolasi NEDI

Algoritma NEDI ditemukan oleh Xin Li dan Michael Orchard yang merupakan pengembangan dari algoritma EDI. Interpolasi NEDI merupakan interpolasi *covariance-based adaptive* sehingga perhitungan pada algoritma interpolasi NEDI berdasarkan perhitungan *covariance*. *Covariance* adalah ukuran seberapa jauh perbedaan dari dua variabel acak.

Ide dasar dari algoritma NEDI adalah memperkirakan koefisien lokal *covariance* dari sebuah citra berukuran kecil dan menggunakan perkiraan *covariance* tersebut untuk melakukan interpolasi pada citra yang berukuran lebih besar (Li dan Orchard, 2001).

Secara umum cara kerja dari interpolasi adalah melakukan pendekatan dengan menghitung bobot rata-rata dari piksel tetangga yang terdekat dari piksel yang belum diketahui nilainya. Cara kerja yang digunakan pada algoritma NEDI adalah menghitung lokal *covariances* dalam citra asli(citra yang berukuran kecil) dan menggunakannya untuk menghitung bobot interpolasi. Dan bobot interpolasi ini digunakan untuk mendapatkan nilai perkiraan piksel.

GAMBAR 2.5 menunjukkan sebuah contoh kasus pembesaran pada sebuah citra sebelum dilakukan interpolasi.



(a) (b)

Gambar 2.5 Pembesaran pada sebuah citra digital *grayscale* sebelum dilakukan interpolasi: (a) citra asli; (b) citra yang diperbesar sebelum dilakukan proses interpolasi. (Anonymous, 2007[b])

Citra pada GAMBAR 2.5 (a) mewakili citra *grayscale* berukuran 3 x 3. Setiap kotak adalah sebuah piksel dan garis diantara mereka adalah hanya untuk ilustrasi. Citra pada GAMBAR 2.5 (b) merupakan hasil pembesaran sebelum dilakukan interpolasi dari citra pada GAMBAR 2.5 (a) sehingga menjadi berukuran 6 x 6 (dengan kolom dan baris terakhir dihilangkan sehingga menjadi 5 x 5). Oleh karena itu pada citra yang berukuran besar akan terlihat 16 piksel lebih banyak dari citra asli. Karena belum dilakukan interpolasi untuk memberi suatu nilai, piksel-piksel tersebut dianggap bernilai nol. Piksel yang diberi tanda "*a* " pada GAMBAR 2.5 (b) sama sekali tidak mempunyai piksel tetangga berdekatan secara diagonal. Piksel yang ditandai " *b* " mempunyai tetangga berdekatan secara diagonal yang nilainya telah ada atau diketahui.

Untungnya, permasalahan tetangga diagonal atau aksial tidak menimbulkan perbedaan di dalam NEDI. *Covariances* dapat dihitung untuk semua orientasi sumbu manapun, jadi untuk piksel *b* dapat digunakan perbedaan sumbu diagonal dan bukan perbedaan sumbu aksial (sumbu xy). Perhitungan *covariances* dilakukan di dalam sebuah *window* tersendiri dengan ukuran tertentu.

Pembesaran citra dengan algoritma interpolasi NEDI adalah menskalakan citra berukuran H x W menjadi sebuah citra berukuran KH x KW dengan K adalah skala pembesaran bernilai 2^n dimana n =1,2,3.... Nilai skala pembesaran bernilai 2^n disebabkan proses kerja interpolasi NEDI yang selalu memperbesar citra dengan skala pembesaran 2. Sehingga untuk nilai skala pembesaran lebih dari 2 dilakukan dengan cara mengulang proses yang sama(memperbesar citra dengan skala pembesaran 2) untuk citra hasil proses pembesaran sebelumnya. Sebagai contoh, jika kita ingin memperbesar citra dengan skala pembesaran 4 maka sebuah citra akan diperbesar dengan skala pembesaran 2 terlebih dahulu dan hasilnya kemudian diperbesar kembali dengan skala pembesaran 2.

Interpolasi NEDI akan dilakukan secara 2 tahap perhitungan. Yang pertama akan dihitung piksel *b* dan kemudian piksel *a*. Piksel *b* mempunyai 4 tetangga interpolasi yang sudah diketahui. Piksel *a* akan mempunyai 4 tetangga yang mempunyai nilai setelah piksel *b* ditentukan.

Jika citra berukuran besar adalah Q dan citra berukuran kecil adalah P maka hubungan antara Q dan P adalah $Q = 2P$ sehingga $Q(2x, 2y) = P(x,y)$. Piksel *b* merupakan semua piksel $Q(x, y)$, dimana

nilai *x* ganjil dan nilai *y* ganjil. Sedangkan piksel *a* merupakan semua piksel *Q*(*x*,y), dimana *x* ganjil dan *y* genap atau *x* genap dan *y* ganjil.

Dalam interpolasi NEDI, beberapa variable yang digunakan adalah:

- ***M*** adalah sebuah variabel bernilai 2^n dimana n =1,2,3....
- ***N*** adalah sebuah *window* lokal berukuran M x M**.** ***N*** berisikan piksel tetangga dari piksel yang akan dicari nilainya
- ***y*** merupakan sebuah vektor kolom berukuran M^2 berisikan piksel-piksel dalam *window* ***N***.
- ***y(k)*** adalah piksel ke ***k*** dalam vektor kolom ***y***.
- ***C*** merupakan sebuah matriks berukuran M^2 x 4 berisikan tetangga diagonal/aksial dari setiap piksel dalam vektor kolom *y*. Baris vektor ke ***k*** berisi 4 piksel tetangga diagonal dari ***y(k)***.

Setelah semua variabel tersebut diketahui maka dilakukan penghitungan *covariances* lokal untuk citra asli dengan menggunakan persamaan 2.1 dan 2.2 yang merupakan metode klasik *covariance* (Li dan Orchard, 2001)

$$R = \frac{1}{M * M} * C^T \cdot C \tag{2.1}$$

$$r = \frac{1}{M * M} * C^T \cdot y \tag{2.2}$$

R akan menghasilkan sebuah matriks berukuran 4x4, dan ***r*** akan menghasilkan sebuah vektor kolom. ***R*** dan ***r*** merupakan *covarience* lokal pada citra asli. Kemudian langkah selanjutnya adalah menggunakan kedua perkiraan *covarience* yang telah didapat untuk melakukan interpolasi. Rumus interpolasi yang digunakan adalah interpolasi linear tingkat ke-4 yaitu nilai piksel didapatkan dengan menjumlahkan hasil perkalian bobot interpolasi linear dengan nilai-nilai piksel keempat tetangga terdekat.

Berdasarkan teori klasik *Wiener filtering,* MMSE (*Minimum Mean Square Error*) bobot interpolasi linear optimal dapat dihitung menggunakan persamaan 2.3.

$$\alpha = R^{-1} \cdot r = (C^T \cdot C)^{-1} * (C^T . y) \tag{2.3}$$

Hasil yang didapatkan merupakan sebuah vektor kolom, $\boldsymbol{\alpha}$, yang berisikan 4 bobot interpolasi untuk keempat tetangga. Bobot ini kemudian dikalikan dengan nilai keempat piksel tetangga di sekeliling piksel yang akan dicari nilainya. Dan hasil perkalian ditambahkan untuk menghasilkan sebuah nilai piksel yang akan diisikan ke dalam piksel yang belum diketahui nilainya. Perhitungan pemberian nilai untuk semua piksel *b* dapat dijelaskan melalui persamaan 2.4.

$$Q_{2x+1,2y+1} = \sum_{k=0}^{1} \sum_{l=0}^{1} \alpha_{2k+l} * Q_{2(x+k),2(y+l)} \tag{2.4}$$

Dimana *Q* adalah citra berukuran besar. Untuk piksel *a* perhitungan pencarian nilai bobot menggunakan tetangga secara aksial. Perhitungan pemberian nilai untuk semua piksel *a* pada citra berukuran besar *Q* dapat dijelaskan melalui persamaan 2.5.

$$\begin{aligned} Q_{2x+1,2y}\ atau\ Q_{2x,2y+1} = {} & \alpha_0 * Q_{2x,2y} + \alpha_1 * Q_{2x+1,2y-1} + \\ & \alpha_2 * Q_{2x+1,2y+1} + \alpha_3 * Q_{2x+2,2y} \end{aligned} \tag{2.5}$$

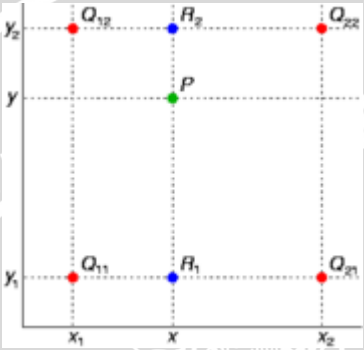
Pada penelitian yang akan dilakukan, algoritma interpolasi NEDI akan diaplikasikan pada citra berwarna dengan menggunakan proses yang sama untuk citra *grayscale* pada tiap-tiap elemen warnanya (*Red*, *Green*, dan *Blue*).

2.3.2 Interpolasi Bilinear

Interpolasi *bilinear* adalah sebuah pengembangan algoritma dari algoritma interpolasi linear yang menginterpolasikan fungsi dari dua variabel pada sebuah bidang. Kunci utama ide algoritma ini adalah

menjalankan interpolasi linear untuk satu arah dan selanjutnya menjalankan interpolasi linear untuk arah yang lain.

GAMBAR 2.6 merupakan contoh interpolasi bilinear untuk sebuah titik.



Gambar 2.6 Interpolasi bilinear untuk sebuah titik.

Pada GAMBAR 2.6 titik merah merupakan titik-titik dengan nilai yang diketahui dan titik hijau merupakan titik yang ingin dicari nilainya. Diasumikan nilai dari keempat titik $Q_{11} = (x_1, y_1)$, $Q_{12} = (x_1, y_2)$, $Q_{21} = (x_2, y_1)$, and $Q_{22} = (x_2, y_2)$ diketahui. Maka untuk mendapatkan nilai dari titik P digunakan persamaan 2.6.

$$\begin{aligned} f(x,y) \approx & \frac{f(Q_{11})}{(x_2-x_1)(y_2-y_1)}(x_2-x)(y_2-y) \\ & + \frac{f(Q_{21})}{(x_2-x_1)(y_2-y_1)}(x-x_1)(y_2-y) \\ & + \frac{f(Q_{12})}{(x_2-x_1)(y_2-y_1)}(x_2-x)(y-y_1) \\ & + \frac{f(Q_{22})}{(x_2-x_1)(y_2-y_1)}(x-x_1)(y-y_1). \end{aligned} \tag{2.6}$$

Interpolasi bilinear akan digunakan jika terjadi kegagalan penghitungan pada interpolasi NEDI saat dilakukan invers matriks

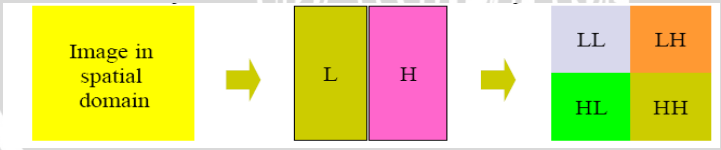
akibat adanya matriks singular yang terbentuk dari perhitungan *covariance* lokal.

2.4 *Discrete Wavelet Transform*

Discrete Wavelet Transform (DWT) pada sebuah citra adalah sebuah representasi sinyal sederhana dari sebuah citra yang menyediakan formasi spasial dan lokalisasi spektrum yang baik (Mohideen,2008).

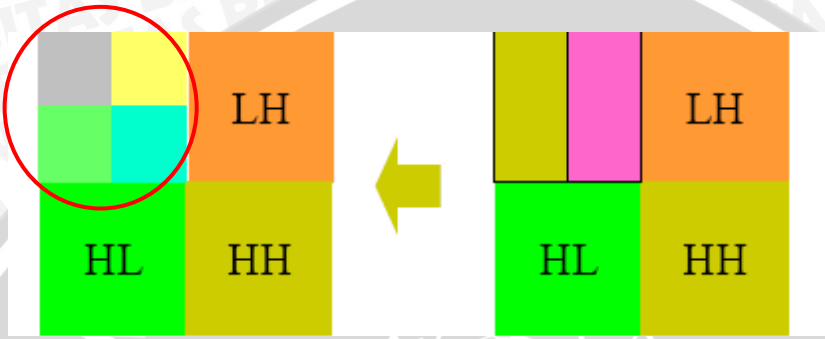
DWT dapat diinterpretasikan sebagai dekomposisi sinyal pada serangkaian kanal frekuensi yang berdiri sendiri. Cara untuk mendekomposisi sebuah sinyal citra adalah melewatkan sinyal itu pada dua filter yang saling melengkapi dan memunculkan dua sinyal yang disebut sebagai perkiraan dan detail. Filter yang digunakan adalah filter *low pass* dan filter *high pass*. Filter *high pass* akan dilewati oleh sinyal berfrekuensi tinggi dan menghasilkan sinyal detail, sedangkan filter low pass akan dilewati oleh sinyal yang berfrekuensi rendah dan menghasilkan sinyal perkiraan.

Sebuah citra dapat didekomposisikan ke dalam sebuah rangkaian citra-citra yang berbeda ukuran resolusi spasialnya menggunakan DWT. Untuk mendekomposisi sebuah citra diperlukan DWT dua dimensi. DWT dua dimensi akan mengaplikasikan DWT terhadap citra secara baris terlebih dahulu baru kemudian diikuti pengaplikasian terhadap kolom. Sebuah N dekomposisi yang dapat dilakukan akan menghasilkan 3N + 1 sekumpulan data frekuensi yang berbeda yang dinamakan LL, LH, HL, dan HH seperti yang tampak pada GAMBAR 2.7. GAMBAR 2.7 merupakan contoh dekomposisi DWT dua dimensi level 1 sebuah citra (Abouzeid, 2004).



Gambar 2.7 Dekomposisi DWT dua dimensi level 1 sebuah citra digital.

Setiap pertambahan level dekomposisi (pada N>1), transformasi wavelet akan diaplikasikan hanya kepada frekuensi LL saja seperti tampak pada GAMBAR 2.8 (Abouzeid, 2004).



Gambar 2.8 Dekomposisi DWT dua dimensi level 2 sebuah citra digital.

Beberapa contoh dari DWT adalah wavelet Haar, wavelet Daubechies, wavelet Symlets, wavelet Coiflets, wavelet Meyer, wavelet Morlet, dan wavelet Mexican Hat.

2.4.1 Wavelet Haar

Salah satu contoh dari DWT yang sering digunakan adalah wavelet Haar yang ditemukan oleh matematikawan asal Hungaria yaitu Alfred Haar. Wavelet Haar merupakan salah satu wavelet yang paling tua dan paling sederhana.

Prinsip kerja dari wavelet Haar adalah adanya 2 proses yang berjalan secara bersamaan yaitu proses pencarian rata-rata dan proses pencarian perbedaaan. Pada wavelet Haar, untuk jumlah data m, jumlah dekomposisi yang dapat dilakukan adalah $^{2}\log m$.

Untuk memudahkan pemahaman terhadap kedua proses dalam wavelet Haar , akan diambil contoh sebuah matriks baris berukuran 8 yang mempunyai nilai [3 5 4 8 13 7 5 3]. Jumlah level dekomposisi yang dapat dilakukan adalah pada baris matriks baris berukuran 8 adalah $^{2}\log 8 = 3$.

- Dekomposisi level 1

  Pencarian rata-rata untuk tiap pasang data dalam baris matrik (contoh: (3 +5) / 2 = 4, (4 +8) /2 = 6 ,dst) dan menempatkan

hasilnya pada ke-4 posisi sebuah matrik baris baru. Sisa elemen matriks baru digunakan untuk selisih nilai dari nilai elemen pertama setiap pasangan data dengan nilai rata-rata pasangan datanya.(Contoh: 3 – 4 = -1, 4 - 6 = -2, dst) .Sehingga dari kedua proses tersebut terbentuk baris matriks [4 6 10 4 -1 -2 3 1].

- Dekomposisi level 2

  Proses pencarian rata-rata dan proses pencarian perbedaan diterapkan terhadap keempat elemen pertama dari baris matrik yang terbentuk pada dekomposisi level 1 sehingga menghasilkan baris matriks baru [5 7 -1 3 -1 -2 3 1].

- Dekomposisi level 3

  Proses pencarian rata-rata dan proses pencarian perbedaan diterapkan terhadap dua elemen pertama dari baris matrik yang terbentuk pada dekomposisi level 2 sehingga menghasilkan baris matriks baru [6 -1 -1 3 -1 -2 3 1].

Contoh penerapan wavelet Haar pada matriks baris tersebut adalah penggunaan wavelet Haar pada satu dimensi sehingga disebut 1-D wavelet Haar. Untuk mendekomposisi sebuah citra, akan digunakan 2-D wavelet Haar. Prinsip kerja dari 2-D wavelet Haar adalah menjalankan 1-D wavelet Haar untuk setiap baris pada matriks citra dan kemudian menjalankan 1-D wavelet Haar untuk setiap kolom matriks citra hasil dari proses 1-D wavelet Haar pada setiap baris matriks citra. Wavelet Haar digunakan dalam perhitungan M-DWT untuk mendekomposisi citra digital.

2.4.2 M-DWT

M-DWT adalah sebuah metode pengukur kualitas citra secara objektif yang menggunakan DWT yang ditemukan oleh Devon Gayle dkk. pada tahun 2006. Metode M-DWT adalah metode yang tidak memperhatikan model HVS (Human Visual System) sehingga mengabaikan asumsi yang mempertimbangkan jarak pandang manusia terhadap objek citra yang diamati. Metode M-DWT bersifat *full-reference* karena citra asli dan citra hasil pengolahan digunakan dalam pengukuran kualitas citra.

Model warna citra yang digunakan oleh metode M-DWT untuk prosesnya adalah model warna Y'UV. Model warna Y'UV adalah sebuah transformasi linear dari komponen RGB yang menghasilkan sebuah sinyal *luminance* (Y') dan sepasang sinyal *chrominance* (U

dan V). Sinyal yang digunakan dalam pengukuran kualitas citra adalah hanya sinyal *luminance* (Y') saja. Secara matematis, model warna Y'UV dapat diperoleh dari model warna RGB dengan persamaan 2.7, 2.8 dan 2.9

$$Y' = 0.299*R+0.114*G+0.587*B \quad (2.7)$$
$$U = 0.436*(B-Y') / (1-0.114) \quad (2.8)$$
$$V = 0.615*(R-Y') / (1-0.299) \quad (2.9)$$

Langkah-langkah dari metode M-DWT adalah sebagai berikut:

- Mencari lapisan *luminance* dari citra asli dengan persamaan 2.7
- Mencari lapisan *luminance* dari citra hasil pembesaran dengan persamaan 2.7
- Pada lapisan luminance dari citra asli diterapkan 2-D DWT(2-D wavelet Haar)
- Pada lapisan luminance dari citra hasil pembesaran diterapkan 2-D DWT(2-D wavelet Haar)
- Untuk setiap frekuensi yang didapat (LL,LH,HL,HH) dilakukan langkah-langkah berikut:
  - Membuat sebuah matriks L(i,j) dimana i=1,...,n dan j=1,...,m dan memasukan nilai-nilai frekuensi ke dalam matriks L(i,j). Matriks ini dipergunakan untuk citra asli.
  - Membuat sebuah matriks L1(i,j) dimana i=1,...,n dan j=1,...,m dan memasukan nilai-nilai frekuensi ke dalam matriks L1(i,j). Matriks ini dipergunakan untuk citra hasil pembesaran
  - Menghitung nilai absolut dari perbedaan kedua nilai-nilai dalam matriks L(i,j) dan L1(i,j) dengan persamaan:

    $$|L(i,j)-L1(i,j)| \quad (2.10)$$

    dimana i=1,...,n dan j=1,...,m
  - Hitung standar deviasi hasil perbedaan dengan persamaan:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\chi_i - \overline{\chi}\right)^2} \quad (2.11)$$

dimana

$$\overline{\chi} = \frac{1}{N}\sum_{i=1}^{N} \chi_i \tag{2.12}$$

N adalah jumlah data.

- Hitung rata-rata untuk keempat nilai standar deviasi dengan persamaan:

$$M - DWT = \frac{\sigma_{LL} + \sigma_{LH} + \sigma_{HL} + \sigma_{HH}}{4} \tag{2.13}$$

Untuk pengukuran hasil pembesaran dengan interpolasi NEDI, DWT yang akan diaplikasikan pada citra asli dan citra hasil pembesaran digunakan wavelet Haar sampai dekomposisi level 1. Nilai yang dihasilkan oleh metode ini mengindikasikan bahwa semakin kecil nilainya berarti semakin bagus citra hasil pembesaran.

2.5 *Mean Squared Error*

Pada Statistika, untuk menghitung kesalahan suatu nilai terhadap nilai yang diharapkan dapat menggunakan *Mean Squared Error* (MSE). Rumusan MSE dapat dilihat pada persamaan 2.14.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(x_i - a\right)^2 \tag{2.14}$$

Error (pada rumusan 2.14) adalah perbedaan nilai yang terjadi antara nilai yang ada (a) dengan nilai yang diharapkan (xi) sejumlah tertentu (n). MSE selain dengan persamaan 2.14, juga dapat dikembangkan dengan rumusan seperti pada persamaan 2.15.

$$MSE = \frac{1}{MN}\sum_{y=1}^{M}\sum_{x=1}^{N}\left(I(x,y) - I'(x,y)\right)^2 \tag{2.15}$$

Persamaan 2.15 akan digunakan untuk proses evaluasi citra hasil pembesaran oleh algoritma NEDI. Pada persamaan 2.15 I(x,y) adalah

citra asli, I'(x,y) adalah citra hasil pembesaran dan M,N adalah dimensi dari citra (Kumar, 2001).

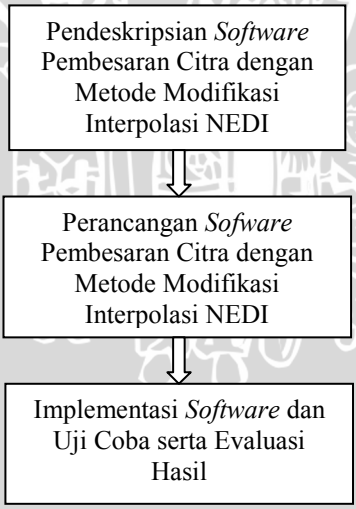
BAB III
METODE DAN PERANCANGAN

Pada bab metode dan perancangan ini akan dibahas penggunaan metode yang digunakan dalam pembuatan perangkat lunak pembesaran citra digital dengan menggunakan modifikasi algoritma NEDI serta langkah-langkah yang dilakukan dalam pembuatan perangkat lunak ini.

