KATA PENGANTAR

Puji Syukur penulis panjatkan kehadirat Allah SWT karena berkat rahmat dan hidayah-Nya, penulis dapat menyelesaikan Laporan Tugas Akhir yang berjudul "*Filter Bandpass* dan *Bandstop* Untuk Menurunkan *Noise* Pada Citra Menggunakan Delphi 7.0" dengan baik dan lancar. Laporan Tugas Akhir ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata Satu (S1) pada program Studi Matematika Ekstensi Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Diponegoro Semarang.

Selama pelaksanaan penyusunan Laporan Tugas Akhir ini, penulis banyak mendapat bimbingan, arahan dan bantuan dari berbagai pihak yang sangat mendukung. Oleh karena itu dengan segala kerendahan hati, penulis ingin mengucapkan terima kasih dengan tulus kepada :

1. Ibu Dra. Rum Hastuti, M.Si selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Diponegoro.
2. Ibu Dr. Widowati, S.Si, M.Si selaku Ketua Jurusan Matematika Fakultas MIPA Universitas Diponegoro.
3. Bapak Bambang Irawanto, S.Si, M.Si selaku Ketua Program Studi Matematika Fakultas MIPA Universitas Diponegoro.
4. Bapak Drs.Eko Adi Sarwoko, M.kom dan Bapak Aris Sugiharto, S.Si, M.Kom selaku dosen pembimbing yang telah memberi petunjuk,

nasehat, pengarahan serta saran dan bimbingan dalam menyelesaikan Laporan Tugas Akhir ini.

5. Bapak dan Ibu dosen Jurusan Matematika atas semua ilmu yang telah diberikan.
6. Ibu dan Bapak tercinta saya yang telah mendoakan dan memberikan semua fasilitas serta semua saudara dan teman yang senantiasa selalu memberikan suport dan semangat dalam penyusunan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih banyak kekurangannya. Untuk itu penulis mengharapkan kritik dan saran yang bersifat membangun demi kesempurnaan Tugas Akhir ini. Semoga Tugas Akhir ini dapat membawa manfaat bagi penulis sendiri khususnya dan bagi para pembaca pada umumnya.

Semarang, 2010

Penulis

ABSTRAK

Pengolahan citra digital adalah suatu proses mengolah citra dengan menggunakan komputer agar didapat sebuah citra yang kualitasnya lebih baik. Di dalam pengolahan citra terdapat berbagai macam metode untuk menghasilkan citra yang kualitasnya lebih baik. Salah satu metodenya adalah *noise filtering*. Pada tugas akhir ini dibahas tentang *noise filtering* dengan menggunakan *filter Bandpass* dan *Bandstop*. Kedua metode ini berfungsi untuk mengurangi *noise* pada citra yang berformat BMP dengan mengganti nilai piksel *noise* dengan nilai baru yang merupakan intensitas warna rata-rata piksel di sekitarnya yang bukan *noise*. Pada program yang telah dibuat diimplementasikan kedua metode tersebut sehingga diperoleh hasil citra yang berkurang *noise*-nya.

Kata kunci : *filtering* citra, *bandpass, bandstop*

ABSTRACT

Digital image processing is a process image using computer to be got a image which is better quality. In processing of image there are assorted of method to yield image which is better quality. One of the method is a filtering noise. At this final duty will be studied about filtering noise using Bandpass filter and Bandstop filter. Both of this method function to lessen noise at image which have bitmap format by changing value of piksel noise with new value which represent mean colour intensity of piksel around him which non noise. At this program which have been made by implementation both of the method so obtained result of image with lessen noise.

Keyword : image *filtering bandpass, bandstop*.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pemanfaatan teknologi *digital* pada bidang pengolahan citra sudah semakin berkembang, hal ini dapat dilihat dengan banyaknya software atau perangkat lunak dalam bidang pengolahan citra seperti Adobe Photoshop, ACDsee dan perangkat lunak yang lainnya. Pengolahan citra secara *digital* adalah memproses suatu citra sehingga menghasilkan citra lain yang lebih baik kualitasnya. Dalam proses perbaikan citra terdapat berbagai metode di antaranya *coding, scalling, sampling dan filtering* dan metode yang lainnya.

Citra *digital* dapat diperoleh melalui scanner, kamera digital dan alat-alat lain yang berfungsi untuk *capture* citra. Pada saat proses *capture* (pengambilan gambar), ada beberapa gangguan yang mungkin terjadi, seperti kamera tidak fokus atau munculnya bintik-bintik yang bisa jadi disebabkan oleh proses *capture* yang tidak sempurna. Setiap gangguan pada citra dinamakan dengan *noise*. *Noise* pada citra tidak hanya terjadi karena ketidak-sempurnaan dalam proses *capture*, tetapi bisa juga disebabkan oleh kotoran-kotoran yang terjadi pada citra, oleh karena itu citra yang memiliki *noise* perlu diadakan pemrosesan melalui pengolahan citra agar dapat diperoleh citra yang kualitasnya lebih baik.

Noise yang terdapat pada citra digital dapat dikurangi atau dihilangkan dengan berbagai macam metode, metode-metode yang sering digunakan antara lain dengan metode median filter. Median filter adalah teknik mereduksi *noise* dengan cara mengganti piksel *noise* dengan nilai median atau nilai tengah dari

piksel-piksel tetangganya, dan piksel itu sendiri. Filter median sangat efektif untuk menghilangkan *noise* jenis *salt and peper* dan juga dapat mempertahankan detail citra karena tidak tergantung dengan nilai-nilai yang berbeda dengan nilai-nilai yang umum dalam lingkungannya. Selain median filter terdapat metode lain yaitu Gaussian filter. Gaussian filter adalah suatu filter dengan nilai pembobotan pada setiap piksel dipilih berdasarkan bentuk fungsi Gaussian. Filter ini sangat baik dan sering digunakan untuk menghilangkan *noise* yang bersifat sebaran normal, yang banyak dijumpai pada citra hasil proses digitalisasi menggunakan kamera, hal ini terjadi karena merupakan fenomena alamiah akibat dari sifat pantulan cahaya dan kepekaan sensor cahaya pada kamera itu sendiri. Selain metode-metode di atas terdapat berbagai macam metode yang sering digunakan untuk menghilangkan *noise*, namun pada tugas akhir ini akan diteliti menggunakan metode bandpass dan bandstop untuk mereduksi *noise* karena metode ini akan menghasilkan citra yang lebih baik dimana efek blur yang dihasilkan lebih kecil dibandingkan dengan metode *Gaussian filter* sehingga citra hasil akan tampak lebih baik.

1.2 Perumusan Masalah

Berdasarkan latar belakang di atas, dapat dirumuskan permasalahan yaitu bagaimana membuat suatu perangkat lunak pengolahan citra yang spesifik untuk mengurangi *noise* yang terdapat pada suatu citra *digital*, menggunakan *filter bandpass* dan *filter bandstop*.

