Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – CI1599

GENERATOR WATERMARK YANG UNIK BERDASARKAN NOMOR DOKUMEN

WACHID ASARI
NRP 5104 100 108

Dosen Pembimbing
Ir. Suhadi Lili
Chastine Fatichah, S. Kom, M. Kom

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2009

TUGAS AKHIR – CI1599

GENERATOR WATERMARK YANG UNIK BERDASARKAN NOMOR DOKUMEN

WACHID ASARI
NRP 5104 100 108

Dosen Pembimbing
Ir. Suhadi Lili
Chastine Fatichah, S. Kom, M. Kom

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2009

[Halaman ini sengaja dikosongkan]

FINAL PROJECT – CI1599

UNIQUE WATERMARK GENERATOR BASED ON DOCUMENT NUMBER

WACHID ASARI
NRP 5104 100 108

Supervisor
Ir. Suhadi Lili
Chastine Fatichah, S. Kom, M. Kom

INFORMATICS DEPARTMENT
Faculty of Information Technology
Sepuluh Nopember Institute of Technology
Surabaya 2009

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

GENERATOR WATERMARK YANG UNIK BERDASARKAN NOMOR DOKUMEN

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Sistem Bisnis Cerdas
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :
WACHID ASARI
NRP : 5104 100 108

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Ir. Suhadi Lili .....
NIP: 132048148 (pembimbing 1)

Chastine Fatichah, S. Kom, M. Kom .....
NIP: 132298829 (pembimbing 2)

SURABAYA
JANUARI 2009

[Halaman ini sengaja dikosongkan]

GENERATOR WATERMARK YANG UNIK BERDASARKAN NOMOR DOKUMEN

Nama Mahasiswa	: WACHID ASARI
NRP	: 5104 100 108
Jurusan	: Teknik Informatika FTIf-ITS
Dosen Pembimbing I	: Ir. Suhadi Lili
Dosen Pembimbing II	: Chastine Fatichah, S. Kom, M. Kom

Abstrak

Pada saat ini, perkembangan teknologi semakin memudahkan manusia untuk memperoleh informasi dalam format dokumen digital. Kemudahan akses terhadap dokumen digital memberikan dampak yang cukup besar terhadap tindakan pemalsuan dokumen. Oleh karena itu, watermark dapat digunakan sebagai salah satu metode untuk melindungi kepemilikan hak cipta atas kekayaan intelektual yang tersedia dalam format dokumen digital.

Pembahasan tugas akhir ini menyajikan suatu metode watermark pada dokumen digital yang memanfaatkan persamaan parametrik untuk menghasilkan pola-pola watermark yang unik berdasarkan nomor dokumen. Persamaan parametrik yang digunakan berdasarkan pada fungsi-fungsi trigonometri dan konsep dari persamaan kurva Lissajous. Disamping itu, nomor dokumen yang merupakan ciri khas dari suatu dokumen digital akan menjadi nilai parameter pada persamaan yang mempengaruhi pola watermark yang terbentuk.

Penggunaan konsep dasar kurva Lissajous dengan modifikasi fungsi-fungsi parametrik trigonometri dapat menghasilkan pola gambar watermark yang unik. Sedangkan penentuan nilai batasan yang sesuai akan menghasilkan pola watermark yang memenuhi aspek estetika.

Kata kunci: *Watermark*, Kurva *Lissajous*, Persamaan Parametrik, ASP.NET, Devexpress, Xtrareport

[Halaman ini sengaja dikosongkan]

UNIQUE WATERMARK GENERATOR

BASED ON DOCUMENT NUMBER

Name : WACHID ASARI
NRP : 5104 100 108
Major : Informatics Engineering IT Department – ITS
Supervisor I : Ir. Suhadi Lili
Supervisor II : Chastine Fatichah, S. Kom, M. Kom

Abstract

Nowadays, the development of technology more facilitate people to obtain information in digital format documents. Ease of access to digital documents give a large impact for an action in document forgery. Therefore, the watermark can be used as a method to protect the copyright ownership of the intellectual property that is available in the format of digital documents.

This final project presents a method of digital watermark on documents that utilize parametric equation to generate a pattern that is unique watermark based on the documents number. Parametric Equation used based on trigonometric functions and the concept of Lissajous curve equation. In addition, the document number that are the characteristic of a digital document will be the parameter value on the equation that affect the form of watermark pattern.

The result of using Lissajous curve concept with the modification of parametric equation and trigonometry is unique watermark pattern. While determining the proper constrain value to the parametric function will produce a watermark pattern that meet an aesthetic aspect.

Keyword: Watermark, Lissajous Curve, Parametric Equation, ASP.NET, Devexpress, Xtrareport

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Segala puji dan syukur, kehadirat Tuhan Yang Maha Esa yang telah memberikan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul:

GENERATOR WATERMARK YANG UNIK BERDASARKAN NOMOR DOKUMEN

Teriring rasa syukur pada Sang Keindahan Yang Diberkati atas segala karunia yang menyelimuti kehidupan penulis. Dengan ini penulis juga ingin menyampaikan rasa terima kasih pada pihak-pihak yang telah mendukung penyelesaian tugas akhir ini:

- Istri tercinta yang telah menjadi inspirasi dalam kehidupan penulis.
- Pihak Keluarga: Bapak, Ibu, dan Adik-adik tercinta yang telah mengisi hari-hari kehidupan penulis.
- Pihak Keluarga dari Kediri: Bapak, Ibu dan Dini yang telah memberikan semangat untuk penyelesaian tugas akhir ini.
- Bapak Ir. Suhadi Lili dan Ibu Chastine Fatichah selaku dosen pembimbing yang telah memberikan bimbingan, petunjuk dan ilmu untuk menyelesaikan tugas akhir ini.
- Bapak Yudhi Purwananto, M. Kom selaku Ketua Jurusan Teknik Informatika ITS.
- Bapak Dr. Agus Zainal Arifin, S. Kom, M. Kom selaku dosen wali penulis.
- Rekan-rekan di Laboratorium Pemrograman Teknik Informatika ITS, terutama para admin angkatan tahun 2002, 2003, 2004, dan 2005.

- ❖ Mas Rully yang telah memberikan kesempatan untuk fokus mengerjakan tugas akhir ini.
- ❖ Rekan-rekan di Laboratorium Komputing dan AJK terutama angkatan tahun 2004.
- ❖ Rekan-rekan di kantor Bratang dan P2KB UNESA atas semua dukungan dan semangat yang telah diberikan untuk menyelesaikan tugas akhir ini.
- ❖ Rekan-rekan di kantor SIMPEG yang telah mewarnai proses pengerjaan tugas akhir ini.

Penulis mengharapkan adanya saran dan kritik yang membangun, sehingga hasil dari tugas akhir ini dapat menjadi sebuah solusi yang bermanfaat bagi masyarakat.

Surabaya, Januari 2009

Penulis

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar Tugas Akhir yang meliputi Latar Belakang, Tujuan dan Manfaat Pembuatan, Rumusan dan Batasan Permasalahan, Metodologi Pembuatan Tugas Akhir, dan Sistematika Penulisan.

1.1 Latar Belakang

Perkembangan teknologi informasi dewasa ini semakin memudahkan manusia memperoleh informasi dari berbagai media. Salah satu media informasi yang sangat populer dalam beberapa tahun terakhir ini adalah media elektronik, salah satunya adalah teknologi internet yang diakses melalui suatu komputer. teknologi internet berperan sebagai salah satu media dan sumber segala informasi yang cukup lengkap. Kecanggihan teknologi tersebut memungkinkan pertukaran informasi antara satu pihak dengan pihak lainnya bisa terjadi dalam waktu yang cepat dengan biaya yang relatif cukup murah walaupun dibatasi oleh jarak yang cukup jauh. Hal ini tentunya selaras dengan kebutuhan manusia terhadap informasi yang semakin kompleks.

Salah satu dampak perkembangan teknologi dan kebutuhan manusia terhadap informasi adalah semakin maraknya pemanfaatan dokumen digital dalam kehidupan sehari-hari. Terobosan baru dalam konsep dokumen digital memungkinkan penyimpanan informasi dalam berbagai macam bentuk, baik itu berupa artikel yang tersedia secara *online*, *ebook*, dokumen presentasi (*powerpoint*), dokumen makalah (pdf), dokumen gambar (*image*), audio, maupun video. Beberapa bentuk dokumen digital tersebut tentunya sudah tidak asing bagi sebagian besar pengguna komputer diseluruh dunia, karena merupakan suatu

tetapan standar untuk suatu dokumen digital yang telah disediakan oleh sistem operasi komputer. Berikut ini adalah logo (*icon*) beberapa format dokumen digital sesuai dengan Gambar 1.1.

Gambar 1.1 Logo Beberapa Format Dokumen Digital

Berikut ini adalah beberapa faktor yang menjadi pertimbangan manusia untuk lebih menggunakan dokumen digital, yakni:

- Memudahkan proses pencarian informasi dari suatu dokumen saat dibutuhkan di masa mendatang.
- Mengurangi resiko kehilangan / kerusakan dokumen.
- Kemudahan akses terhadap dokumen, penggunaan yang fleksibel, dan kemudahan distribusi dokumen.
- Penghematan biaya dibandingkan dengan proses pembuatan dokumen *hardcopy*.
- Penggunaan dokumen digital memungkinkan adanya suatu *sharing* dokumen, sehingga bisa diakses oleh beberapa orang pada saat yang sama.

Faktor-faktor tersebut diatas telah menjadikan dokumen digital sebagai suatu solusi yang cukup brilian untuk menghadapi masa

depan yang semakin menuntut adanya kemudahan dan kecepatan penyebaran suatu informasi. Adapun dampak negatif yang diakibatkan dari penggunaan dokumen digital ini adalah sifatnya yang dapat di *copy* persis sama dengan aslinya, oleh karena itu proses pemalsuan dokumen digital dapat dengan mudah dilakukan. Tentunya akan cukup sulit untuk menentukan keaslian suatu dokumen digital yang dihasilkan dari tindakan pemalsuan, terlebih lagi jika dokumen tersebut telah beredar luas di masyarakat.

Salah satu teknik dalam proses verifikasi keaslian suatu dokumen digital dikenal dengan istilah *watermark*. Metode *watermark* bertujuan untuk melindungi hak cipta atas kekayaan intelektual yang tersedia pada format dokumen digital. Proses *watermarking* pada suatu dokumen digital akan memasukkan sejumlah nilai-nilai bit (*binary*) maupun suatu bentuk yang berpola pada dokumen digital, sehingga bisa dilakukan identifikasi terhadap kepemilikan dokumen digital tersebut.

Proses identifikasi *watermark* pada umumnya menggunakan suatu alat atau mesin yang dirancang khusus untuk membaca adanya informasi tentang kepemilikan dan keaslian dokumen digital. Sehingga hasil proses pemeriksaan tersebut diharapkan memiliki tingkat keakuratan yang tinggi.

Secara umum konsep *watermark* ini bisa dianggap sebagai bentuk *background* atau logo pada suatu dokumen *hardcopy*, lazimnya berupa tanda khusus yang mewakili lambang dari suatu unit organisasi tertentu. Namun, untuk penerapan dan implementasi *watermark* pada dokumen digital tentunya terdapat perbedaan metode dan proses yang dilakukan, agar pola *watermark* yang terbentuk mampu memberikan bukti keaslian pada dokumen digital tersebut.

Berikut ini adalah contoh dokumen dengan format *image* yang diberi logo *watermark* sesuai Gambar 1.2.

Gambar 1.2 Contoh Dokumen dengan *Watermark*

Tugas akhir ini akan membahas solusi dari permasalahan keaslian suatu dokumen digital dengan menggunakan teknik *watermarking* yang memanfaatkan nomor dokumen sebagai input nilai pada fungsi parametrik untuk menghasilkan gambar *watermark* yang unik namun tetap memenuhi aspek estetika.

1.2 Rumusan Permasalahan

Berdasarkan penjelasan tentang latar belakang di atas, penulis kemudian merumuskan beberapa detil permasalahan yang akan dibahas pada Tugas Akhir ini, yakni:

1. Bagaimana mendapatkan persamaan-persamaan parametrik yang menghasilkan gambar atau bentuk *watermark* yang indah dengan variasi yang cukup.
2. Bagaimana menentukan variasi parameter untuk mengubah bentuk dan warna gambar *watermark*.
3. Bagaimana melakukan proses transformasi nomor dokumen menjadi nilai-nilai parameter.
4. Bagaimana proses implementasi generator gambar *watermark* diatas kanvas *Xtrareport* berdasarkan *framework* Devexpress pada ASP.NET 2008.

