BAB II
LANDASAN TEORI

Bab ini akan menjelaskan teori – teori yang dipakai sebagai acuan dalam tugas akhir ini. Berikut adalah penjelasan mengenai teori – teori tersebut.

2.1. Citra Digital

Dalam kehidupan sehari-hari, disadari atau tidak, citra memegang peranan yang cukup penting. Data atau informasi tidak hanya disajikan dalam bentuk teks, tetapi juga dapat berupa gambar, audio (bunyi, suara, musik), dan video. Keempat macam data atau informasi ini sering disebut multimedia. Era teknologi informasi saat ini tidak dapat dipisahkan dari multimedia. Situs web (*website*) di Internet dibuat semenarik mungkin dengan menyertakan visualisasi berupa gambar atau video.

Citra atau istilah lainnya adalah gambar sebagai salah satu komponen multimedia memegang peranan sangat penting sebagai bentuk informasi visual. Citra terbagi atas dua yaitu citra diam dan citra bergerak. Citra diam (*still images*) adalah citra tunggal yang tidak bergerak. Sedangkan citra bergerak (*moving images*) adalah rangkaian citra diam yang ditampilkan secara beruntun sehingga memberi kesan pada mata sebagai gambar yang bergerak.

Citra digital merupakan citra yang diambil berdasarkan *sampling* dan kuantisasi tertentu sehingga citra digital ini terbentuk dari piksel – piksel yang besarnya tergantung pada besar kecilnya sampling dan nilainya (besarnya derajat keabuan) tergantung pada kuantisasi. Dengan kata lain, dapat dinyatakan bahwa citra digital adalah citra yang didefinisikan sebagai fungsi f(x,y) dimana x menyatakan nomor baris, y menyatakan nilai kolom, dan f menyatakan nilai derajat keabuan dari citra. Dengan demikian (x,y) adalah posisi dari piksel dan f adalah nilai derajat keabuan pada titik (x,y). Pemrosesan citra dengan komputer digital salah satunya membutuhkan citra digital sebagai masukannya. Citra digital dapat dianggap sebagai suatu matriks, dimana indeks baris dan kolomnya menyatakan koordinat sebuah titik

pada citra tersebut. Dan nilai masing-masing elemennya menyatakan intensitas cahaya pada titik tersebut. Suatu titik pada sebuah citra digital sering disebut sebagai "*picture elemen*" atau "*piksel*". Semakin kecil ukuran piksel yang mewakili suatu citra, maka makin tinggi resolusinya dan makin halus gambarnya.

2.1.1 Sampling dan Kuantisasi

Supaya citra dapat direpresentasikan secara numerik, maka sebuah citra harus didigitalisasi, baik terhadap ruang (koordinat(x,y)) maupun terhadap derajat keabuannya (f(x,y)). Proses digitalisasi koordinat (x,y) dikenal sebagai pencuplikan citra (*image sampling*), sedangkan proses digitalisasi derajat keabuan f(x,y) disebut sebagai "Kuantisasi Derajat Keabuan" *(graylevel quantization*).

Citra digital dapat dinyatakan dalam matriks dua dimensi f(x,y) dimana (x,y) merupakan koordinat piksel dalam matriks dan f merupakan derajat intensitas piksel tersebut. Citra digital berbentuk matriks dengan ukuran M x N akan tersusun sebagai berikut:

$$F(x,y) = \begin{bmatrix} f(1,1) & f(1,2) & f(1,3) & \dots & (f(1,N-1) \\ f(1,1) & f(1,2) & f(1,3) & \dots & (f(1,N-1) \\ f(1,1) & f(1,2) & f(1,3) & \dots & (f(1,N-1) \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ f(M-1,1) & f(M-1,2) & f(M-1,3) & \dots & (f(M-1,N-1) \end{bmatrix}$$

Gambar 2.1 Fungsi Citra f(x,y) Berukuran m x n

2.1.2 Sistem RGB (Red Green Blue)

Gambar standar (bitmap) dengan ukuran 512 x 512 dan mempunyai kedalaman warna 24 bit atau dikenal dengan istilah true color, dalam hal ini

kedalaman warna yang dimaksud yaitu jumlah bit yang dipakai untuk merepresentasikan tiap titik pada gambar tersebut dinyatakan dalam bit per piksel.

Komputer mempunyai 3 komponen warna merah, komponen warna hijau, dan komponen warna biru. Ketiga komponen warna dasar ini dikenal dengan istilah warna RGB (*Red Green Blue*).

2.1.3 Konversi RGB ke Grayscale (Y)

Konversi warna RGB kedalam warna *luminance* (Y) atau dikenal dengan istilah *grayscale*, yaitu gambar dengan derajat keabuan yang mempunyai intensitas warna 0 sampai 255, dimana 0 adalah untuk merepresentasikan warna hitam dan 255 adalah merepresentasikan warna putih. Untuk mengubah citra berwarna yang mempunyai nilai matriks masing – masing r, g, dan b menjadi citra *grayscale* dengan nilai Y, maka konversi dapat dilakukan dengan mengambil rata – rata dari nilai r, g, b sehingga dapat dituliskan sebagai berikut [5] :

$$Y = \frac{r + g + b}{3}$$

2.2 Pengolahan Citra

Pengolahan citra (*image processing*) memegang peranan yang cukup penting dalam memproses suatu gambar sehingga menghasilkan gambar lain yang lebih sesuai dengan yang diharapkan. Dalam kehidupan sehari – hari pengolahan citra digunakan dalam bidang fotografi dan perfilman, bidang kedokteran, bidang teknologi dan komunikasi dan dalam dunia game.