1.3 Batasan Masalah

Pembuatan aplikasi, dibuat sesuai dengan tujuannya yaitu mengurangi *noise* pada citra dengan *filter bandpass* dan *bandstop,* adapun batasan masalahnya adalah sebagai berikut :

- Teknik *filtering* yang digunakan adalah *bandpass* dan *bandstop*.
- Citra *digital* yang diolah dalam program menggunakan format BMP.
- Pemasukan nilai batas atas dan batas bawah di antara 0 sampai 255.
- Cara pemberian *noise* dengan di-*brush* menggunakan aplikasi *paint* yang terdapat pada *windows.*
- Kemampuan yang dimiliki program adalah pemanggilan *file*, menyimpan *file* dan melakukan pengolahan citra dengan *filter bandpass* dan *bandstop.*
- Pembuatan program menggunakan Borland Delphi 7.0.

1.4 Tujuan

Tujuan dari tugas akhir ini adalah menghasilkan suatu perangkat lunak pengolahan citra yang yang lebih mudah digunakan oleh user untuk mengurangi *noise* yang terdapat pada suatu citra *digital* dan untuk mengetahui seberapa jauh keberhasilan *filter bandpass* dan *filter bandstop* dalam memfilter *noise.*

1.5 Manfaat

Manfaat dari pembuatan tugas akhir ini adalah didapatkannya citra yang kualitasnya lebih baik setelah dilakukan proses pengolahan citra menggunakan *Filter Bandpass* dan *Filter Bandstop*.

1.6 Sistematika Penulisan

Penulisan tugas akhir ini dilakukan secara sistematis dalam beberapa bab yang akan mempermudah dalam memberikan gambaran dan mengetahui isi dari uraian-uraian di dalamnya, bab-bab tersebut meliputi, Bab I akan di uraikan mengenai apa yang menjadi latar belakang, perumusan dan pembatasan masalah, tujuan, manfaat dan sistematika penulisan dari tugas akhir ini. Pada Bab II di dalamnya diuraikan teori-teori yang ada hubungannya dengan pokok permasalahan yang dipilih, dan akan dijadikan sebagai landasan dalam penulisan tugas akhir ini. Bab III berisikan penjelasan tentang analisa kebutuhan dan rancangan pembuatan serta tampilan program. Pada bab IV akan diuraikan mengenai pembahasan serta implementasi dari perancangan sistem. Dan pada Bab V berisi tentang kesimpulan dari uraian-uraian bab yang telah dibahas sebelumnya.

BAB II

DASAR TEORI

2.1 Rekayasa Perangkat Lunak

Perangkat lunak adalah

1. Instruksi atau program computer yang bila dieksekusi dapat menjalankan fungsi tertentu.
2. Struktur data yang dapat membuat program manipulasi informasi.
3. Dokumen yang menjelaskan operasi dan penggunaan program. (Pressman, 2000)

Menurut Fritz, Gerald, Jerry, Tahun 1981 :

Rekayasa Perangkat Lunak adalah pengembangan dan penggunaan prinsip pengembangan suara untuk memperoleh perangkat lunak secara ekonomis yang reliable dan bekerja secara efisien pada mesin nyata.

Menurut IEEE (Institute of Electrical and Electronik Engineers) :

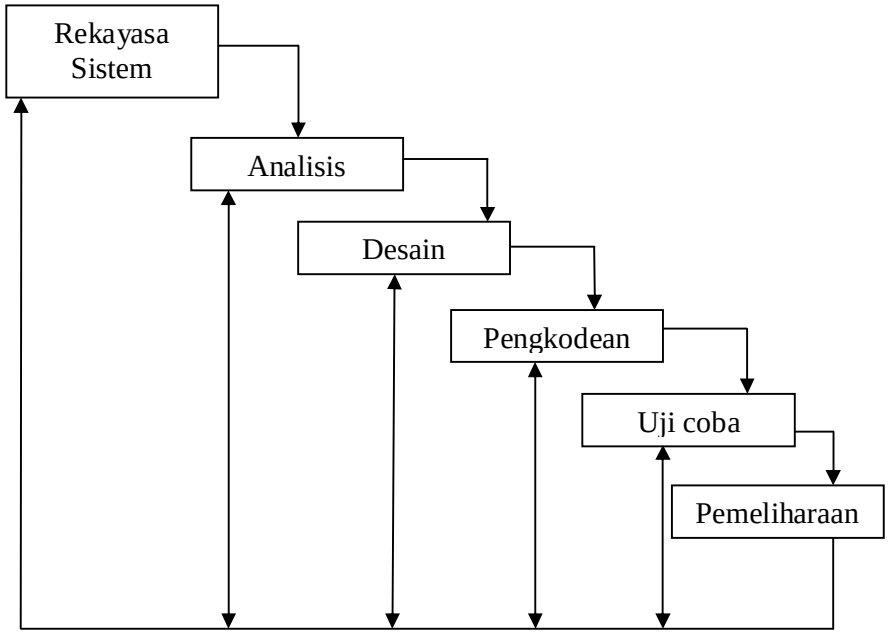
Rekayasa Perangkat Lunak adalah aplikasi dari rancangan yang sistematik, berdisiplin dan *quantifiable* (dapat diukur) terhadap pengembangan, operasi dan perawatan perangkat lunak.

Di dalam merekayasa perangkat lunak terdapat tujuan yang hendak dicapai atau diinginkan yaitu bagaimana menghasilkan produk perangkat lunak yang baik, yang dimaksud dengan lebih baik adalah perangkat lunak yang mudah di gunakan, dirawat dapat diandalkan, bekerja secara efisien dan mempunyai antarmuka pemakai yang baik dan juga di tinjau dari segi biaya yang ekonomis dan efisien.

Perangkat lunak memiliki karakteristik yang berupa elemen logika dan bukan merupakan elemen fisik**.** Dengan demikian, perangkat lunak memiliki cirri berbeda dengan perangkat keras, menurut pressman 2000 karakteristik rekayasa perangkat lunak adalah sebagai berikut :

a) Perangkat lunak terciptakan adanya pengembangan
b) Perangkat lunak tidak akan menjadi ”kuno” (tidak dipakai lagi)
c) Kebanyakan perangkat lunak dibuat secara khusus untuk tujuan tertentu dari pada disusun dengan komponen yang ada.

Perangkat lunak memiliki yang sering disebut “*waterfall mode*” atau siklus klasik, yang meliputi urutan sistematika dalam rangka pengembangan perangkat lunak. Langkah-langkah proses waterfal mode dapat dilihat pada gambar 2.1.