1.3 Batasan Permasalahan

Batasan dalam Tugas Akhir ini adalah sebagai berikut:

1. Tugas Akhir ini hanya membahas tentang proses pembuatan *watermark* pada dokumen digital dengan memanfaatkan fungsi-fungsi parametrik dan trigonometri.
2. Fungsi-fungsi parametrik yang digunakan mengacu pada konsep kurva *Lissajous*.
3. *Watermark* yang dihasilkan pada dokumen digital bersifat dapat dilihat secara langsung karena digunakan sebagai *background* pada dokumen yang akan dicetak.
4. Aplikasi ini dibangun diatas platform ASP.NET dengan menggunakan *framework* Devexpress.

5. Implementasi metode *watermark* ini merupakan suatu *Class Component* yang dirancang untuk dapat diintegrasikan diatas kanvas Xtrareport.
6. *Interface Drawing* pada *Class Component* yang dibuat ini akan mengacu pada interface yang dibawa oleh Xtrareport, yakni: IGraphics yang sedikit berbeda dari *Interface Drawing* pada platform ASP.NET.
7. *Watermark* yang digambar pada kanvas Xtrareport nantinya dapat diubah ke format lain sesuai dengan pilihan format dokumen digital yang disediakan oleh DevExpress.

1.4 Tujuan dan Manfaat

Tujuan Tugas Akhir ini adalah untuk mendapatkan metode dalam pembuatan gambar *watermark* yang unik berdasarkan nomor dokumen dengan tetap memenuhi aspek estetika dalam menjamin keaslian suatu dokumen digital.

Tugas Akhir ini diharapkan mampu memberikan gambaran yang mencukupi untuk mengetahui proses implementasi metode *watermark* yang dibangun diatas *framework* Devexpress pada *platform* ASP.NET.

Adapun manfaat dari aplikasi yang dihasilkan dari implementasi metode *watermark* ini adalah untuk proses verifikasi keaslian dokumen digital.

1.5 Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

A. Studi literatur

Mencari dan mempelajari berbagai macam literatur yang berkaitan dengan rumusan masalah, teori-teori yang berhubungan dengan system dan metode *watermark* yang akan dibangun, desain sistem, struktur framework serta bahasa pemrogramannya.

B. Perencanaan pembangunan perangkat lunak

Mengadakan studi tentang metode *watermark* yang akan dikembangkan serta menganalisa bagaimana implementasi metode ini akan dibangun, infrastruktur yang diperlukan dan hal-hal yang lain yang berkaitan.

C. Perancangan perangkat aplikasi

Perancangan perangkat lunak meliputi 2 aktivitas, yakni antara lain sbb:

a) Spesifikasi kebutuhan perangkat lunak, tahap ini merupakan tahap pemetaan dari problem domain perangkat lunak yang akan dikembangkan.
b) Deskripsi Pengembangan perangkat lunak, tahap ini merupakan tahap pemetaan dari *solution* domain perangkat lunak yang akan dikembangkan.

D. Pembuatan (implementasi) perangkat lunak

Pada tahap ini dilakukan proses analisa dan desain untuk implementasi konsep ke dalam sebuah program. Terdapat beberapa poin penting yang akan diimplementasikan dalam sistem, yakni:

- Program dapat menerima input nomor dokumen sehingga bisa dilakukan proses pemilahan nilai-nilai parameter.
- Program dapat menentukan pola yang sesuai dengan karakteristik nomor dokumen.
- Program dapat memberikan output gambar *watermark* yang unik dengan memperhatikan unsur estetika.
- Program dapat terintegrasi dalam framework DevExpress.

E. Uji coba dan Evaluasi

Pada tahap ini, aplikasi sudah selesai dibuat dan siap digunakan untuk diuji kebenarannya berdasarkan tujuan pembuatan program tersebut.

F. Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan penyusunan buku sebagai laporan dan dokumentasi dari perangkat lunak secara keseluruhan, mulai dari tahap awal hingga tahap akhir pembuatan Tugas Akhir.

1.6 Sistematika Penulisan

Buku tugas akhir ini terdiri dari beberapa bab, yang dijelaskan sebagai berikut:

BAB I. PENDAHULUAN

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan tugas akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan tugas akhir.

BAB II. TINJAUAN PUSTAKA

Bab ini membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan Tugas Akhir ini, yakni: konsep *watermark*, proses perhitungan persamaan parametrik, konsep menggambar dan integrasi gambar *watermark* pada kanvas di lingkungan *framework* DevExpress.

BAB III. METODOLOGI

Bab ini membahas desain dari sistem yang akan dibuat meliputi: desain algoritma, arsitektur, proses, antarmuka perangkat lunak, dan implementasi dari desain sistem yang dilakukan pada tahap desain.

BAB IV. UJI COBA DAN EVALUASI

Bab ini membahas uji coba dari aplikasi yang dibuat dengan melihat output yang dihasilkan oleh aplikasi, dan evaluasi untuk mengetahui kemampuan aplikasi.

BAB V. PENUTUP

Bab ini berisi kesimpulan dari hasil uji coba yang dilakukan serta saran untuk pengembangan aplikasi selanjutnya.

BAB II
TINJAUAN PUSTAKA

Pada bab ini akan dibahas mengenai tinjauan pustaka yang menjadi dasar dari pembuatan Tugas Akhir. Pokok-pokok permasalahan yang dibahas diantaranya mengenai konsep *watermark*, konsep fungsi-fungsi persamaan trigonometri, persamaan parametrik serta kurva *Lissajous*, konsep grafika vektor, konsep raster vektor, dan konsep *drawing* pada kanvas di *platform* .NET dengan *framework* Devexpress Xtrareport.

2.1 Watermark

Perkembangan teknologi informasi dewasa ini semakin memudahkan manusia memperoleh informasi dari berbagai media. Salah satu media informasi yang sangat populer dalam beberapa tahun terakhir ini adalah media elektronik, yang menawarkan terobosan baru dalam konsep dokumen digital, baik itu berupa artikel yang tersedia secara online, ebook, audio, maupun video. Dari sudut pandang pengguna (*end user*) keberadaan media informasi elektronik tentunya sangat bermanfaat dan cukup banyak menghemat waktu dan biaya. Di sisi lain, sifat dokumen digital dapat di *copy* persis sama dengan aslinya, karenanya proses pemalsuan dokumen digital bisa dengan mudah dilakukan sehingga cukup sulit untuk menentukan keaslian dari suatu dokumen digital yang sudah beredar luas di masyarakat.

Salah satu teknik dalam proses verifikasi keaslian suatu dokumen digital dikenal dengan istilah *watermark*. Pada media informasi dokumen digital, *watermark* bertujuan untuk melindungi hak cipta atas kekayaan intelektual yang tersedia dalam format digital. Proses *watermarking* pada suatu dokumen digital akan

memasukkan sejumlah nilai-nilai bit (*binary*) yang berpola pada dokumen digital, sehingga bisa dilakukan identifikasi terhadap kepemilikan dokumen digital tersebut.

2.1.1 Konsep Watermark

Dalam sebuah buku, Juergen Seitz [Seitz 2004] memberikan definisi tentang *digital watermarking*, yakni:

"Digital watermarking" means embedding information into digital material in such a way that it is imperceptible to a human observer but easily detected by computer algorithm. A digital watermark is a transparent, invisible information pattern that is inserted into a suitable component of the data source by using a specific computer algorithm. Digital watermarks are signals added to digital data (audio, video, or still images) that can be detected or extracted later to make an assertion about the data.

Adapun penjelasan lain tentang *watermark* oleh Goldstein dan David [Goldstein 2002], yakni:

A digital watermark is a pattern of binary digits inserted into the artefact that provides information about copyright. It is not visible and must be robust enough to continue to exist if the digital artefact is changed in some way.

Pernyataan lainnya yang mendukung tentang definisi *watermark* diberikan oleh Devid Andriyano [Andriyano 2002], yakni:

Watermark merupakan tanda yang diberikan pada sebuah data. seperti suara, gambar, atau video. Tanda ini ada pada data namun kcbcradaannya tidak dapat dirasakan oleh indera manusia, serta tidak menambah ukuran data. Tujuan utama

penggunaan watermark adalah untuk menunjukkan kepemilikan sebuah data secara sah, dengan dcmikian watermark harus dapat merepresentasikan infomasi tcrtentu.

Berdasarkan beberapa definisi tersebut maka metode *watermark* adalah suatu cara penyembunyian atau penanaman data maupun informasi tertentu (baik hanya berupa catatan umum maupun bentuk dokumen rahasia) kedalam suatu data digital lainnya, tetapi tidak diketahui kehadirannya oleh indera manusia (indera penglihatan atau indera pendengaran), dan mampu menghadapi proses-proses pengolahan sinyal digital sampai pada tahap tertentu. Sehingga mampu menyediakan informasi yang sah tentang kepemilikan dokumen digital.

Dalam kehidupan nyata, penggunaan konsep *watermark* pada dokumen digital berbeda dengan *watermark* yang terdapat pada uang kertas. *Watermark* pada uang kertas masih dapat dilihat dengan jelas oleh mata telanjang manusia, tetapi *watermark* pada media digital dimaksudkan agar tidak dapat dirasakan kehadirannya oleh manusia, sehingga dibutuhkan suatu alat bantu mesin pengolah digital seperti komputer, dan sejenisnya untuk dapat mengetahui adanya suatu *watermark* pada dokumen digital. Konsep *watermark* pada dokumen digital memanfaatkan kekurangan-kekurangan sistem indera manusia seperti mata dan telinga. Dengan adanya kekurangan inilah, metode *watermark* dapat diterapkan pada berbagai media digital..

Implementasi metode *watermark* pada dokumen digital diharapkan dapat mengurangi tindakan pemalsuan yang semakin marak di masyarakat. Tentunya diperlukan metode yang handal agar *watermark* yang dibentuk memiliki kualitas terhadap segala macam bentuk manipulasi dan pemalsuan.

2.1.2 Pemanfaatan Watermark

Watermark dapat dimanfaatkan untuk berbagai tujuan, seperti :

- ***Tamper proofing***, yakni penggunaan *watermark* sebagai alat untuk identifikasi atau alat indikator yang menunjukkan data digital telah mengalami perubahan dari aslinya.
- ***Feature location***, yakni menggunakan metode *watermark* sebagai alat untuk identifikasi isi dari data digital pada lokasi-lokasi tertentu, misalnya penamaan objek tertentu dari beberapa objek yang lain pada suatu citra digital.
- ***Annotation / caption***, yakni penggunaan *watermark* sebagai keterangan tentang data digital itu sendiri.
- ***Copyright Labeling***, yakni penggunaan *watermark* sebagai metode untuk penyembunyikan label hak cipta pada data digital sebagai bukti otentik kepemilikan karya digital tersebut.

Konsep *watermark* dapat diterapkan pada berbagai domain dari suatu dokumen digital. Oleh karena itu penerapan *watermark* pada data digital seperti teks, citra, video, dan audio dapat dilakukan langsung pada jenis data digital tersebut. Misalnya untuk dokumen digital citra dan video dapat menggunakan *watermark* pada domain spasial, sedangkan untuk dokumen digital audio dapat menggunakan *watermark* pada domain waktu, maupun dapat terlebih dahulu dilakukan transformasi ke dalam domain yang lain.

Selain digunakan untuk proteksi kepemilikan serta untuk memonitor distribusi material dokumen digital ataupun untuk otentikasi, ada beberapa bentuk aplikasi lainnya yang menerapkan konsep *watermark*. Misalnya, pemanfaatan digital *watermark*

pada industri *broadcasting* untuk memonitor iklan yang ditayangkan. Melalui aplikasi tersebut, pihak pemasang iklan bisa mendeteksi dan menghitung apakah iklannya sudah ditayangkan sesuai dengan kesepakatan kontrak kerja. Sedangkan dari pihak produser atau pemegang hak milik dari suatu film dapat memonitor distribusi dan penayangan filmnya. Sehingga, penayangan ulang yang tidak sesuai kontrak dapat terdeteksi.

2.1.3 Nomor Dokumen untuk Watermark

Pada sebuah paper oleh Jeremy Hylton [Hylton 1994], dipaparkan tentang konsep penggunaan *document ID*, yakni:

The purpose or function of a document ID is to provide a globally unique, persistent identifier used for recognition, for access to characteristics of the document or for access to the document itself. A document may have several names. In particular, we distinguish between the name of a document and its location. The combination of a (FQDN of a) storage service and the name used by that storage service constitutes a location, not a name. A particular document ID belongs to a namespace. There can be any number of namespaces, and thus a document can have different names in different namespaces. There may be, but isn't necessarily, a way to translate between different namespaces.

Setrag Khoshafian [Khoshafian 1996] menyebutkan tentang konsep *document ID* pada bukunya, yakni:

Another method for identifying objects is unique keys (also called identifier keys). This mechanism is commonly used in DBMSs. For example, for the database table storing Documents, the key could be a document's name--if the name is unique and uniquely

identifies a document or the document ID that is a unique identifier for documents.