Meskipun sebuah citra kaya informasi, namun seringkali citra yang dimiliki mengalami penurunan mutu (degradasi), misalnya mengandung cacat atau derau (*noise*), warnanya terlalu kontras, kurang tajam, kabur, dan sebagainya. Agar citra yang mengalami gangguan mudah diinterpretasi (baik oleh manusia maupun mesin),

maka citra tersebut perlu dimanipulasi menjadi citra lain yang kualitasnya lebih baik. Inilah yang disebut dengan pengolahan citra.

Pengolahan citra adalah pemrosesan citra, khususnya dengan menggunakan komputer, menjadi citra yang kualitasnya lebih baik. Pengolahan citra mempunyai dua tujuan utama, yakni sebagai berikut:

1. Memperbaiki kualitas citra, dimana citra yang dihasilkan dapat menampilkan informasi secara jelas atau dengan kata lain manusia dapat melihat informasi yang diharapkan dengan menginterpretasikan citra yang ada. Dalam hal ini interpretasi terhadap informasi yang ada tetap dilakukan oleh manusia. [5]
2. Mengekstraksi informasi ciri yang menonjol pada suatu citra, dimana hasilnya adalah informasi citra dimana manusia mendapatkan informasi ciri dari citra secara numerik atau dengan kata lain komputer (mesin) melakukan interpretasi terhadap informasi yang ada pada citra melalui besaran-besaran data yang dapat dibedakan secara jelas (besaran-besaran ini berupa besaran numerik). [5]

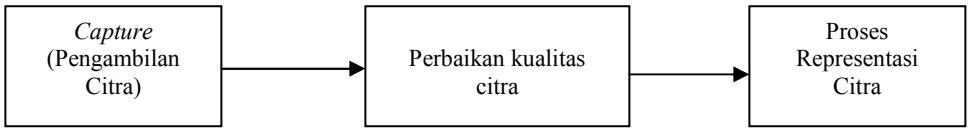
2.2.1 Dasar-Dasar Pengolahan Citra Digital

Ada beberapa hal yang penting didalam pengolahan citra digital, antara lain teknik-teknik pengambilan citra, model citra digital, sampling dan kuantisasi, *threshold*, *histogram*, proses *filtering*, perbaikan citra sampai pada pengolahan citra digital yang lebih lanjut seperti segmentasi *dan image clustering*. [5]

Ada perbedaan yang sangat mendasar mengenai citra digital dan citra analog. Citra digital merupakan representatif dari citra yang diambil oleh mesin dengan bentuk pendekatan berdasarkan *sampling* dan kuantisasi. *Sampling* menyatakan besarnya kotak – kotak yang disusun dalam baris dan kolom. Dengan kata lain *sampling* pada citra menyatakan besar kecilnya ukuran piksel atau titik pada citra, dan kuantisasi menyatakan besarnya nilai tingkat kecerahan yang dinyatakan dalam nilai tingkat keabuan (*grayscale*) sesuai dengan jumlah bit biner yang digunakan oleh

mesin dengan kata lain kuantisasi pada citra menyatakan jumlah warna yang ada pada citra.

Proses pengolahan citra secara diagram proses dimulai dari pengambilan citra, perbaikan kualitas citra, sampai dengan pernyataan representatif citra dicitrakan dengan gambar dibawah ini.



Gambar 2.2 Proses Pengolahan Citra

2.2.2 Teknik – Teknik Pengambilan Citra Digital

Ada beberapa teknik pengambilan digital yang bisa dilakukan, antara lain dengan menggunakan kamera digital, Web-cam, atau menggunakan scanner (bisa *scanner* umum, *finger print scanner*, atau *bar-code scanne)r*. Hasil dari citra dengan menggunakan teknik kamera atau *scanner* berupa citra *raster* (pola bintik – bintik pengganti warna). Teknik pengambilan citra selain membutuhkan peralatan input, juga dibutuhkan suatu interface yang disebut dengan *frame grabber* yang berupa rangkaian untuk mengolah citra secara hardware.

Selain itu pengambilan citra dapat dilakukan dengan menggunakan teknik grafika komputer, yaitu dengan membentuk objek citra komputer sesuai dengan teknik mengambil citra. Teknik ini banyak digunakan untuk keperluan *entertainment* seperti pembuatan animasi.

2.2.3 Teknik Pengolahan Citra Digital

Secara umum teknik pengolahan citra digital dibagi menjadi 3 tingkat pengolahan, yakni sebagai berikut:

1. Tahap pertama yang dinamakan dengan *low level processing* (pengolahan tingkat rendah). Pengolahan ini operasional-operasional dasar dalam pengolahan citra, seperti pengurangan *noise*, perbaikan citradan restorasi citra.
2. Tahap kedua dinamakan dengan *Mid-Level Processing* (Pengolahan tingkat menengah). Pengolahan ini meliputi segmentasi pada citra, deskripsi objek, dan klasifikasi objek secara terpisah.
3. Tahap ketiga yang dinamakan dengan *High-Level Processing* (pengolahan tingkat tinggi), yang meliputi analisis citra.

Dari ketiga tahap pengolahan citra diatas, dapat dinyatakan suatu gambaran mengenai teknik-teknik pengolahan citra digital dan macam-macamnya. Kelompok ini disusun berdasarkan urutan yang pada umumnya dilakukan pada proses pengolahan citra:

- Perbaikan citra (*Image restoration*)
- Peningkatan kualitas citra (*Image enhancement*)
- Pemampatan data citra (*Image data compaction*)
- Analisa citra (*Image analysis*)
- Segmentasi citra (*image segmentation*)
- Rekonstruksi citra (*Image reconstruction*)

2.2.3.1 Image Restoration

Image restoration atau perbaikan citra berhubungan dengan minimalisasi atau penghilangan degradasi tertentu yang terdapat dalam citra sehingga didapatkan kembali citra aslinya. Degradsi ini dapat diakibatkan oleh lingkungan penginderaan citra, misalnya derau yang diakibatkan sensor citra, buram *(blur*) akibat kamera yang tidak fokus, keadaan atmosfir atau pencahayaan ketika citra ditangkap, dan sebagainya.