Gambar 2.1 Model Pengembangan *waterfall (presman 2000)*

1. Rekayasa sistem dibutuhkan karena perangkat lunak selalu menjadi bagian dari sebuah sistem yang lebih besar. Hal ini dimulai dengan melakukan

penyusunan terhadap kebutuhan-kebutuhan tersebut ke dalam perangkat lunak. Pandangan terhadap sistem ini diperlukan ketika perangkat lunak harus dihubungkan dengan elemen lain seperti *hardware* dan *database*. rekayasa sistem meliputi pengumpulan kebutuhan pada tingkat sistem dengan sejumlah kecil desain dan analisis tingkat atas.

2. Tahap Analisis (*Analysis Requirement*)

   Analisis kebutuhan perangkat lunak merupakan proses pengumpulan kebutuhan yang dikhususkan pada perangkat lunak, untuk memahami ruang lingkup informasi perangkat lunak tersebut seperti fungsi-fungsi yang dibutuhkan, cara kerja, dan antarmuka (*interface*). Kebutuhan akan sistem ini didokumentasikan dan ditinjau kembali bersama user atau pemakai perangkat lunak.

3. Tahap Desain (*Desain*)

   Desain perangkat lunak merupakan proses langkah-langkah yang dipusatkan pada empat atribut program yang berbeda, yaitu struktur data, arsitektur perangkat lunak, perincian prosedur dan karakteristik antarmuka. Proses desain menerjemahkan kebutuhan ke dalam perangkat lunak yang dapat diperkirakan untuk kualitas sebelum dimulai pengkodean. Desain juga didokumentasikan dan menjadi bagian dari konfigurasi perangkat lunak.

4. Tahap Pengkodean (*Coding*)

   Tahap pengkodean bertujuan untuk menerjemahkan desain ke dalam bentuk bahasa pemrograman yang dapat di jalankan oleh mesin khususnya *computer*.

5. Tahap Uji Coba (*Testing*)

   Proses uji coba dititik beratkan pada logika internal perangkat lunak, untuk menjamin bahwa semua perintah telah dicoba. Dan fungsi-fungi eksternal, uji coba dilakukan untuk menemukan kesalahan (*error*), serta memastikan bahwa dengan input yang didefinisikan akan menghasilkan output yang sesuai dengan keinginan user.

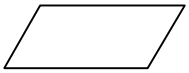
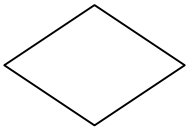
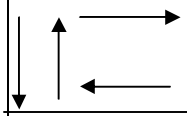
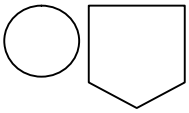
6. Pemeliharaan (*Maintenance*)

   Pada tahap ini, perangkat lunak akan mengalami perubahan-perubahan setelah digunakan. Hal ini dapat terjadi pada saat di temukannya kesalahan, proses adaptasi perangkat lunak pada sistem operasi dan perangkat lunak lainnya yang dapat menimbulkan kesalahan. Pemeliharaan perangkat lunak menggunakan kembali setiap langkah daur hidup (*life cycle*) yang terdahulu untuk sebuah program yang sudah ada. Hal ini lebih baik daripada membuat suatu program baru lagi.

Di dalam permodelan suatu sistem memiliki alat bantu untuk menggambarkan suatu diagram aliran instruksi yang berisi serangkaian simbol yang menunjukkan secara logika tentang hubungan antar elemenn-elemen yang disebut diagram alir atau *flowchart*. Diagram alir atau *flowchart* dapat melukiskan sebuah kagiatan pengolahan informasi yang berkisar tentang konfigurasi komputer sampai tahap rinci dalam program.

Pada Tabel 2.1 akan dijelaskan mengenai simbol-simbol yang terdapat dalam diagram alir program.

Tabel 2.1 smbol-simbol diagaram alir

Gambar	Deskripsi
	**Simbol Masukan / keluaran (input / output)** Untuk mewakili input dan output.
	**Simbol Proses** Digunakan untuk menyatakan suatu kegiatan manipulasi data.
	**Simbol Kondisi** Digunakan untuk pengambilan keputusan terhadap altrnatif.
	**Simbol Proses Terdefinisi** Menunjukan suatu proses program lain yang ditunjukan di tempat lain
	**Simbol Arah Aliran** Menunjukan arah alur pemrosesan
	**Simbol Konektor / Penghubung** Digunakan untuk menghubungkan antar bagian dalam diagram alir
	**Simbol Persiapan** Untuk memberikan nilai awal suatu besaran

2.2 Citra Digital

Citra merupakan suatu signal digital dua dimensi yang dapat di observasi oleh sistem visual manusia. Ditinjau dari sudut pandang matematis, citra merupakan fungsi menerus (*continue*) dari intensitas cahaya pada bidang dwimatra. Sumber cahaya menerangi obyek, obyek memantulkan kembali sebagian dari berkas cahaya tersebut. Pantulan cahaya ini ditangkap oleh alat-alat

optik, misalnya mata manusia, kamera pemindai (*scanner*) kamera digital, dan sebagainya, sehingga banyak obyek citra tersebut terekam.

Citra digital merupakan suatu larik / *array* dua dimensi atau suatu matrik yang elemen-elemennya menyatakan tingkat keabuan dari elemen gambar; jadi informasi yang terkandung di dalamnya bersifat diskrit (Arymurthy dan Setiawan, 1992).

Pengolahan citra digital adalah pemrosesan citra, khususnya dengan menggunakan komputer, menjadi citra yang kualitasnya lebih baik (Rinaldi Munir, 2004). Teknik-teknik pengolahan citra mentransformasikan citra menjadi citra lain. Jadi masukannya adalah citra dan keluarannya juga citra, namun keluaran mempunyai kualitas yang lebih baik dari pada citra masukan.

Terdapat dua jenis citra yaitu citra diam dan citra bergerak. Citra bergerak adalah rangkaian citra diam yang ditampilkan secara beruntun (sekuensial) sehingga memberi kesan pada mata sebagai gambar yang bergerak. Sedangkan citra diam adalah citra yang tidak bergerak. Gambar 2.2 merupakan contoh dari citra diam.

Gambar 2.2 Citra Diam (citra bunga dan citra wajah)

Agar dapat diolah dengan komputer *digital,* suatu citra harus direpresentasikan secara numerik dengan nilai-nilai diskrit. Representasi citra dari

fungsi kontinu menjadi nilai-nilai diskrit disebut pencitraan (*imaging*) atau digitalisasi. Citra yang dihasilkan inilah yang disebut citra digital(*Digital Image*), dinyatakan sebagai kumpulan piksel dalam matrik dua dimensi. Pada umumnya citra digital berbentuk empat persegi panjang dan dimensi ukurannya dinyatakan tinggi dikalikan dengan lebar atu lebar dikalikan dengan panjang.