Tahapan pembuatannya adalah sebagai berikut:

1. Mendeskrpisikan secara umum perangkat lunak yang akan dibuat.
2. Membuat perangkat lunak berdasarkan analisis dan perancangan yang dilakukan.
3. Uji coba perangkat lunak dengan menggunakan citra standar yang berwarna dan tidak melanggar batasan yang akan dijelaskan selanjutnya.
4. Evaluasi citra hasil pembesaran dengan modifikasi algoritma NEDI yang dilakukan oleh perangkat lunak.

Langkah-langkah pembuatan dapat digambarkan pada GAMBAR 3.1.



Gambar 3.1 Diagram Alir Pembuatan Perangkat Lunak.

3.1 Analisis Perangkat Lunak

Pada subbab analisis perangkat lunak akan dibahas mengenai semua hal yang diperlukan dalam proses pembuatan aplikasi pembesaran citra mencangkup deskripsi secara umum

3.1.1 Deskripsi Umum Perangkat Lunak

Perangkat lunak yang akan dibuat digunakan untuk memperbesar sebuah citra digital berwarna. Hasil perbesaran itu diharapkan sama dengan citra aslinya yang berukuran kecil. Perangkat lunak yang akan dibuat menggunakan salah satu metode pembesaran citra digital. Metode yang akan dipakai dalam perangkat lunak ini adalah metode interpolasi. Algoritma interpolasi yang akan digunakan adalah algoritma interpolasi NEDI yang telah dimodifikasi dengan menggunakan interpolasi bilinear saat interpolasi NEDI tidak dapat mengahsilkan sebuah nilai interpolasi. Selain itu, aplikasi yang akan dibuat dapat digunakan untuk mengevaluasi citra hasil pembesaran yang telah dilakukan oleh aplikasi. Melalui hasil evaluasi dapat diketahui kualitas citra hasil pembesaran.

Citra masukan berupa citra berwarna berukuran HxW atau dapat juga HxH. Citra ini akan dibesarkan dengan skala pembesaran n, sehingga menjadi nHxnW atau HxH. Nilai skala pembesaran *n* akan dimasukkan oleh *user* dimana $n = 2^k$, untuk k = bilangan asli (dalam aplikasi ini dipergunakan nilai k = 1, 2, 3, 4, 5, dan 6). Selain itu, *user* akan memasukkan nilai *m* yang merupakan jumlah tetangga yang akan digunakan dalam proses pembesaran dimana $m = (2^i) \times (2^i)$, untuk i = bilangan asli (dalam aplikasi ini dipergunakan nilai i = 1, 2, 3, 4, dan 5).

Metode evaluasi hasil pembesaran citra dilakukan dengan cara membandingkan citra hasil pembesaran dengan sebuah citra asli yang berukuran sama. Citra asli tersebut akan dikecilkan terlebih dahulu dengan memakai aplikasi lain di luar aplikasi yang akan dibuat (*ACDsee 8*). Citra yang telah diperkecil akan dibesarkan kembali ke ukuran semula memakai aplikasi yang akan dibuat. Citra hasil pembesaran inilah yang akan dibandingkan dengan citra aslinya. Untuk proses perbandingan akan dilakukan dengan 2 metode yaitu menghitung nilai MSE untuk setiap elemen warna dan

mengukur kualitas citra hasil pembesaran menggunakan metode M-DWT.

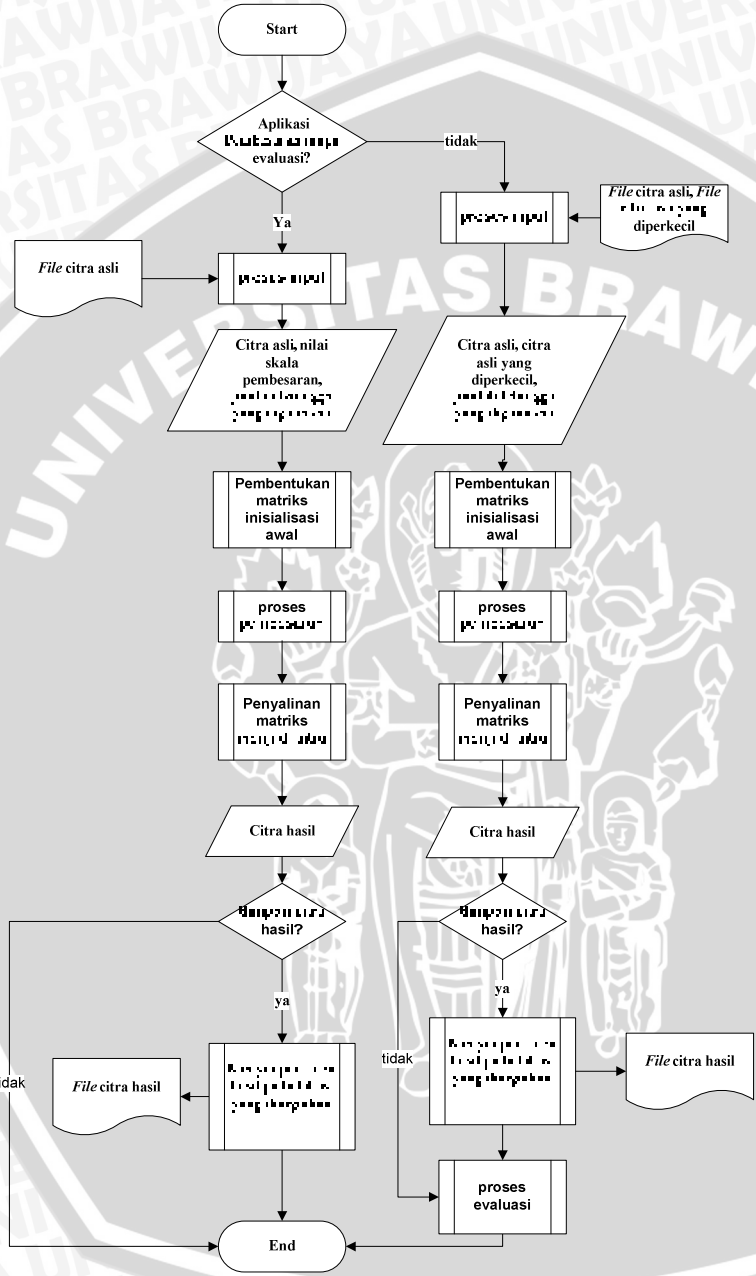
Ketika *user* memasuki perangkat lunak maka proses yang terjadi adalah u*ser* dihadapkan pada 2 pilihan yaitu pembesaran citra digital tanpa proses evaluasi atau pembesaran citra digital dengan proses evaluasi untuk hasil pembesarannya.

Proses yang terjadi jika user memilih pembesaran citra digital tanpa proses evaluasi adalah sebagai berikut:

- Pembesaran citra digital dimulai dengan proses masukan sebuah citra digital berwarna yang sesuai dengan kriteria..
- *User* akan memasukkan sebuah nilai *n* sebagai nilai skala pembesaran dan sebuah nilai *m* yang merupakan jumlah tetangga yang terlibat dalam proses perhitungan.
- Setelah citra asli, skala pembesaran, dan jumlah tetangga dimasukkan oleh *user*, citra yang telah dimasukkan diubah ke dalam bentuk matriks. Proses ini dinamakan pembentukan matriks inisialisasi awal.
- Proses pembesaran citra digital menggunakan modifikasi algoritma interpolasi NEDI (proses perhitungan). Proses pembesaran dibagi ke dalam 3 sub proses yaitu pembentukan matriks hitung, proses modifikasi NEDI, dan pembentukan matriks inisialisasi akhir. Ketiga proses ini akan diulas lebih jauh pada sub bab 3.2.3.
- Pengubahan kembali matriks ke dalam bentuk citra
- *User* akan diperbolehkan untuk menyimpan hasil pembesaran citra digital.
- Keluar dari aplikasi.

Proses yang terjadi jika user memilih pembesaran citra digital dengan proses evaluasi adalah hampir sama dengan sub menu pembesaran citra digital tanpa proses evaluasi karena hanya berbeda pada proses masukkan yang membuat user harus memasukkan citra asli, citra asli yang telah diperkecil, dan jumlah tetangga yang digunakan dan adanya proses evaluasi citra digital hasil pembesaran sebelum keluar dari aplikasi.

Proses secara keseluruhan yang terjadi pada perangkat lunak yang akan dibuat dapat diilustrasikan dengan sebuah diagram alir yang terdapat dalam GAMBAR 3.2



Gambar 3.2 Diagram alir perangkat lunak secara keseluruhan.

3.2 Perancangan Perangkat Lunak

Berdasarkan analisis yang telah dilakukan, selanjutnya akan dibahas mengenai arsitektur dan proses yang terjadi pada aplikasi yang akan dibuat ini. Perancangan proses yang akan dibahas pada subbab ini meliputi proses masukan, proses penyalinan citra pada matriks, proses pembentukan matriks hitung, proses modifikasi NEDI, proses pembentukan matriks inisialisasi akhir, proses penyalinan matriks menjadi citra, proses penyimpanan hasil pembesaran dan proses evaluasi hasil.

3.2.1 Perancangan Proses Masukan

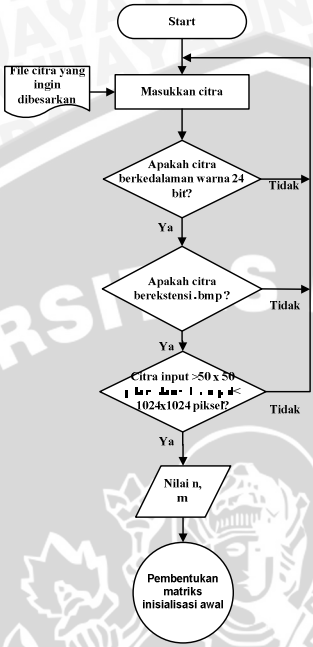
Proses masukan merupakan proses paling awal pada perangkat lunak yang akan dibangun. Proses masukan akan dibedakan menjadi 2 bagian karena setiap sub aplikasi (pembesaran tanpa evaluasi dan pembesaran dengan evaluasi) mempunyai input yang berbeda.

3.2.1.1 Proses Masukan pada Sub Aplikasi Pembesaran Citra Digital tanpa Evaluasi

Pada proses masukan pada aplikasi pembesaran tanpa evaluasi, yang akan dimasukkan oleh seorang *user* adalah sebuah citra, nilai *n* sebagai nilai skala pembesaran citra dan nilai *m* sebagai jumlah tetangga yang digunakan.

Pada proses masukan, citra yang dimasukkan ke dalam perangkat lunak dapat berupa citra *RGB* berformat *Bitmap (.bmp)* dan mempunyai kedalaman warna 24 bit (16.777.216 warna). Selain format dan kedalaman warna, citra masukan juga harus berukuran lebih besar dari 50 x 50 piksel atau lebih kecil sama dengan 1024x1024. Jika tidak, *user* tidak akan dapat melakukan proses selanjutnya.

GAMBAR 3.3 adalah diagram alir dari proses masukan pada sub aplikasi pembesaran citra digital tanpa evaluasi.



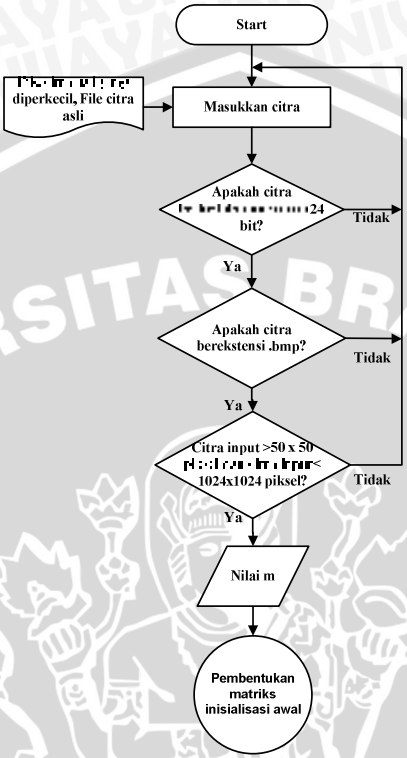
Gambar 3.3 Diagram alir proses masukan pada sub aplikasi pembesaran citra digital tanpa evaluasi.

3.2.1.2 Proses Masukan pada Sub Aplikasi Pembesaran Citra Digital dengan Evaluasi

Pada proses masukan pada aplikasi pembesaran dengan evaluasi yang akan dimasukkan oleh seorang *user* adalah dua buah citra yaitu cita asli dan citra asli yang telah diperkecil dan nilai *m* sebagai jumlah tetangga yang digunakan.

Pada proses masukan, citra yang dimasukkan ke dalam perangkat lunak dapat berupa citra *RGB* berformat *Bitmap (.bmp)* dan mempunyai kedalaman warna 24 bit (16.777.216 warna). Selain format dan kedalaman warna, citra masukan juga harus berukuran lebih besar dari 50 x 50 piksel atau lebih kecil sama dengan 1024x1024. Jika tidak, *user* tidak akan dapat melakukan proses selanjutnya.

GAMBAR 3.4 adalah ilustrasi dari proses masukan pada sub aplikasi pembesaran citra digital dengan evaluasi.



Gambar 3.4 Diagram alir proses masukan pada sub aplikasi pembesaran citra digital dengan evaluasi.

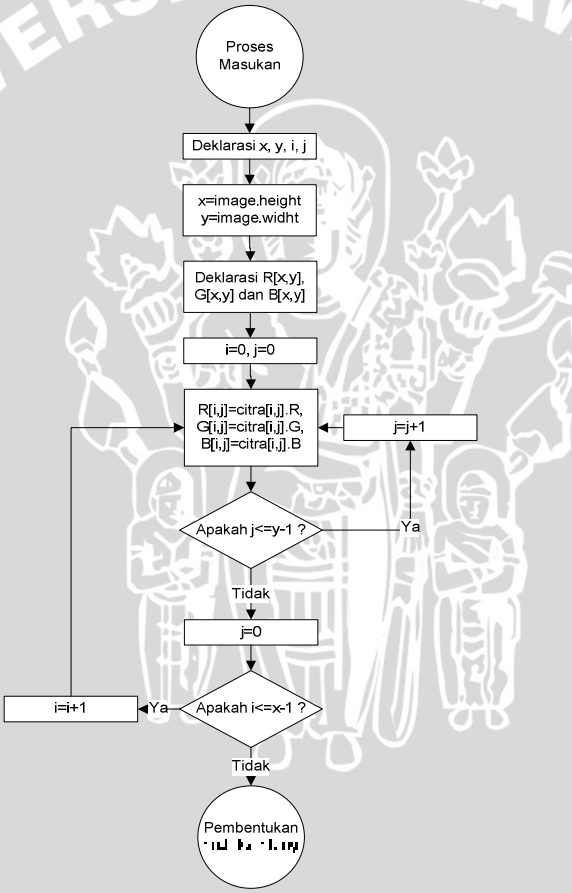
3.2.2 Perancangan Proses Pembentukan Matriks Inisialisasi Awal

Pada umumnya, waktu komputasi yang dibutuhkan oleh fungsi *getPixel()/setPixel()* akan jauh lebih lama jika dibandingkan dengan proses pengambilan atau pemberian nilai pada matriks. Oleh karena itu, sebelum citra memasuki proses lebih lanjut, alangkah baiknya apabila citra digital yang berbentuk *image* atau *bitmap* disalin terlebih dahulu ke dalam matriks.

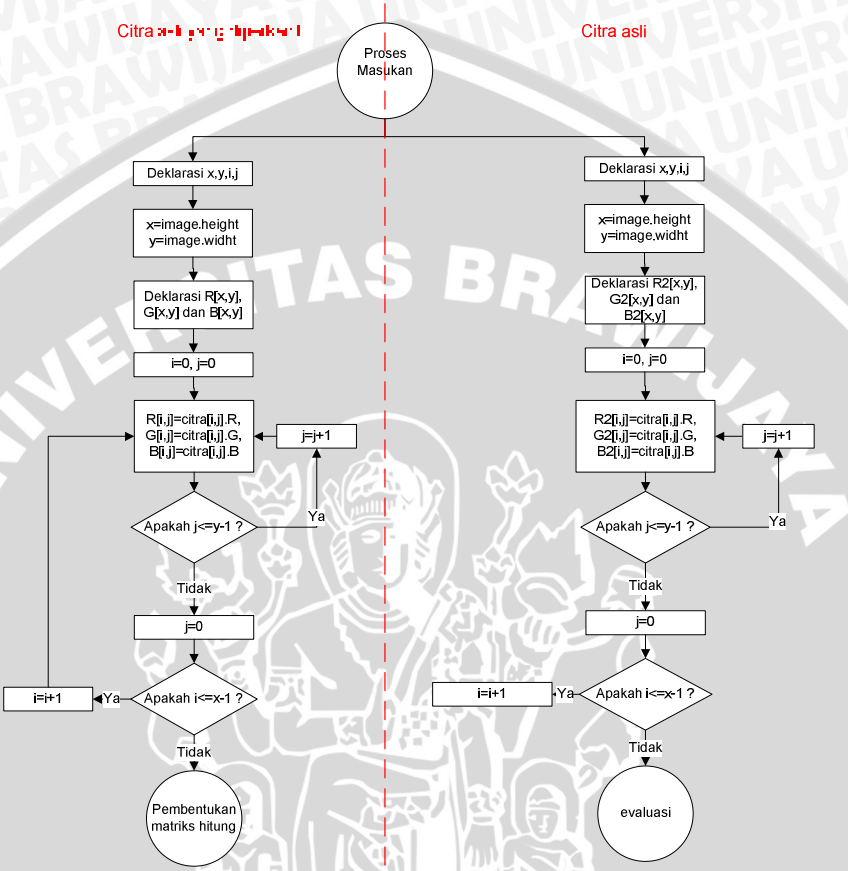
Pada proses ini, nilai intensitas warna (untuk citra 24 bit) dari masing-masing piksel akan disimpan kedalam bentuk matriks dua dimensi (R[,],G[,],B[,]) dengan tipe data *byte* dan bersifat dinamis. Ukuran lebar dan tinggi matriks sama dengan lebar dan tinggi citra yang akan disalin ke dalam matriks. Ketiga matriks

tersebut dinamakan matriks inisialisasi awal. Misalkan citra berukuran 800x600 dan maka ukuran matriks adalah 800x600.

Penyalinan nilai piksel akan dimulai dari koordinat (0,0) atau dari kiri atas sampai dengan kanan bawah. Proses penyalinan dilakukan terhadap semua citra masukan. GAMBAR 3.5 mengilustrasikan proses penyalinan citra pada sub aplikasi pembesaran citra digital tanpa evaluasi. GAMBAR 3.6 mengilustrasikan proses penyalinan citra pada sub aplikasi pembesaran citra digital dengan evaluasi.



Gambar 3.5 Diagram alir proses pembentukan matriks inisialisasi awal pada sub aplikasi pembesaran citra digital tanpa evaluasi.

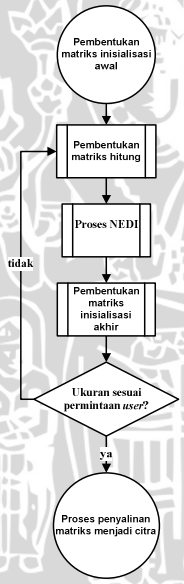


Gambar 3.6 Diagram alir proses pembentukan matriks inisialisasi awal pada sub aplikasi pembesaran citra digital dengan evaluasi.

3.2.3 Perancangan Proses Pembesaran Citra dengan Modifikasi Interpolasi NEDI

Proses pembesaran dengan memakai modifikasi algoritma NEDI dapat diilustrasikan dengan GAMBAR 3.7. Langkah pertama adalah membuat 3 matriks berukuran 2 kali ukuran matriks inisialisasi awal. Matriks-matriks ini disebut matriks hitung. Kemudian nilai-nilai yang terdapat pada matriks inisialisasi awal dipetakan ke dalam matriks hitung dengan aturan hitung (x_{2*i}, y_{2*j}) = inisialisasi_awal (x_i,y_j). Matriks hitung masih bertipe data *byte* dan bersifat dinamis.

Langkah ketiga adalah melakukan proses modifikasi NEDI. Langkah keempat adalah matriks hasil perhitungan dijadikan matriks inisialisasi akhir. Langkah kelima adalah mengecek apakah hasil pembesaran tersebut sudah sesuai dengan skala pembesaran yang diinginkan (sub-aplikasi pembesaran citra digital tanpa evaluasi) atau sesuai dengan citra ukuran panjang dan lebar citra asli (sub-aplikasi pembesaran citra digital dengan evaluasi). Jika tidak sesuai, maka matriks inisialisasi akhir diubah menjadi matriks hitung dengan aturan hitung (x_{2*i}, y_{2*j}) = inisialisasi_akhir (x_i,y_j). Kemudian langkah ketiga sampai langkah kelima diulangi kembali. Jika sesuai maka proses pembesaran citra dengan modifikasi interpolasi NEDI berhenti dan dilanjutkan proses selanjutnya(proses penyalinan matriks menjadi citra).

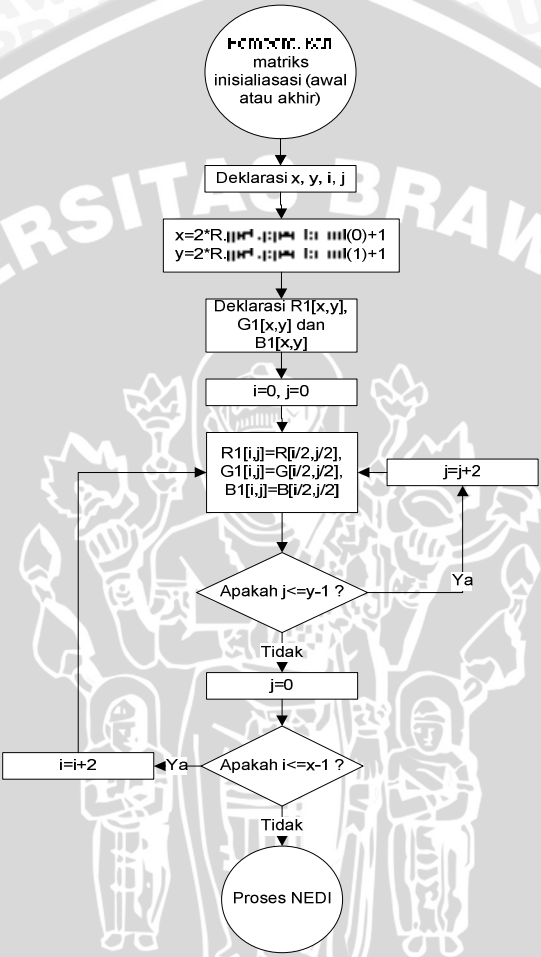


Gambar 3.7 Diagram alir proses pembesaran citra dengan modifikasi interpolasi NEDI.