2.2.3.2 Image Enhancement

Berbeda dengan proses perbaikan citra, proses *image enhancement* atau peningkatan kualitas citra hanya didasarkan pada prosedur-prosedur yang sangat *heuristic* (eksperimental, *trial* and *error*). Tujuannya adalah untuk memperjelas atau mempertajam ciri-ciri tertentu dari citra untuk suatu pengguna yang spesifik. Istilah spesifik disini penting artinya, karena teknik-teknik yang dilakukan sangat berbeda antara keperluan yang satu dengan yang lainnya. Proses ini tidak meningkatkan informasi yang terkandung dalam data citra digital, melainkan meningkatkan daerah cakupan data ciri yang diinginkan agar lebih mudah dideteksi.

2.2.3.3 Image Data Compaction

Karena besarnya jumlah data informasi visual didalam suatu citra digital, maka sistem pengolahannya juga membutuhkan kemampuan penyimpanan yang besar. Misalnya, citra yang dihasilkan televisi bisa mengandung data mencapai 10 mega byte per detiknya. Hal ini menimbulkan masalah, baik dalam proses penyimpanan (tergantung kapasitas memori yang tersedia) maupun dalam proses pengiriman data dari suatu tempat ketempat yang lain. Oleh karena itu dibutuhkan suatu cara untuk mereduksi jumlah bit yang dibutuhkan untuk menyimpan atau mentransmisi citra dengan syarat informasi yang penting tidak hilang. Untuk mengurangi jumlah data atau memanfaatkan suatu citra, maka citra asli ditransformasikan dahulu kedalam bidang lain. Proses ini bisa disebut dengan pengkodean citra (*image encoding*). Proses ini relatif tidak akan mengurangi informasi citra aslinya. Citra hasil transformasi tersebut kemudian dikodekan kembali.

2.2.3.4 Image Segmentation

Image Segmentation atau segmentasi citra memiliki tujuan untuk memecah suatu citra ke dalam beberapa segmen sehingga diperoleh citra sesuai dengan yang diinginkan. Jenis proses ini berkaitan erat dengan pengenalan pola

2.2.3.5 Image Analysis

Image Analysis atau analisa citra berhubungan dengan proses penghitungan kuantitatif dari sebuah citra untuk mendapatkan deskripsi tentang citra tersebut. Sebelum melakukan analisis pada suatu citra, ciri-ciri khusus (*features*) dari citra diekstrak terlebih dahulu. Kemudian objek yang menjadi perhatian khusus dalam citra diisolasi dengan menggunakan teknik segmentasi. Setelah itu, dilakukan perhitungan kuantitatif terhadap ciri khusus objek tersebut untuk mendapatkan deskripsinya atau untuk kebutuhan klasifikasi.

2.2.3.6 Image Reconstruction

Image Reconstruction atau rekonstruksi citra adalah kasus khusus dari restorasi citra, dimana objek berdimensi dua atau lebih akan direkonstruksi beberapa proyeksi berdimensi satu. Contoh aplikasi proses ini adalah pada dunia medis yaitu foto sinar-x, dimana citra diperoleh dari proyeksi sinar-x melalui suatu objek. Dari foto-foto sinar-x tersebut dapat dibentuk objek tiga dimensi dengan proses rekonstruksi.

2.3 Cropping

Cropping merupakan salah satu cara yang digunakan dalam pengenalan pola untuk bisa mengenali pola yang ada. Sebagai contoh, dalam pengenalan pola wajah *cropping* telah digunakan untuk bisa mengenali pola wajah seseorang dengan menggunakan metode Semantik Wajah. Salah satu komponen yang ada dalam *cropping* adalah komponen hbp.dll.

Kegunaan dari komponen hbp.dll ini adalah untuk membuat citra terdiri dari beberapa bagian – bagian. Sebagai contoh, dalam pengenalan pola wajah, komponen hbp.dll ini dapat digunakan untuk membagi citra menjadi bagian – bagian tertentu sehingga dapat mengambil bagian – bagian tertentu dari wajah sesuai dengan yang diinginkan. Dalam komponen hbp.dll ini juga ada komponen – komponen lain yang menjadi pendukung bagi komponen hbp.dll.

Adapun komponen – komponen yang terdapat dalam hbp.dll adalah sebagai berikut:

RChbp_update	*RChbp_getNormalisedFaceImage*
RChbp_updateImage	*RChbp_getPersonIdentity*
RChbp_update2	*RChbp_getPersonID*
RChbp_setDetectionMode	*RChbp_getPersonName*
RChbp_getImage	*RChbp_getPortrait*
RChbp_getImagePixel	*RChbp_enableLearning*
RChbp_getBodyPosture	*RChbp_faceDetected*
RChbp_getFaceImage	*RChbp_save*
	RChbp_load

2.4 Ekstraksi ciri

Pada dasarnya, teknik – teknik ekstraksi yang ada terbagi dalam metode statistik, spektral, dan struktural. Metode statistik menggunakan perhitungan statistik distribusi derajat keabuan (histogram) dengan mengukur tingkat kekontrasan, granularitas, dan kekasaran suatu daerah dari hubungan ketetanggaan antar piksel di dalam citra. Paradigma statistik ini penggunaannya tidak terbatas, sehingga sesuai untuk tekstur – tekstur alami yang tidak terstruktur dari sub pola sub pola dan himpunan aturan (mikrostruktur). Metode spektral berdasarkan pada fungsi autokorelasi suatu daerah atau *power distribution* pada dominan trasformasi Fourier dalam mendeteksi periodisitas tekstur. Metode struktural menggunakan deskripsi primitif tekstur dan aturan sintak dalam melakukan analisa. Biasanya metode ini banyak digunakan untuk pola – pola makrostruktur.