Citra digital yang berukuran N x M lazim dinyatakan dengan matriks yang berukuran N baris dan M kolom seperti pada gambar 2.3.

$$F(x,y) \approx \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,M-1) \\ f(1,0) & f(1,1) & \dots & f(1,M-1) \\ f(2,0) & f(2,1) & \dots & f(2,M-1) \\ \vdots & \vdots & \vdots & \vdots \\ f(N-1,0) & f(N-1,1 & \dots & f(N-1,M-1) \end{bmatrix}$$

Gambar 2.3 : Representasi citra digital dalam matriks N x M (Gonzalez, 1987)

Masing masing elemen pada citra digital (berarti elemen matriks) disebut sebagai *picture element* atau piksel (*piksel*). Jadi citra yang berukuran N x M mempunyai NM buah piksel. Misalkan sebuah citra *digital* berukuran 256 x 256 piksel dengan derajat keabuan 256 level dan dipresentasikan secara numerik dengan matriks terdiri dari 256 baris (di-indeks dari 0 sampai 255) dan 256 kolom seperti gambar 2.4.

(0,0)

$$\begin{bmatrix} 0 & 134 & 145 & \dots & 201 \\ 10 & 110 & 145 & \dots & 212 \\ 90 & 78 & 152 & \dots & 199 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 132 & 154 & 128 & \dots & 222 \end{bmatrix}$$

(255,255)

(0,0)

0	134	145	...	210
10	110	145	...	212
90	78	152	...	199
⋮	⋮	⋮	⋮	⋮
132	154	128	...	222

(255,255)

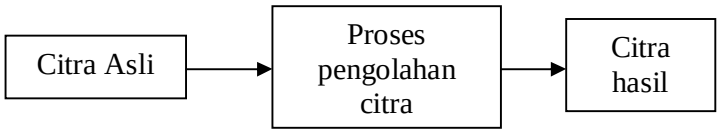
Gambar 2.4: Contoh Representasi citra dalam matriks N x M

Piksel pertama pada koordinat (0,0) mempunyai intensitas 0 yang berarti warna piksel tersebut hitam, piksel kedua pada koordinat (0,1) mempunyai intensitas 134 yang berarti warna antara hitam dan putih. Citra hitam putih disebut juga citra satu kanal, karena warna hanya ditentukan oleh satu fungsi saja. Citra berwarna (*color images*) dikenal dengan nama citra spektral, karena warna pada citra disusun oleh tiga komponen warna yang disebut komponen RGB, yaitu merah (*red*), hijau (*green*), dan biru (*blue*). Intensitas suatu titik pada citra berwarna merupakan kombinasi dari tiga intensitas : derajat keabuan merah (f_{merah} (x,y)), hijau (f_{hijau} (x,y)) dan biru (f_{biru} (x,y)).

2.3 Pengolahan Citra

Pengolahan citra adalah pemrosesan citra, khususnya menggunakan komputer, menjadi citra yang kualitasnya lebih baik dan sesuai dengan keinginan pemakai. Pengolahan citra bertujuan memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia atau mesin (dalam hal ini komputer).

Teknik-teknik pengolahan citra mentransformasikan citra ke citra yang lain. Jadi masukannya adalah citra dan keluarannya juga citra, namun citra keluaran atau hasil mempunyai kualitas lebih baik dari pada citra masukan. Jalanya proses pengolahan citra dapat dilihat pada gambar 2.5.



Gambar 2.5 Alur Proses pengolahan citra (Munir 2004)

Terdapat beberapa operasi di dalam pengolahan citra yang dapat diklasifikasi dalam beberapa jenis, antara lain :

1. Perbaikan Kualitas Citra (*Image Enhancement*)

Jenis operasi ini bertujuan untuk memperbaiki citra dengan cara memanipulasi parameter parameter citra. Dengan operasi ini, ciri-ciri khusus yang terdapat di dalam citra lebih ditonjolkan. Contoh-contoh perbaikan citra adalah:

a. Perbaikan kontras gelap atau terang.
b. Perbaikan tepian obyek (*edge enchancement*).
c. Penajaman citra (*sharpening*).
d. Pemberian warna semu (*peseudocoloring*).
e. Penipisan derau (*noise filtering*).

2. Pemugaran Citra (*Image restoration*)

Operasi ini bertujuan untuk menghilangkan atau meminimumkan cacat pada citra. Tujuannya hmpir sama dengan operasi perbaikan citra, bedanya pada pemugaran citra penyebab degradasi gambar diketahui.

Contoh operasinya adalah :

a. Penghilangan kesamaran (deblurring)
b. Penghilangan derau (*noise*)

3. Pemampatan Citra (*Image Compression*)

Operasi ini dilakukan agar citra dapat dipresentasikan dalam bentuk yang lebih kompak sehingga memerlukan memori yang lebih sedikit. Hal penting yang harus diperhatikan dalam pemampatan citra adalah citra yang telah dimampatkan harus tetap mempunyai

kualitas gambar yang bagus. Contoh metode pemampatan citra adalah metode JPEG.

4. Segmentasi Citra (*Image Segmentation*)

   Jenis operasi ini bertujuan untuk memecahkan suatu citra ke dalam beberapa segmen dengan suatu kriteria tertentu. Jenis operasi ini berkaitan erat dengan pengenalan pola.

5. Analisa Citra (Image Analysis)Jenis operasi ini bertujuan menghitung besaran kuantitatif dari citra untuk menghasilkan deskripsinya. Teknik analisa citra mengekstraksi ciri-ciri tertentu membantu dalam identifikasi obyek. Proses segmentasi terkadang diperlukan untuk mengalokasi obyek yang diinginkan dari sekelilingnya. Contoh-contoh analisa citra antara lain:

   a. Pendeteksi tepi obyek (*edge detection*)
   b. Ekstraksi batas (*boundary*)
   c. Representasi daerah (*region*)

6. Rekonstruksi Citra (Image Reconstruction) Jenis operasi ini bertujuan untuk membentuk ulang obyek dari beberapa citra hasil proyeksi. Opersi rekonstruksi citra banyak digunakan dalam dunia medis. Misalnya beberapa foto rontgen dengan sinar X digunakan untuk membentuk ulang gambar organ tubuh.

2.4 Format File Bitmap (BMP)

Format citra yang baku di lingkungan sistem operasi Microsoft Windows adalah file bitmap (BMP). Pada saat ini format BMP kurang begitu populer dan

mulai jarang digunakan dibanding format JPG atau GIF, karena file BMP pada umumnya tidak dimampatkan, sehingga ukuran relatif lebih besar dari pada file JPG atau GIF.

Terjemahan bebas bitmap adalah pemetaan bit. Artinya nilai intensitas piksel di dalam citra dipetakan ke sejumlah bit tertentu. Peta bit umumnya adalah 8, yang berarti setiap piksel panjangnya 8 bit. Delapan bit ini mempresentasikan nilai intensitas piksel. Dengan demikian ada sebanyak 2^8 =256 derajat keabuan, mulai dari 0 (00000000) sampai 255 (11111111). Setiap berkas bitmap terdiri atas header berkas, header bitmap, informasi palet dan data bitmap. Header adalah data yang terdapat pada awal bagian berkas citra.