3.2.3.1 Perancangan Proses Pembentukan Matriks Hitung

Proses ini dilakukan dengan membentuk matriks dua dimensi baru(R1[,], G1[,], B1[,]) yang berukuran dua kali dari matriks inisialisasi(awal maupun akhir). Kemudian nilai-nilai dari matriks

inisialisasi dipetakan ke dalam matrik hitung. GAMBAR 3.8 adalah ilustrasi dari proses pembentukan matriks hitung.

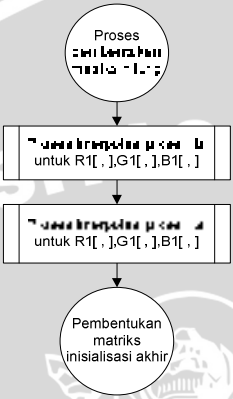


Gambar 3.8 Diagram alir proses pembentukan matriks hitung.

3.2.3.2 Perancangan Proses Modifikasi NEDI

Pada proses NEDI yang pertama dicari adalah seluruh nilai piksel *b* baru kemudian piksel *a*. Untuk matriks hitung R1(i,j), G1(i,j), dan B1(i,j) yang dimaksud dengan piksel *b* adalah piksel yang terletak pada nilai *i* ganjil dan nilai *j* ganjil. Sedangkan yang

dimaksud dengan piksel *a* adalah piksel yang terletak pada *i* ganjil dan *j* genap atau *i* genap dan *j* ganjil GAMBAR 3.9 adalah ilustrasi proses NEDI secara umum.



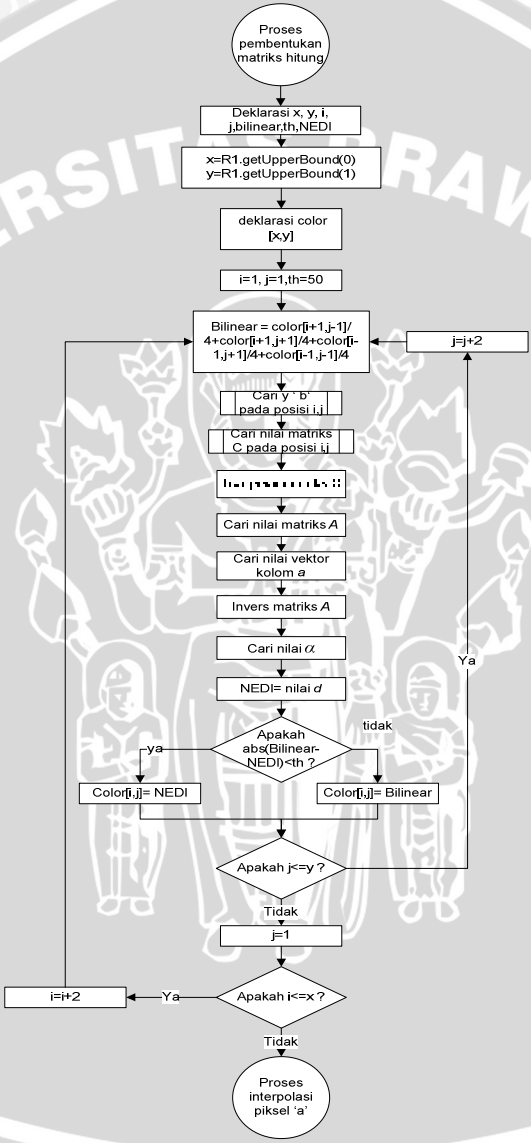
Gambar 3.9 Diagram alir proses modifikasi NEDI secara umum.

3.2.3.2.1 Perancangan Proses Interpolasi Piksel 'b'

Seperti yang telah dijelaskan pada bab 2.3.1 Variabel yang ditentukan nilainya pada awal perhitungan adalah vektor kolom *y*, matriks *C*, dan matriks C^T. Dari variabel tersebut maka dapat ditentukan nilai–nilai untuk *covariance* lokal dari persamaan 2.1 dan 2.2. Jika terbentuk matriks singular dari perhitungan *covariance* lokal maka invers matriks menghasilkan nilai 0. Hal ini akan mengakibatkan perhitungan NEDI juga akan menghasilkan nilai 0. Untuk itu digunakan sebuah variabel dan interpolasi bilinear. Penggunaan variabel *th* bertujuan untuk mengukur perbedaan nilai hasil interpolasi NEDI dengan nilai hasil interpolasi bilinear. Jika terjadi perbedaan yang sangat jauh misalnya perbedaannya lebih besar dari 50, maka yang digunakan adalah nilai dari interpolasi bilinear.

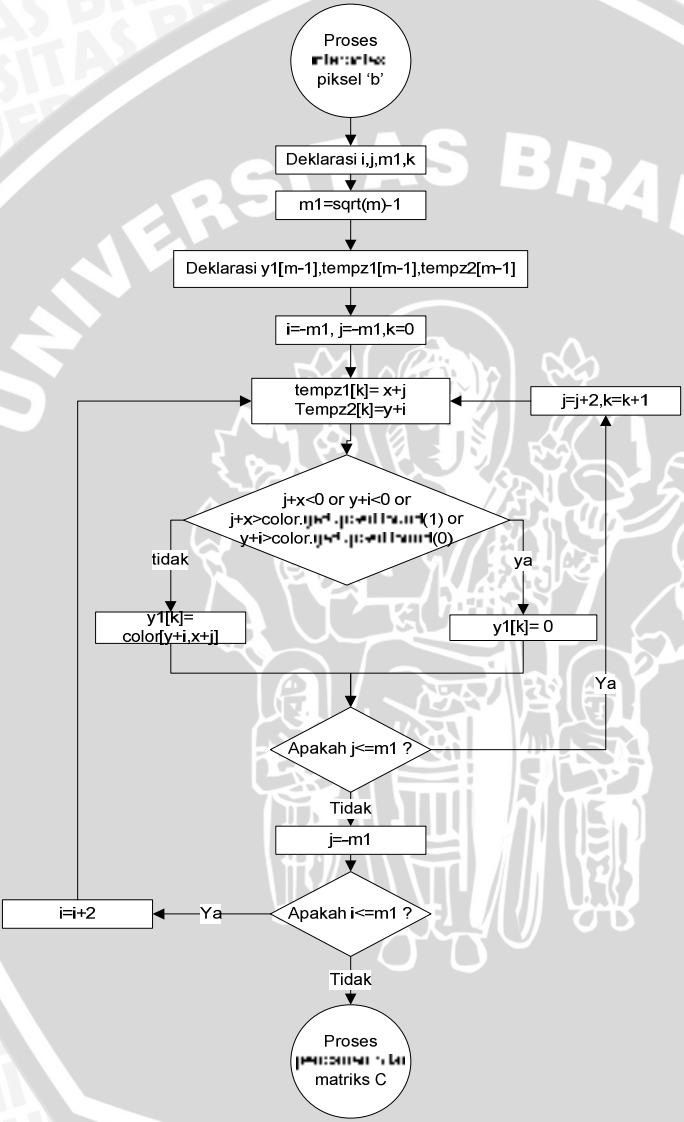
Setelah *covariance* lokal didapatkan maka langkah selanjutnya adalah menentukan nilai bobot interpolasi dalam vektor kolom α dengan rumus persamaan 2.3

Untuk mendapatkan sebuah nilai yang akan diisikan pada piksel *b* yang belum bernilai digunakan persamaan 2.4. GAMBAR 3.10 merupakan diagram alir proses interpolasi piksel *b*.



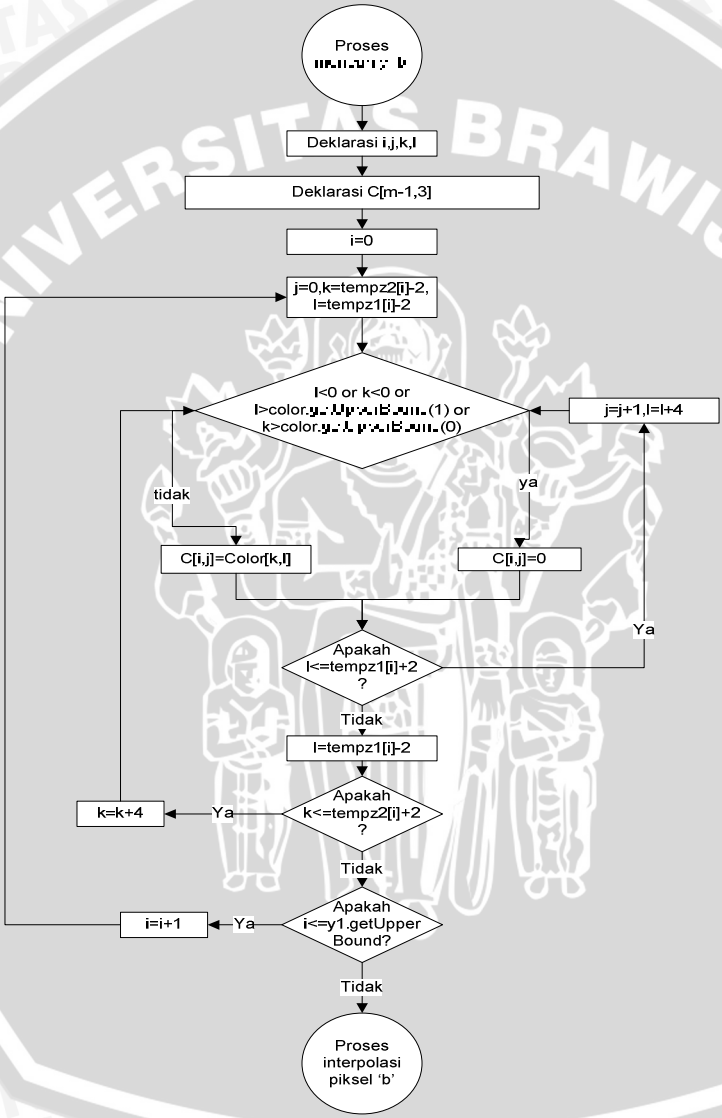
Gambar 3.9 Diagram alir proses interpolasi piksel *b*.

Pencarian nilai tetangga untuk piksel *b* (mencari nilai-nilai vektor kolom *y*) dapat diilustrasikan dengan digram alir seperti pada GAMBAR 3.11.



Gambar 3.11 Diagram alir proses pencarian vektor kolom *y* untuk piksel *b*.

Aturan penentuan nilai-nilai yang terdapat dalam matriks *C* yang berisi 4 tetangga diagonal dari piksel yang ada di vektor kolom *y* dapat dilustrasikan dengan diagram alir pada GAMBAR 3.12.

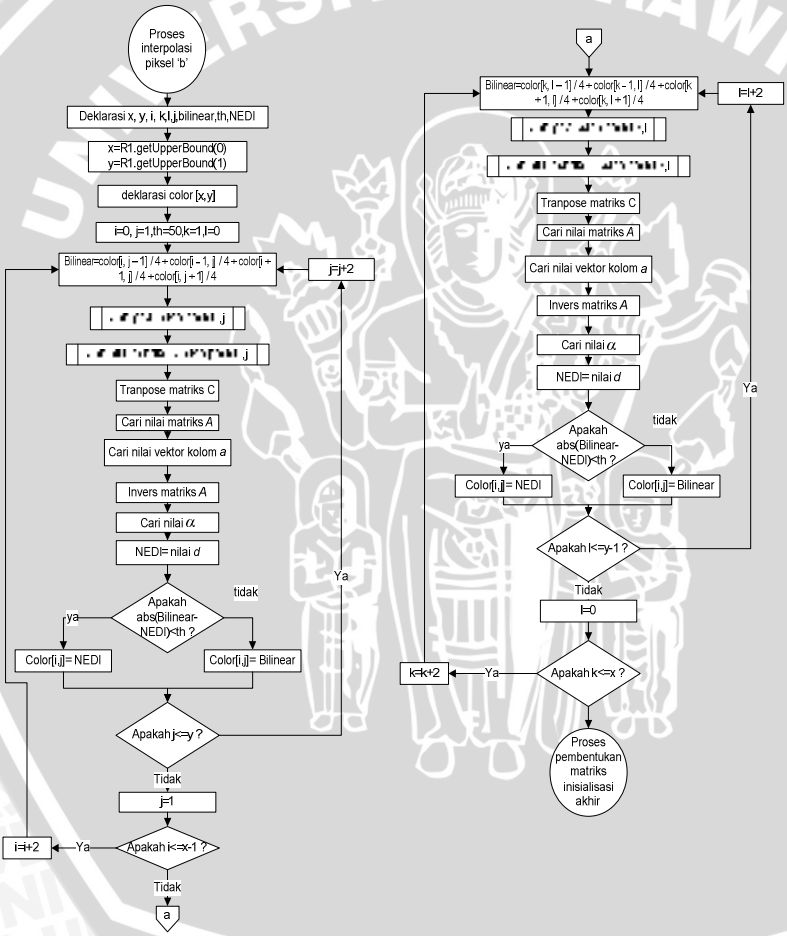


Gambar 3.12 Diagram alir proses pencarian matriks *C* piksel *b*.

3.2.3.2.2 Perancangan Proses Interpolasi Piksel 'a'

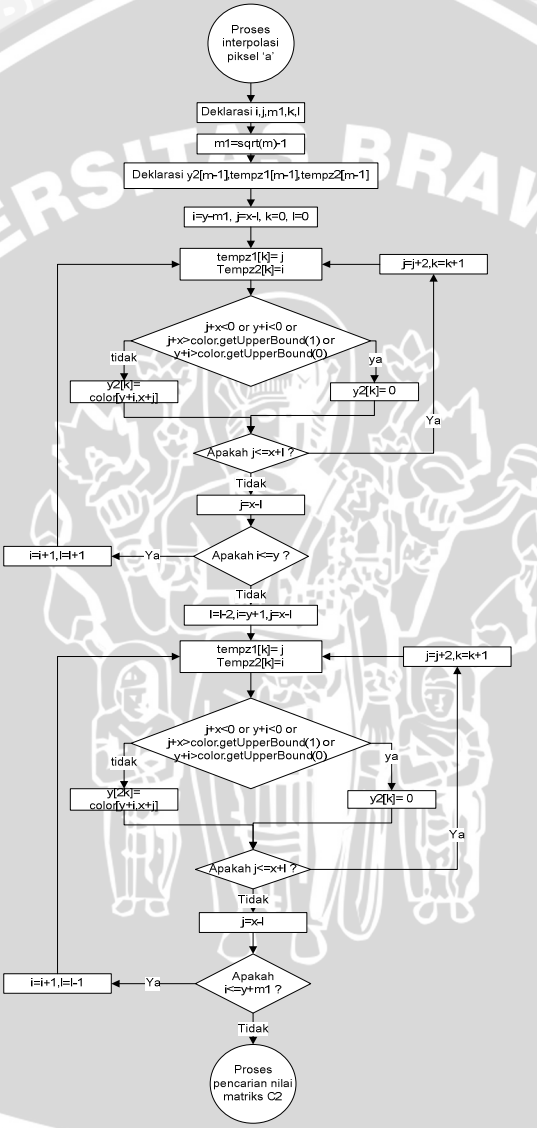
Keseluruhan proses interpolasi untuk piksel *a* hampir sama dengan proses interpolasi pada piksel *b*. Perbedaannya hanya terletak pada cara penentuan nilai-nilai yang dimasukkan pada vektor kolom *y* dan matriks *C*.

Untuk mendapatkan sebuah nilai yang akan diisikan pada piksel *a* digunakan persamaan 2.5. GAMBAR 3.13 merupakan diagram alir proses interpolasi piksel *a*.



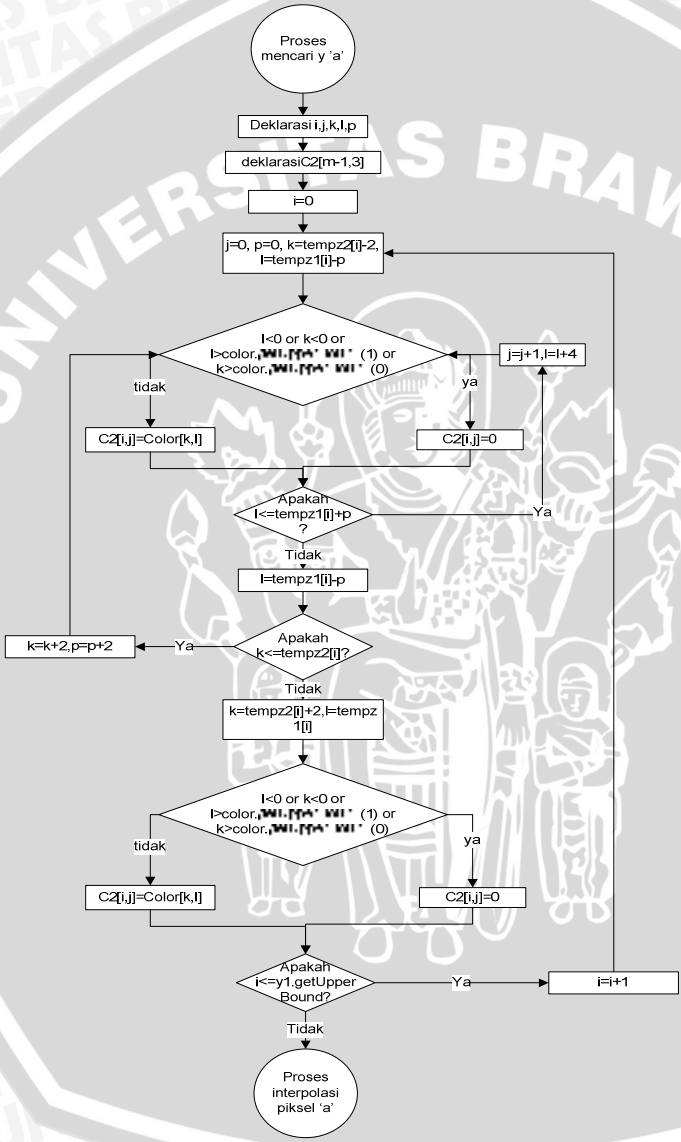
Gambar 3.13 Diagram alir proses interpolasi piksel *a*.

Pencarian nilai tetangga untuk piksel *a* (mencari nilai-nilai vektor kolom *y*) dapat diilustrasikan dengan digram alir seperti pada GAMBAR 3.14.



Gambar 3.14 Diagram alir proses pencarian vektor kolom *y* untuk piksel *a*.

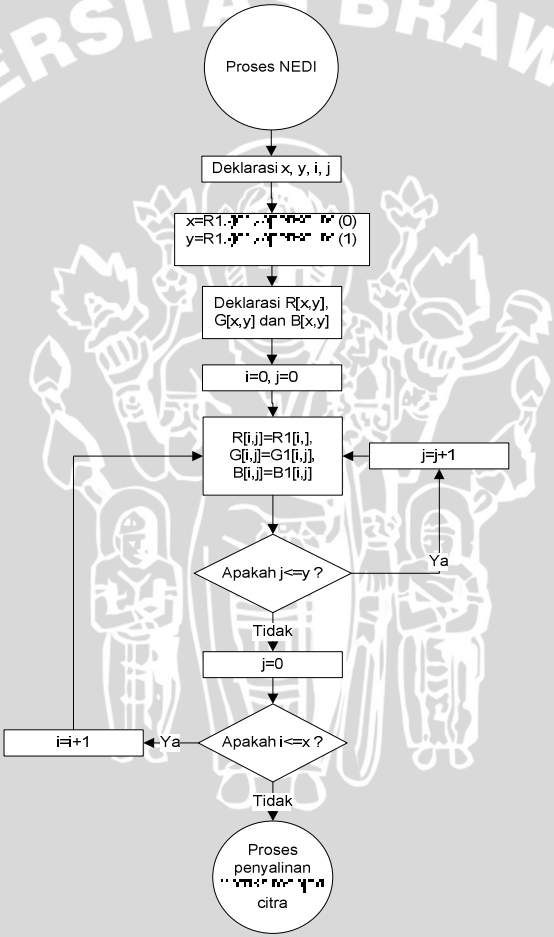
Aturan penentuan nilai-nilai yang terdapat dalam matriks C yang berisi tetangga aksial dari piksel yang ada di vektor kolom *y* dapat dilustrasikan dengan diagram alir pada GAMBAR 3.15.



Gambar 3.15 Diagram alir proses pencarian matriks *C* piksel *a*.

3.2.3.3 Perancangan Proses Pembentukan Matriks Inisialisasi Akhir

Proses ini dilakukan dengan membentuk matriks dua dimensi baru (R1[,],G1[,],B1[,]) yang berukuran sama dengan matriks hitung. Kemudian nilai-nilai dari matriks hitung dipetakan ke dalam matrik inisialisasi akhir. GAMBAR 3.16 adalah ilustrasi dari proses pembentukan matriks inisialisasi akhir.