Cara ekstraksi ciri dalam mendeskripsikan informasi pada tekstur mengikuti cara manusia dalam menginterpretasi ciri secara visual. Dengan menentukan nilai piksel citra sebagai primitif dan jumlah gabungan spasial antara tekstur primitif sehingga menghasilkan suatu orde analisa tekstur.

2.5 Pengenalan pola

Pola adalah entitas yang terdefinisi dan dapat diidentifikasi melalui ciri-cirinya. Ciri-ciri tersebut digunakan untuk membedakan suatu pola dengan pola lainnya. Ciri yang bagus adalah ciri yang memiliki daya pembeda yang tinggi, sehingga pengelompokkan pola berdasarkan ciri yang dimiliki dapat dilakukan dengan keakuratan yang tinggi.

Pengenalan pola bertujuan untuk mengenali suatu objek dengan mengenali ekstraksi informasi penting yang terdapat pada suatu citra, sehingga proses klasifikasi citra dapat dilakukan. Dari sejumlah citra diolah sehinggga citra dengan ciri yang sama akan dikelompokkan pada suatu kelompok tertentu. Berikut ini merupakan diagram blok dasar dari sebuah sistem pengenalan pola:



Gambar 2.5 Diagram Pengenalan Pola

2.6 Biometrik

Teknologi Biometrik adalah sistem yang menggunakan bagian tubuh manusia untuk kepastian pengenalan. Teknologi ini menggunakan bagian tubuh manusia yang unik dan tetap seperti sidik jari, mata dan wajah seseorang.

Sampai saat ini, teknologi yang sering digunakan adalah sidik jari, sementara pengenalan iris mata atau titik khusus wajah belum banyak diaplikasikan karena alasan faktor ekonomis. Penggunaan sidik jari paling sering digunakan untuk umum, karena lebih komersial, dan mudah untuk digunakan semua orang dari sistem kepastian pengenalan menggunakan wajah ataupun mata seseorang. Pada saat menggunakan fasilitas pengenalan sidik jari, harus dipastikan terlebih dahulu jari-jarinya tidak kotor, karena alat pencocokan sidik jari tidak akan mengenalinya.

Keamanan biometrik telah sangat meningkat, dengan adanya pengetatan sistem keamanan negara-negara, terutama setelah peristiwa 11 September di Amerika. Salah satu bidang yang mengalami peningkatan adalah sistem pengenalan, seperti misalnya teknologi pencitraan wajah tiga dimensi. Dengan menggunakan sistem komputer untuk memetakan titik-titik unik wajah, teknologi ini satu generasi lebih maju dibandingkan teknologi pengenalan yang sudah kita kenal saat ini, yaitu yang menggunakan sidik jari dan pengenalan pola selaput pelangi mata atau iris.

Berdasarkan sifat – sifatnya, biometrik terbagi atas 2 yaitu biometrik dengan ciri khas yang berubah terhadap waktu seperti pengenalan wajah dan ciri khas yang tidak berubah oleh waktu seperti pengenalan sidik jari, retina, tanda tangan.

2.6.1 Pengenalan wajah

Pengenalan bentuk-bentuk dan posisi dari ciri-ciri wajah seseorang adalah tugas yang kompleks. Pertama sebuah kamera menangkap gambar dari sebuah wajah dan kemudian *software* memilah-milah pola informasi yang selanjutnya dibandingkan dengan *template user.*

Pengaplikasian ini dipengaruhi oleh gangguan cahaya dan kesulitan-kesulitan yang ditemukan saat wajah membentuk sudut tertentu dari kamera. Faktor-faktor ini menyebabkan rendahnya kemampuan pengenalan (kegagalan pengenalan). Untuk memecahkan masalah tersebut perusahaan computer membangun 3D *face acquisition system*. Secara umum berbicara mengenai dua cara untuk menentukan pengenalan, didasari oleh *feature* wajah dan pendekatan area wajah.

Pada saat ini penelitian tentang pengenalan wajah sedang terus dikembangkan salah satunya adalah *Nevenvision. Nevenvision* merupakan salah satu pengemban software pengenalan wajah dari Amerika yang telah sukses mengimplementasikan sistem pengenalan biometric yang bersifat *mobile*. Aplikasi tersebut telah diterapkan pada *Los Angeles Police Department (LAPD)*. Melalui media *Pocket PC,* Telepon selular, PDA, dan lain-lain. Sistem aplikasi ini sangat membantu kepolisian dalam berbagai hal, contohnya dalam hal kriminalitas, pencarian orang hilang, pengamanan

bandara. Dengan diterapkannya aplikasi ini dikepolisian, diharapkan dapat menekan angka kriminalitas dimasyarakat.

2.6.2 Pengenalan sidik jari

Sistem ini meliputi sebuah perangkat keras *scanner* dan perangkat lunak. Merekam karakteristik sidik jari yang spesifik, menyimpan data tiap-tiap *user* ke dalam sebuah *template*, ketika *user* mencoba lagi menguatkan akses maka perangkat lunak akan membandingkan data yang tersimpan pada *template* dengan pembacaan sidik jari dari *scanner*.

Sistem sidik jari sangat akurat tetapi dapat dipengaruhi oleh perubahan-perubahan di dalam sidik jari (terbakar, bekas luka dan lain sebagainya), kotoran dan faktor-faktor lain yang menimbulkan gangguan pada gambar. Beberapa jenis aplikasi yang dapat diterapkan pada teknologi sidik jari adalah: *Police Investment System, Door Access System, Card Access System, Software Access System, Banking System,* dan sistem keamanan yang lain.