Terdapat tiga macam citra dalam format BMP, yaitu citra biner, citra berwarna dan citra hitam-puth (*grayscale*). Citra biner hanya memiliki dua nilai keabuan 0 dan 1. Oleh kerena itu 1 bit telah cukup untuk mempresentasikan nilai piksel. Citra berwarna adalah citra yang lebih umum. Warna yang terlihat di dalam citra bitmap merupakan kombinasi dari tiga komponen warna, yaitu : R (Red), G (Green) dan B (Blue). Kombinasi dari tiga warna RGB tersebut menghasilkan warna yang khas untuk piksel yang bersangkutan. Pada citra 256 warna, setiap piksel memiliki panjang 8-bit, akan tetapi komponen RGBnya disimpan dalam tabel RGB yang disebut *palet.* Pada tabel 2.5 berikut akan memperlihatkan panjang informasi palet untuk setiap versi *bitmap*, masing-masing untuk citra 16 warna, 256 warna dan 16,7 juta warna. Berkas citra 24-bit tidak mempunyai palet RGB, karena langsung diuraikan ke dalam data *bitmap*.

Tabel 2.2 : Panjang informasi *palet bitmap* berwarna

Citra m warna	*Palet bitmap*
Citra 16 warna	64 *byte*
Citra 256 warna	1024 *byte*
Citra 16,7 juta warna	0 *byte*

Informasi *palet* warna terletak sesudah *header bitmap*. Informasi *palet* warna dinyatakan dalam satu tabel RGB. Setiap *entry* pada tabel terdiri atas tiga buah *field* yaitu, R (*Red*), G (*Green*), dan B (*Blue*). Data bitmap diletakan sesudah informasi *palet*.

Format citra 8-*bit* dapat dilihat pada gambar 2.7. Format citra 4-*bit* (16 warna), hampir sama dengan format citra 8-*bit*. Pada citra 4-*bit* dan citra 8-*bit*, warna suatu piksel diacu dari tabel informasi palet *entry* ke-k (*k* merupakan nilai rentang 0-15 untuk citra 16 warna dan 0-155 untuk citra 256 warna). Sebagai contoh pada gambar 2.6, piksel pertama bernilai 2, warna piksel pertama ini ditentukan oleh komponen RGB pada *palet* warna *entry* ke-2, yaitu R=14, G=13 dan B=16.piksel kedua serupa dengan piksel pertama. Piksel ketiga bernilai 1, warna ditentukan oleh komponen RGB pada *palet* warna *entry* ke-1, yaitu R=20, G=45 dan B=24. Demikian seterusnya untuk piksel-piksel lainnya. Khusus untuk citra hitam-putih 8-*bit*, komponen R,G dan B suatu piksel bernilai sama dengan data bitmap piksel tersebut. Jadi piksel dengan nilai data bitmap 129, memiliki nilai R=129, G=129 dan B=129.

<header berkas>
<header bitmap>
<palet warna>

	R	G	B
1	20	45	24
2	14	13	16
3	12	17	15
...	...	...	...
255	46	78	25

<data bitmap>
2 2 1 1 1 3 5…

Gambar 2.6: Format citra 8-*bit* (256 warna) (Munir, 2004)

Citra yang lebih kaya warna adalah citra 24-*bit*. Setiap piksel panjangnya 24-*bit*, karena setiap bit langsung menyatakan komponen warna merah (8-bit), komponen warna hijau (8-*bit*) dan komponen warna biru (8-*bit*). Citra 24-*bit* juga disebut citra 16 juta warna karena mampu menghasilkan 2^{24} =16.777.216 kombinasi warna. Contohnya seperti pada Gambar 2.8 berikut ini, dimana piksel pertama memiliki nilai R=20, G=19 dan B=21. Piksel kedua memiliki nilai R=24, G=23 dan B=24 dan demikian seterusnya.

<header berkas>
<header bitmap>

<data bitmap>
20 19 21 24 24 23 24

Gambar 2.7 : Format citra 24-*bit* (16,7 juta warna) (Munir, 2004)

2.5 Image Histogram Citra Digital

Image Histogram adalah suatu fungsi yang menyatakan jumlah kemunculan dari setiap nilai (Achmad Basuki, 2005). Jadi misalkan diketahui data

X = 1 3 2 5 3 0 2 1 2 4 2

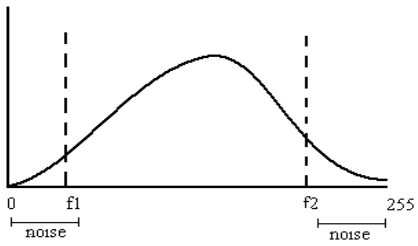
Histogram adalah munculnya setiap nilai, yaitu nilai 0 muncul 1 kali, nila 1 muncul 2 kali, nilai 2 muncul 4 kali dan seterusnya. Karena citra mempunyai 256 derajat keabuan yaitu (0-255) maka histogram menyatakan jumlah kemunculan nilai 0-255.

2.6 Filter Bandpass dan Bandstop

Proses *filtering* adalah salah satu jenis operasi pada pengolahan citra. Salah satu metode yang dipakai untuk menangani *noise* adalah *filter bandpass* dan *filter bandstop*. *Filter* ini berfungsi untuk mengurangi titik-titik *noise* atau membuat citra menjadi tampak lebih bersih, *noise* pada citra *digital* adalah suatu piksel atau kumpulan piksel yang memiliki nilai *digital* acak, letaknya menyebar secara acak dan mengganggu citra atau merusak citra trsebut.

2.6.1 Filter Bandpass

Kata *band* berarti pita, sedangkan *pass* memiliki arti lewat. Sedangkan menurut kamus TI (Devlopment Site for proposed Revisions to American National Standard, TI 523-2001, http://www.its.bldrdoc.gov/projects/devglossary/) band memiliki arti spektrum frekuensi di antara dua batas terdefinisi. Sehingga *filter bandpass* di dalam citra mempunyai arti proses *filter* dilakukan apabila nilai piksel melewati nilai 0 sampai batas bawah (f1) atau batas atas (f2) sampai 255. Sehingga piksel yang berada pada nilai 0 sampai batas bawah (f1) atau batas atas (f2) sampai 255 dianggap sebagai *noise*. Kemudian nilai *digital* dari *noise* tersebut akan diganti dengan melakukan rata-rata nilai *digital* piksel bukan *noise* yang berada di sekitarnya.



Gambar 2.8 : Contoh batas bawah dan atas *filter bandpass*.