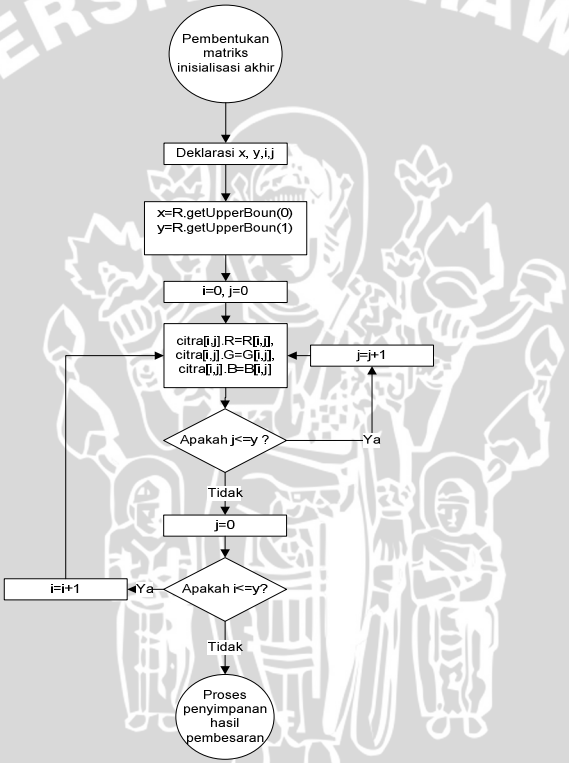


Gambar 3.16 Diagram alir proses pembentukan matriks inisialisasi akhir.

3.2.4 Perancangan Proses Penyalinan Matriks Menjadi Citra

Untuk dapat menampilkan citra hasil pembesaran, matriks yang dihasilkan dari proses-proses sebelumnya harus dikembalikan ke dalam bentuk citra.

Penyalinan kembali nilai piksel akan dimulai dari koordinat (0,0) atau dari kiri atas sampai dengan kanan bawah. GAMBAR 3.17 adalah ilustrasi dari proses penyalinan kembali matriks ke dalam bentuk citra.

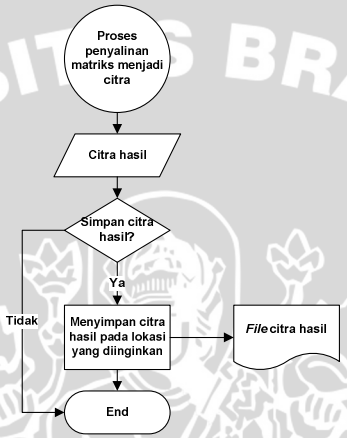


Gambar 3.17 Diagram alir proses penyalinan matriks menjadi citra.

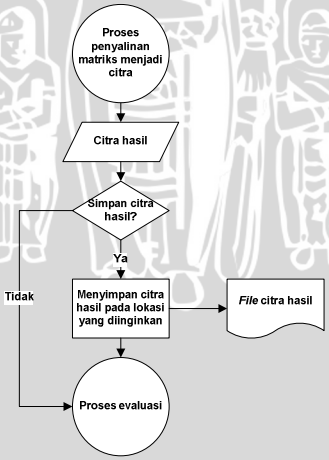
3.2.5 Perancangan Proses Penyimpanan Hasil Pembesaran

Proses penyimpanan citra hasil hanya akan dijalankan apabila *user* menginginkan citra agar disimpan kedalam sebuah *file*. Jika user

menginginkan hal tersebut, maka perangkat lunak akan menyimpan citra hasil ke dalam sebuah *file* yang ditentukan oleh *user*. GAMBAR 3.18 adalah ilustrasi untuk proses penyimpanan citra hasil pada sub aplikasi pembesaran citra digital tanpa evaluasi. GAMBAR 3.19 adalah ilustrasi untuk proses penyimpanan citra hasil pada sub aplikasi pembesaran citra digital dengan evaluasi.



Gambar 3.18 Diagram alir proses penyimpanan hasil pembesaran pada sub aplikasi pembesaran citra digital tanpa evaluasi.

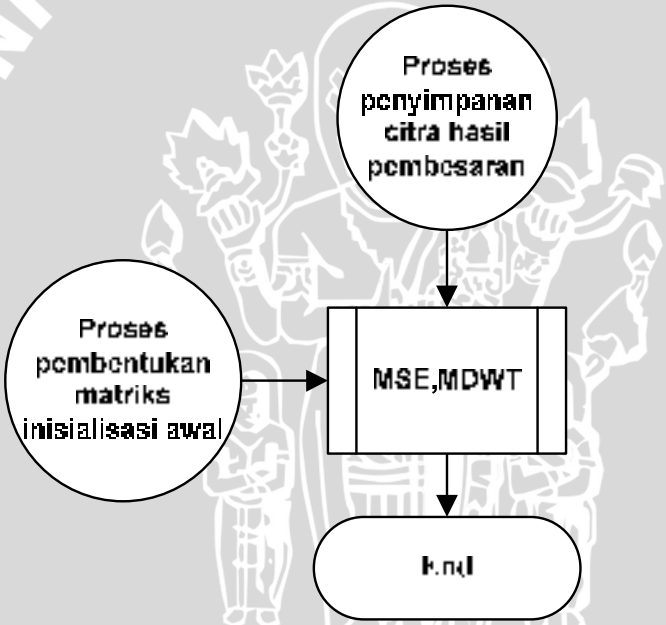


Gambar 3.19 Diagram alir proses penyimpanan hasil pembesaran pada sub aplikasi pembesaran citra digital dengan evaluasi.

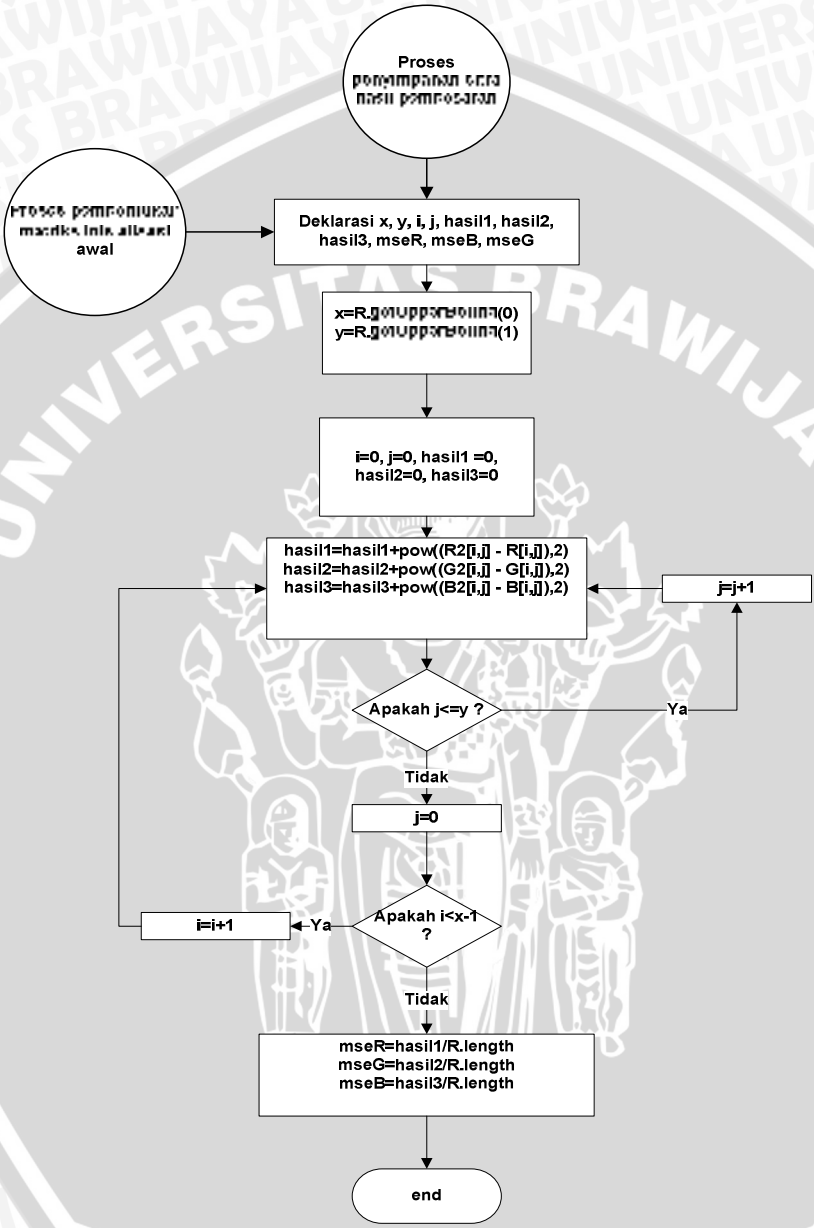
3.2.6 Perancangan Proses Evaluasi

Proses evaluasi digunakan untuk mengetahui kualitas hasil pembesaran. Ada 2 metode yang akan digunakan untuk mengevalusi hasil pembesaran yaitu metode MSE dan M-DWT. MSE digunakan untuk mengetahui *error* yang terjadi pada citra hasil pembesaran. Metode M-DWT digunakan untuk mengukur kualitas citra hasil pembesaran. GAMBAR 3.20 menunjukkan diagram alir dari evaluasi hasil pembesaran citra secara umum.

GAMBAR 3.21 menunjukkan diagram alir dari metode MSE. GAMBAR 3.22 menunjukkan diagram alir dari metode M-DWT secara umum.



Gambar 3.20 Diagram alir proses evaluasi.



Gambar 3.21 Diagram alir proses MSE.

3.2.6.1 Perancangan Proses M-DWT

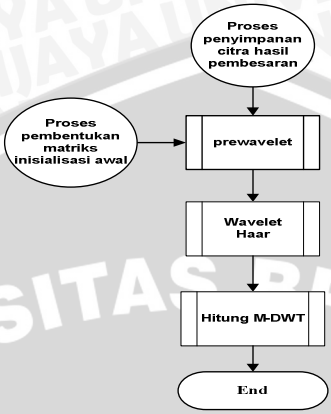
Seperti tampak pada GAMBAR 3.22, metode M-DWT dibagi ke dalam 3 sub proses, yaitu proses prewavelet, proses wavelet Haar, dan proses penghitungan M-DWT.

GAMBAR 3.23 menunjukkan diagram alir dari proses prewavelet. Proses ini digunakan untuk mengambil sinyal luminance(Y') dari citra berwarna. DWT akan diaplikasikan pada sinyal luminance. Setelah dikenakan proses prewavelet maka proses selanjutnya adalah proses wavelet Haar.

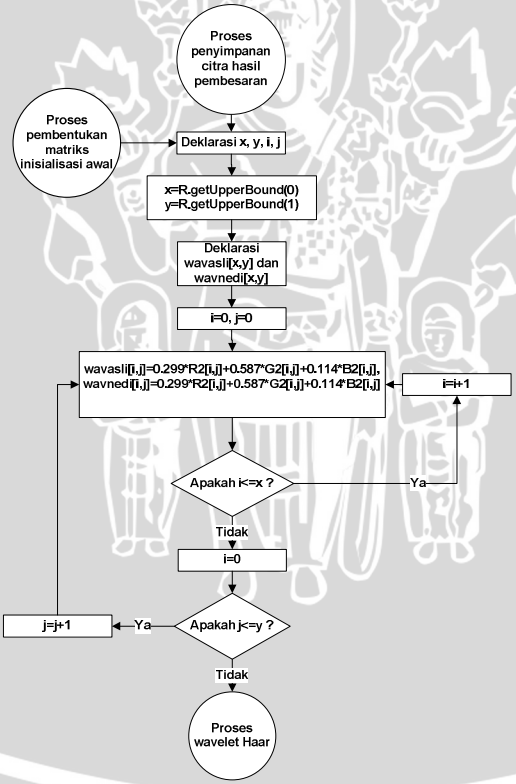
Proses wavelet Haar adalah proses untuk mengaplikasikan DWT dua dimensi pada sinyal luminance yang telah didapat dari proses prewavelet. Proses wavelet Haar diaplikasikan pada sinyal luminace dari citra asli maupun citra hasil pembesaran. Proses wavelet Haar mengaplikasikan DWT secara baris terlebih dahulu, dan kemudian secar kolom. Untuk lebih jelas GAMBAR 3.24 menunjukkan diagram alir dari proses wavelet Haar.

Pengaplikasian DWT yang dimaksud adalah proses dekomposisi. GAMBAR 3.25 merupakan proses hitung wavelet (proses dekomposisi wavelet) sampai level 1 dekomposisi.

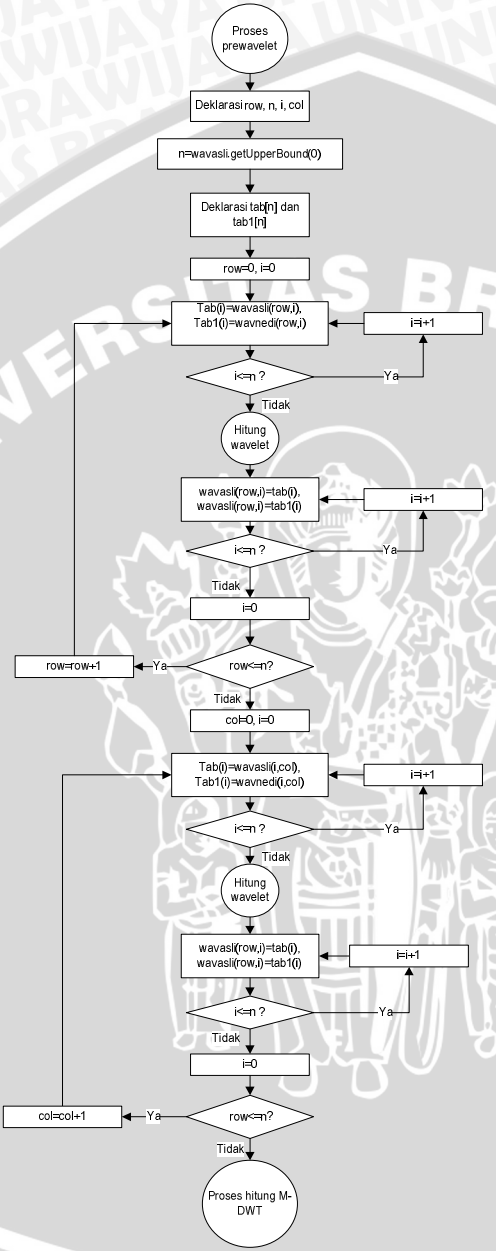
Proses penghitungan M-DWT akan dilakukan setelah didapatkan matriks dari hasil proses wavelet. Dalam proses M-DWT, setiap matriks citra akan dibagi 4 bagian yang merupakan representasi dari frekuensi LL, LH, HL, HH yang timbul akibat proses wavelet. Untuk setiap frekuensi, dicari perbedaan absolut antara citra asli dan citra hasil pembesaran. Kemudian dari hasil perbedaan setiap frekuensi itu dicari standar deviasinya. Untuk dapat menghasilkan sebuah nilai yang dapat digunakan untuk mengukur kualitas citra hasil pembesaran, dicari rata-rata dari keempat hasil standar deviasi. Nilai rata-rata inilah yang menggambarkan kualitas citra secara objektif. Untuk lebih jelas GAMBAR 3.26 menunjukkan diagram alir dari proses penghitungan M-DWT dan GAMBAR 3.27 menunjukkan diagram alir proses penghitungan standar deviasi.



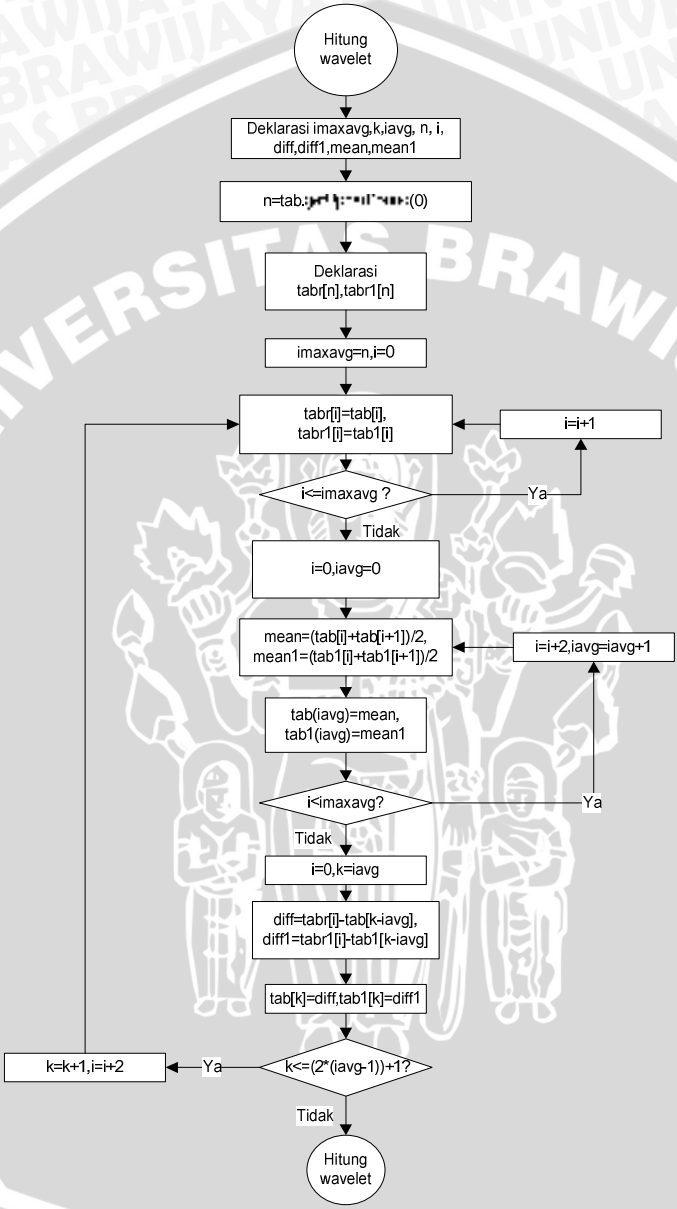
Gambar 3.22 Diagram alir proses M-DWT.



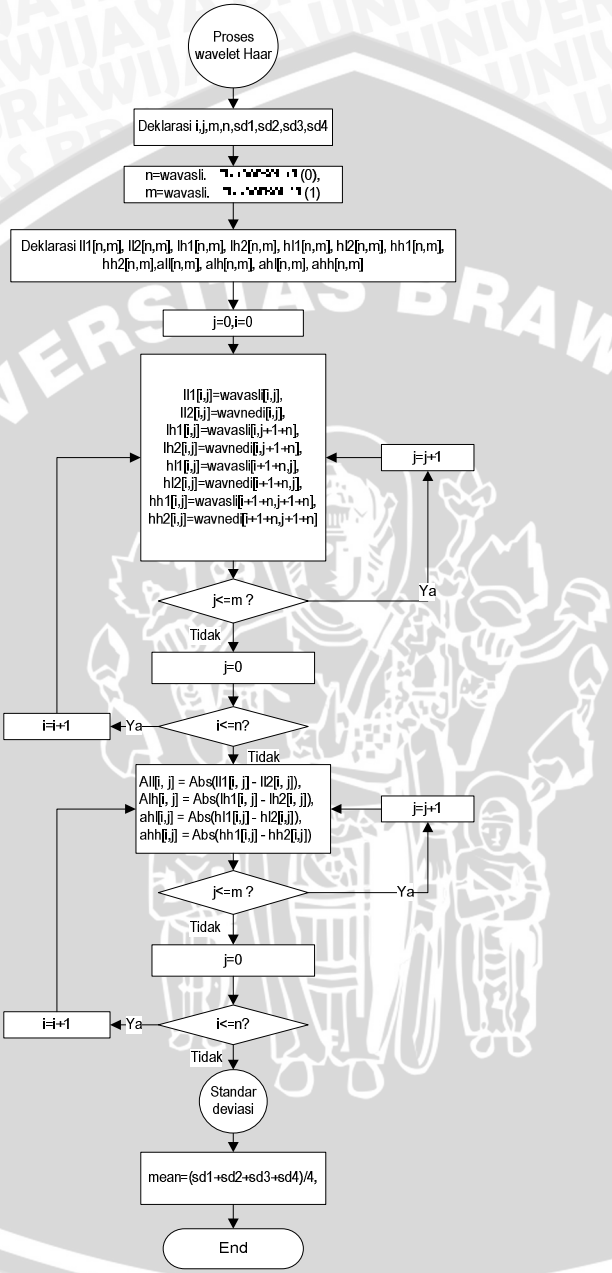
Gambar 3.23 Diagram alir proses prewavelet.



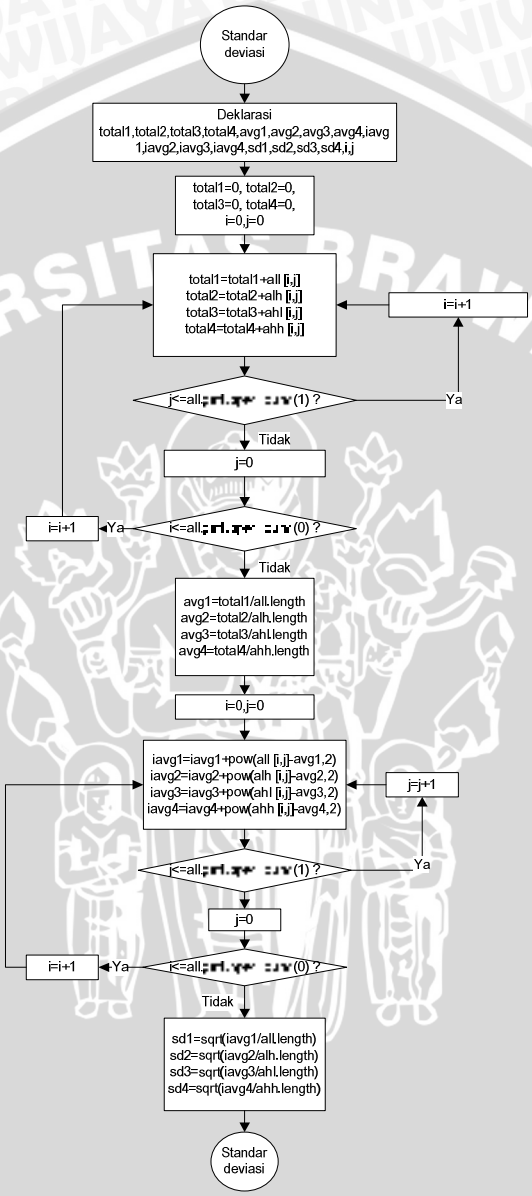
Gambar 3.24 Diagram alir proses wavelet Haar.



Gambar 3.25 Diagram alir proses hitung wavelet.



Gambar 3.26 Diagram alir proses hitung M-DWT.



Gambar 3.27 Diagram alir proses penghitungan standar deviasi.

3.3 Perancangan Uji Coba

Pada subbab ini akan dijelaskan mengenai data yang digunakan dalam penelitian, dan skenario pengujian datanya.