2.6.3 Pengenalan bagian mata.

Pengenalan pola bagian mata adalah yang termasuk rumit karena keanekaragaman dari karakteristik yang unik dari tiap-tiap orang. Sebuah sistem pengenalan iris menggunakan sebuah kamera video untuk menangkap contoh dan perangkat lunak membandingkan data hasil dengan *template-template* yang disimpan.

2.6.4 Pengenalan retina.

Gambar retina sulit untuk ditangkap dan selama pendataan user harus memusatkan sebuah titik serta mempertahankannya sehingga kamera dapat melaksanakan penangkapan gambar dengan baik. Hal yang sebenarnya ditentukan pola dari pembuluh-pembuluh darah. Tetapi ketika pola-pola ini pada tiap-tiap orang, identifikasi dapat menjadi lebih presisi. Sistem yang didasarkan pada dua bagian

mata, iris dan retini adalah dipertimbangkan untuk menawarkan tingkat keamanan terbaik.

2.7 Eigen Value (Nilai Eigen)

Pada dasarnya matriks merupakan alat yang dapat memecahkan sistem persamaan linier yang banyak ditemui diberbagai bidang seperti bidang teknik, ekonomi, ilmu fisika, dan lain – lain. Akan tetapi pada kenyataannya hanya matriks bujursangkar saja yang berperan penting, sehingga diperlukan pengertian tentang nilai – nilai eigen atau nilai – nilai karakteristik dari matriks tersebut.

Jika A adalah suatu matriks bujursangkar sebarang berukuran atau berorde n x n, maka vektor tak nol x dalam R^n dinamakan vektor eigen dari A, jika A.x adalah kelipatan skalar dari x yaitu:

$$\mathbf{A\ x = \lambda\ x}$$

Untuk skalar λ, skalar λ tersebut dinamakan *eigen value* atau nilai karakteristik dari matriks A dan x dikatakan vektor eigen yang bersesuaian dengan λ.

Jadi pada dasarnya masalah *eigen value* adalah masalah untuk menentukan semua *eigen value* λ dari sebuah matriks bujursangkar agar persamaan $\mathbf{A\ x = \lambda\ x}$ dipenuhi.

Salah satu cara untuk menentukan *eigen value* dari suatu matriks adalah dengan mencari akar – akar dari persamaan eigennya. Untuk mencari *eigen value* suatu matriks A orde n x n, maka kita tulis kembali persamaan A x = λ x sebagai :

$$\mathbf{A\ x = \lambda\ I\ x}$$

Atau secara ekivalen:

$$[\lambda I - A]\, x = 0$$

dimana I adalah suatu matriks Identitas ordo n.

Supaya λ dapat menjadi *eigen value*, maka harus ada pemecahan tak nol dari persamaan tersebut. Persamaan tersebut akan mempunyai pemecahan tak nol jika dan hanya jika

$$\det\ [\lambda I - A] = 0$$

Persamaan tersebut dinamakan persamaan eigen atau persamaan karakteristik dan skalar yang memenuhi persamaan ini adalah *eigen value* dari matriks A. *Eigen Value* suatu matriks bujursangkar A sebarang berordo n x n, dimana $|\lambda I - A|$ dapat ditentukan :

$$\begin{bmatrix} \lambda & 0 & \cdots & 0 \\ 0 & \lambda & & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \lambda \end{bmatrix} - \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} \lambda - a_{11} & -a_{12} & \cdots & -a_{1n} \\ -a_{21} & \lambda - a_{22} & \cdots & -a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -a_{n1} & -a_{n2} & \cdots & \lambda - a_{nn} \end{bmatrix}$$

Apabila persamaan eigen tersebut dikembangkan lagi maka akan menghasilkan sebuah suku banyak dalam λ yang dinamakan suku banyak atau polinom eigen dari matriks A.

Contoh:

Carilah *eigen value* (nilai karakteristik) dari matriks :

$$A = \begin{bmatrix} 6 & -4 & 0 \\ -4 & 6 & 0 \\ 0 & 0 & 10 \end{bmatrix}$$

Penyelesaian:

Cari matriks [λ I - A] yaitu :

$$[\lambda I - A] = \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 6 & -4 & 0 \\ -4 & 6 & 0 \\ 0 & 0 & 10 \end{bmatrix}$$

$$= \begin{bmatrix} \lambda - 6 & 4 & 0 \\ 4 & \lambda - 6 & 0 \\ 0 & 0 & \lambda - 10 \end{bmatrix}$$

Maka suku banyak eigennya adalah :

$$\det [\lambda I - A] = \det \begin{bmatrix} \lambda - 6 & 4 & 0 \\ 4 & \lambda - 6 & 0 \\ 0 & 0 & \lambda - 10 \end{bmatrix}$$

$$= (\lambda - 2)\,(\lambda - 10)^2$$

dan persamaan eigennya adalah :

$$(\lambda - 2)\ (\lambda - 10)^2 = 0$$

Jadi *eigen value* dari matriks A tersebut adalah $\lambda_1 = 2$, $\lambda_2 = 10$ dan $\lambda_3 = 10$

Dari contoh diatas, jelas terlihat bahwa *eigen value* suatu matriks dapat dihitung dengan cara memecahkan persamaan eigennya untuk mendapatkan suku banyak eigennya kemudian cari akar – akar yang memenuhinya yang juga merupakan *eigen value* matriks tersebut.

Cara tersebut diatas akan menjadi tidak efektif lagi bila akan mencari *eigen value* dari matriks – matriks yang berukuran atau berordo besar, karena akan menghasilkan suatu perhitungan yang rumit dan cukup memakan waktu, dan apabila menggunakan komputer sebagai alat bantu maka akan dihadapkan pada kesulitan dalam memformulasikan permasalahannya.