Berdasarkan paparan referensi tersebut diatas tentang *document ID* atau yang lebih dikenal dengan istilah nomor dokumen, maka suatu nomor dokumen memiliki peranan yang sangat vital dalam penentuan ciri khas suatu dokumen, terutama dalam implementasi dokumen digital. Nomor dokumen memberikan penekanan pada karakteristik yang membedakan antara dua atau lebih dokumen dengan judul yang sama.

Dalam kehidupan sehari-hari, konsep nomor dokumen telah menjadi dasar pada sistem perpustakaan. Apabila diperhatikan dengan seksama, maka setiap buku pada perpustakaan pasti memiliki nomor dokumen yang berbeda walaupun memiliki judul yang sama. Hal ini akan membuat struktur dan manajemen buku menjadi lebih mudah dipilah-pilah sesuai dengan kebutuhan sistem.

Pada pembahasan Tugas Akhir ini, konsep nomor dokumen akan digunakan sebagai penanda, ciri khas atau karakteristik dari suatu pola *watermark*. Nomor dokumen tersebut akan menjadi nilai masukan yang menentukan pola bentuk dan warna yang dihasilkan pada *watermark*.

Secara umum, nomor dokumen akan diolah sedemikian rupa sehingga menjadi nilai parameter pada fungsi-fungsi parametrik. Fungsi-fungsi parametrik inilah yang bertugas untuk membentuk pola-pola *watermark* yang memenuhi aspek estetika. Oleh karena itu, diperlukan adanya klasifikasi dan penentuan nilai parameter yang tepat agar tujuan tersebut dapat tercapai.

2.2 Matematika untuk Watermark

Kata matematika berasal dari kata *máthema* dalam bahasa Yunani yang memiliki arti yakni: "sains, ilmu pengetahuan, atau belajar", maupun sering juga disebut dengan *mathematikós* yang diartikan sebagai "suka belajar". Disiplin utama dalam matematika didasarkan pada kebutuhan perhitungan, pengukuran, dan melakukan prediksi. Ketiga kebutuhan tersebut secara umum berkaitan dengan pembagian umum bidang matematika, yakni: studi tentang struktur, ruang dan perubahan.

Studi tentang struktur dimulai dengan bilangan. Hal pertama dan yang sangat umum adalah bilangan natural, bilangan bulat, dan operasi arimetikanya. Semua cakupan tentang hal tersebut dijabarkan dalam aljabar dasar. Sifat bilangan bulat yang lebih mendalam dipelajari dalam teori bilangan. Adapun studi tentang metode-metode untuk memecahkan persamaan matematika dipelajari dalam aljabar abstrak. Konsep vektor, digeneralisasi menjadi vektor ruang yang dipelajari dalam aljabar linier, yang termasuk dalam dua cabang studi, yakni: struktur dan ruang.

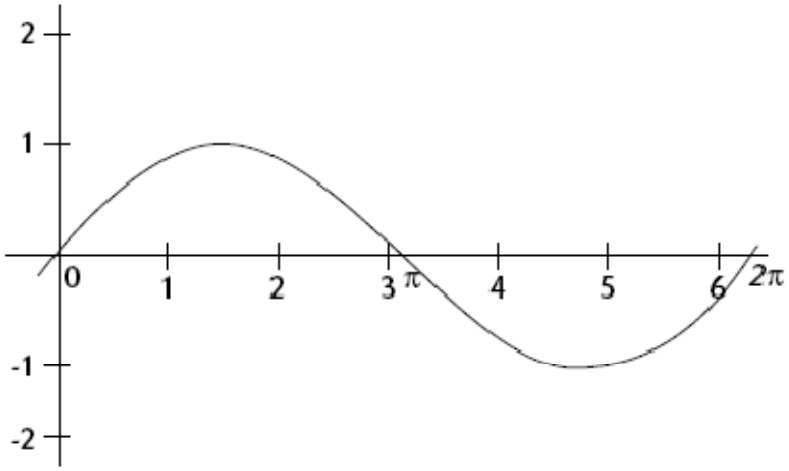
Studi tentang ruang pada ilmu matematika mencakup tentang pembelajaran geometri dan trigonometri. Konsep trigonometri merupakan cabang ilmu matematika yang nantinya akan menjadi dasar dan terkait erat dengan proses pembuatan *watermark* yang terdapat dalam pembahasan tugas akhir ini. Hal lain yang penting untuk dicermati adalah penggunaan konsep bilangan untuk menghasilkan suatu pola yang indah pada *watermark*.

2.2.1 Persamaan Trigonometri

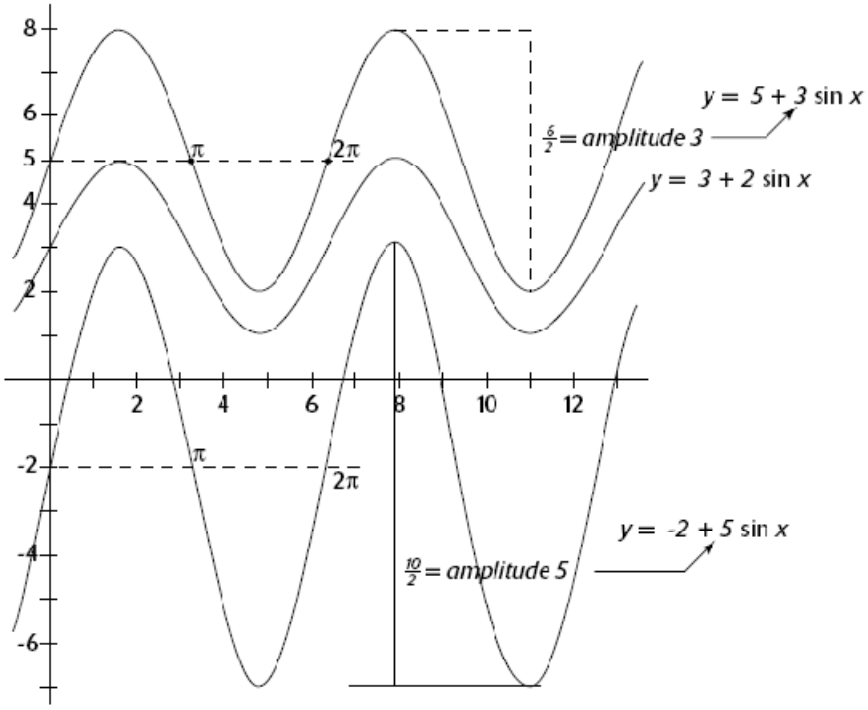
Berikut ini akan dipaparkan beberapa hal terkait mengenai trigonometri berdasarkan sebuah buku oleh David Alan Herzog [Herzog 2005], yakni:

The building blocks of trigonometry are based on the characteristics of similar triangles that were first formulated by Euclid. He discovered that if two triangles have two angles of equal measure, then the triangles are similar. In similar triangles, the ratios of the corresponding sides of one to the other are all equal. Since all right triangles contain a 90° angle, proving two of them similar only requires having one acute angle of one triangle equal in measure to one acute angle of the second. Having established that, we easily find that in two similar right triangles, the ratio of each side to another in one triangle is equal to the ratio between the two corresponding sides of the other triangle. It is no long stretch from there to realize that this must be true of all similar triangles. Those relationships led to the ***trigonometric ratios.***

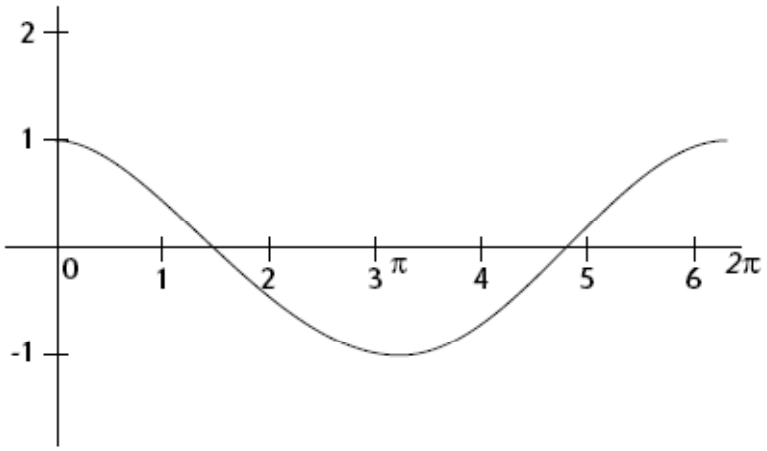
Berdasarkan rasio trigonometri maka akan didapatkan tiga fungsi yang sangat terkenal pada bidang trigonometri, yakni: fungsi sinus, cosinus, dan tangen. Apabila dilakukan penelusuran dalam satu periode putaran lingkaran dengan acuan pengukuran radian serta penerapan beberapa perubahan amplitudo maupun frekuensi fungsi, maka ketiga fungsi tersebut akan memberikan bentuk kurva seperti pada gambar-gambar berikut ini (sesuai dengan ilustrasi Gambar 2.1 hingga 2.4, yang menerangkan tentang konsep kurva trigonometri).



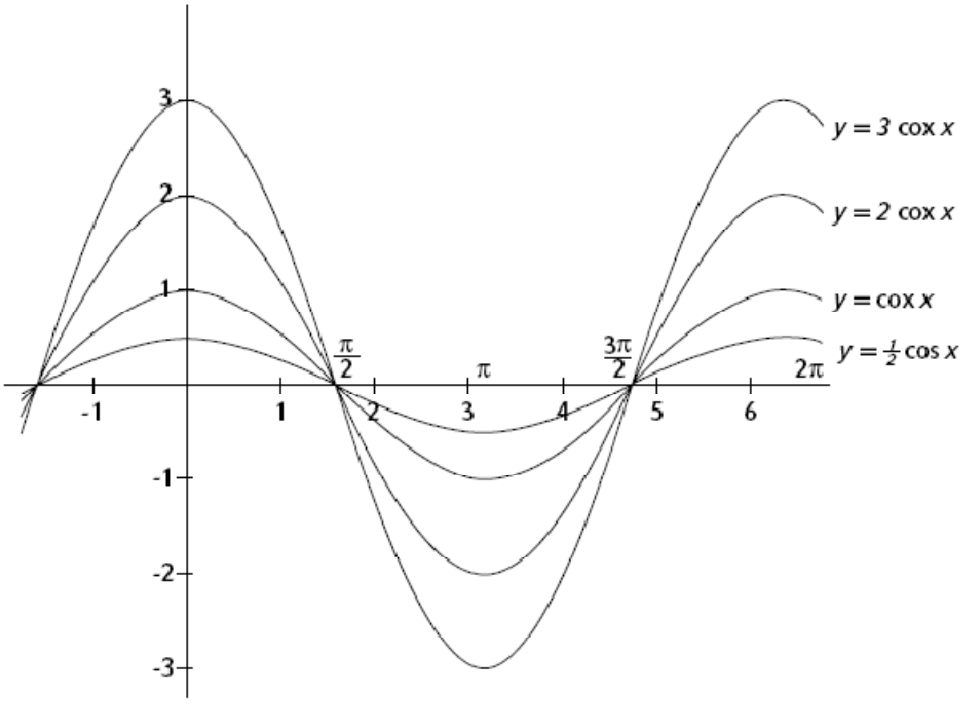
Gambar 2.1 Grafik Fungsi Sinus Satu Periode Putaran



Gambar 2.2 Grafik Fungsi Sinus dengan Modifikasi Amplitudo dan Frekuensi



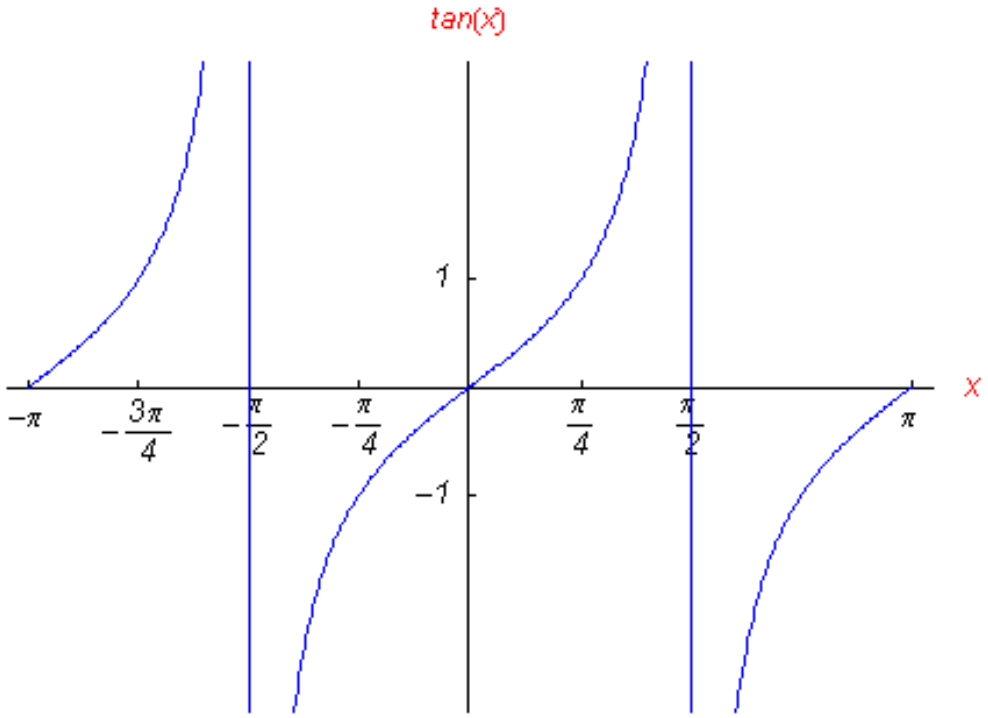
Gambar 2.3 Grafik Fungsi Cosinus Satu Periode Putaran



Gambar 2.4 Grafik Fungsi Cosinus dengan Modifikasi Amplitudo

Proses manipulasi amplitudo pada fungsi sinus maupun cosinus akan membuat nilai maksimal dan minimal fungsi tersebut akan semakin meningkat. Sedangkan manipulasi frekuensi akan mengakibatkan semakin banyaknya fase putaran yang dilalui oleh suatu grafik fungsi tersebut. Berdasarkan karakteristik fungsi sinus dan cosinus tersebut, maka hal ini bisa dijadikan dasar pembuatan fungsi-fungsi parametrik (akan dibahas lebih lanjut pada sub bab berikutnya) untuk membentuk pola *watermark*.