3.3.1 Lingkungan Pengujian

Perangkat lunak yang digunakan untuk menunjang penelitian ini adalah *ACDSee 8.0* untuk memperkecil citra asli yang digunakan untuk pengujian. Algoritma *resizing* dalam *ACDSee 8.0* yang digunakan adalah algoritma *bicubic*.

3.3.2 Pengujian Kualitas Citra Hasil Pembesaran

Citra yang diujikan diambil dari citra standar berwarna dan juga citra yang merupakan hasil pemotretan kamera digital. Penggunaan citra yang bukan citra standar adalah untuk melihat kualitas citra hasil pembesaran pada berbagai macam citra. Citra standar berwarna diambil dari alamat-alamat berikut ini:
http://www.dip.ee.uct.ac.za/imageproc/stdimages/colour/,
http://www.cs.cmu.edu/~chuck/lennapg/lena_std.tif.

Pengujian kualitas citra hasil pembesaran (bilinear maupun modifikasi NEDI) dilakukan dengan cara mencari nilai MSE untuk tiap elemen warna dan mencari nilai kualitas dengan metode M-DWT. Pengukuran MSE menunjukan rata-rata perbedaan antara intensitas citra asli *f(x,y)* dan intensitas citra hasil pembesaran *g(x,y)*. Semakin kecil nilai yang didapat dari metode MSE dan metode M-DWT, maka citra hasil pembesaran semakin bagus (menyerupai citra asli).

File gambar disimpan dalam *field* NE, nilai skala pembesaran disimpan dalam *field* Sc, nilai *m*(jumlah tetangga yang terlibat dalam proses pembesaran) disimpan dalam *field m* dan nilai *MSE* dari tiap nilai *Red, Green,* dan *Blue* disimpan dalam *field* MSER, MSEG, MSEB, dan nilai dari hasil perhitungan M-DWT disimpan dalam *field* M-DWT. Sehingga bentuk tabel pengujian akan disajikan seperti yang tampak dalam TABEL 3.1:

Tabel 3.1 Tabel Hasil Uji Kualitas Citra Digital Hasil Pembesaran Menggunakan Modifikasi Interpolasi NEDI

NE	M	Sc	MSER	MSEG	MSEB	M-DWT

Hasil pengujian ini akan menunjukkan kualitas citra hasil pembesaran menggunakan modifikasi interpolasi NEDI berdasarkan nilai skala pembesaran, nilai variabel M dan hasil dari MSEnya.

TABEL 3.2 merupakan tabel hasil pengujian terhadap citra hasil pembesaran menggunakan interpolasi bilinear. *File* gambar disimpan dalam *field* BL, nilai *MSE* dari tiap nilai *Red, Green,* dan *Blue* disimpan dalam *field* MSER, MSEG, MSEB, dan nilai dari hasil perhitungan M-DWT disimpan dalam *field* M-DWT.

Tabel 3.2 Tabel Hasil Uji Kualitas Citra Digital Hasil Pembesaran Menggunakan Interpolasi Bilinear

BL	MSER	MSEG	MSEB	M-DWT

3.4 Contoh Perhitungan Manual

Pada subbab ini akan ditunjukkan cara perhitungan yang dilakukan dalam proses interpolasi citra untuk pembesaran citra. Contoh perhitungan yang pada subbab ini hanya akan dilakukan kepada nilai *Red* citra berwarna berukuran 6 x 6 seperti tampak pada TABEL 3.3 untuk mempermudah pemahaman cara kerja dari algoritma interpolasi NEDI. Selain itu juga dibuat perhitungan dengan algoritma interpolasi bilinear dengan memakai nilai *Red* pada TABEL 3.3 sebagai perbandingan hasil interpolasi kedua algoritma

Tabel 3.3 Nilai *Red* pada Citra Digital Berukuran 3 x 3

	0	***1***	***2***
0	192	145	133
1	144	147	110
2	127	125	99

Nilai *Red* pada citra berwarna pada TABEL 3.3 dibesarkan dengan skala pembesaran 2 sehingga menjadi berukuran 12 x 12 seperti yang tampak pada TABEL 3.4. Kemudian nilai-nilai pada TABEL 3.3 dipetakan ke dalam TABEL 3.4.

Tabel 3.4 Nilai *Red* Citra Digital Setelah Diperbesar

	0	*1*	*2*	*3*	*4*	*5*
0	**192**	a_1	**145**	a_2	**133**	a_3
1	a_4	b_1	a_5	b_2	a_6	b_3
2	**144**	a_7	**147**	a_8	**110**	a_9
3	a_{10}	b_4	a_{11}	b_5	a_{12}	b_6
4	**127**	a_{13}	**125**	a_{14}	**99**	a_{15}
5	a_{16}	b_7	a_{17}	b_8	a_{18}	b_9

3.4.1 Perhitungan Manual Modifikasi Algoritma Interpolasi NEDI

Piksel *a* dan *b* adalah piksel yang belum diketahui nilainya dan akan dicari nilainya menggunakan interpolasi. Yang pertama dicari adalah seluruh nilai piksel b. Untuk contoh perhitungan diambil salah satu contoh dari piksel b yang diwakili oleh piksel b_1 pada nilai *Red* pada citra (TABEL 3.4). Setelah semua nilai piksel *b* ditemukan maka perhitungan nilai piksel *a* akan dilakukan. Untuk contoh perhitungan diambil sebuah piksel yang diwakili oleh piksel a_5 pada nilai *Red* pada citra (TABEL 3.4).

Contoh perhitungan piksel b_1. Telah ditentukan memakai jumlah tetangga 4 sehingga didapatkan vektor kolom *y* dengan nilai-nilainya seperti yang tampak pada TABEL 3.5.

Tabel 3.5 Vektor Kolom *y* dari Piksel *b1*

y_1	**Tetangga diagonal 1 (X_{i-1},Y_{j-1})**	192
y_2	**Tetangga diagonal 2 (X_{i+1},Y_{j-1})**	145
y_3	**Tetangga diagonal 3 (X_{i-1},Y_{j+1})**	144
y_4	**Tetangga diagonal 4 (X_{i+1},Y_{j+1})**	147

Nilai-nilai yang terdapat dalam vektor kolom pada TABEL 3.5 adalah nilai-nilai tetangga dari piksel *b1* secara diagonal seperti tampak pada tabel 3.6 (piksel yang berwarna merah).

Tabel 3.6 Piksel yang Digunakan pada Vektor Kolom *y* Piksel *b1*

	0	***1***	***2***	***3***	***4***	***5***
0	**192**	a_1	**145**	a_2	**133**	a_3
1	a_4	b_1	a_5	b_2	a_6	b_3
2	**144**	a_7	**147**	a_8	**110**	a_9
3	a_{10}	b_4	a_{11}	b_5	a_{12}	b_6
4	**127**	a_{13}	**125**	a_{14}	**99**	a_{15}
5	a_{16}	b_7	a_{17}	b_8	a_{18}	b_9

Kemudian didapatkan nilai-nilai untuk matriks C yang berisi tetangga digonal dari piksel yang ada di vektor kolom *y* (TABEL 3.5).Yang akan ditampilkan pada TABEL 3.7 adalah matriks C dari piksel *b1*. `Baris ke-*k* merupakan nilai keempat tetangga dari piksel yang ada dalam vektor kolom *y* dengan *k* =1,2,...,m(jumlah tetangga yang digunakan).

Tabel 3.7 Matriks *C* dari Piksel *b1*

Tetangga piksel **192**	0	0	0	147
Tetangga piksel **145**	0	0	144	110
Tetangga piksel **144**	0	145	0	125
Tetangga piksel **147**	192	133	127	99

Menghitung nilai matriks *R* dengan persamaan 2.1 dan matriks *r* dengan persamaan 2.2. TABEL 3.8 menunjukkan hasil perhitungan yang menghasilkan nilai-nilai matriks *R* dari piksel *b1*. TABEL 3.9 menunjukkan hasil perhitungan yang menghasilkan nilai-nilai vektor kolom *r* dari piksel *b1*.

Tabel 3.8 Matriks *R* dari Piksel *b1*

9216.00	6384.00	6096.00	4752.00
6384.00	9678.50	4222.75	7823.00
6096.00	4222.75	9216.25	7103.25
4752.00	7823.00	7103.25	14783.75

Tabel 3.9 Vektor kolom *r* dari Piksel *b1*

7056.00
10107.75
9887.25
19181.75

Dari perhitungan dengan persamaan 2.3 didapatkan hasil nilai-nilai vektor kolom α seperti tampak pada TABEL 3.10

Tabel 3.10 Vektor kolom α dari Piksel *b1*

0.1781
-0.1329
0.0092
1.3061

Kemudian dilakukan perhitungan sebagai berikut:

d = 0.1781* 192 -0.1329* 145 + 0.0092* 144 + 1.3061* 147
d = 208.2462≈ 208

Menurut perhitungan diatas didapatkan bahwa nilai piksel *b1* adalah 208. Berdasarkan contoh langkah-langkah perhitungan di atas maka didapatkan semua nilai piksel *b* adalah seperti yang tampak pada TABEL 3.11(piksel berwarna merah).

Tabel 3.11 Hasil Interpolasi Nilai Semua Piksel *b*

	0	***1***	***2***	***3***	***4***	***5***
0	**192**	a_1	**145**	a_2	**133**	a_3
1	a_4	**208**	a_5	**168**	a_6	**61**
2	**144**	a_7	**147**	a_8	**110**	a_9
3	a_{10}	**153**	a_{11}	**111**	a_{12}	**52**
4	**127**	a_{13}	**125**	a_{14}	**99**	a_{15}
5	a_{16}	**63**	a_{17}	**56**	a_{18}	**25**

Contoh perhitungan piksel *a5*. Telah ditentukan memakai jumlah tetangga 4 sehingga didapatkan vektor kolom *y* dengan nilai-nilainya seperti yang tampak pada TABEL 3.12.

Tabel 3.12 Vektor Kolom y dari Piksel *a5*

y_1	**Tetangga aksial 1 (X_i,Y_{j-1})**	145
y_2	**Tetangga aksial 2 (X_{i-1},Y_j)**	208
y_3	**Tetangga aksial 3 (X_{i+1},Y_j)**	168
y_4	**Tetangga aksial 4 (X_i,Y_{j+1})**	147

Nilai- nilai yang terdapat dalam vektor kolom pada TABEL 3.12 adalah nilai- nilai tetangga dari piksel *a5* secara aksial seperti tampak pada tabel 3.13 (piksel yang berwarna biru).

Tabel 3.13 Piksel yang Digunakan pada Vektor Kolom *y* Piksel *a5*

	0	***1***	***2***	***3***	***4***	***5***
0	**192**	a_1	**145**	a_2	**133**	a_3
1	a_4	**208**	a_5	**168**	a_6	**61**
2	**144**	a_7	**147**	a_8	**110**	a_9
3	a_{10}	**153**	a_{11}	**111**	a_{12}	**52**
4	**127**	a_{13}	**125**	a_{14}	**99**	a_{15}
5	a_{16}	**63**	a_{17}	**56**	a_{18}	**25**

Kemudian didapatkan nilai-nilai untuk matriks *C* yang berisi tetangga aksial dari piksel yang ada di vektor kolom *y* (TABEL 3.12) seperti yang tampak pada TABEL 3.14.

Tabel 3.14 Matriks *C* dari Piksel *a5*

Tetangga piksel **145**	0	192	133	147
Tetangga piksel **208**	0	0	168	153
Tetangga piksel **168**	0	208	0	111
Tetangga piksel **147**	145	144	110	125

Menghitung nilai matriks *R* dengan persamaan 2.1 dan matriks *r* dengan persamaan 2.2. TABEL 3.15 menunjukkan hasil perhitungan yang menghasilkan nilai-nilai matriks *R* dari piksel *a5*. TABEL 3.16 menunjukkan hasil perhitungan yang menghasilkan nilai-nilai vektor kolom *r* dari piksel *a5*.

Tabel 3.15 Matriks *R* dari Piksel *a5*

5256.25	5220.00	3987.50	4531.25
5220.00	25216.00	10344.00	17328.00
3987.50	10344.00	14503.25	14751.25
4531.25	17328.00	14751.25	18241.00

Tabel 3.16 Vektor kolom *r* dari Piksel *a5*

5328.75
20988.00
17599.75
22540.50

Dari perhitungan dengan persamaan 2.3 didapatkan hasil nilai-nilai vektor kolom α seperti tampak pada TABEL 3.17.

Tabel 3.17 Matriks α dari Piksel *a5*

0.0911
-0.4099
-0.8398
2.2816

Kemudian dilakukan perhitungan sebagai berikut:

d = 0.0911*145 -0.4099* 177 -0.8398*168 + 2.2816* 147
d = 122.2591≈ 122

Menurut perhitungan diatas didapatkan bahwa nilai piksel *a5* adalah 122. Berdasarkan contoh langkah-langkah perhitungan sebelumnya maka didapatkan semua nilai piksel *a* dan matriks nilai *Red* setelah dilakukan perhitungan menggunakan modifikasi interpolasi NEDI adalah seperti yang tampak pada TABEL 3.18

Tabel 3.18 Matriks Nilai *Red* Setelah Dilakukan Perhitungan Modifikasi Interpolasi NEDI

	0	1	2	3	4	5
0	192	136	145	112	133	49
1	136	208	122	168	118	61
2	144	182	147	148	110	56
3	106	153	149	111	93	52
4	127	117	125	98	99	44
5	48	63	61	56	45	25

Perhitungan yang sama juga berlaku pada nilai–nilai *Green* dan *Blue* dari citra tersebut.

3.4.2 Perhitungan Manual Algoritma Interpolasi Bilinear

Hasil interpolasi untuk semua nilai piksel *b* (berwarna merah) dengan menggunakan algoritma interpolasi bilinear tampak dalam TABEL 3.19.

Tabel 3.19 Hasil Interpolasi Semua Nilai Piksel *b* Perhitungan Interpolasi Bilinear

	0	1	2	3	4	5
0	192	a_1	145	a_2	133	a_3
1	a_4	157	a_5	134	a_6	61
2	144	a_7	147	a_8	110	a_9
3	a_{10}	135	a_{11}	120	a_{12}	52
4	127	a_{13}	125	a_{14}	99	a_{15}
5	a_{16}	63	a_{17}	56	a_{18}	25

Hasil interpolasi untuk semua nilai piksel *a* (berwarna biru) dengan menggunakan algoritma interpolasi bilinear tampak dalam TABEL 3.20.TABEL 3.20 merupakan matriks nilai *Red* setelah dilakukan perhitungan menggunakan algoritma bilinear

.

Tabel 3.20 Matriks Nilai *Red* Setelah Dilakukan Perhitungan Interpolasi Bilinear

	0	*1*	*2*	*3*	*4*	*5*
0	192	123	145	103	133	49
1	123	157	146	134	110	61
2	144	146	147	128	110	56
3	102	135	132	120	95	52
4	127	113	125	100	99	44
5	48	63	61	56	45	25

BAB IV
HASIL DAN PEMBAHASAN

Pada bab ini akan dilakukan implementasi sistem, serta analisa hasil yang dikeluarkan oleh perangkat lunak, untuk selanjutnya dilakukan analisa hasil untuk mengevaluasi kualitas citra hasil pembesaran menggunakan modifikasi interpolasi NEDI.

4.1 Lingkungan Implementasi

Lingkungan implementasi yang akan dijelaskan dalam subbab ini meliputi lingkungan implementasi perangkat keras dan perangkat lunak.

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan aplikasi pembesaran citra digital dengan metode modifikasi algoritma interpolasi NEDI adalah:

1. Prosesor Intel® Pentium(R) M 1,86 GHz
2. Memori 512 Mb
3. Harddisk dengan kapasitas 120 GB
4. Monitor 14" (1280x768 pixel)
5. VGA Mobile Intel(R) 915GM/GMS, 910GML Express 128 MB
6. Keyboard
7. Touchpad

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan aplikasi pembesaran citra digital dengan metode modifikasi algoritma interpolasi NEDI adalah:

1. Sistem operasi Microsoft® Windows XP Home Edition.
2. *Compiler* Microsoft® Visual Basic 2003.
3. MSDN Library – July 2003

4.2 Implementasi Program

Berdasarkan perancangan perangkat lunak pada subbab 3.2., maka pada subbab ini akan dibahas mengenai implementasi dari

perancangan tersebut menggunakan bahasa pemrograman Visual Basic .NET.

4.2.1 Proses Masukan

Proses masukan merupakan proses paling awal pada perangkat lunak yang akan dibangun. Citra yang dimasukkan berasal dari sebuah *file* yang berekstensi *.bmp* dengan kedalaman warna 24 bit (16.777.216 warna). Selain format dan kedalaman warna, citra masukan juga harus lebih besar sama dengan 50 x 50 piksel atau lebih kecil sama dengan 1024x1024

4.2.1.1 Masukan Citra Proses Pembesaran Citra Digital tanpa Evaluasi

Kode untuk memasukkan citra yang berasal dari *file .bmp* adalah:

```
With OpenFileDialog1
       .Filter = "Bitmap(*.BMP)|*.BMP"
       .ShowDialog()
       If .FileName = "" Then Exit Sub
       Picgambar.Image = Image.FromFile(.FileName)
End With
With Picgambar
   If  (.Image.Width <= 50) Or (.Image.Width >= 1024) Or
       (.Image.Height <= 50) Or(.Image.Height >= 1024)Or
       (.Image.PixelFormat <> PixelFormat.Format24bppRgb)
   Then
            MsgBox("Masukkan citra salah",
            MsgBoxStyle.Critical, "Error")
   End If
End With
```

4.2.1.2 Masukan Citra Proses Pembesaran Citra Digital dengan Evaluasi

Kode untuk memasukkan citra asli dan citra asli yang diperkecil yang berasal dari *file.bmp* sama dengan kode untuk memasukkan citra pada menu pembesaran citra digital tanpa evaluasi seperti pada subbab 4.2.1.1.

4.2.2 Pembentukan Matriks Inisialisasi Awal

Proses penyalinan citra kepada matriks atau dapat disebut proses pembentukan matriks inisialisasi awal untuk masing-masing warna (*Red*, *Green* dan *Blue*) diambil dalam baris yang terpisah. Kode untuk pembentukan matriks inisialisasi awal pada sub aplikasi pembesaran citra digital tanpa evaluasi dan kode untuk pembentukan matriks inisialisasi awal pada sub aplikasi pembesaran citra digital dengan evaluasi untuk citra asli yang diperkecil adalah:

```
Private Sub salintomatriks(ByVal citra As Image)
        Dim i, j, k, l, m As Integer
        Dim bmp As New Bitmap(citra)
        ReDim R(citra.Height - 1, citra.Width - 1)
        ReDim G(citra.Height - 1, citra.Width - 1)
        ReDim B(citra.Height - 1, citra.Width - 1)

        For i = 0 To citra.Width - 1
            For j = 0 To citra.Height - 1
                R(j, i) = bmp.GetPixel(i, j).R
                G(j, i) = bmp.GetPixel(i, j).G
                B(j, i) = bmp.GetPixel(i, j).B
            Next
        Next
    End Sub
```

Kode untuk pembentukan matriks inisialisasi awal pada sub aplikasi pembesaran citra digital dengan evaluasi untuk citra asli adalah:

```
Private Sub salintomatriks2(ByVal citra As Image)
        Dim i, j, k, l, m As Integer
        Dim bmp As New Bitmap(citra)
        ReDim R2(citra.Height - 1, citra.Width - 1)
        ReDim G2(citra.Height - 1, citra.Width - 1)
        ReDim B2(citra.Height - 1, citra.Width - 1)

        For i = 0 To citra.Width - 1
            For j = 0 To citra.Height - 1
                R2(j, i) = bmp.GetPixel(i, j).R
                G2(j, i) = bmp.GetPixel(i, j).G
                B2(j, i) = bmp.GetPixel(i, j).B
            Next
        Next
    End Sub
```

4.2.3 Proses Pembesaran Citra dengan Modifikasi Interpolasi NEDI

Pada subbab ini akan dibahas mengenai implementasi proses pembesaran citra dengan modfikasi interpolasi NEDI seperti yang telah dijelaskan pada subbab 3.2.3. dimulai dari proses pembentukan matriks hitung, proses modifikasi NEDI, dan proses pembentukan matriks inisialisasi akhir

4.2.3.1 Proses Pembentukan Matriks Hitung

Seperti telah dijelaskan pada subbab 3.2.3.1, kode untuk pembentukan matriks hitung adalah sebagai berikut:

```
Private Sub inisialisasitohitung()
        Dim i, j, k, l As Integer
        k = 2 * R.GetUpperBound(0) + 1
        l = 2 * R.GetUpperBound(1) + 1
        ReDim R1(k, l)
        ReDim G1(k, l)
        ReDim B1(k, l)

        For i = 0 To k - 1 Step 2
            For j = 0 To l - 1 Step 2
                R1(i, j) = R(i / 2, j / 2)
                G1(i, j) = G(i / 2, j / 2)
                B1(i, j) = B(i / 2, j / 2)
            Next j
        Next i
    End Sub
```

Sedangkan kode untuk menghitung energi pada citra digital 24 bit adalah:

4.2.3.2 Proses Modifikasi NEDI

Seperti telah dibahas pada subbab 3.2.3.2, maka selanjutnya akan dijelaskan implementasi proses interpolasi piksel *b* dan proses interpolasi piksel *a*