Oleh karena itu selain cara diatas, para ahli matematika juga mengembangkan metode dan algoritma yang lain guna menentukan *eigen value* suatu matriks yang lebih bersifat pendekatan numerik. Hal tersebut didasarkan pada pertimbangan agar terhindar dari perhitungan yang rumit tadi, dan lebih mudah memformulasikan permasalahan dan penyelesaian masalah *eigen value* apabila dikerjakan dengan menggunakan komputer sebagai alat bantu. Adapun metode dan algoritma yang dimaksud adalah Metode QR Baku

2.7.1 Maximum Eigen Value

Maximum eigen value adalah nilai maximum dari suatu *eigen value* yang diperoleh dari hasil penyelesaian suatu matriks. Dari contoh diatas diperoleh *eigen value* dari suatu matriks A adalah $\lambda_1 = 2$, $\lambda_2 = 10$ dan $\lambda_3 = 10$. Maka yang disebut sebagai *maximum eigen value* adalah nilai λ yang terbesar yaitu $\lambda_2 = \lambda_3 = 10$.

Eigen value ini dapat digunakan untuk mengenali pola wajah seseorang. Setiap orang memiliki struktur wajah yang berbeda. Inilah yang kemudian menjadi penyebab pola wajah dapat digunakan untuk mengenali seseorang. Salah satu metode pengenalan pola wajah adalah dengan *eigen value*. *Eigen value* yang dimiliki setiap orang digunakan untuk mengenali pola wajah orang tersebut.

2.7.2 Matriks Segitiga Atas

Suatu matriks dikatakan matriks segitiga atas adalah suatu matriks bujursangkar yang seluruh unsur-unsur di bawah dibawah diagonal utama matriks tersebut bernilai nol, atau dengan kata lain, $a_{ij} = 0$, apabila i>j.
Sebagai contoh:

$$A = \begin{bmatrix} 2 & 8 & 3 & 6 & 7 \\ 0 & -3 & 8 & 1 & 2 \\ 0 & 0 & 1 & 3 & 7 \\ 0 & 0 & 0 & 4 & 5 \\ 0 & 0 & 0 & 0 & -2 \end{bmatrix}$$

2.7.3 Matriks Tridiagonal

Suatu matriks dikatakan matriks tridiagonal apabila matriks tersebut berbentuk bujursangkar dan unsur-unsur $a_{ij} = 0$, bilamana $|i - j|>1$.
Sebagai contoh:

$$A = \begin{bmatrix} 2 & 8 & 0 & 0 & 0 \\ 3 & -3 & 8 & 0 & 0 \\ 0 & 1 & 1 & 3 & 0 \\ 0 & 0 & 7 & 4 & 5 \\ 0 & 0 & 0 & 3 & -2 \end{bmatrix}$$

2.7. 4 Matriks Similar

Suatu matriks bujursangkar A dan B dikatakan similar apabila ada matriks dapat dibalik P sehingga $B = P^{-1} A P$.

2.7.5 Matriks Ortogonal

Definisi

Dua buah vector x dan y dikatakan saling orthogonal apabila memenuhi $x^T.y=0$ dan $y^T.x=0$.

Contoh :

$x^T = [\ 2\ \ 4\ \ 4\ \ 8\ ]$

$y^T = [\ 12\ \ 6\ \ -4\ \ -4\ ]$, maka

$$x^T . y = [\ 2\ \ 4\ \ 4\ \ 8\ ] \begin{bmatrix} 12 \\ 6 \\ -4 \\ -4 \end{bmatrix} = 24 + 24 - 16 - 32 = 0$$

$$y^T . x = [\ 12\ \ 6\ \ -4\ \ -4\ ] \begin{bmatrix} 2 \\ 4 \\ 4 \\ 8 \end{bmatrix} = 24 + 24 - 16 - 32 = 0$$

Definisi

Suatu matriks bujur sangkar A dikatakan matriks orthogonal apabila memenuhi persamaan : $A.A^T = A^T.A = I$

Dari defenisi tersebut dapat disimpulkan bahwa suatu matriks orthogonal A berarti :

1. A adalah suatu matriks bujur sangkar yang mempunyai baris-baris serta kolom-kolomnya saling ortonormal.
2. Jika A adalah suatu matriks orthogonal maka A^T juga adalah matriks ortogonal
3. Karena $A.A^T = A^T.A = I$, maka ini berarti $A^T = A^{-1}$

2.7.6 Transformasi Householder

Transformasi Householder adalah suatu metode yang dapat digunakan untuk mereduksi suatu bentuk matriks, apabila matriks simetri ke bentuk matriks tridiagonal dan apabila bukan matriks simetri ke bentuk matriks Hessenberg atas. Pada Bab ini hanya akan membahas mengenai matriks tridiagonal saja.

Definisi

Matriks Transformasi Householder atau matriks Householder adalah suatu matriks yang berbentuk : $H = I - 2F.F^T$

Dengan $F^T = [\ 0, \ldots 0, F_{r+1}, \ldots, F_n\ ]$

Dimana F adalah vector kolom dengan norm Euclides Kesatuan

$| F |_2 = F^T.F = 1$

Teorema

Jika H adalah suatu matriks householder, maka dipenuhi sifat sebagai berikut :

1. H adalah matriks simetri atau $H^T = H$.
2. H adalah matriks orthogonal atau $H^T.H = H.\ H^T = I$
3. H involuntary atau $H^{-1} = H$

Bukti :

1. H adalah matriks simetri atau $H^T = H$

   $H^T = (I-2F.F^T)^T$

   $= I-2(F^T)^T.F^T$

$= I-2F.F^T$

$= H$

Jadi terbukti bahwa matriks haouseholder adalah matriks simetri.