Sedangkan pola grafik fungsi tangen memiliki perbedaan jika dibandingkan dengan fungsi sinus dan cosinus, seperti yang ditunjukkan pada Gambar 2.5 berikut ini.



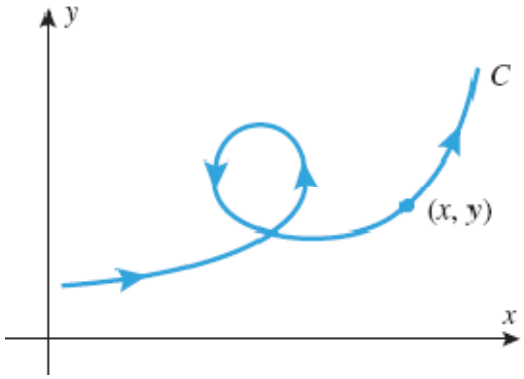
Gambar 2.5 Grafik Fungsi Tangen

2.2.2 Persamaan Parametrik

Suatu partikel bergerak sepanjang kurva C pada bidang *cartesian* *xy* yang merupakan titik-titik koordinat *x* dan *y*, dalam suatu fungsi waktu yang diberikan notasi sebagai berikut:

$$x = f(t),\ y = g(t)$$

Hal tersebut dikenal dengan istilah persamaan parametrik dari pergerakan suatu partikel yang melewati lintasan kurva C, yang diilustrasikan pada Gambar 2.6 berikut ini.



Gambar 2.6 Pergerakan Partikel pada Lintasan Kurva C

Berikut ini diberikan suatu studi kasus untuk menggambar grafik dari suatu persamaan parametrik, yakni:

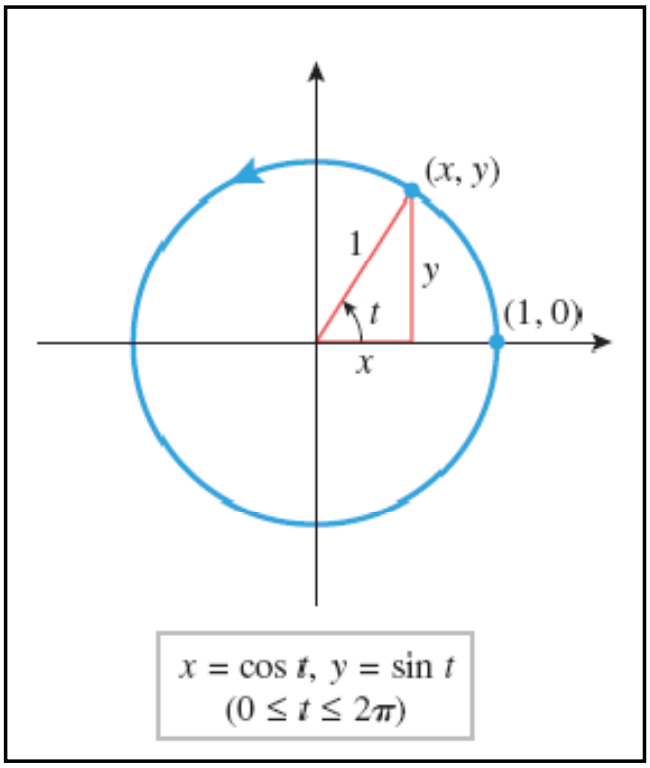
$$x = \cos(t)\ ,\ y = \sin(t) \quad (0 \le t \le 2\pi)$$

Salah satu cara untuk dapat menyelesaikan persoalan diatas adalah dengan proses substitusi nilai parameter *t*, tentunya dengan mengingat konsep identitas trigonometri, sehingga:

$$x^2 + y^2 = \sin^2 t + \cos^2 t = 1$$

Gambar 2.7 Penerapan Konsep Identitas Trigonometri

Berikut ini adalah kurva grafik yang dihasilkan oleh persamaan parametrik tersebut diatas (ilustrasi Gambar 2.8).



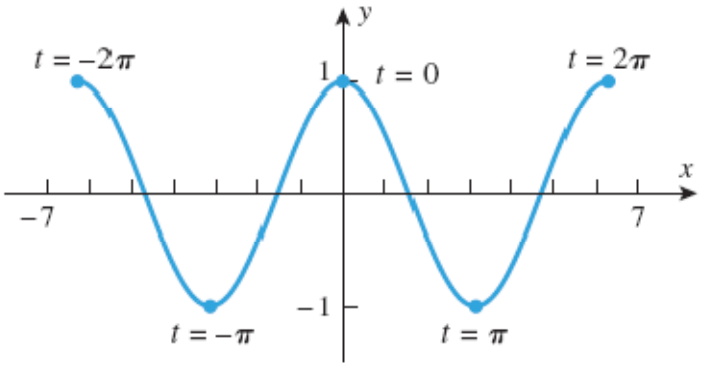
Gambar 2.8 Kurva Hasil Persamaan Parametrik

Suatu persamaan $y = f(x)$ dapat diubah dalam bentuk persamaan parametrik, yakni dengan memberikan suatu parameter $t = x$. Sebagai contoh, diberikan suatu persamaan $y = \cos(x)$ pada interval [-2 π, 2 π]. Maka, bentuk persamaan parametrik yang sesuai adalah sebagai berikut:

$$x = t, \quad y = \cos t \qquad (-2\pi \leq t \leq 2\pi)$$

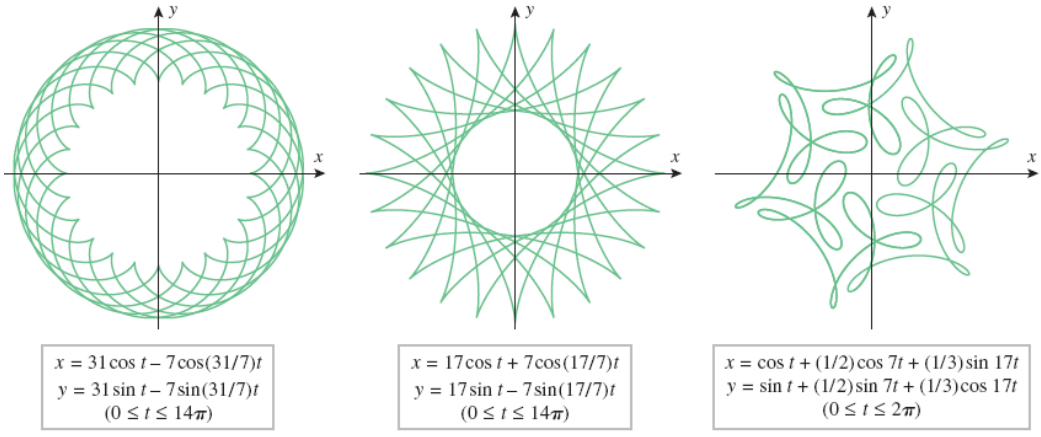
Gambar 2.9 Bentuk Persamaan Parametrik

Sehingga akan dihasilkan suatu bentuk grafik sesuai Gambar 2.10.



Gambar 2.10 Kurva Persamaan Parametrik

Beberapa fungsi parametrik memiliki bentuk kurva yang sangat kompleks, sehingga sangat sulit untuk digambar dengan metode manual. Oleh karena itu diperlukan adanya perhitungan otomatis menggunakan sistem komputer untuk menghasilkan gambar dari persamaan parametrik tersebut. Gambar 2.11 adalah beberapa contoh penggambaran kurva persamaan parametrik.



Gambar 2.11 Beberapa Contoh Kurva Persamaan Parametrik

2.2.3 Kurva Lissajous

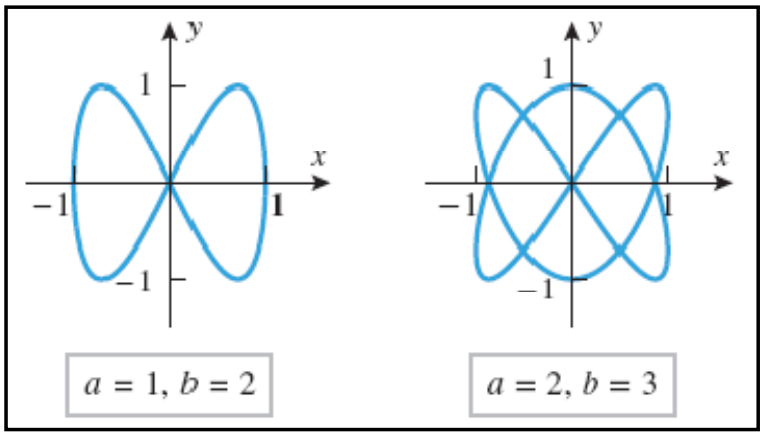
Pada pertengahan abad 19, seorang fisikawan Perancis yang bernama Jules Antoine Lissajous (1822–1880) sangat tertarik pada bentuk persamaan parametrik berikut ini:

$$x = \sin at, \quad y = \sin bt$$

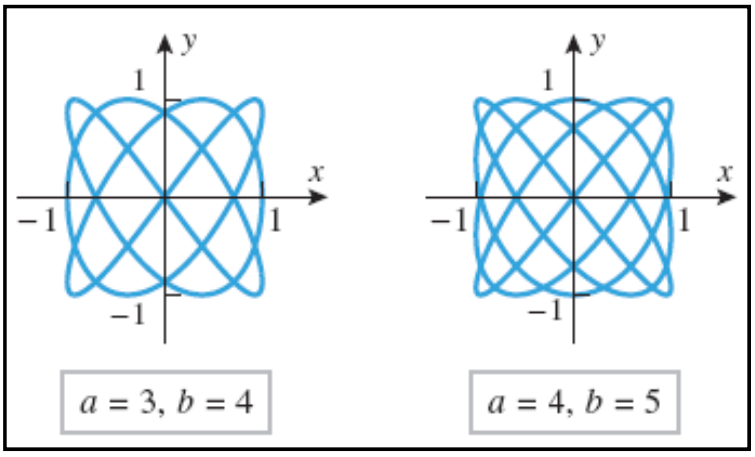
Gambar 2.12 Persamaan Parametrik Kurva Lissajous

Beliau mengembangkan fungsi tersebut pada suatu pembelajaran tentang getaran dengan menggabungkan dua gerakan sinusoidal yang saling tegak lurus. Persamaan diatas menggambarkan adanya getaran sinusoidal pada sumbu x dengan frekuensi a/2 π dan getaran sinusoidal pada sumbu y dengan frekuensi b/2 π. Jika nilai perbandingan antara a dengan b adalah bilangan rasional, maka akan menghasilkan efek getaran yang bergerak sepanjang lintasan kurva, yang dikenal dengan kurva *Lissajous*.

Berikut ini akan diberikan perbandingan gambar kurva Lissajous dengan perbedaan konstanta a dan b sesuai dengan ilustrasi Gambar 2.13 dan 2.14.