4.2.3.2.1 Proses Interpolasi Piksel 'b'

Implementasi dari proses interpolasi piksel *b* dalam bahasa pemrograman adalah:

```
Public Sub kovarien(ByVal color(,) As Byte)
      Dim i, j, a As Integer
      Dim warna(,), nedi, th, bilinear As Byte
```

```
Dim c1(,), e(,), rbes(,), d(,), rkec(,), f(,),
g(,) As Double
If TC2.Visible = True Then
      a = CInt(CB2.Text)
Else
      a = CInt(CB3.Text)
End If
th = 50
ReDim warna(color.GetUpperBound(0),
color.GetUpperBound(1))
For i = 0 To color.GetUpperBound(0)
    For j = 0 To color.GetUpperBound(1)
        warna(i, j) = color(i, j)
    Next
Next
inedi = 0
bilp = 0
bile = 0
For i = 1 To color.GetUpperBound(0) - 2 Step 2
    For j = 1 To color.GetUpperBound(1) - 2 Step 2
       bilinear = CByte(color(i + 1, j - 1) / 4 +
       color(i + 1, j + 1) / 4 + color(i - 1, j -
       1) / 4 + color(i - 1, j + 1) / 4)
       cariyc1(warna, j, i)
       c1 = tranpose_matrik(C)
       e = kali_matriks(c1, C)
       rbes = kali_matriks(a, e)    'covariance1
       d = kali_matriks(c1, y1)
       rkec = kali_matriks(a, d)    'covariance2
       f = hitung_invers2(rbes)
       g = kali_matriks(f, rkec)    'nilai bobot
       Try
       nedi = CByte(g(0, 0) * color(i - 1, j - 1)
       + g(1, 0) * color(i + 1, j - 1) + g(2, 0) *
       color(i - 1, j + 1) + g(3, 0) * color(i +
       1, j + 1))
             If Math.Abs(CInt(bilinear) -
             CInt(nedi)) < th Then
                 color(i, j) = nedi
                 inedi = inedi + 1
             Else
                 color(i, j) = bilinear
                 bilp = bilp + 1 ' jika
       menghasilkan matriks singular maka diadakan
       penyesuaian dengan mengggunakan bilinear
             End If
       Catch exc As OverflowException
```

```
                    color(i, j) = bilinear     '
                    perhitungan  hasil absolut(NEDI)>255
                    bile = bile + 1
                End Try
            Next
        Next

        For i = 1 To color.GetUpperBound(1) Step 2
            color(color.GetUpperBound(0), i) =
            color(color.GetUpperBound(0) - 2, i)
        Next

        For j = 1 To color.GetUpperBound(0) Step 2
            color(j, color.GetUpperBound(1)) = color(j,
            color.GetUpperBound(1) - 2)
        Next

    End Sub
```

Implementasi dari proses pencarian vektor kolom y dan matriks C perhitungan piksel *b* dalam bahasa pemrograman adalah:

```
Private Sub cariyc1(ByVal col(,) As Byte, ByVal x As
Integer, ByVal y As Integer)
        Dim tempz() As titik
        Dim i, j, d, e, l, k, m, n As Integer
        If TC2.Visible = True Then
            n = CInt(CB2.Text)
        Else
            n = CInt(CB3.Text)
        End If
        m = Math.Sqrt(n) - 1
        ReDim y1(n - 1, 0)
        ReDim C(n - 1, 3)
        ReDim tempz(n - 1)
        i = 0
        For k = -m To m Step 2
            For l = -m To m Step 2
               tempz(i).hori = x + l
               tempz(i).vert = y + k
               If (l + x) < 0 Or (y + k) < 0 Or (x + l)
               > col.GetUpperBound(1) Or (k + y) >
               col.GetUpperBound(0) Then
                    y1(i, 0) = 0
               Else
                    y1(i, 0) = CDbl(col(y + k, x + l))
                End If
                i = i + 1
            Next
        Next
```

```
        For i = 0 To y1.GetUpperBound(0)
          j = 0
          e = tempz(i).vert
          d = tempz(i).hori
          For k = e - 2 To e + 2 Step 4
            For l = d - 2 To d + 2 Step 4
                If (l < 0) Or (k < 0) Or (l >
                col.GetUpperBound(1)) Or (k >
                col.GetUpperBound(0)) Then
                        C(i, j) = 0
            Else
                        C(i, j) = CDbl(col(k, l))
            End If
                        j = j + 1
            Next
          Next
        Next
End Sub
```

4.2.3.2.2 Proses Interpolasi Piksel 'a'

Implementasi dari proses interpolasi piksel *a* dalam bahasa pemrograman adalah:

```
Private Sub proseskovarien2(ByVal color(,) As Byte)
        Dim i, j, a As Integer
        Dim warna(,), nedi, th, bilinear As Byte
        Dim c1(,), e(,), rbes(,), d(,), rkec(,), f(,),
        g(,) As Double
        If TC2.Visible = True Then
              a = CInt(CB2.Text)
        Else
             a = CInt(CB3.Text)
        End If
        th = 50
        ReDim warna(color.GetUpperBound(0),
        color.GetUpperBound(1))
        For i = 0 To color.GetUpperBound(0)
           For j = 0 To color.GetUpperBound(1)
               warna(i, j) = color(i, j)
           Next
        Next

        For i = 1 To color.GetUpperBound(0) - 2 Step 2
           For j = 2 To color.GetUpperBound(1) - 2 Step 2
              bilinear = CByte(color(i, j - 1) / 4 +
              color(i - 1, j) / 4 + color(i + 1, j) / 4 +
              color(i, j + 1) / 4)
              cariyc2(warna, j, i)
```

```
            c1 = tranpose_matrik(C)
            e = kali_matriks(c1, C)
            rbes = kali_matriks(a, e) 'covariance1
            d = kali_matriks(c1, y1)
            rkec = kali_matriks(a, d) 'covariance2
            f = hitung_invers2(rbes)
            g = kali_matriks(f, rkec)    ' nilai bobot
            Try
                nedi = CByte(g(0, 0) * color(i, j - 1) +
                g(1, 0) * color(i - 1, j) + g(2, 0) *
                color(i + 1, j) + g(3, 0) * color(i, j +
                1))
                If Math.Abs(CInt(bilinear) - CInt(nedi))
                < th Then
                    color(i, j) = nedi
                    inedi = inedi + 1
                Else
                    color(i, j) = bilinear    ' jika
                    menghasilkan matriks singular maka
                    diadakan penyesuaian dengan
                    mengggunakan bilinear
                    bilp = bilp + 1
                End If
                Catch exc As OverflowException
                    color(i, j) = bilinear     '
                    perhitungan  hasil absolut(NEDI)>255
                    bile = bile + 1
                End Try
        Next
    Next
    For i = 2 To color.GetUpperBound(0) - 2 Step 2
      For j = 1 To color.GetUpperBound(1) - 2 Step 2
          bilinear = CByte(color(i, j - 1) / 4 +
          color(i - 1, j) / 4 + color(i + 1, j) / 4 +
          color(i, j + 1) / 4)
          cariyc2(warna, j, i)
          c1 = tranpose_matrik(C)
          e = kali_matriks(c1, C)
          rbes = kali_matriks(a, e)    'covariance1
          d = kali_matriks(c1, y1)
          rkec = kali_matriks(a, d)    'covariance2
          f = hitung_invers2(rbes)
          g = kali_matriks(f, rkec)    'nilai bobot
          Try
              nedi = CByte(g(0, 0) * color(i, j - 1) +
              g(1, 0) * color(i - 1, j) + g(2, 0) *
              color(i + 1, j) + g(3, 0) * color(i, j +
              1))
```

```
                If Math.Abs(CInt(bilinear) - CInt(nedi)) <
                th Then
                     color(i, j) = nedi
                     inedi = inedi + 1
                Else
                     color(i, j) = bilinear     ' jika
                     menghasilkan matriks singular maka
                     diadakan penyesuaian dengan
                     mengggunakan bilinear
                     bilp = bilp + 1
                End If
                Catch exc As OverflowException
                     color(i, j) = bilinear          '
                     perhitungan  hasil absolut(NEDI)>255
                     bile = bile + 1
                End Try
            Next
        Next

        For i = 1 To color.GetUpperBound(1) Step 2
             color(color.GetUpperBound(0) - 1, i) =
             color(color.GetUpperBound(0) - 2, i)
             color(color.GetUpperBound(0), i - 1) =
             color(color.GetUpperBound(0) - 1, i - 1)
             color(0, i) = color(1, i)
        Next

        For j = 1 To color.GetUpperBound(0) Step 2
             color(j, color.GetUpperBound(1) - 1) =
             color(j, color.GetUpperBound(1) - 2)
             color(j - 1, color.GetUpperBound(1)) =
             color(j - 1, color.GetUpperBound(1) - 1)
             color(j, 0) = color(j, 1)
        Next
    End Sub
```

Implementasi dari proses pencarian vektor kolom y dan matriks C perhitungan piksel *a* dalam bahasa pemrograman adalah:

```
Private Sub cariyc2(ByVal col(,) As Byte, ByVal x As
Integer, ByVal y As Integer)
      Dim tempz() As titik
      Dim i, j, d, e, l, k, m, n, o As Integer
      If TC2.Visible = True Then
           n = CInt(CB2.Text)
      Else
           n = CInt(CB3.Text)
      End If
      m = Math.Sqrt(n) - 1
      ReDim y1(n - 1, 0)
```

```
ReDim C(n - 1, 3)
ReDim tempz(n - 1)
i = 0
o = 0
For k = y - m To y
    For l = x - o To x + o Step 2
        tempz(i).hori = l
        tempz(i).vert = k
        If l < 0 Or k < 0 Or l >
        col.GetUpperBound(1) Or k >
        col.GetUpperBound(0) Then
                y1(i, 0) = 0
         Else
                y1(i, 0) = CDbl(col(k, l))
         End If
        i = i + 1
    Next
    o = o + 1
Next
o = o - 2
For k = y + 1 To y + m
    For l = x - o To x + o Step 2
        tempz(i).hori = l
        tempz(i).vert = k
        If l < 0 Or k < 0 Or l >
        col.GetUpperBound(1) Or k >
        col.GetUpperBound(0) Then
                y1(i, 0) = 0
        Else
                y1(i, 0) = CDbl(col(k, l))
        End If
        i = i + 1
    Next
o = o - 1
Next

For i = 0 To y1.GetUpperBound(0)
  e = tempz(i).vert
  d = tempz(i).hori
  j = 0
  o = 0
  For k = e - 2 To e Step 2
    For l = d - o To d + o Step 4
        If l < 0 Or k < 0 Or l >
        col.GetUpperBound(1) Or k >
        col.GetUpperBound(0) Then
                C(i, j) = 0
        Else
                C(i, j) = CDbl(col(k, l))
```

```
                End If
                j = j + 1
            Next
            o = o + 2
        Next
        k = e + 2
        l = d
         If l < 0 Or k < 0 Or l >
         col.GetUpperBound(1) Or k >
         col.GetUpperBound(0) Then
                C(i, j) = 0
         Else
                C(i, j) = CDbl(col(k, l))
         End If
    Next
End Sub
```

4.2.3.3 Proses Pembentukan Matriks Inisialisasi Akhir

Berikut ini merupakan implementasi dari proses pembentukan matriks inisialisasi akhir:

```
Private Sub inisialisasi()
    Dim i, j As Integer
    ReDim R(R1.GetUpperBound(0), R1.GetUpperBound(1))
    ReDim G(G1.GetUpperBound(0), G1.GetUpperBound(1))
    ReDim B(B1.GetUpperBound(0), B1.GetUpperBound(1))
     For i = 0 To R.GetUpperBound(0)
        For j = 0 To R.GetUpperBound(1)
            R(i, j) = R1(i, j)
            G(i, j) = G1(i, j)
            B(i, j) = B1(i, j)
        Next
     Next
End Sub
```

4.2.4 Proses Penyalinan Kembali Matriks Menjadi Citra

Untuk dapat menampilkan citra hasil *resizing*, matriks yang dihasilkan dari proses-proses sebelumnya harus dikembalikan ke dalam bentuk citra. Oleh karena itu perlu dilakukan proses penyalinan kembali matriks kedalam bentuk citra seperti telah dijelaskan pada subbab 3.2.4. Agar lebih jelas, berikut merupakan kode untuk menyalin kembali matriks ke dalam bentuk citra:

```
Private Sub backtocit()
    Dim i, j As Integer
    Dim red1, green1, blue1 As Integer
```

```
            Dim b3 As New Bitmap(R.GetUpperBound(1) +
            1, R.GetUpperBound(0) + 1,
            PixelFormat.Format24bppRgb)
        Dim im As Image
        For i = 0 To R.GetUpperBound(0)
            For j = 0 To R.GetUpperBound(1)
                red1 = CInt(R(i, j))
                green1 = CInt(G(i, j))
                blue1 = CInt(B(i, j))
                b3.SetPixel(j, i, Color.FromArgb(red1,
                green1, blue1))
            Next
        Next
       Picgambar4.Image = b3
    End Sub
```

4.2.5 Proses Penyimpanan Citra Hasil

Seperti telah dijelaskan pada subbab 3.2.5. bahwa apabila *user* menginginkan, citra akan disimpan kedalam sebuah *file* yang ditentukan oleh *user* itu sendiri. Oleh karena itu, dibuatlah proses untuk menyimpan citra hasil sebagai berikut:

```
With SaveFileDialog1
     .Filter = " Bitmap(*.BMP)|*.BMP"
     .OverwritePrompt = True
End With
If SaveFileDialog1.ShowDialog() = DialogResult.OK Then
           NamaFile = SaveFileDialog1.FileName
End If
```

4.2.6 Proses Evaluasi

Seperti yang dijelaskan pada subbab 3.2.6, terdapat 2 metode yang digunakan untuk mengevaluasi hasil citra pembesaran yaitu MSE dan M-DWT. Berikut adalah kode dari metode MSE:

```
Private Sub evaluasi()
       Dim i, j, m, n As Integer
       Dim hasil1, hasil2, hasil3, hasilall, mseR, mseG,
       mseB As Double
       m = R.GetUpperBound(0)
       n = R.GetUpperBound(1)
       hasil1 = 0
      hasil2 = 0
      hasil3 = 0

    For i = 0 To m
       For j = 0 To n
```

```
                hasil1 = hasil1 + Math.Pow(CDbl(R2(i, j)) -
                CDbl(R(i, j)), 2)
                hasil2 = hasil2 + Math.Pow(CDbl(G2(i, j)) -
                CDbl(G(i, j)), 2)
                hasil3 = hasil3 + Math.Pow(CDbl(B2(i, j)) -
                CDbl(B(i, j)), 2)
            Next
        Next
        mseR = Math.Round(hasil1 / R.Length, 4)
        mseG = Math.Round(hasil2 / R.Length, 4)
        mseB = Math.Round(hasil3 / R.Length, 4)
End Sub
```

Untuk metode M-DWT terdapat 3 proses utama yaitu proses prewavelet, proses wavelet haar dan proses hitung M-DWT. Dalam proses wavelet Haar terdapat subproses dekomposisi. Berikut merupakan kode dari proses prewavelet:

```
Private Sub prewavelet()
        Dim i, j, m, n As Integer
        m = R.GetUpperBound(0)
        n = R.GetUpperBound(1)
        ReDim wavasli(m, n)
        ReDim wavnedi(m, n)
        For i = 0 To m
          For j = 0 To n
                wavasli(i, j) = CDbl(CByte(0.299 * R2(i, j)
                + 0.587 * G2(i, j) + 0.114 * B2(i, j)))
                wavnedi(i, j) = CDbl(CByte(0.299 * R(i, j)
                + 0.587 * G(i, j) + 0.114 * B(i, j)))
          Next
        Next
    End Sub
```

Kode dari proses dekomposisi wavelet Haar adalah:

```
Private Sub wavelethaar(ByVal tabr() As Double, ByVal n
As Integer)
        Dim IMaxAvg, i, k, iavg, ulang, langkah As
        Integer
        Dim mean, diff As Double
        Dim Tab() As Double
        ReDim Tab(wavasli.GetUpperBound(0))
        IMaxAvg = n
        For i = 0 To IMaxAvg
             Tab(i) = tabr(i)
        Next
        i = 0
        iavg = 0
```

```
    While (i < IMaxAvg)
        mean = (tabr(i) + tabr(i + 1)) / 2
        tabr(iavg) = mean
        i += 2
        iavg += 1
    End While
    i = 0
    For k = iavg To (2 * (iavg - 1)) + 1
        diff = Tab(i) - tabr(k - iavg)
        i += 2
        tabr(k) = diff
    Next
End Sub
```

Kode dari proses wavelet Haar adalah:

```
Private Sub proseswavelet()
    Dim i, row, n, m, col As Integer
    Dim tab(), tab1() As Double
    prewavelet()
    n = wavasli.GetUpperBound(0)
    m = wavasli.GetUpperBound(1)
    ReDim tab(n)
    ReDim tab1(n)
    For row = 0 To m
        For i = 0 To m
            tab(i) = wavasli(row, i)
            tab1(i) = wavnedi(row, i)
        Next
        wavelethaar(tab, m)
        wavelethaar(tab1, m)
        For i = 0 To n
            wavasli(row, i) = tab(i)
            wavnedi(row, i) = tab1(i)
        Next
    Next
    For col = 0 To n
        For i = 0 To n
            tab(i) = wavasli(i, col)
            tab1(i) = wavnedi(i, col)
        Next
        wavelethaar(tab, n)
        wavelethaar(tab1, n)
        For i = 0 To n
            wavasli(i, col) = tab(i)
            wavnedi(i, col) = tab1(i)
        Next
    Next
End Sub
```

Fungsi standar deviasi digunakan pada saat proses penghitungan M-DWT. Kode dari fungsi standar deviasi adalah:

```
Private Function sd(ByVal devi(,) As Double) As Double
        Dim i, j As Integer
        Dim total, avg, iavg, stand As Double
        avg = 0
        total = 0
        iavg = 0
        For i = 0 To devi.GetUpperBound(0)
            For j = 0 To devi.GetUpperBound(1)
                total = total + devi(i, j)
            Next
        Next
        avg = total / devi.Length
        For i = 0 To devi.GetUpperBound(0)
            For j = 0 To devi.GetUpperBound(1)
                    iavg = iavg + Math.Pow((devi(i, j) -
                    avg), 2)
            Next
        Next
        stand = Math.Sqrt(iavg / devi.Length)
        Return stand
 End Function
```

Kode dari proses penghitungan M-DWT adalah:

```
Private Sub mdwt()
         Dim ll1(,), ll2(,), lh1(,), lh2(,), hl1(,),
         hl2(,), hh1(,), hh2(,) As Double
         Dim all(,), alh(,), ahl(,), ahh(,), sd1, sd2,
         sd3, sd4, mean As Double
        Dim i, j, n, m As Integer
        n = wavasli.GetUpperBound(0) \ 2
        m = wavasli.GetUpperBound(1) \ 2
        ReDim ll1(n, m)
        ReDim ll2(n, m)
        ReDim lh1(n, m)
        ReDim lh2(n, m)
        ReDim hl1(n, m)
        ReDim hl2(n, m)
        ReDim hh1(n, m)
        ReDim hh2(n, m)
        ReDim all(n, m)
        ReDim alh(n, m)
        ReDim ahl(n, m)
        ReDim ahh(n, m)
        For i = 0 To n
            For j = 0 To m
                ll1(i, j) = wavasli(i, j)
```

```
                ll2(i, j) = wavnedi(i, j)
            Next
        Next

        For i = 0 To n
            For j = 0 To m
                lh1(i, j) = wavasli(i, j + 1 + m)
                lh2(i, j) = wavnedi(i, j + 1 + m)
            Next
        Next

        For i = 0 To n
            For j = 0 To m
                hl1(i, j) = wavasli(i + 1 + n, j)
                hl2(i, j) = wavnedi(i + 1 + n, j)
            Next
        Next

        For i = 0 To n
            For j = 0 To m
                  hh1(i, j) = wavasli(i + 1 + n, j + 1 +
                  m)
                 hh2(i, j) = wavnedi(i + 1 + n, j + 1 +
                 m)
            Next
        Next
        For i = 0 To n
            For j = 0 To m
              all(i, j) = Math.Abs(ll1(i, j) - ll2(i, j))
              alh(i, j) = Math.Abs(lh1(i, j) - lh2(i, j))
              ahl(i, j) = Math.Abs(hl1(i, j) - hl2(i, j))
              ahh(i, j) = Math.Abs(hh1(i, j) - hh2(i, j))
            Next
        Next
        sd1 = sd(all)
        sd2 = sd(alh)
        sd3 = sd(ahl)
        sd4 = sd(ahh)
    mean = Math.Round(((sd1 + sd2 + sd3 + sd4) / 4), 4)
End Sub
```

4.3 Implementasi Antarmuka

Untuk mempermudah *user* dalam menggunakan perangkat lunak, maka dibuatlah antarmuka seperti pada perangkat lunak yang ada pada umumnya. Terdapat 4 macam antarmuka yang dibuat, masing-masing adalah *form* menu, *form* input, *form* proses, *form* hasil, dan *form* evaluasi citra hasil pembesaran.

4.3.1 Form Menu

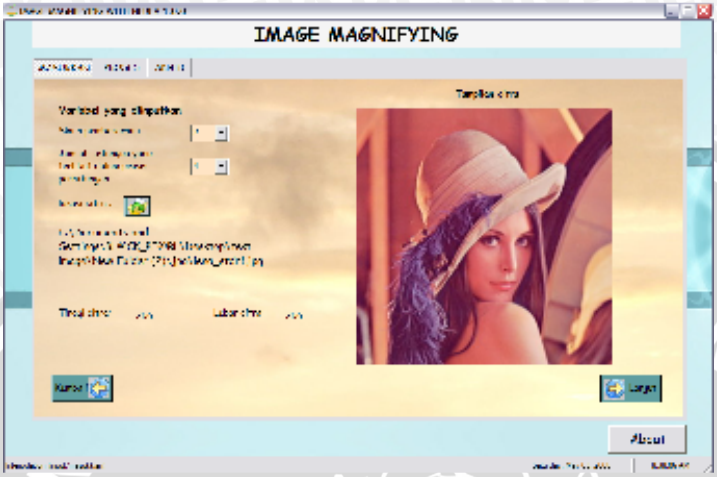
Form menu adalah antarmuka yang akan memandu *user* untuk memilih menu. Menu pilihan yang tersedia adalah menu pembesar citra tanpa evaluasi hasil dan menu pembesar citra dengan evaluasi. GAMBAR 4.1. merupakan tampilan *form* menu pada saat perangkat lunak dijalankan.

Gambar 4.1 *Form* menu.

4.3.2 Form Masukkan

GAMBAR 4.2 merupakan *form* masukkan apabila *user* memilih menu pembesar citra tanpa evaluasi. Melalui *form* ini seorang *user* akan memasukkan citra dan nilai-nilai variebel yang dibutuhkan(nilai skala pembesaran dan nilai jumlah tetangga yang digunakan).

GAMBAR 4.3 merupakan *form* masukkan apabila *user* memilih menu pembesar citra dengan evaluasi. Melalui *form* ini seorang *user* akan memasukkan 2 buah citra yaitu citra asli dan citra asli yang telah diperkecil. Selain itu, *user* akan memasukkan nilai jumlah tetangga yang digunakan.