2. H adalah matriks orthogonal atau $H^T.H = H.H^T = I$

Sebelumnya telah dibuktikan bahwa $H^T=H$, maka :

$H^T.H = H.H^T = H.H$

$= (I-2F.F^T).(I-2F.F^T)$

$= I-4I.F.F^T + 4\ (F.F^T)2$

$= I-4F.F^T + 4\ (\ F.F^T)(F.F^T)$

$= I-4F.F^T + 4F.I.F^T$

2.7.6 Metode QR Baku

Misalkan suatu matriks real sebarang A, dan setelah ditransformasi Householder disebut A_1, akan dicari *eigen valuenya* dengan menggunakan algoritma QR baku, maka :

Dekomposisi QR kan matriks A, tersebut ke dalam bentuk :

$$A_1= Q_l\ ,\ R_l$$

di mana Q adalah sua.tu maxriks ortogonal, dan

R adalah suatu matriks segitiga atas.

Kemudian bentuk matriks A_2 dengan membalik urutan perkalian matriks A_1 yaitu

$$A_2 = R_1\ .\ Q_1$$

dari persamaan $A_1= Q_l\ ,\ R_l$ dapat pula ditulis :

$$R_1 = Q_1^T\ .\ A_1$$

dan dengan substitusi pada persamaan $A_2 = R_1 \, . \, Q_1$ mengakibatkan :

$$A_2 = Q_1{}^T \, . \, A_1 \, . \, Q_1$$

dan itu berarti bahwa matriks A_1 dan A_2 adalah similar.

Kemudian matriks A_2 tersebut dapat didekomposisikan kembali ke dalam bentuk :

$$A_2 = Q_2 \, . \, R_2$$

sehingga dari persamaan - persamaan tersebut di atas tadi terbentuklah algoritma sebagai berikut .

$$A_k = Q_k \, . \, R_k$$

$$A_{k+1} = R_k \bullet Q_k$$

$$= Q_{k+1} \, . \, R_{k+1} \text{ , untuk } k = 1, 2, 3, \ldots, n$$

di mana Q_k adalah matriks ortogonal

R_k adalah matriks segitiga atas

Kemudian, karena

$$A_{k+1} = R_k \bullet Q_k = Q_k{}^T \, . \, A_k . \, Q_k$$

Lebih jauh diperoleh :

$$A_{k+1} = Q_k{}^T \bullet A_k \bullet Q$$

$$= Q_k{}^T \bullet Q_{k-1}{}^T A_{k-1} \bullet Q_{k-1} \bullet Q_k$$

$$= (Q_k{}^T \ldots Q_1{}^T) . \, A_1 . \, (Q_1 \ldots Q_1)$$

Untuk mencari matriks similar dari A_1 hingga ke A_k, di mana A_k nantinya merupakan suatu matriks segitiga atas, maka lakukan algoritma QR baku yang langkah-langkahnya sebagai berikut :

Langkah kerja algoritma QR baku adalah :

1. Reduksikan matriks simetri, misalkan matriks T menggunakan transformasi Householder ke dalam bentuk matriks tridiagonal. Misalkan matriks T yang telah direduksi tadi adalah matriks A_k, k = 1.
2. Lakukan dekomposisi QR untuk matriks A_k, untuk k = 1, 2, 3, . . . , n dengan menggunakan serangkaian rotasi bidang untuk mengeliminasi unsur-unsur di bawah diagonal utamanya, sehingga nantinya memenuhi persamaan :

$$A_k = Q_k . R_k$$

3. Dari bentuk tersebut diatas kita bentuk matriks similar A_{k+1} dengan cara :

$$A_{k+1} = Q_k{}^T \bullet A_k \bullet Q_k$$

4. Kemudian kontrol unsur ke (n,n-1) dari matriks A_{k+1} apakah mendekati nol dengan toleransi yang telah ditentukan (berapa angka di belakang koma).

   Misalkan toleransi tersebut adalah ε, jika :

   - $\left|a_{n,n-1}^{(k+1)}\right| > \varepsilon$, maka diulangi lagi langkah 2 sampai 4, dimana $A_{k+1} = A_k$ dan dihitung sebagai satu iterasi.
   - $\left|a_{n,n-1}^{(k+1)}\right| > \varepsilon$, Konvergen ke nol, maka dilanjutkan langkah selanjutnya, di mana unsur a,,,. adalah nilai eigen dari matriks T.

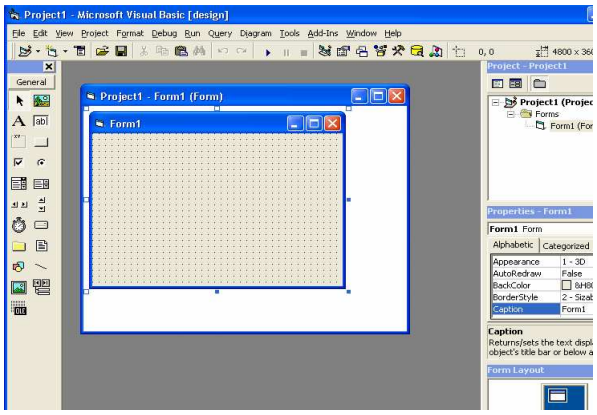
5. Deflasikan matriks A_{k+1} menjadi matriks A_1 berukuran (n - 1) x (n - 1) dengan menghilangkan kolom dan baris ke - n, dan ini dihitung sebagai satu langkah.
6. Ulangi langkah 2 sampai 5 hingga diperoleh semua *eigen value* yang dicari.