Algoritma yang digunakan untuk *filter bandpass* adalah sebagai berikut:

- Langkah 1: *Load* citra yang akan diproses.
- Langkah 2 : Masukan nilai batas bawah dan batas atas untuk nilai R, G dan B.
- Langkah 3 : Lakukan *loop* untuk memproses seluruh citra. Pencarian *noise* di antara 0 sampai nilai batas bawah (f1) atau di antara nilai batas atas (f2) sampai 255. Jika nilai (R, G, dan B) bernilai benar, maka *noise* ditemukan. Selanjutnya hasil bagi nilai *digital* titik atau piksel yang bukan *noise* digunakan untuk menggantikan nilai digital warna yang ber-*noise*.
- Langkah 4 : Tampilkan citra hasil yang telah di proses.

Contoh 1

Berikut adalah contoh dari *filter bandpass* dalam melakukan proses *filtering noise* pada suatu citra. Misalkan terdapat suatu bagian citra *digital bitmap* 8-*bit* berbentuk matriks 5x5 seperti gambar 2.9.

130	134	135	137	138
132	135	136	138	139
133	**25**	137	139	140
134	135	138	140	141
137	154	128	140	142

132	135	136
133	**25**	137
134	135	138

Gambar 2.9 Contoh (1a). Matrik Citra Asli *Filter Bandpass* dan Matrik 3x3 *Filter Bandpass*

130	134	135	137	138
132	135	136	138	139
133	**135**	137	139	140
134	135	138	140	141
137	154	128	140	142

Gambar 2.10 Contoh (1a). Matrik Citra Hasil *Filter Bandpass*

Jika nilai batas bawah yang dimasukan misal 30 dan nilai batas atas yang dimasukan adalah 200. Maka nilai *digital* yang lebih kecil dari 30 atau lebih besar dari 200 dianggap sebagai *noise*. Pada gambar di atas yang dianggap sebagai *noise* adalah piksel dengan nilai 25. Kemudian dibentuk matrik 3x3 berdasarkan piksel-piksel yang mengelilingi piksel yang dianggap *noise*, yang dapat dilihat pada gambar 2.10. Perhitungan dilakukan dengan cara menjumlah nilai *digital* piksel-piksel yang bukan *noise*, dibagi jumlah piksel yang mengelilingi *noise*. Sehingga perhitungannya menjadi :

$$\mathrm{P} = \frac{1}{s}\sum_{n=1}^{s} pn$$

P = nilai *digital* yang menggantikan *noise*

s = jumlah titik yang bukan *noise*

pn = jumlah nilai *digital* yang bukan *noise*

$P = \frac{1}{8}\sum_{1}^{8} pn$

$P = \frac{1}{8} \times (132+135+136+133+137+134+135+138)$

P = 135

Nilai yang dihasilkan adalah 135, kemudian digantikan ke nilai digital piksel yang dianggap sebagai *noise*, sehingga hasilnya dapat dilihat seperti Gambar 2.10 Contoh 1 Matrik Citra Hasil *Filter Bandpass*.

Contoh 2

Sebagai contoh lain, misalkan terdapat suatu citra *digital bitmap* 24-*bit* berbentuk matriks 5x15 seperti gambar 2.11.

R	G	B	R	G	B	R	G	B	R	G	B	R	G	B
130	134	135	137	135	139	138	138	138	139	139	140	142	142	145
133	135	137	138	136	138	139	139	139	140	140	141	141	142	144
20	**10**	**40**	139	137	140	140	140	141	142	142	143	143	143	145
134	135	135	141	137	141	141	141	141	142	143	144	145	145	147
137	138	138	140	128	140	142	142	142	143	143	145	145	146	148

Gambar 2.11 Contoh (2a) Matrik Citra Bitmap 24-bit *Filter Bandpass*.

Nilai batas bawah yang dimasukan misal 30 dan nilai batas atas yang dimasukan adalah 200. Pada matrik citra di atas terdapat *noise* yaitu pada piksel yang di cetak tebal yaitu R=20, G=10, B=40. Karena di

sebelah kiri piksel yang ber-*noise* tidak terdapat piksel yang mengelilingi kemudian akan membentuk matrik 5x6 seperti Gambar 2.12.

R	G	B	R	G	B
130	134	135	137	135	139
133	135	137	138	136	138
20	**10**	**40**	139	137	140
134	135	135	141	137	141
137	138	138	140	128	140

Gambar 2.12 Contoh (2a). Matrik 5x6 *Filter Bandpass*

Kemudian dikelompokan berdasarkan warna R,G,B, sehingga menjadi matrik 2x3 seperti Gambar 2.13.

133	138
20	139
134	141

R

135	136
10	137
135	137

G

135	139
40	140
135	141

B

Gambar 2.13 Contoh 2 Matrik 3x2 *Filter Bandpass*

Perhitungan dapat dilakukan dengan cara menjumlah nilai digital piksel-piksel yang bukan *noise*, dibagi jumlah piksel yang mengelilingi piksel *noise* untuk masing-masing warna R,G dan B.

Perhitungannya untuk warna R sebagai berikut:

$$P = \frac{1}{s}\sum_{n=1}^{s} pn$$

$$P = \frac{1}{5} \times (133 + 138 + 139 + 134 + 141)$$

$$P = \frac{1}{5} \times 685$$

$$P = 137$$

Perhitungannya untuk warna G sebagai berikut

$$P = \frac{1}{s}\sum_{n=1}^{s} pn$$

$$P = \frac{1}{5} \times (135 + 136 + 137 + 135 + 137)$$

$$P = \frac{1}{5} \times 680$$

$$P = 136$$

Perhitungannya untuk warna B sebagai berikut

$$P = \frac{1}{s}\sum_{n=1}^{s} pn$$

$$P = \frac{1}{5} \times (135 + 139 + 140 + 135 + 141)$$

$$P = \frac{1}{5} \times 690$$

$$P = 138$$

Jadi diperoleh warna **R=137**, **G=136**, **B=138**

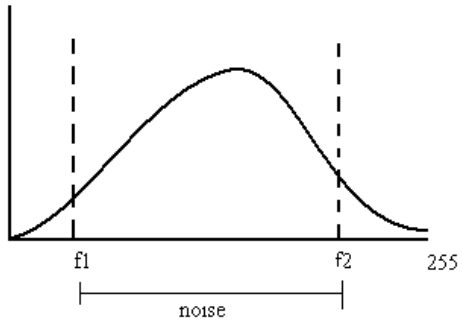
Setelah perhitungan dilakukan, penggantian nilai digital piksel yang ber-*noise* dilakukan, sehingga hasilnya menjdi seperti gambar 2.14.