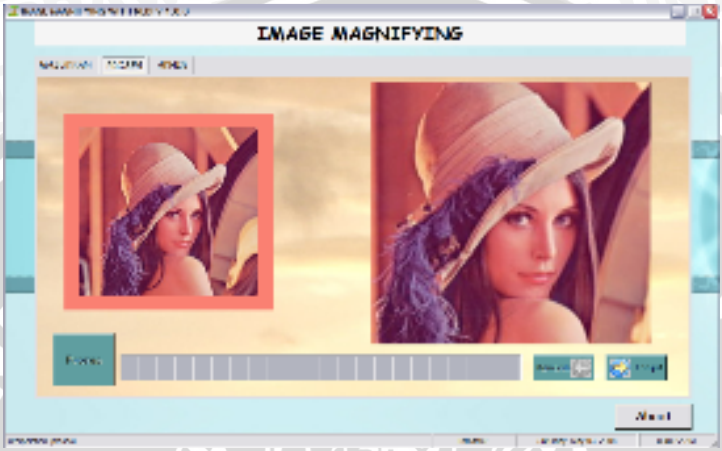
Gambar 4.2 *Form* masukkan menu pembesar citra tanpa evaluasi.

Gambar 4.3 *Form* masukkan menu pembesar citra dengan evaluasi.

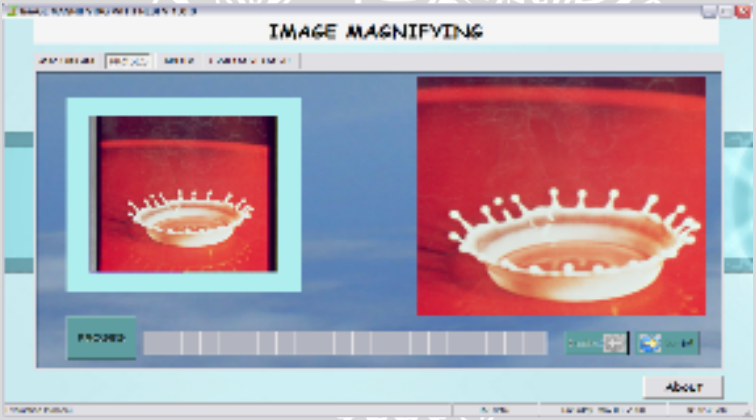
4.3.3 Form Proses

Form proses adalah *form* yang digunakan user untuk memulai proses pembesaran citra. GAMBAR 4.4 merupakan tampilan *form* proses pada menu menu pembesar citra tanpa evaluasi dan

GAMBAR 4.5 merupakan tampilan *form* proses pada menu menu pembesar citra dengan evaluasi.



Gambar 4.4 *Form* proses menu pembesar citra tanpa evaluasi.

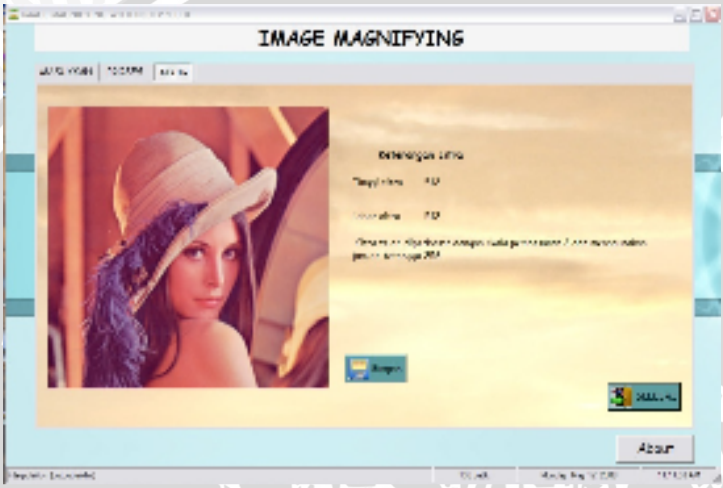


Gambar 4.5 *Form* proses menu pembesar citra dengan evaluasi.

4.3.4 Form Hasil

Form hasil adalah *form* yang berfungsi untuk menunjukkan kepada *user* citra hasil (setelah dilakukan pembesaran). GAMBAR 4.6 merupakan tampilan *form* hasil pada menu menu pembesar citra

tanpa evaluasi dan GAMBAR 4.7 merupakan tampilan *form* hasil pada menu menu pembesar citra dengan evaluasi. Pada GAMBAR 4.7 citra yang ada pada bagian sebelah kiri dari form adalah citra asli, sedangkan citra yang berada di bagian sebelah kanan merupakan citra hasil.

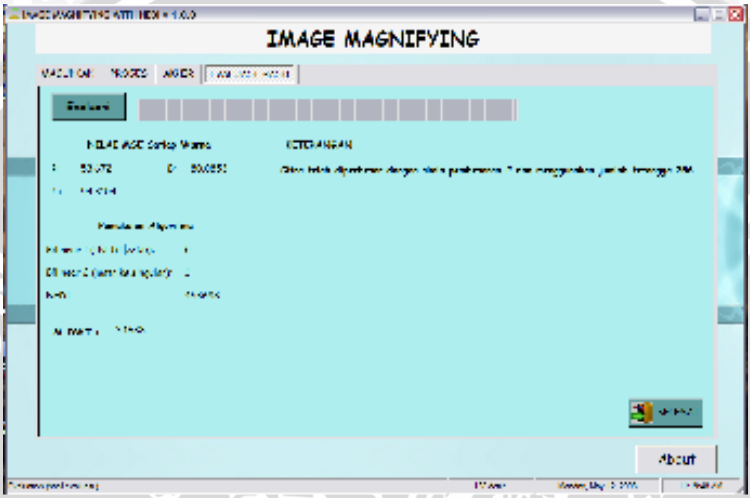


Gambar 4.6 *Form* hasil menu pembesar citra tanpa evaluasi.

Gambar 4.7 *Form* hasil menu pembesar citra dengan evaluasi.

4.3.5 Form Evaluasi

Form evaluasi adalah *form* yang digunakan *user* untuk memulai proses evaluasi dan menampilkan hasil evaluasi citra hasil. GAMBAR 4.8 merupakan tampilan *form* evaluasi.



Gambar 4.8 *Form* evaluasi.

4.4 Implementasi Uji Coba

Pada subbab ini akan dilakukan pembahasan mengenai pengujian yang telah dilakukan pada sistem dan hasil evaluasi dari hasil yang dikeluarkan sistem.

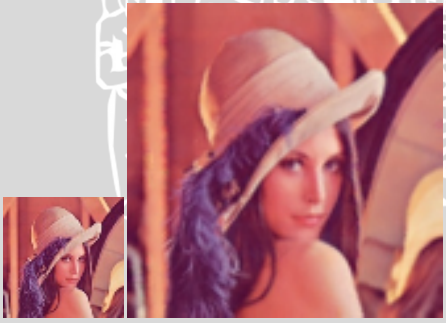
4.4.1 Pengaplikasian Modifikasi Algoritma Interpolasi NEDI

Penelitian dengan menggunakan algoritma interpolasi NEDI oleh Xin Li dan Michael Orchard yang berkaitan dengan pembesaran citra telah diterapkan pada citra digital *grayscale*. GAMBAR 4.9 merupakan contoh hasil penelitian oleh Xin Li dan Michael Orchard. GAMBAR 4.9(a) adalah citra digital asli. GAMBAR 4.9(b) adalah citra digital hasil pembesaran dengan skala pembesaran 4. Sedangkan pada penelitian yang dilakukan kali ini adalah pembesaran citra pada citra berwarna (memakai model warna RGB).

(a) (b)

Gambar 4.9 Hasil penelitian Xin Li dan Michael Orchard: (a) citra digital asli; (b) citra digital hasil pembesaran

Pengaplikasian algoritma interpolasi NEDI pada pembesaran citra berwarna dapat dilakukan dengan cara mengaplikasikan algoritma interpolasi NEDI yang telah dimodifikasi untuk setiap elemen warnanya yaitu pada elemen warna R, elemen warna G, dan elemen warna B. Hal ini dilakukan karena setiap elemen warna mengontrol *luminance* dan *chrominance* secara bersamaan. GAMBAR 4.10 merupakan contoh dari hasil pembesaran citra digital berwarna menggunakan modifikasi algoritma interpolasi NEDI . GAMBAR 4.10(a) adalah citra digital asli. GAMBAR 4.9(b) adalah citra digital hasil pembesaran dengan skala pembesaran 4.



(a) (b)

Gambar 4.10 Hasil pembesaran citra berwarna menggunakan aplikasi yang dibuat: (a) citra digital asli; (b) citra digital hasil pembesaran

4.4.2 Evaluasi Citra Hasil Pembesaran Menggunakan Modifikasi Algoritma NEDI

Pengujian yang dilakukan adalah untuk melihat kualitas citra hasil pembesaran menggunakan modifikasi algoritma interpolasi NEDI. Untuk pengujian digunakan 3 citra uji. 2 dari ketiga citra uji merupakan citra standar dan 1 citra lainnya merupakan hasil pemotretan kamera digital. Pemakaian citra bukan citra standar dalam citra uji dimaksudkan untuk mengetahui kualitas citra hasil pembesaran pada berbagai macam citra. Keterangan untuk masing-masing citra yang akan diujikan terdapat pada TABEL 4.1.

Tabel 4.1. Keterangan citra yang akan diujikan.

Nama Citra	Ukuran Citra		Sumber
	Lebar	Tinggi	
lena	512	512	Citra standar
laut	1024	768	kamera digital
baboon	512	512	Citra standar

Masing- masing citra uji akan dibesarkan dengan 3 skala pembesaran yaitu 2, 4, dan 8 dan untuk setiap nilai skala pembesaran akan menggunakan nilai jumlah tetangga yang berbeda yaitu 4, 16, 64, 256, dan 1024.

(a)

(b)

(c)

Gambar 4.11 Citra uji: (a) citra lena; (b) citra baboon; (c) citra laut.

Pada pengujian skala pembesaran 2, citra lena dan citra baboon diperkecil terlebih dahulu menjadi 256x256 dan untuk citra laut diperkecil menjadi 512x384. GAMBAR 4.12 merupakan contoh citra hasil pembesaran dengan nilai skala pembesaran 2 dengan jumlah tetangga 4 dan 1024.

(a) (b)

(c)

(d)

(e)

(f)

Gambar 4.12 Contoh citra hasil pembesaran dengan skala pembesaran 2(modifikasi NEDI): (a) lena dengan m=4; (b) lena dengan m=1024; (c) laut dengan m=4; (d) laut dengan m=1024; (e) baboon dengan m=4; (f) baboon dengan m=1024.

Pengukuran kualitas citra untuk skala pembesaran 2 dapat dilihat pada TABEL 4.2.

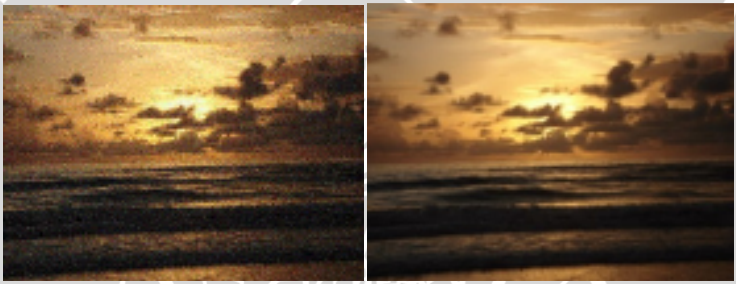
Tabel 4.2 Tabel Hasil Uji Kualitas Citra Digital Hasil Pembesaran Menggunakan Modifikasi Algoritma Interpolasi NEDI dengan Skala Pembesaran 2

NE	m	Sc	MSER	MSEG	MSEB	M-DWT
Lena	4	2	181.07	242.71	234.92	3.997
Lena	16	2	72.26	119.00	95.81	3.549
Lena	64	2	58.01	98.57	82.23	3.243
Lena	256	2	53.67	93.31	80.08	3.169
Lena	1024	2	52.57	93.42	79.59	3.159
laut	4	2	193.61	173.20	155.74	3.698
laut.	16	2	87.92	65.86	45.59	2.918
laut	64	2	69.09	49.78	34.01	2.497
laut	256	2	64.78	46.44	32.35	2.358
laut	1024	2	63.89	45.93	32.18	2.336
baboon	4	2	625.50	779.66	755.62	7.959
baboon	16	2	464.72	624.33	612.89	7.898
baboon	64	2	431.52	577.70	560.32	7.634
baboon	256	2	426.34	570.48	552.76	7.582
baboon	1024	2	425.05	568.38	550.23	7.567

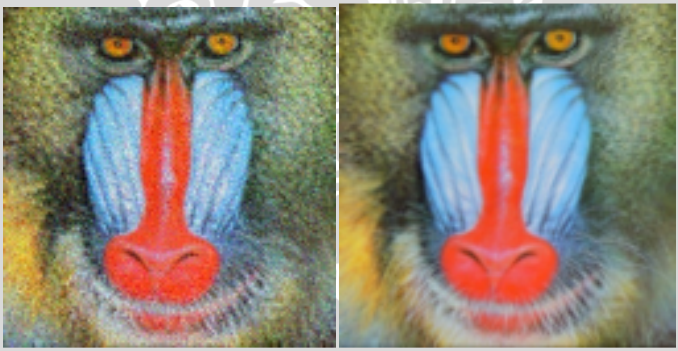
Pada TABEL 4.2 dapat dilihat bahwa semakin banyak jumlah tetangga yang digunakan maka kualitas citra hasil pembesaran juga semakin baik dan berlaku untuk ketiga citra uji. Sebagai contoh, pada citra lena untuk skala pembesaran(Sc) 2 dengan m=4 nilai M-DWTnya 3.997, m=16 nilai M-DWTnya 3.549, m=64 nilai M-DWTnya 3.243, m=256 nilai M-DWTnya 3.169, m=1024 nilai M-DWTnya 3.159. Pada TABEL 4.2 dapat juga dilihat bahwa kualitas yang terbaik dimiliki oleh citra laut dan yang paling buruk dimiliki oleh citra baboon. Hal ini terlihat dari nilai M-DWT 2.336 yang dimiliki oleh citra laut sedangkan nilai M-DWT citra baboon adalah 7.567 untuk nilai Sc dan m yang sama yaitu 2 dan 1024.

Pada pengujian skala pembesaran 4, citra lena dan citra baboon diperkecil terlebih dahulu menjadi 128x128 dan untuk citra laut diperkecil menjadi 256x192. GAMBAR 4.13 merupakan contoh citra hasil pembesaran dengan nilai skala pembesaran 4 dengan jumlah tetangga 4 dan 1024.

(a) (b)



(c) (d)



(e) (f)

Gambar 4.13 Contoh citra hasil pembesaran dengan skala pembesaran 4(modifikasi NEDI): (a) lena dengan m=4; (b) lena dengan m=1024; (c) laut dengan m=4; (d) laut dengan m=1024; (e) baboon dengan m=4; (f) baboon dengan m=1024.

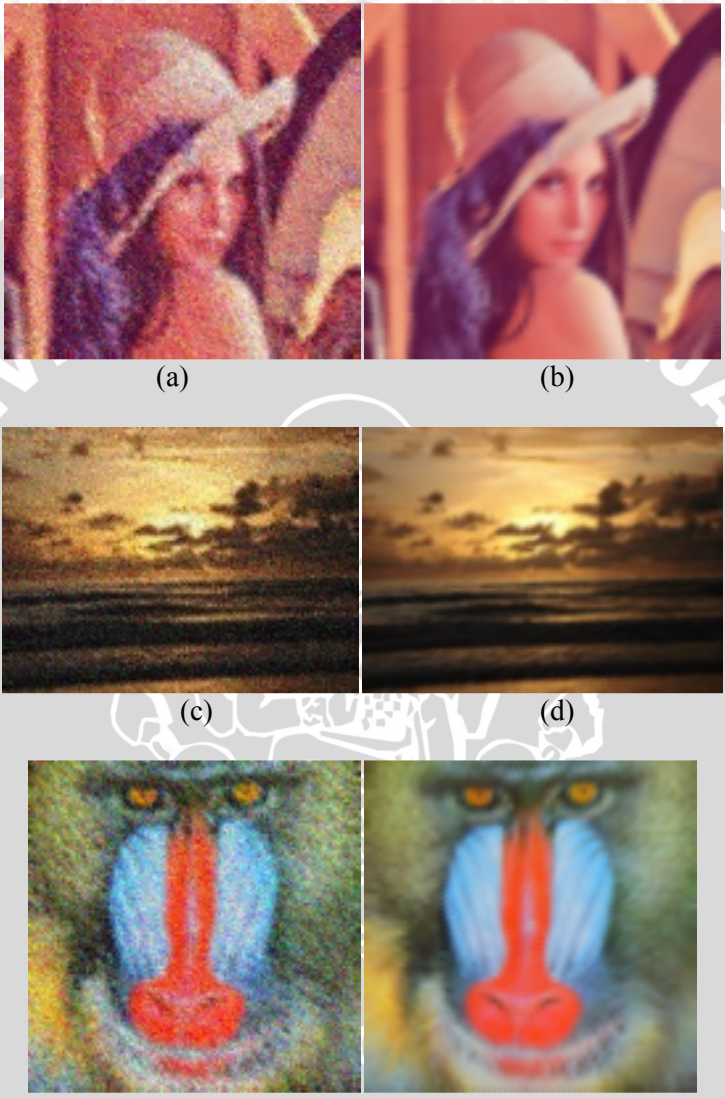
Pengukuran kualitas citra untuk skala pembesaran 4 dapat dilihat pada TABEL 4.3.

Tabel 4.3 Tabel Hasil Uji Kualitas Citra Digital Hasil Pembesaran Menggunakan Modifikasi Algoritma Interpolasi NEDI dengan Skala Pembesaran 4

NE	M	Sc	MSER	MSEG	MSEB	M-DWT
Lena	4	4	453.44	579.87	488.93	5.877
Lena	16	4	223.99	332.83	209.58	5.313
Lena	64	4	192.1	295.50	182.60	5.004
Lena	256	4	182.43	282.20	176.99	4.929
Lena	1024	4	179.84	277.75	175.40	4.890
laut	4	4	505.79	429.73	338.32	5.165
laut.	16	4	267.95	189.77	99.85	4.108
laut	64	4	229.01	157.14	75.10	3.725
laut	256	4	215.16	146.97	70.98	3.548
laut	1024	4	210.11	144.42	70.36	3.499
baboon	4	4	1012.65	1256.27	1262.80	9.416
baboon	16	4	681.00	911.61	943.14	8.972
baboon	64	4	638.97	861.56	889.72	8.816
baboon	256	4	633.52	855.69	881.98	8.794
baboon	1024	4	632.11	854.43	879.70	8.791

Seperti hasil pada TABEL 4.2, melalui TABEL 4.3 juga dapat dilihat bahwa semakin banyak jumlah tetangga yang digunakan maka kualitas citra hasil pembesaran juga semakin baik. Sebagai contoh, pada citra laut untuk Sc 4 dengan m=4 nilai M-DWTnya 5.165, m=16 nilai M-DWTnya 4.108, m=64 nilai M-DWTnya 3.725, m=256 nilai M-DWTnya 3.548, m=1024 nilai M-DWTnya 3.499. Kualitas yang terbaik dimiliki oleh citra laut dan yang paling buruk dimiliki oleh citra baboon. Hal ini terlihat dari nilai M-DWT 3.499 yang dimiliki oleh citra laut sedangkan nilai M-DWT citra baboon adalah 8.791 untuk nilai Sc dan m yang sama yaitu 4 dan 1024.

Pada pengujian skala pembesaran 8, citra lena dan citra baboon diperkecil terlebih dahulu menjadi 64x64 dan untuk citra laut dfiperkecil menjadi 128x96. GAMBAR 4.14 merupakan contoh citra hasil pembesaran dengan nilai skala pembesaran 8 dengan jumlah tetangga 4 dan 1024.



Gambar 4.14 Contoh citra hasil pembesaran dengan skala pembesaran 8(modifikasi NEDI): (a) lena dengan m=4; (b) lena dengan m=1024; (c) laut dengan m=4; (d) laut dengan m=1024; (e) baboon dengan m=4; (f) baboon dengan m=1024 .

Pengukuran kualitas citra untuk skala pembesaran 8 dapat dilihat pada TABEL 4.4.

Tabel 4.4 Tabel Hasil Uji Kualitas Citra Digital Hasil Pembesaran Menggunakan Modifikasi Algoritma Interpolasi NEDI dengan Skala Pembesaran 8

NE	M	Sc	MSER	MSEG	MSEB	M-DWT
Lena	4	8	838.14	1043.75	818.11	7.345
Lena	16	8	482.51	663.40	372.41	6.588
Lena	64	8	410.37	587.61	317.76	6.347
Lena	256	8	403.00	575.32	309.42	6.297
Lena	1024	8	398.16	566.04	304.95	6.253
Laut	4	8	972.06	807.01	573.92	6.551
laut.	16	8	606.75	417.79	193.12	5.371
laut.	64	8	521.44	352.80	150.02	4.900
Laut	256	8	487.90	333.49	141.83	4.733
Laut	1024	8	484.64	331.70	140.03	4.715
Baboon	4	8	1355.93	1644.63	1706.65	10.088
Baboon	16	8	891.99	1125.32	1272.86	9.428
Baboon	64	8	823.33	1058.98	1182.01	9.307
Baboon	256	8	808.93	1046.41	1166.11	9.280
baboon	1024	8	805.83	1044.40	1160.33	9.272

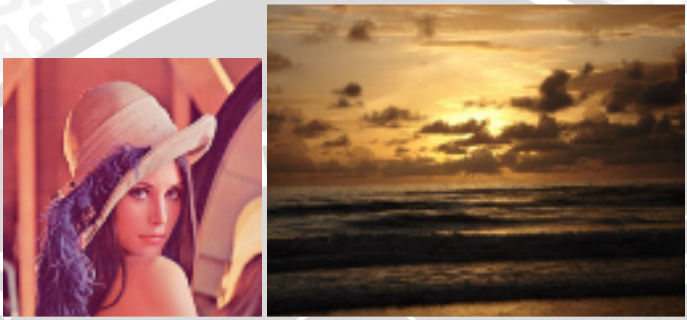
Seperti hasil pada TABEL 4.2, hasil yang terdapat pada TABEL 4.4 juga dapat dilihat bahwa semakin banyak jumlah tetangga yang digunakan maka kualitas citra hasil pembesaran juga semakin baik. Selain itu, kualitas yang terbaik masih dimiliki oleh citra laut dan yang paling buruk dimiliki oleh citra baboon.