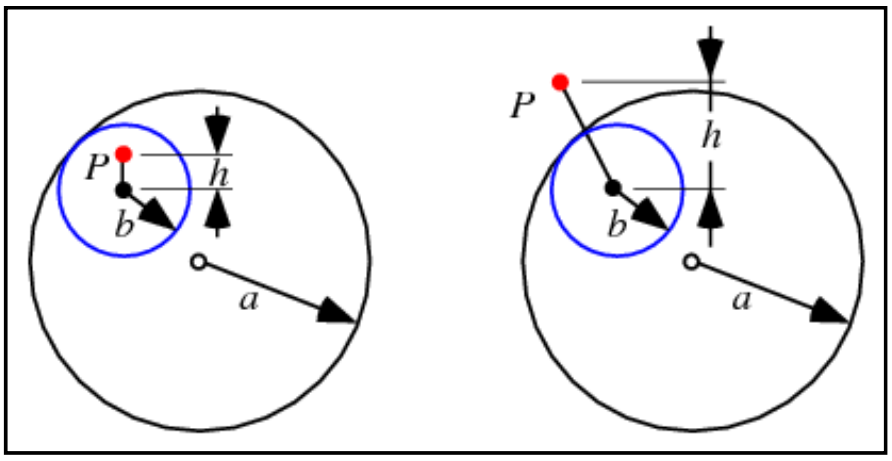
Gambar 2.13 Perbandingan Kurva Lissajous Bagian 1



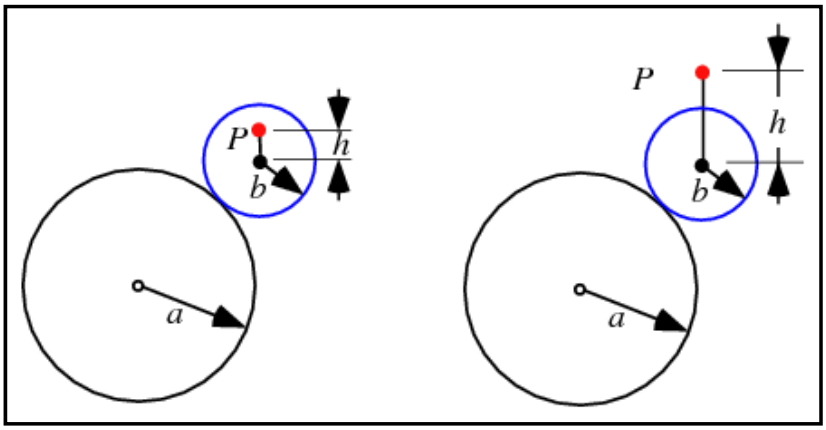
Gambar 2.14 Perbandingan Kurva Lissajous Bagian 2

Diperlukan variasi perbandingan konstanta, maupun parameter nilai lainnya (termasuk proses modifikasi persamaan parametrik) pada persamaan kurva *Lissajous* sehingga menghasilkan bentuk pola gambar yang cukup indah dan variatif.

2.2.4 Hypotrochoid dan Epitrochoid



Gambar 2.15 Ilustrasi Hypotrochoid



Gambar 2.16 Ilustrasi Epitrochoid

Ilustrasi Gambar 2.15 dan 2.16 merupakan proses pembentukan kurva *Hypotrochoid* dan *Epitrochoid*. Kurva *Hypotrochoid* dan *Epitrochoid* merupakan garis kurva yang menyatakan kumpulan titik-titik *P* akibat pergerakan lingkaran kecil dengan jari-jari *b* pada area lingkaran besar dengan jari-jari *a*.

Berikut ini adalah persamaan parametrik untuk penggambaran kurva *Hypotrochoid* dan *Epitrochoid* sesuai dengan Gambar 2.17 dan Gambar 2.18.

$$x = ((a-b) * \cos(\theta)) + \left(\mathrm{h} * \cos\left(\frac{a-b}{b} * \theta\right)\right)$$

$$y = ((a-b) * \sin(\theta)) - \left(\mathrm{h} * \sin\left(\frac{a-b}{b} * \theta\right)\right)$$

Gambar 2.17 Persamaan Parametrik Kurva Hypotrochoid

$$x = ((a+b) * \cos(\theta)) - \left(\mathrm{h} * \cos\left(\frac{a+b}{b} * \theta\right)\right)$$

$$y = ((a+b) * \sin(\theta)) - \left(\mathrm{h} * \sin\left(\frac{a+b}{b} * \theta\right)\right)$$

Gambar 2.18 Persamaan Parametrik Kurva Epitrochoid

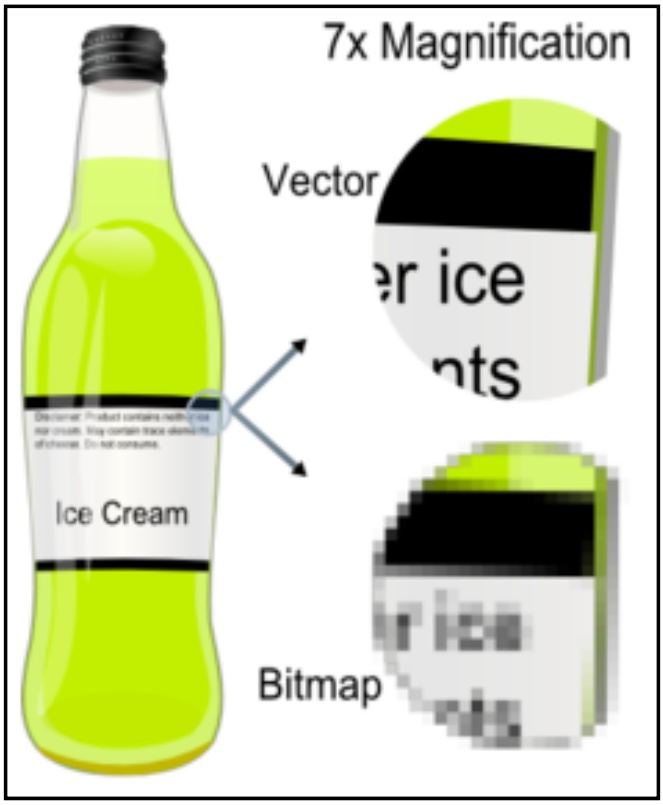
2.3 Vector Graphic

Vector Graphics adalah penggunaan konsep bentuk-bentuk geometri primitif seperti titik, garis, lengkungan, dan poligon yang berdasarkan pada persamaan matematika, untuk mewakili gambar dalam grafika komputer. Format *Vector Graphics* melengkapi konsep pada raster *graphics* (merupakan perwakilan dari gambar sebagai serangkaian pixel-pixel).

Tampilan pada layar komputer terdiri atas titik-titik kecil yang disebut sebagai *pixel*. Suatu gambar akan dihasilkan berdasarkan kumpulan titik-titik tersebut. Semakin tinggi kerapatan antar titik, maka gambar yang dihasilkan akan lebih berkualitas. Apabila gambar tersebut diperbesar, maka akan dapat dilihat dengan jelas bahwa gambarnya menjadi kasar sehingga perbedaan antar *pixel* dapat terlihat secara jelas.

File *Vector Graphics* menyimpan detil garis, bentuk, dan warna dalam gambar pada suatu formula matematika. Formula tersebut menentukan posisi yang tepat untuk peletakan suatu titik sehingga memungkinkan pembentukan kualitas gambar yang sangat baik pada tampilan layar komputer. Apabila dilakukan pencetakan pada kertas, maka akan dihasilkan gambar yang tajam dengan kualitas yang tinggi.

Berikut ini digambarkan tentang perbedaan antara *vector graphic* dan raster *graphic*, setelah melalui proses pembesaran gambar dapat dilihat perbedaan kualitas *image* yang dihasilkan oleh tiap metode *graphic* tersebut (ilustrasi pada Gambar 2.19).



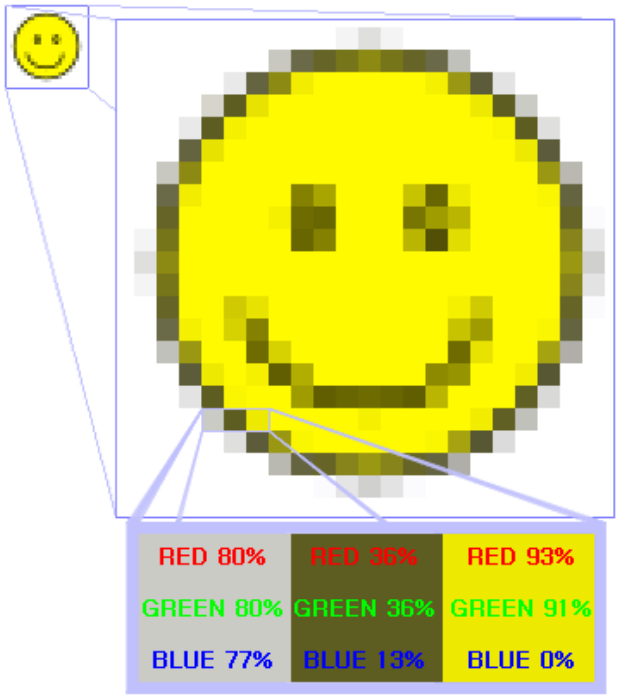
Gambar 2.19 Perbandingan Vector dan Raster Graphic

2.4 Raster Graphic

Raster merupakan kata yang berasal dari bahasa latin, yakni rastrum yang dapat diartikan sebagai "yang menyapu". Apabila ditinjau dari skema penempatan *pixel* pada area *image*, maka raster merupakan suatu metode untuk menempatkan titik-titik yang dikenal sebagai *pixel* pada area gambar.

Dalam bidang komputer grafis, *raster graphic* adalah struktur data yang mewakili suatu kotak persegi (*pixel*), atau suatu titik warna yang dapat dilihat melalui monitor, kertas, ataupun media lain. Gambar Raster dapat disimpan pada file gambar dalam berbagai format. Sebuah bitmap terkait dengan bit-bit pada gambar yang ditampilkan di layar monitor. Bitmap ditentukan oleh lebar dan tinggi gambar dalam *pixel* dan oleh jumlah bit per *pixel* (kedalaman warna, yang menentukan jumlah warna yang dapat mewakili gambar tersebut).

Pada Gambar 2.20 berikut ini adalah sketsa *image* pada *raster graphic*:



Gambar 2.20 Sketsa Raster Graphic

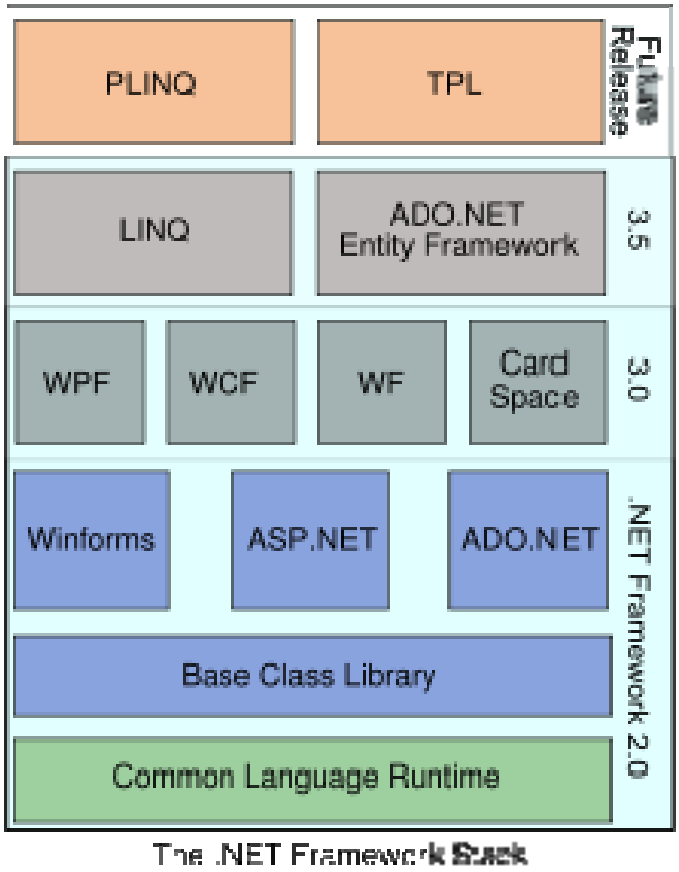
2.5 .NET Framework

Secara umum, *framework* merupakan konsep dasar yang digunakan untuk menyelesaikan suatu masalah. Dalam konteks perangkat lunak, *framework* merupakan sebuah desain sistem yang *reusable*. *Framework* dapat dinyatakan sebagai aplikasi pendukung, *library*, atau perangkat lunak untuk membantu pengembangan dan penyatuan komponen yang berbeda dalam suatu sistem maupun proyek.

Pengembangan *framework* memungkinkan para *developer* untuk membuat aplikasi yang handal berdasarkan konsep-konsep dan fungsi-fungsi yang telah didefinisikan oleh sistem pada *framework* tersebut. Sehingga proses pembuatan aplikasi menjadi lebih mudah, cepat, dan efisien.