R	G	B	R	G	B	R	G	B	R	G	B	R	G	B
130	134	135	137	135	139	138	138	138	139	139	140	142	142	145
133	135	137	138	136	138	139	139	139	140	140	141	141	142	144
137	**136**	**138**	139	137	140	140	140	141	142	142	143	143	143	145
134	135	135	141	137	141	141	141	141	142	143	144	145	145	147
137	138	138	140	128	140	142	142	142	143	143	145	145	146	148

Gambar 2.14 Contoh (2a). Matrik Citra Hasil *Filter Bandpass*

2.6.2 Filter Bandstop

Filter bandstop merupakan kebalikan dari *filer bandpass*, yaitu proses filter dilakukan apabila nilai piksel melewati batas bawah sampai nilai batas atas dianggap sebagai *noise*. Untuk penggantian nilai *digital noise* tersebut sama dengan *filter bandpass* yaitu nilai *digital* dari *noise* akan diganti dengan melakukan rata-rata nilai *digital* piksel bukan *noise*. Yang berada di sekitarnya(kumpulan titik yang berbatasan dengan *noise* yang berupa matrik 3x3).



Gambar 2.15 Contoh batas bawah dan batas atas *filter bandstop*

Algoritma yang digunakan untuk *filter bandstop* adalah sebagai berikut:

- Langkah 1: *Load* citra yang akan di proses.
- Langkah 2 : Masukan nilai batas bawah dan batas atas untuk nilai R, G dan B.
- Langkah 3 : Lakukan *loop* untuk memproses seluruh citra. Pencarian *noise* di antara 0 sampai nilai batas bawah atau di antara nilai batas atas sampai 255. Jika nilai (R, G, dan B) bernilai benar, maka *noise* di temukan. Selanjutnya hasil bagi nilai *digital* titik

atau piksel yang bukan *noise* digunakan untuk menggantikan nilai digital warna yang ber-*noise*.

- Langkah 4 : Tampilkan citra hasil yang telah di proses.

  Contoh 3

  Berikut adalah contoh dari *filter bandstop* dalam melakukan proses *filtering noise* pada suatu citra. Misalkan terdapat suatu bagian citra *digital bitmap* 8-bit berbentuk matrik 5x5 dan kemudian dirubah menjadi matrik 3x3 seperti Gambar 2.16

130	134	135	137	138
132	135	136	138	139
133	134	137	**50**	140
134	135	138	140	141
137	154	128	140	142

→

136	138	139
137	**50**	140
138	140	141

Gambar 2.16 Contoh (3a). Matrik Citra Asli dan Matrik 3x3 *Filter Bandstop*.

130	134	135	137	138
132	135	139	138	139
133	134	137	**50**	140
134	135	138	140	141
137	154	128	140	142

Gambar 2.17 contoh (3a). Matrik Citra Hasil *Filter Bandstop*

Nilai batas bawah yang dimasukan misal 30 dan nilai batas atas yang dimasukan adalah 100. Maka nilai *digital* di antara 30 sampai dengan 100 dianggap sebagai *noise*. Pada gambar di atas yang dianggap sebagai

noise adalah piksel dengan nilai 50. Kemudian dibentuk matrik 3x3 berdasarkan piksel-piksel yang mengelilingi piksel yang dianggap *noise* seperti pada gambar 2.17 . Perhitungan dilakukan dengan cara menjumlah nilai *digital* piksel-piksel yang bukan *noise*, dibagi jumlah piksel yang mengelilingi *noise*.

Sehingga perhitungannya menjadi seperti berikut :

$$P = \frac{1}{8}\sum_{1}^{8} pn$$

$$P = \frac{1}{8} \times (139+138+139+137+140+138+140+141)$$

$$P = \frac{1}{8} \times 112$$

$$P = 139$$

Nilai yang dihasilkan adalah **139**, kemudian digantikan ke nilai *digital* piksel yang dianggap sebagai *noise*, sehingga hasilnya dapat dilihat seperti Gambar 2.19 Matrik Citra Hasil *Filter Bandstop*.

Contoh 4

Sebagai contoh berikut adalah contoh dari *filter bandstop* dalam melakukan proses *filtering noise* pada suatu citra. Misalkan terdapat suatu citra *digital bitmap* 24-bit berbentuk matriks 5x15 seperti Gambar 2.18.

R	G	B	R	G	B	R	G	B	R	G	B	R	G	B
130	134	135	133	135	137	138	138	138	139	139	140	142	142	145
60	**65**	**70**	138	136	138	139	139	139	140	140	141	141	142	144
130	138	137	139	137	138	140	140	141	142	142	143	143	143	145
134	135	138	140	138	140	141	141	141	142	143	144	145	145	147
137	138	138	140	128	140	142	142	142	143	143	145	145	146	148

Gambar 2.18 contoh (4a). Matrik Citra Bitmap 24-bit *Filter Bandstop*

Matrik pada gambar 2.20 dapat dianggap sebagai matrik 5x5, karena untuk setiap piksel memiliki 3 buah nilai *digital* untuk warna merah (R), hijau (G), dan biru (B). Sehingga pada gambar tersebut piksel pertama memiliki nilai R=130, G=134, dan B=135. Piksel kedua memiliki nilai R=137, G=135, dan B=137, sedemikian seterusnya.

Misal nilai batas bawah yang dimasukan untuk nilai R=50, G=50, dan B=50. Sedangkan nilai batas atas yang dimasukan untuk R=100, G=100, dan B=100. Maka nilai digital RGB yang pikselnya di antara 50 sampai dengan 100 dianggap sebagai *noise*. Sehingga ditemukan *noise* pada piksel yang dicetak tebal, yaitu R=60, G=65, dan B=70.

Dalam melakukan perhitungan untuk mengganti nilai digital dapat dibentuk matrik 5x6 dan 3x2 seperti gambar berikut karena piksel yang ber-*noise* tidak memiliki piksel yang mengelilinginya di sebelah kiri seperti pada gambar 2.19.

R	G	B	R	G	B
130	134	135	133	135	137
60	**65**	**70**	138	136	138
130	138	137	139	137	138
134	135	138	140	138	140
137	138	138	140	128	140

Gambar 2.19 contoh (4a). Matrik 5x6 Yang Memiliki *Noise*

Kemudian dikelompokan berdasarkan warna R,G,B, sehingga menjadi matrik 3x2 seperti pada gambar 2.20.

130	133
60	138
130	139

R

134	135
65	136
138	137

G

135	137
70	138
137	138

B

Gambar 2.20 contoh (4a). Matrik 3x2 *Filter Bandstop*

Perhitungan dapat dilakukan dengan cara menjumlah nilai digital piksel-piksel yang bukan *noise*, dibagi jumlah piksel yang mengelilingi piksel *noise* untuk masing-masing warna R,G dan B.