2.8 Teori Dasar Bahasa Pemrograman Visual Basic

Visual Basic sangat popular pada tahun – tahun terakhir ini. *Visual Basic* dikenal untuk *Application*, *VBScript* dan versi lengkap VB. Tidak seperti bahasa pemrograman yang lain, seperti Pascal misalnya, dimana harus menuliskan kode program untuk segala sesuatunya, *Visual Basic* mampu menambahkan sendiri sebagian kode program secara otomatis ke dalam program sehingga memudahkan penggunanya. *Visual Basic* mempunyai banyak sarana untuk membangun program aplikasi berbasis windows dengan cepat dan efisien. *Visual Basic* versi 6.0 dapat digunakan untuk membuat program aplikasi yang sederhana maupun yang kompleks, database dan DHTML.

Seperti program berbasis windows lainnya, *Visual basic* terdiri dari banyak windows. Ketika akan memulai *Visual basic* sekumpulan windows yang paling berkaitan inilah yang disebut dengan *Integrated Development Environtment* (IDE). Program yang berbasis windows bersifat *event driven*, artinya program bekerja berdasarkan *event* dalam objek tersebut, misalnya jika seorang *user* mengklik sebuah tombol maka program akan memberikan 'reaksi' terhadap *event* klik tersebut. Program akan memberikan reaksi sesuai dengan kode – kode program yang dibuat untuk suatu *event* pada objek tertentu.

Pada waktu memulai *Visual basic*, beberapa windows kecil berada disebuah windows besar (windows induk), bentuk inilah yang dikenal dengan format *Multiple Document Interface* (MDI). Berikut ini adalah contoh tampilan IDE pada sebuah *project Visual basic* dengan sebuah *form* dan sebuah *command button*.

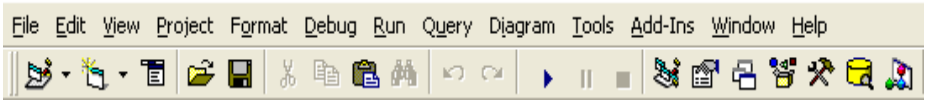


Gambar 2.4 Tampilan IDE Visual Basic

Windows – windows Visual basic :

1. Menu/Toolbar

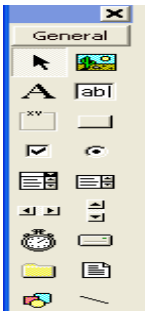
Windows ini adalah satu – satunya elemen dari IDE yang selalu tampak berguna untuk memilih ataupun menambah elemen IDE lain, menambah *form* dan menambah kontrol pada suatu *project.(* Gambar 2.5)



Gambar 2.5 Menu/Toolbar

2. Toolbox

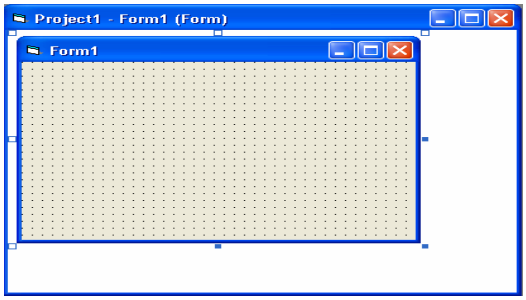
Toolbox adalah sekumpulan kontrol yang dapat diletakkan pada form program aplikasi yang dibuat. *Toolbox* dapat disembunyikan untuk memberikan ruangan bagi elemen IDE lainnya, sehingga lebih mempermudah proses desain maupun penulisan programnya. **(**Gambar 2.6)



Gambar 2.6 Toolbox

3. Form

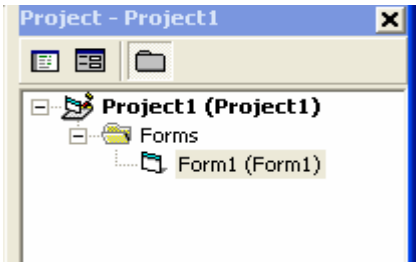
Form adalah sebuah atau beberapa windows untuk pembuatan program aplikasi. *Form* ini dapat memuat berbagai macam kontrol (tobol – tombol, maupun teks) yang diperlukan dalam desain program yang sesuai dengan kebutuhan program. (Gambar 2.7)



Gambar 2.7 Form

4. Project Windows

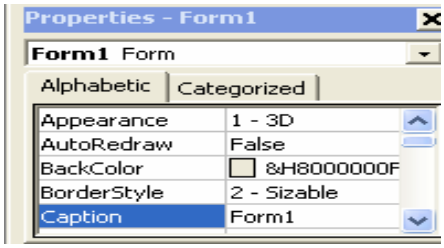
Windows ini menampilkan seluruh *form*, *class, class module*, dan komponen lain yang ada pada sebuah *project*.(Gambar 2.8)



Gambar 2.8 Project Windows

5. Property Windows

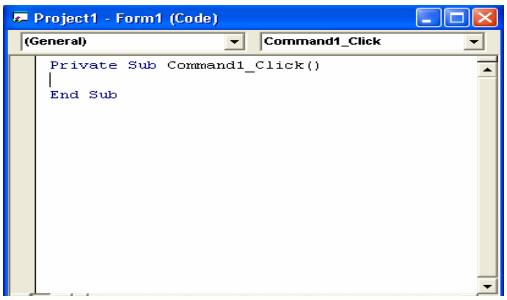
Windows ini berisi seluruh *property* dari masing – masing objek pada sebuah *project* yang meliputi *property form* dan kontrol – kontrol yang ada pada *form* tersebut. Beberapa *property* dapat diisikan pada tahap desain dan ada pula *property* yang harus diisikan dengan menuliskan kode selama program dijalankan.(Gambar 2.9)



Gambar 2.9 Property Windows

6. Code Windows

Pada windows inilah semua kode/perintah tentang program dituliskan dengan memperhatikan event apa saja yang diperlukan (Gambar 2.10)



Gambar 2.10 Code Windows