4.4.3 Evaluasi Citra Hasil Pembesaran Menggunakan Algoritma Bilinear

Untuk melakukan pengujian menggunakan algoritma interpolasi bilinear digunakan citra uji seperti pada TABEL 4.1. Masing-masing citra uji akan diperbesar dengan skala pembesaran 2, 4, dan 8.

Pada pengujian skala pembesaran 2, citra lena dan citra baboon diperkecil terlebih dahulu menjadi 256x256 dan untuk citra laut

diperkecil menjadi 512x384. GAMBAR 4.15 merupakan contoh citra hasil pembesaran dengan nilai skala pembesaran 2. Pengukuran kualitas citra untuk skala pembesaran 2 dapat dilihat pada TABEL 4.5.



(a) (b)

(c)

Gambar 4.15 Contoh citra hasil pembesaran dengan skala pembesaran 2(bilinear): (a) lena (b) laut (c) baboon.

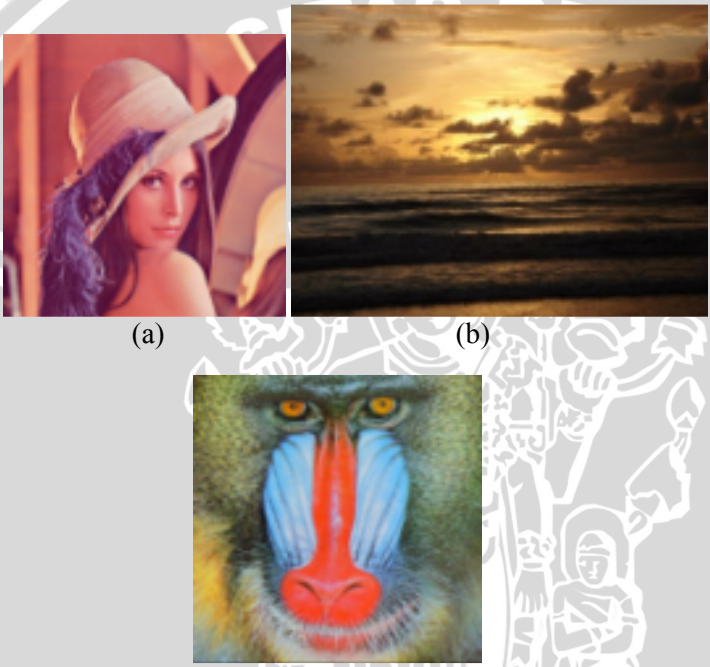
Tabel 4.5 Tabel Hasil Uji Kualitas Citra Digital Hasil Pembesaran Menggunakan Interpolasi Bilinear dengan Skala Pembesaran 2

BL	MSER	MSEG	MSEB	M-DWT
Lena	42.767	79.103	70.556	2.865
Laut	36.767	26.584	24.150	1.774
Baboon	359.567	484.781	474.674	7.050

Dari hasil pada TABEL 4.5 diketahui bahwa kualitas yang terbaik dimiliki oleh citra laut dan yang paling buruk dimiliki oleh

citra baboon. Hal ini terlihat dari nilai M-DWT 1.774 yang dimiliki oleh citra laut sedangkan nilai M-DWT citra baboon adalah 7.050.

Pada pengujian skala pembesaran 4, citra lena dan citra baboon diperkecil terlebih dahulu menjadi 128x128 dan untuk citra laut diperkecil menjadi 256x192. GAMBAR 4.16 merupakan contoh citra hasil pembesaran dengan nilai skala pembesaran 4.Pengukuran kualitas citra untuk skala pembesaran 4 dapat dilihat pada TABEL 4.6.



(a) (b)

(c)

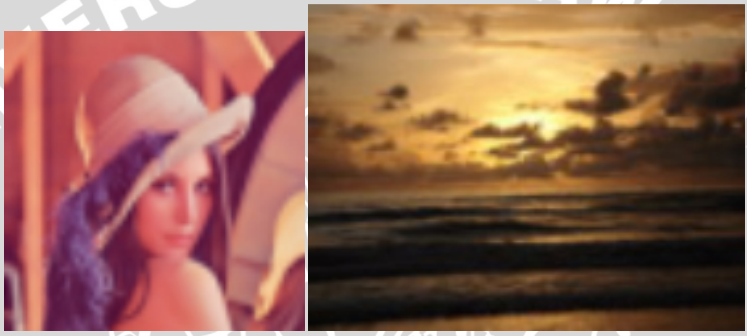
Gambar 4.16 Contoh citra hasil pembesaran dengan skala pembesaran 4(bilinear): (a) lena (b) laut (c) baboon.

Tabel 4.6 Tabel Hasil Uji Kualitas Citra Digital Hasil Pembesaran Menggunakan Interpolasi Bilinear dengan Skala Pembesaran 4

BL	MSER	MSEG	MSEB	M-DWT
Lena	144.621	237.330	153.658	4.471
Laut	112.130	77.325	49.023	2.721
Baboon	567.995	777.087	794.614	8.481

Dari hasil pada TABEL 4.6 diketahui bahwa kualitas yang terbaik dimiliki oleh citra laut dan yang paling buruk dimiliki oleh citra baboon. Hal ini terlihat dari nilai M-DWT 2.721 yang dimiliki oleh citra laut sedangkan nilai M-DWT citra baboon adalah 8.481.

Pada pengujian skala pembesaran 8, citra lena dan citra baboon diperkecil terlebih dahulu menjadi 64x64 dan untuk citra laut dfiperkecil menjadi 128x96. GAMBAR 4.17 merupakan contoh citra hasil pembesaran dengan nilai skala pembesaran 8. Pengukuran kualitas citra untuk skala pembesaran 8 dapat dilihat pada TABEL 4.7.



(a) (b)

(c)

Gambar 4.17 Contoh citra hasil pembesaran dengan skala pembesaran 8(bilinear): (a) lena (b) laut (c) baboon.

Tabel 4.7 Tabel Hasil Uji Kualitas Citra Digital Hasil Pembesaran Menggunakan Interpolasi Bilinear dengan Skala Pembesaran 8

BL	MSER	MSEG	MSEB	M-DWT
Lena	341.942	512.885	282.329	5.878
Laut	252.801	173.487	86.858	3.661
Baboon	742.199	985.536	1094.526	9.091

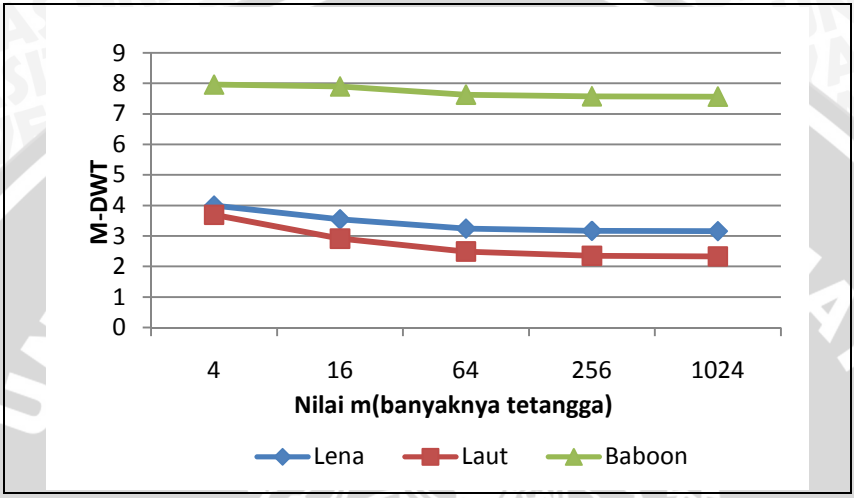
Seperti pada TABEL 4.5, kualitas yang terbaik dimiliki oleh citra laut dan yang paling buruk dimiliki oleh citra baboon. Hal ini terlihat dari nilai M-DWT 3.661 yang dimiliki oleh citra laut sedangkan nilai M-DWT citra baboon adalah 9.091.

4.5 Analisa Hasil

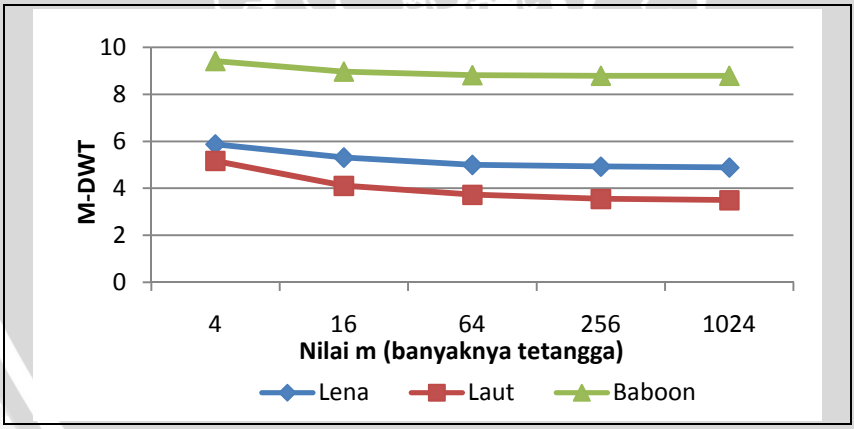
Pada skripsi ini, pengujian kualitas citra hasil pembesaran dilakukan dalam dua metode, yaitu MSE dan M-DWT. Kedua metode ini berkaitan satu sama lain secara tidak langsung. Pada hasil percobaan menunjukkan bahwa jika nilai MSE untuk ketiga elemen warna menurun, maka nilai M-DWT juga mengecil. Hal ini menunjukkan apabila *error*nya sedikit maka kualitas citra semakin baik. Sebagai contoh pada citra lena dengan skala pembesaran(Sc) 2 didapatkan hasil untuk m=4 mempunyai nilai MSER=181.07, MSEG=242.71, MSEB= 234.92, M-DWT= 3.997; m=16 mempunyai nilai MSER=72.26, MSEG=119.00, MSEB= 95.81, M-DWT= 3.549; m=64 mempunyai nilai MSER=58.01, MSEG=98.57, MSEB= 82.23, M-DWT= 3.243; m=256 mempunyai nilai MSER=53.67, MSEG=93.31, MSEB= 80.08, M-DWT= 3.169; m=1024 mempunyai nilai MSER=52.57, MSEG=93.42, MSEB= 79.59, M-DWT= 3.159.

Ketiga buah citra uji memiliki perbedaan karakteristik, yaitu banyaknya perbedaan nilai piksel yang besar pada piksel–piksel yang berdekatan yang dimiliki atau banyaknya variasi nilai piksel yang dimiliki oleh sebuah citra. Ternyata, perbedaan karakteristik ketiga buah citra uji memberikan hasil yang bervariasi dalam hal kualitas citra hasil pembesaran. Agar lebih mudah, penulis membuat tiga buah grafik untuk menunjukkan hubungan antara jumlah tetangga yang digunakan (nilai m) dan nilai kualitas citra berdasarkan metode M-DWT untuk pembesaran citra digital menggunakan modifikasi algoritma interpolasi NEDI. Ketiga buah grafik tersebut masing-

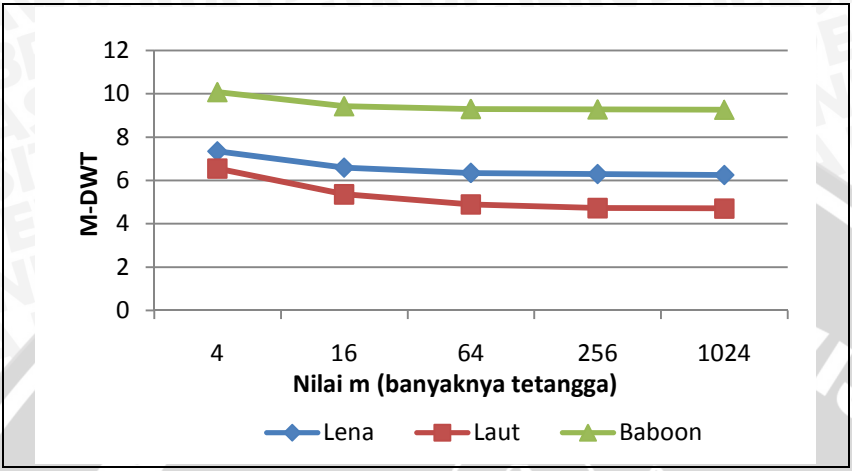
masing adalah untuk perbesaran dengan skala 2 (GAMBAR 4.18), perbesaran dengan skala 4 (GAMBAR 4.19) dan perbesaran dengan skala 8 (GAMBAR 4.20).



Gambar 4.18 Grafik kualitas citra hasil pembesaran untuk skala pembesaran 2 dengan menggunakan modifikasi algoritma NEDI.



Gambar 4.19 Grafik kualitas citra hasil pembesaran untuk skala pembesaran 4 dengan menggunakan modifikasi algoritma NEDI.



Gambar 4.20 Grafik kualitas citra hasil pembesaran untuk skala pembesaran 8 dengan menggunakan modifikasi algoritma NEDI.

Citra baboon mempunyai kualitas citra hasil pembesaran yang paling buruk diantara ketiga citra uji di setiap skala pembesaran. Sebagai contoh pada skala pembesaran 2 dan jumlah tetangga 1024 M-DWT citra lena 3.159, M-DWT citra laut 2.336, M-DWT citra baboon 7.567; pada skala pembesaran 4 dan jumlah tetangga 1024 M-DWT citra lena 4.890, M-DWT citra laut 3.499, M-DWT citra baboon 8.791; pada skala pembesaran 8 dan jumlah tetangga 1024 M-DWT citra lena 6.253, M-DWT citra laut 4.715, M-DWT citra baboon 9.272. Hal ini dikarenakan citra baboon memiliki banyak nilai piksel yang sangat berbeda untuk piksel-piksel yang berdekatan (nilai pikselnya sangat bervariasi) jika dibandingkan dengan dua citra uji lainnya. Perbedaan nilai piksel yang besar pada piksel–piksel yang berdekatan mengakibatkan hasil interpolasi tidak sama dengan citra asli (mempunyai perbedaan nilai yang besar) sehingga citra hasil pembesaran tampak blur seperti yang terlihat pada GAMBAR 4.21

(a) (b)

Gambar 4.21 Perbandingan citra baboon dan citra baboon hasil pembesaran: (a) bagian pada citra asli; (b) bagian pada citra hasil pembesaran dengan skala 2 dan m 1024

Melalui ketiga grafik yang telah dibuat(GAMBAR 4.18, GAMBAR 4.19, dan GAMBAR 4.20) dapat dilihat bahwa semakin banyak jumlah tetangga yang digunakan maka kualitas citra hasil pembesaran juga semakin baik. Sebagai contoh, pada citra laut untuk Sc 4 dengan m=4 nilai M-DWTnya 5.165, m=16 nilai M-DWTnya 4.108, m=64 nilai M-DWTnya 3.725, m=256 nilai M-DWTnya 3.548, m=1024 nilai M-DWTnya 3.499. Dengan sedikitnya jumlah tetangga yang digunakan maka semakin mudah mengarah pada kesalahan pada saat mencari bobot-bobot interpolasi.

Selain itu, melalui ketiga grafik tersebut dapat dilihat bahwa semakin besar skala pembesaran maka kualitas citra hasil pembesaran semakin buruk. Sebagai contoh untuk jumlah tetangga 1024 pada citra lena skala pembesaran 2 nilai M-DWTnya 3.159, skala pembesaran 4 nilai M-DWT nya 4.890, skala pembesaran 8 nilai M-DWTnya 6.253. Hal tersebut diakibatkan cara kerja interpolasi NEDI dalam memperbesar citra. Cara kerja interpolasi NEDI untuk pembesaran citra untuk skala pembesaran lebih dari 2 yaitu dengan memperbesar dengan skala pembesaran 2 hasil pembesaran sebelumnya. Contoh jika kita ingin memperbesar citra dengan skala 4 maka sebuah citra akan diperbesar dengan skala pembesaran 2 dan hasilnya kemudian diperbesar kembali dengan

skala pembesaran 2. Cara kerja seperti ini membuat nilai hasil interpolasi tidak akurat jika dibandingkan dengan citra asli karena sebagian besar nilai–nilai piksel yang digunakan pada pembesaran selanjutnya adalah hasil interpolasi yang merupakan nilai perkiraan untuk piksel-piksel yang belum bernilai saat citra diperbesar.

Jika dibandingkan dengan hasil pembesaran dengan menggunakan algoritma interpolasi bilinear, modifikasi algoritma interpolasi NEDI mempunyai hasil pembesaran lebih jelek untuk ketiga citra uji. Perbandingan dilakukan terhadap hasil terbaik yang dapat dihasilkan oleh modifikasi algoritma intepolasi NEDI (hasil pembesaran menggunakan jumlah tetangga 1024) dengan hasil pembesaran menggunakan interpolasi bilinear untuk semua citra uji. Seperti pada skala pembesaran 2, untuk citra laut diperoleh nilai M-DWT 1.774 dengan menggunakan algoritma interpolasi bilinear sedangkan jika memakai modifikasi algoritma NEDI didapatkan nilai M-DWT 2.336(jumlah tetangga 1024).

BAB V
PENUTUP

5.1 Kesimpulan

Kesimpulan yang didapat dalam pengerjaan Tugas Akhir ini adalah :

1. Modifikasi Algoritma interpolasi NEDI dapat dimanfaatkan sebagai salah satu metode untuk memperbesar citra digital berwarna. Pembesaran citra digital berwarna dengan menggunakan modifikasi algoritma interpolasi NEDI dapat dilakukan dengan menerapkan algoritma ini pada setiap elemen warnanya (R,G, dan B).
2. Kualitas citra hasil pembesaran ditentukan oleh jumlah tetangga yang digunakan. Semakin banyak jumlah tetangga yang digunakan maka kualitas citra yang dihasilkan semakin bagus. Sebagai contoh pada citra lena untuk skala pembesaran(Sc) 2 dengan m=4 nilai M-DWTnya 3.997 dan nilai M-DWT ini cenderung menurun dengan semakin banyaknya jumlah tetangga yang digunakan sehingga pada niali m=1024 nilai M-DWTnya 3.159.
3. Citra yang memiliki banyak nilai piksel yang sangat berbeda untuk piksel-piksel yang berdekatan atau pikselnya sangat bervariasi nilainya jika dibesarkan memakai modifikasi algoritma interpolasi NEDI, hasilnya akan tampak blur dan memiliki kualitas yang buruk.Sebagai contoh pada hasil pengujian pada skala pembesaran 2 dan jumlah tetangga 1024 M-DWT citra lena 3.159, M-DWT citra laut 2.336, M-DWT citra baboon 7.567.
4. Jika dibandingkan dengan hasil pembesaran dengan menggunakan algoritma interpolasi bilinear, modifikasi algoritma interpolasi NEDI mempunyai hasil pembesaran lebih jelek. Seperti pada skala pembesaran 2, untuk citra laut diperoleh nilai M-DWT 1.774 dengan menggunakan algoritma interpolasi bilinear sedangkan jika memakai modifikasi algoritma NEDI didapatkan nilai M-DWT 2.336 (jumlah tetangga 1024).

5.2 Saran

Beberapa saran untuk penelitian lebih lanjut yang dapat diberikan oleh penulis adalah :

1. Disarankan tidak menggunakan jumlah tetangga yang sedikit pada pemakaian modifikasi algoritma interpolasi NEDI untuk pembesaran citra digital karena menurut penelitian ini dengan sedikitnya jumlah tetangga yang digunakan kualitas citra hasil pembesaran tidak bagus .
2. Disarankan tidak menggunakan citra yang mempunyai banyak nilai piksel yang sangat berbeda untuk piksel-piksel yang berdekatan atau citra yang nilai pikselnya bervariasi seperti citra baboon yang digunakan pada citra uji.

DAFTAR PUSTAKA

Abouzeid, Alhusein A. dan Wu, Huaming. 2004. *Energy Efficient Distributed JPEG2000 Image Compression in Multihop Wireless Networks*. http://csr.bu.edu/aswn2004/slides/Tuesday/Session6/2-Alhussein.pdf. Diakses tanggal 22 April 2008.

Ahmad, Balza dan Firdausy, Kartika. *Teknik Pengolahan Citra Digital Menggunakan Delphi*. Yogyakarta: Ardi Publishing, 2005.

Anonymous. 1999. *RGB*. http://searchsmb.techtarget.com/sDefinition/0,,sid44_gci212900,00.html. Diakses tanggal 1 Oktober 2007.

Anonymous. 2007[a]. *Digital Image Interpolaton*. http://www.cambridgeincolour.com/tutorials/image-interpolation.htm. Diakses tanggal 4 September 2007.

Anonymous. 2007[b]. *Edge Directed Interpolation*. http://chiranjivi.tripod.com/EDITut.html. Diakses tanggal 4 September 2007.

Gayle, Devon dkk. 2006. *A Full-Reference Color Image Quality Measure in The DWT Domain.* http://www.ee.bilkent.edu.tr/~signal/defevent/papers/cr1039.pdf. Diakses tanggal 22 April 2008.

Gozalez, Rafael C.. *Digital Image Processing*. USA: Addison-Wesley Publishing Company Inc., 1992.

Jähne, Bernd. *Digital Image Processing*. Germany: Springer, 2002.

Kumar, Satish. 2001**.** *An Introduction to Image Compression*. http://www.debugmode.com/imagecmp/An%20Introduction%20to%20Image%20Compression.htm. Diakses tanggal 11 Desember 2007.

Li, Xin dan Orchard, M.. 2001. *New Edge Directed Interpolation*. http://neuron2.net/library/nedi.pdf. diakses tanggal 4 September 2007.

Mohideen, S. Kother. 2008. *Image De-Noising using Discrete Wavelet Transform*. http://paper.ijcsns.org/07_book/200801/20080129.pdf. Diakses tanggal 22 April 2008

Woo,Chi.2007. *Interpolation*. http://planetmath.org/encyclopedia/Interpolation.html. Diakses tanggal 1 Oktober 2007