Perhitungannya untuk warna R sebagai berikut:

$$P = \frac{1}{s}\sum_{n=1}^{s} pn$$

$$P = \frac{1}{5} \times (130 + 133 + 138 + 130 + 149)$$

$$P = \frac{1}{5} \times 670$$

$$P = 134$$

Perhitungannya untuk warna G sebagai berikut

$$P = \frac{1}{s}\sum_{n=1}^{s} pn$$

$$P = \frac{1}{5} \times (135 + 135 + 136 + 138 + 137)$$

$$P = \frac{1}{5} \times 680$$

$$P = 136$$

Perhitungannya untuk warna B sebagai berikut

$$P = \frac{1}{s}\sum_{n=1}^{s} pn$$

$$P = \frac{1}{5} \times (135 + 137 + 138 + 137 + 138)$$

$$P = \frac{1}{5} \times 685$$

P = 137

Jadi diperoleh warna **R=134**, **G=136**, **B=137**

Setelah perhitungan dilakukan, penggantian nilai digital piksel yang ber-*noise* dilakukan, sehingga hasilnya menjdi seperti gambar 2.21.

R	G	B	R	G	B	R	G	B	R	G	B	R	G	B
130	134	135	137	135	137	138	138	138	139	139	140	142	142	145
154	**136**	**137**	138	136	138	139	139	139	140	140	141	141	142	144
133	135	137	139	137	139	140	140	141	142	142	143	143	143	145
134	135	138	140	138	140	141	141	141	142	143	144	145	145	147
137	138	138	140	128	140	142	142	142	143	143	145	145	146	148

Gambar 2.21 contoh (4a) Matrik Citra Hasil *Filter Bandstop*

2.7 Pemrograman Borland Delphi 7

Delphi berasal dari bahasa pemrograman yang cukup terkenal, yaitu bahasa *pascal*. Bahasa *pascal* diciptakan pada tahun 1971 oleh ilmuwan dari swiss, yaitu Niklaus Wirth. Nama *pascal* diambil dari ahli matematika dan filsafat Perancis, yaitu *Blasie Pascal* (1623-1622).

Karena pemrograman *windows* dengan *turbo pascal* masih dirasa cukup sulit, maka sejak tahun 1993 *Borland International* mengembangkan bahasa *pascal* yang bersifat visual. Hasil dari pengembangan ini adalah dirilisnya Delphi 1 pada tahun 1995. Perkembangan Delphi tidak berhenti sampai di situ. Pada tahun berikutnya 1996, *Borland International* merilis Delphi 2 untuk *windows* 95/NT. Dan kemudian dalam tahun-tahun berikutnya, *Borland International* merilis beberapa versi pengembangan Delphi yang memiliki tambahan fitur baru dibandingkan dengan versi sebelumnya.

2.7.1 Bagian Utama Borland Delphi 7

Pada dasarnya IDE Delphi dibagi menjadi tujuh bagian utama yaitu *Menu*, *Speedbar*, *Component Palette*, *Form Designer*, *Code Explorer*, *Object Tree View* dan *Object Inspector*.

2.7.1.1 Menu

Menu pada Delphi memiliki kegunaan seperti pada aplikasi windows lainnya. Dari menu ini, bisa digunakan untuk memanggil atau menyimpan program, menjalankan program, dan lain sebagainya. Sesuatu yang berhubungan dengan IDE Delphi dapat dilakukan dari menu.

2.7.1.2 *Spee dbar*

Speed bar atau sering juga disebut *toolbar* berisi kumpulan tombol yang tidak lain adalah pengganti dari beberapa item menu yang sering digunakan. Dengan kata lain, setiap tombol pada speedbar menggantikan salah satu item menu.

2.7.1.3 *Component Palette*

Component palette berisi kumpulan icon yang melambangkan komponen-komponen pada VCL (*Visual Component Library*) atau CLX (*Component Library for Cross Platform*). Pda komponen palette terdapatbeberapa tab yaitu : *standard*, *additional*, *data access*, dan tab yang lainnya. *Icon* yang ditampilkan pada *Component Palette* tidak memiliki keterangan yangmenyatakan nama komponen yang dapat dilihat pada gambar 2.24.

Gambar 2.22 : Component Palette Delphi

2.7.1.4 Form Designer

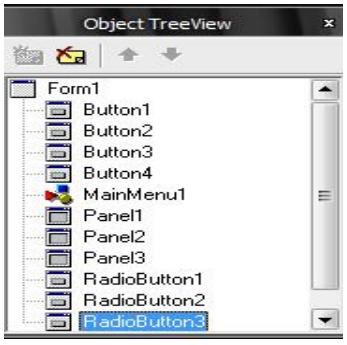
Sesuai dengan namanya, *form designer* merupakan tempat untuk merancang jendela aplikasi. Perancangan form dilakukan dengan meletakan komponen-komponen yang diambil dari *component palette*.

2.7.1.5 Code Explorer

Code explorer adalah tempat dimana akan menuliskan program. Disini diletakan pernyataan-pernyataan dalam bahasa *pascal*. Yang perlu diperhatikan dalam code explorer adalah tidak perlu menuliskan semua kode sumber. IDE Delphi telah menuliskan semacam kerangka program.

2.7.1.6 Object Treeview

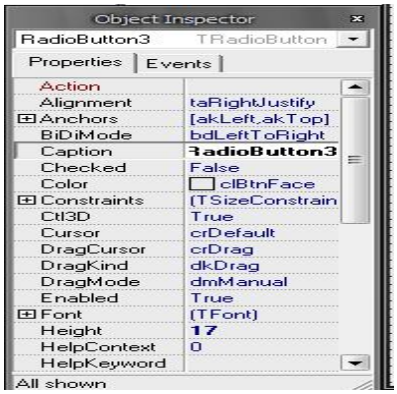
Object treeview berisi daftar komponen yang telah diletakan pada form designer.



Gambar 2.23 : *Object Treeview Delphi*

2.7.1.7 Object Inspector

Object Inspector digunakan untuk mengubah karakteristik sebuah komponen, pada *object inspector* dapat dilihat terdapat dua tag yaitu properties dan events. *User* dapat mengaktifkan salah satu tag dengan mengklik properties atau events. Pada tag properties digunakan untuk mengubah properti dari komponen. Pada tag events digunakan untuk menyisipkan kode untuk menangani kejadian tertentu.



Gambar 2.24 : *Object Inspector Delphi*

2.7.2 Variabel

Dalam dunia pemrograman, variabel digunakan untuk menyimpan data. Pada Delphi, pendeklarasian variabel mengikuti sintak di bawah ini

Var nama_variabel1:tipe_variabel;

Variabel pada Delphi harus mengikuti beberapa aturan sebagai berikut :

- Nama variabel maksimum terdiri dari 63 karakter.
- Nama variabel hanya boleh mengandung huruf, angka garis bawah (_) dan tidak boleh diawali dengan angka.

- Tidak bisa menggunakan kata kunci milik Delphi. Sebagai contoh variabel dengan nama *if, else for* tidak diperbolehkan.