Pada saat ini, cukup banyak *framework* yang dikembangkan baik oleh perseorangan maupun suatu organisasi. Setiap *framework* mempunyai dasar, tujuan, platform dan bahasa yang berbeda-beda, meskipun secara konsep terdapat persamaan yang mendasar. Pada sub bab ini akan dibahas tentang *.NET Framework* sebagai salah satu *framework* yang cukup banyak dipakai dikalangan developer maupun para mahasiswa.

Sebagai salah satu raksasa bidang teknologi informasi dan *software development*, *Microsoft* mulai mengembangkan *framework* ini pada akhir tahun 1990-an. Pada awalnya dinamakan sebagai *Next Generation Windows Service* (NGWS). Berdasarkan proses *development* yang cukup panjang, maka *.NET Framework* 1.0 versi *beta* diluncurkan pada akhir tahun 2000. Pada saat ini versi 3.5 merupakan rilis terbaru dari *.NET framework*. Pada Gambar 2.21 merupakan gambaran tentang perkembangan *.NET framework* dan teknologi yang didukung pada setiap versi.



Gambar 2.21 Perkembangan Teknologi .NET Framework

.NET framework merupakan komponen yang menyatu dengan sistem *Windows* yang mendukung proses pembuatan dan *deployment* suatu aplikasi serta *XML web services*. *Framework* ini dirancang untuk dapat memenuhi beberapa hal berikut ini:

- Menyediakan lingkungan pemrograman berorientasi obyek yang konsisten, sehingga suatu kode *object* dapat dieksekusi secara lokal maupun *remoting*.

- Menyediakan lingkungan eksekusi kode-kode program yang mampu meminimalkan proses *software deployment* dan masalah akibat perbedaan versi.
- Menyediakan lingkungan eksekusi kode-kode program yang mampu mengurangi kendala pada tingkat performa yang terjadi pada lingkungan *scripting* atau *interpreter*.
- Menyediakan suatu tipe development yang konsisten terhadap berbagai macam bentuk aplikasi, misalnya aplikasi berbasis Windows dan aplikasi berbasis web.
- Menyediakan standar komunikasi pada lingkungan industri, yang menjamin adanya fleksibilitas dan integrasi yang ditawarkan oleh .NET *framework*.

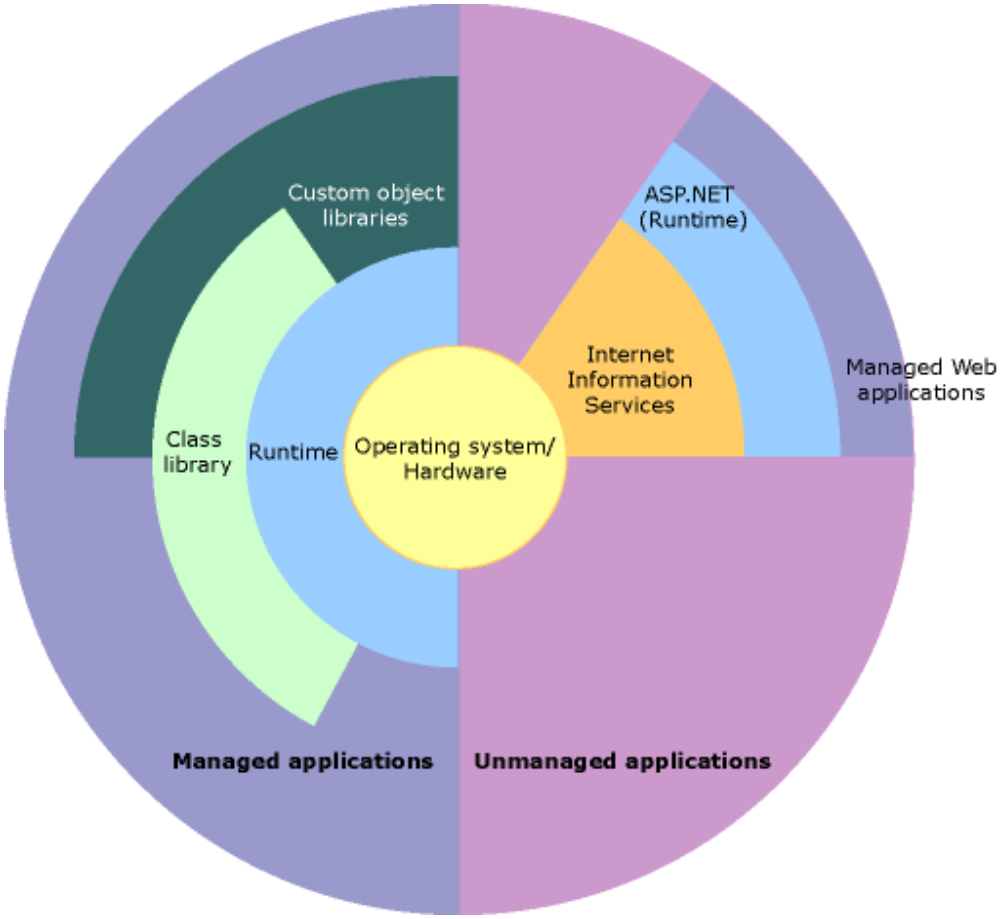
.NET *framework* memiliki 2 komponen utama, yakni: *Common Language Runtime* (lebih dikenal dengan singkatan CLR) dan *.NET Framework Class Library*. CLR merupakan dasar dari semua konsep yang terdapat pada sistem .NET framework. CLR dapat dijelaskan sebagai *agent* yang mengatur proses eksekusi pada kode-kode program, menyediakan service utama seperti: pengaturan memori, thread, dan *remoting*. Disamping itu, CLR mengutamakan proses keamanan pada saat eksekusi kode-kode program, sehingga menghasilkan kualitas program yang handal.

Kode program yang ditujukan untuk mengakses CLR dikenal dengan istilah *managed code*. Sebaliknya, *unmanage code* merupakan kode program yang tidak ditujukan secara langsung untuk mengakses CLR.

Class library merupakan salah komponen utama yang terdapat pada .NET Framework. Komponen ini merupakan kumpulan tipe *object-oriented* yang *reuseable* sehingga memudahkan pengguna untuk membuat aplikasi baik itu berupa *command-line* maupun

bentuk aplikasi GUI (*graphical user interface*) yang berdasarkan inovasi terbaru yang disediakan oleh WinForms, ASP.NET, dan XML Web services.

Pada Gambar 2.22 menjelaskan tentang relasi antara CLR dan Class Library terhadap sistem aplikasi maupun sistem pada komputer secara umum, disamping itu juga menjelaskan tentang proses *managed code* pada sistem arsitektur skala besar.



Gambar 2.22 Relasi CLR, Class Library dan Sistem Komputer

2.5.1 ASP.NET

ASP.NET merupakan suatu kumpulan teknologi yang tercakup dalam *Microsoft .NET Framework* untuk proses pembuatan aplikasi web dan *XML Web services*. Proses eksekusi suatu halaman ASP.NET terjadi di sisi server yang akan menghasilkan kode-kode *mark-up* (HTML, WML, atau XML) yang akan dikirim ke *browser desktop* atau *mobile*.

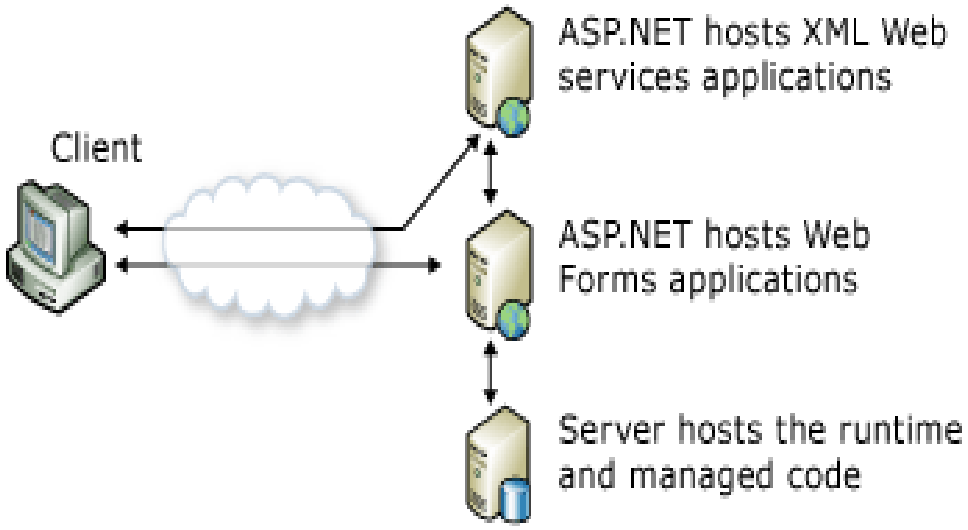
Halaman ASP.NET menerapkan proses kompilasi, serta konsep *event-driven programming* yang meningkatkan performa dan memisahkan antara *application logic* dan *user interface*. Halaman ASP.NET dan *file XML Web services* diciptakan dengan menggunakan ASP.NET *server-side logic* yang dibuat pada Visual Basic, C#, maupun tipe bahasa pemrograman lainnya yang sesuai dengan lingkungan *.NET framework*. Aplikasi web dan *XML Web services* menggunakan kelebihan-kelebihan yang terdapat pada fungsi CLR, seperti: keamanan proses eksekusi kode-kode program, konsep *inheritance*, konsep *language interoperability*, konsep *versioning*, dan sistem keamanan yang terpadu.

Teknologi ASP.NET meliputi beberapa hal berikut ini:

- Halaman dan *controls framework*
- ASP.NET *compiler*
- Infrastruktur keamanan
- Fasilitas *state-management*
- Konfigurasi pada aplikasi web
- Fitur untuk memonitor tingkat performa dan kehandalan sistem aplikasi

- Mendukung proses *debugging*
- *Framework* untuk *XML Web Services*
- Menyediakan pengaturan *life cycle application*

Gambar berikut ini menjelaskan tentang skema suatu jaringan yang menggunakan konsep *manage code* yang berjalan pada lingkungan server yang berbeda. Server IIS dan SQL Server dapat menjalankan proses, sementara *application logic* pada halaman ASP.NET dieksekusi melalui suatu *manage code*.



Gambar 2.23 Konsep Manage Code pada ASP.NET

ASP.NET menyediakan lingkungan *hosting* sehingga para *developer* dapat menggunakan *.NET framework* sebagai proses pembuatan aplikasi berbasis web. Namun, ASP.NET tidak hanya berperan sebagai *runtime host*, ini merupakan arsitektur yang sangat lengkap untuk proses pembuatan aplikasi berbasis web dan distribusi *object* menggunakan *manage code*.

Berikut ini adalah contoh penggunaan kode pada suatu halaman ASP.NET sesuai dengan ilustrasi Gambar 2.24.

```
<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">

    protected void Page_Load(object sender, EventArgs e)
    {
        Label1.Text = DateTime.Now.ToLongDateString();
    }

</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Welcome to ASP.NET Web Application</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        The current time is:
  <asp:Label runat="server" id="Label1" />
    </div>
    </form>

</body>
</html>
```

Gambar 2.24 Contoh Kode pada Halaman ASP.NET

2.5.2 System.Drawing dengan C# pada .NET Framework

Pada sebuah artikel yang ditulis oleh Budi Kurniawan [Kurniawan 2002], berikut ini adalah konsep tentang *System.Drawing*:

The System.Drawing namespace, which contains types that help you with drawing, plays an important role in Windows programming. You need its members to draw a custom control user interface and for sending text and graphics to a printer. Even when you are only using standard controls on your form, you have used some of its members, probably without realizing it. Understanding the System.Drawing namespace enables you to write better -- and probably faster -- code. There are probably only 10 members of the System.Drawing namespace that you use in 99% of your programs.

Berdasarkan paparan referensi tersebut, maka *namespace System.Drawing* pada *.NET framework* yang mengandung tipe-tipe dan fungsi maupun struktur yang memudahkan proses menggambar dan memiliki peranan yang sangat penting pada pemrograman *Windows*. Fungsi-fungsi yang terdapat pada *namespace* tersebut memudahkan para programmer untuk mengirimkan *text* maupun *graphic* ke peralatan printer.

Berikut ini akan dijelaskan tentang struktur *Point*, *Size*, *Color, Pen Class, Brush Class, Graphics Class,* dan *Bitmap Class*. Struktur tersebut seringkali dipakai untuk pembuatan aplikasi yang berbasis *Windows Form* maupun web untuk proses manipulasi menggambar *graphic*.

Untuk memulai suatu *project* pemrograman dengan bahasa C# yang memanipulasi *object graphic*, baris kode berikut merupakan *library* yang digunakan (ilustrasi Gambar 2.25):

```
using System.Drawing;
```

Gambar 2.25 Include Namespace System.Drawing

Struktur Point dan PointF

Object pada struktur *Point* menyatakan suatu koordinat pada sistem 2 dimensi. Contoh yang paling mudah untuk mendefinisikan struktur *Point* adalah menggunakan *constructor* dengan menyatakan nilai *integer* pada absis dan ordinatnya, seperti dalam ilustrasi Gambar 2.26 berikut ini.

```
Point myPoint = new Point(10, 20);
```

Gambar 2.26 Constructor Struktur Point

Apabila diperlukan nilai yang membutuhkan tingkat presisi lebih detil, maka bisa digunakan struktur *PointF*. Pada struktur *PointF*, constructor dinyatakan dengan penggunaan tipe data *float*. Berikut ini adalah definisi *constructor* pada Gambar 2.27.

```
Public PointF (float x, float y);
```

Gambar 2.27 Constructor Struktur PointF

Struktur Size dan SizeF

Object pada struktur *Size* menyatakan suatu panjang dan lebar pada area *rectangle* atau persegi. Contoh mudah untuk mendefinisikan struktur *Size* adalah menggunakan *constructor* dengan menyatakan nilai *integer* pada nilai panjang dan lebarnya, seperti dalam ilustrasi Gambar 2.28 berikut ini.

```
Size mySize = new Size(100, 200);
```

Gambar 2.28 Constructor Struktur Point

Apabila diperlukan nilai yang membutuhkan tingkat presisi lebih detil, maka bisa digunakan struktur *SizeF*. Pada struktur *SizeF*, constructor dinyatakan dengan penggunaan tipe data *float*. Berikut ini adalah definisi *constructor* Gambar 2.29.

```
Public SizeF (float x, float y);
```

Gambar 2.29 Constructor Struktur PointF

Struktur Color

Pada sebuah artikel yang ditulis oleh Budi Kurniawan [Kurniawan 2002], berikut ini diuraikan tentang konsep struktur *Color*:

The Color Structure represents a color that you can use in drawing shapes or to assign to the BackColor or ForeColor properties of a control. To obtain a Color object, you don't use any constructor, because this structure does not have one. Instead, the structure includes a large number of static properties that represent various colors. For example, the Brown property represents a brown Color object. Because these properties are static, you don't need to instantiate a Color object to use them.

Berdasarkan paparan tersebut, maka proses mendefinisikan struktur *Color* bisa dengan mudah dilakukan. Tersedia lebih dari

140 properti struktur *Color* yang bisa langsung digunakan seperti pada ilustrasi baris kode pada Gambar 2.30 berikut ini.

```
Color myColor = Color.Blue;
```

Gambar 2.30 Deklarasi Struktur Color

Disamping struktur yang dapat secara langsung diakses melalui tipe properti *Color*, terdapat juga fungsi yang menggunakan pengaturan nilai parameter komponen RGB. Tiap komponen RGB merupakan suatu nilai *integer* yang berkisar antara 0-255 yang akan menghasilkan *color* 32-bit dengan adanya penambahan nilai *alpha*. Berikut ini adalah ilustrasi baris kode tentang hal tersebut pada Gambar 2.31.

```
// Deklarasi fungsi Color.FromArgb dengan komponen RGB
Color myColor = Color.FromArgb(255, 255, 255);

// Deklarasi fungsi Color.FromArgb dengan komponen RGB
// dan nilai alpha
Public static Color FromArgb(int alpha, int red,
                             int green, int blue);
```

Gambar 2.31 Deklarasi Struktur Color dengan Komponen RGB

Struktur Pen Class

Pen Class merupakan suatu *object* yang mewakili representasi sebuah pena yang dapat digunakan untuk menggambar bentuk maupun menulis teks. Suatu object pena dapat memiliki warna, ketebalan, sebuah *brush*, dan tipe properti lainnya. Cara paling mudah untuk mendefinisikan suatu *object* pena adalah dengan memberikan nilai parameter *color*, berikut ini adalah contoh baris

kode untuk definisi sebuah pena dengan warna kuning pada ilustrasi Gambar 2.32.

```
// Deklarasi object pena dengan nilai default ketebalan
// pena adalah 1f
Pen myPen = new Pen(Color.Yellow);

// Deklarasi object pena dengan nilai ketebalan pena
Pen myPen = new Pen(Color.Yellow, 2f);
```

Gambar 2.32 Deklarasi Struktur Object Pena

Struktur Brush Class

Suatu *object* pena digunakan untuk menggambar garis, bentuk 2 dimensi, maupun menulis suatu teks. Untuk mewarnai interior pada bentuk yang telah digambar dapat digunakan *object brush*. Pada *.NET Framework Class Library* terdapat *Brush Class* (merupakan suatu *abstract class*) dan lima turunan *class* yang dapat digunakan untuk mewarnai interior suatu bentuk.

Berikut ini adalah penjelasan tentang lima *class* yang merupakan *class* turunan dari *Brush*, yakni:

- HatchBrush
  Merupakan representasi *brush* dengan tipe *hatch* pada suatu *rectangular brush*.
- LinearGradientBrush
  Merupakan suatu *brush* yang mewarnai dengan konsep gradasi warna linier.
- PathGradientBrush
  Merupakan tipe *brush* digunakan untuk mewarnai suatu *GraphicPath* dengan gradasi warna.

- SolidBrush
  Merupakan tipe *brush* yang akan mewarnai suatu area secara merata dengan satu warna.
- TextureBrush
  Merupakan tipe *brush* yang akan mewarnai suatu area dengan tekstur gambar.

Berikut ini adalah ilustrasi baris kode yang akan mewarnai suatu area persegi dengan *SolidBrush* pada Gambar 2.33.

```
// Deklarasi object graphic dari kanvas form
Graphics g = form.CreateGraphics();
SolidBrush solidBrush = new SolidBrush(Color.Blue);
g.FillRectangle(solidBrush, 10, 20, 100, 100);
```

Gambar 2.33 Deklarasi Struktur SolidBrush

Struktur Graphic Class

Graphics class merepresentasikan suatu area persegi dimana berbagai macam bentuk dapat digambarkan dan teks dapat dituliskan pada area tersebut. *Constructor* pada class ini bertipe *private*, sehingga tidak diperbolehkan melakukan deklarasi dengan *instantiate object* dari *Graphic class*. Akan tetapi, proses deklarasi yang dapat digunakan adalah melalui *object Graphic* yang didapatkan dari *object* yang lain, misalnya *object Graphic* dari suatu *window form* atau *control* yang lain yang memiliki fungsi *CreateGraphics* untuk mengembalikan suatu *object Graphic*.

Berikut ini akan dijelaskan tentang beberapa fungsi yang terdapat pada *Graphic class* dan merupakan fungsi utama yang akan dipakai pada proses implementasi pola *watermark*, yakni:

- Fungsi DrawLine
  Merupakan fungsi untuk melakukan proses menggambar garis berdasarkan titik-titik koordinat yang telah ditentukan pada suatu kanvas.
- Fungsi DrawString
  Merupakan fungsi untuk menuliskan teks pada area yang telah ditentukan diatas suatu kanvas.
- Fungsi FillRectangle
  Merupakan fungsi untuk mewarnai suatu area persegi (dengan menggunakan salah satu tipe *brush*) yang telah ditentukan pada area kanvas.

Berikut ini adalah penerapan fungsi *Graphic class* sesuai dengan ilustrasi Gambar 2.34.

```
// Deklarasi object graphic dari kanvas form
Graphics g = this.CreateGraphics();

Font font = new Font("Arial", 14);
Pen pen = new Pen(Color.Red, 2);
SolidBrush solidBrush = new SolidBrush(Color.Blue);


// Deklarasi proses menggambar garis pada kanvas
g.DrawLine(pen, 0, 0, 100, 200);


// Deklarasi proses menuliskan teks pada kanvas
g.DrawString("Hello", font, solidBrush, 10, 100);


// Deklarasi proses mewarnai area persegi pada kanvas
g.FillRectangle(solidBrush, 10, 10, 100, 300);
```

Gambar 2.34 Penerapan Fungsi Graphic Class

Struktur Bitmap Class

Bitmap class merepresentasikan sebuah tipe raster atau bitmap yang berbasis pada gambar. Cara yang mudah untuk melakukan proses *instantiate object bitmap* adalah dengan memberikan parameter berupa *complete path* yang merujuk pada suatu *file bitmap*. Sebagai contoh, baris kode berikut ini merupakan *instantiates object bitmap* dari sebuah *file* TV.bmp, yang akan memberikan suatu *object reference* pada *background image* dari suatu form (ilustrasi Gambar 2.35).

```
// Deklarasi object bitmap dengan referensi file bitmap
Image bitmap = new Bitmap("C:\\TV.bmp");
this.BackgroundImage = bitmap;
```

Gambar 2.35 Deklarasi Struktur Bitmap Class

Fungsi yang cukup penting pada *Bitmap class* adalah fungsi untuk proses manipulasi pixel. Sehingga memungkinkan untuk pembuatan aplikasi dengan konsep *image processing*. Berikut ini adalah contoh penerapan manipulasi pixel pada suatu *object bitmap* sesuai ilustrasi Gambar 2.36.

```
Bitmap bitmap = new Bitmap("C:\\TV.bmp");
int width  = bitmap.Width;
int height = bitmap.Height;

int i, j;
for (i = 0; i < width; i++)
{
  for (j = 0; j < height; j++)
  {
     Color pixelColor = bitmap.GetPixel(i, j);
     int r = pixelColor.R; // the Red component
     int b = pixelColor.B; // the Blue component
```

```
      Color newColor = Color.FromArgb(r, 0, b);
      bitmap.SetPixel(i, j, newColor);
   }
}
this.BackgroundImage = bitmap;
```

Gambar 2.36 Proses Manipulasi Pixel pada Bitmap Class

BAB III
ANALISIS DAN PERANCANGAN

Pada bab ini akan dibahas mengenai analisa, desain, dan implementasi perangkat lunak dari aplikasi *Generator Watermark yang Unik Berdasarkan Nomor Dokumen*. Pembahasan pada bab berikut ini akan menyajikan konsep dan desain sistem untuk menghasilkan pola-pola *watermark* yang unik dengan menggunakan konsep persamaan parametrik yang berdasarkan kurva *Lissajous* dan atau melakukan proses modifikasi terhadap fungsi dan parameter persamaan parametrik.

3.1 Analisis Permasalahan

Tugas Akhir ini berupaya untuk menemukan solusi terhadap permasalahan otentikasi dokumen digital. Metode *watermark* merupakan salah satu solusi yang sering dipakai untuk proses otentikasi dokumen digital. Oleh karena itu, terdapat berbagai macam metode untuk menghasilkan bentuk *watermark* yang khas pada suatu dokumen (termasuk *watermark* yang terdapat pada dokumen cetak atau *hardcopy*).

Metode *watermark* yang dipakai pada Tugas Akhir ini akan menerapkan persamaan parametrik sebagai dasar untuk membentuk pola gambar yang unik dan memenuhi nilai estetika. Oleh karena itu, diperlukan nilai parameter yang merupakan tanda pengenal (*identifier*) pada dokumen tersebut. Nilai parameter tersebut merupakan gabungan antara huruf dan atau angka yang terdapat pada setiap dokumen.

Beberapa hal yang menjadi fokus perhatian pada proses implementasi sistem agar Tugas Akhir ini memenuhi persyaratan

keunikan pola gambar dan nilai estetika pada *watermark* akan dibahas pada sub bab berikut ini.

3.1.1 Faktor Variasi Pola Watermark

Hal ini merupakan faktor utama yang diimplementasikan dalam struktur komponen program *Generator Watermark*. Berbagai macam variasi bentuk pola *watermark* akan dihasilkan oleh penentuan nilai parameter yang terdapat pada setiap dokumen. Hal tersebut akan membuat bentuk pola *watermark* yang dihasilkan lebih dinamis.

Variasi bentuk pola *watermark* yang dinamis tersebut disebabkan oleh adanya variasi penentuan nilai pada fungsi parametrik, sehingga bisa dihasilkan lebih dari satu pola *watermark* yang sama, akan tetapi perbedaan antara tiap bentuk pola tersebut dapat dilihat dengan jelas.

3.1.2 Faktor Bentuk Pola Watermark

Bentuk pola *watermark* yang dihasilkan oleh komponen program hasil implementasi lebih diutamakan dengan pola simetris. Dengan berbagai macam bentuk variasi, pola simetris pada suatu bentuk akan menampakkan segi estetika dan keindahan bentuk tersebut. Sedangkan pola asimetris kurang mampu membentuk sisi keindahan suatu bentuk yang terdiri atas berbagai macam pola yang digabung menjadi satu.

Bentuk dasar pola simetris yang akan sering menjadi acuan adalah bentuk lingkaran. Adapun bentuk lain yang digunakan adalah bentuk pola bintang dengan ujung tumpul, persegi banyak, pola bunga, dan bentuk ornamental yang menggabungkan beberapa bentuk dasar tersebut diatas.

3.1.3 Faktor Pola Garis Watermark

Bentuk *watermark* yang dihasilkan merupakan rangkaian dari beberapa garis yang digambar pada dokumen berdasarkan nilai-nilai parameter yang diberikan pada fungsi-fungsi parametrik. Proses penggabungan beberapa garis tersebut ditetapkan memenuhi rentang fase tertentu, sehingga dihasilkan pola garis yang seimbang pada gambar *watermark*. Pola garis yang dihindari adalah gabungan beberapa garis yang terlalu menumpuk pada suatu area tertentu.

Rentang fase pada setiap fungsi memiliki karakteristik yang berbeda-beda, oleh karena itu ditentukan nilai-nilai batasan yang sesuai sehingga terbentuk pola garis yang seimbang, indah, dan elegan.

3.1.4 Faktor Warna Gambar Watermark

Bentuk *watermark* yang dihasilkan merupakan rangkaian dari beberapa warna dasar RGB yang digambar pada dokumen berdasarkan nilai-nilai parameter yang diberikan pada fungsi-fungsi parametrik. Proses pewarnaan pada pola *watermark* akan membentuk pola gradasi warna yang disebabkan oleh perubahan nilai yang dihasilkan oleh fungsi sinusoidal pada persamaan parametrik.

Gradasi warna yang terbentuk pada pola *watermark* diharapkan dapat memenuhi batasan-batasan tertentu agar tidak dihasilkan warna yang cenderung pudar pada suatu area tertentu. Sehingga, hasil gradasi warna pada pola *watermark* akan cenderung menyerupai miniatur pola warna pelangi.

3.1.5 Penggunaan Teknik Emboss

Bentuk pola *watermark* yang dihasilkan oleh komponen program akan dilengkapi oleh pola teks yang digambar dengan menggunakan teknik *Emboss*. Teknik *Emboss* tersebut akan memberikan efek teks yang mempunyai bayangan (menyerupai bentuk 3D). Bentuk teks yang dihasilkan oleh metode *Emboss* tersebut akan menjadi dasar untuk penggambaran pola *background* garis pada dokumen.

Background pada dokumen akan memiliki tanda yang khas berupa nomor dokumen (*identifier*) dan nama instansi terkait pembuat dokumen tersebut. Sehingga pola *watermark* yang terbentuk akan menjadi lebih kaya akan variasi corak dan bentuk.

3.1.6 Penggunaan Teknik Masking

Dalam proses penggambaran bentuk pola *watermark* dan area *background* dokumen diperlukan adanya proses deteksi area, sehingga dihasilkan kualitas corak yang serasi antara padanan bentuk *watermark* dan *background* dokumen. Salah satu metode untuk proses deteksi area tersebut adalah dengan menggunakan teknik *Masking*.

Proses masking pada bentuk *watermark* akan memanfaatkan substitusi area bitmap, sehingga bisa mendeteksi area dengan warna tertentu pada bentuk pola *watermark*. Proses *masking* yang dipakai akan menentukan batasan kerapatan warna pada area bitmap, sehingga proses *clipping* pola *watermark* pada *background* garis menjadi selaras dan serasi.

3.1.7 Integrasi Komponen Program ke Sistem Terkait

Struktur implementasi program dibuat sesuai dengan kaidah teknik pemrograman berorientasi obyek, sehingga proses integrasi pada sistem terkait menjadi lebih mudah. Sistem terkait pada implementasi Tugas akhir ini adalah *Framework Devexpress* versi 8.2.2 (*trial*) yang berjalan pada *platform* ASP.NET. Sehingga, salah satu fokus implementasi pada tahap akhir pengerjaan Tugas Akhir ini adalah proses ekspor dokumen menjadi *file* digital. Oleh karena itu, penggunaan *Framework Devexpress* dapat memudahkan proses integrasi ekspor dokumen menjadi file digital.

Xtrareport merupakan salah satu komponen yang terdapat pada *Framework Devexpress* untuk memudahkan proses manajemen dokumen digital (*report* dokumen dan ekspor *file*). Oleh karena itu, proses implementasi Tugas Akhir ini akan mengutamakan pembuatan komponen (*library* program) yang mampu memfasilitasi proses integrasi pembentukan pola *watermark* pada area kanvas *Xtrareport*.

3.2 Perancangan Sistem

Pada sub bab berikut ini akan dibahas tentang struktur perancangan sistem yang meliputi hal-hal sebagai berikut:

1. Konteks sistem yang menjelaskan tentang struktur *input*, proses, dan *output* sistem.
2. Deskripsi tentang garis besar sistem.
3. Algoritma yang digunakan dalam implementasi sistem yang digambarkan dalam struktur diagram alir
4. Perancangan komponen program *Generator Watermark*.

3.2.1 Konteks Sistem

Konteks sistem digunakan untuk menentukan struktur dan format data yang tepat untuk aplikasi atau perangkat lunak sehingga dapat dioperasikan dengan baik dan benar. Terdapat tiga macam struktur yang diperlukan dalam pengoperasian perangkat lunak yaitu adanya input (nilai masukan) yang didapatkan dari pengguna perangkat lunak, struktur proses (algoritma program) yang dibutuhkan dan dihasilkan selama proses eksekusi perangkat lunak dan output (nilai keluaran) yang memberikan hasil proses eksekusi perangkat lunak bagi pengguna yang menjalankannya.

3.2.1.1 Input

Nilai masukan (*input*) yang diterima oleh sistem adalah *identifier* atau nomor dokumen serta teks yang menyatakan nama instansi terkait yang akan menggunakan dokumen tersebut. Nomor dokumen tersebut merupakan gabungan antara huruf dan atau angka.

Tabel 3.1 berikut ini merupakan keterangan tentang struktur *input* pada sistem.

Tabel 3.1 Struktur Nilai Input

No	Nama Input	Tipe Data	Keterangan
1	Nomor dokumen	String	Merupakan *identifier* yang dimiliki oleh tiap dokumen.
2	Nama instansi	String	Nama pengguna (*user*) dokumen digital.

3.2.1.2 Proses

Struktur proses akan meliputi transformasi nomor dokumen dan pembentukan pola-pola *watermark* pada *background* area *file* digital. Berdasarkan nilai masukan yang telah diberikan pada tahap awal, maka proses transformasi dokumen bisa dijalankan dengan memilah nomor dokumen sehingga menjadi nilai-nilai parameter yang berfungsi untuk menghasilkan pola-pola *watermark*. Proses berikutnya adalah fungsi untuk melakukan fungsi *masking* dan *emboss* pada *background* pola *watermark*, dan tahap akhir adalah fungsi ekspor dokumen menjadi file digital.

Tabel 3.2 berikut ini merupakan struktur nilai parameter yang dihasilkan dari proses transformasi nomor dokumen:

Tabel 3.2 Struktur Nilai Proses

No	Nama Proses	Tipe Data	Keterangan
1	Nilai parameter	Integer	Merupakan rangkaian nilai yang berdasarkan segmen-segmen pada nomor dokumen, sehingga menjadi dasar proses menghasilkan pola-pola *watermark*.
2	Nilai Koordinat	Float	Menjadi dasar proses *masking* area *background* dan *embossing* teks nama instansi pada pola *background* garis.

3.2.1.3 Output

Hasil akhir yang diberikan oleh sistem adalah suatu dokumen digital yang memiliki bentuk pola *watermark*. Dokumen tersebut merupakan hasil proses ekspor yang difasilitasi oleh *Framework Devexpress*.

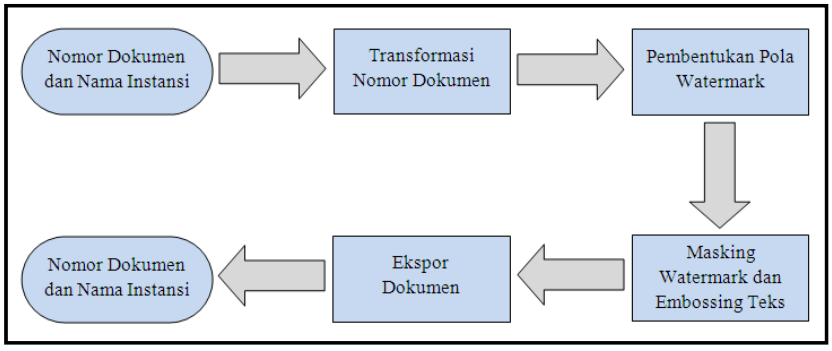
Tabel 3.3 berikut ini merupakan struktur *output* sistem:

Tabel 3.3 Struktur Nilai Output

No	Nama Output	Tipe Data	Keterangan
1	File digital	File (PDF)	File digital yang memiliki bentuk pola-pola watermark

3.2.2 Garis Besar Sistem

Garis besar sistem meliputi beberapa fungsi berikut: fungsi transformasi nomor dokumen, fungsi pembentukan pola *watermark*, fungsi *masking* dan *embossing* teks, dan fungsi ekspor dokumen menjadi file digital. Berikut ini adalah skema garis besar sistem sesuai ilustrasi Gambar 3.1.



Gambar 3.1 Garis Besar Sistem

3.2.2.1 Transformasi Nomor Dokumen

Proses ini merupakan tahapan pemilahan input nomor dokumen menjadi beberapa segmen nilai. Segmen nilai tersebut akan dipisahkan sesuai dengan fungsi yang digunakan pada tiap metode penggambaran *watermark*. Sehingga dapat tercipta cukup banyak variasi nilai parameter pada tiap fungsi dengan penggambaran pola *watermark* yang dinamis.

Berikut ini adalah skema umum tentang proses transformasi nomor dokumen:

1. Pemisahan nomor dokumen menjadi beberapa segmen utama, yakni 4 segmen berdasarkan total karakter yang dihasilkan oleh proses *hashing*.
2. Setiap segmen string tersebut akan terbagi dalam beberapa kategori parameter.
3. Segmen pertama akan ditransformasikan menjadi nilai parameter pertama, kedua, ketiga, dan keempat.
4. Segmen kedua akan ditransformasikan menjadi nilai parameter kelima, dan kedelapan.
5. Segmen ketiga akan ditransformasikan menjadi nilai parameter keenam, dan kesembilan.
6. Segmen keempat akan ditransformasikan menjadi nilai parameter ketujuh, dan kesepuluh.

3.2.2.2 Pembentukan Pola Watermark

Pola *watermark* akan terbentuk berdasarkan nilai input dari segmen-segmen nomor dokumen. Proses pembentukan pola *watermark* akan diolah berdasarkan persamaan parametrik yang memanipulasi fungsi trigonometri.

Persamaan parametrik berikut ini adalah fungsi-fungsi yang akan menjadi dasar dalam pembuatan pola-pola *watermark*, yakni:

$$x = \left(Wx + (Aw * \cos(n * \theta))\right) * (\sin(\theta \pm fa))$$

$$y = \left(Wy + (Aw * \cos(n * \theta))\right) * (\sin(\theta \pm fa))$$

Gambar 3.2 Rumus Dasar I Pola Watermark

$$x = (Rr * \cos(n * \theta)) \pm (Rp * \cos(sv * n * \theta \pm fa))$$

$$y = (Rr * \sin(n * \theta)) \pm (Rp * \sin(sv * n * \theta \pm fa))$$

Gambar 3.3 Rumus Dasar II Pola Watermark

Berikut ini adalah penjelasan gambar rumus persamaan parametrik diatas:

- Konstanta *Wx* menyatakan panjang area bentuk kurva pada area sumbu x.
- Konstanta *Wy* menyatakan panjang area bentuk kurva pada area sumbu y.
- Konstanta *Aw* menyatakan besarnya puncak dan lembah berdasarkan kurva sinus dan cosinus.
- Konstanta *n* menyatakan banyaknya puncak dan lembah berdasarkan kurva sinus dan cosinus.
- Konstanta *fa* menyatakan pergeseran kurva berdasarkan nilai frekuensi yang diberikan pada fungsi tersebut.
- Konstanta *Rr* menyatakan besarnya area panjang dan lebar kurva pada kanvas.
- Konstanta *Rp* menyatakan besarnya ketebalan kurva yang terdiri atas *inside* dan *outside boundary* kurva.

Berikut ini adalah gambaran umum tentang proses pembentukan pola *watermark*:

1. Inisialisasi pengaturan bitmap untuk proses *masking* dan *embossing*.
2. Menggambar *string* nama instansi dan kode nomor dokumen pada *object* bitmap.
3. Pemilahan beberapa fungsi parametrik sesuai dengan inputan nilai nomor dokumen.
4. Fungsi parametrik menerima input nilai-nilai parameter berdasarkan nomor dokumen.
5. Penentuan nilai perbandingan ketebalan garis untuk pembentukan pola watermark pada kanvas dan pada bitmap adalah 1 : 20.
6. Proses pembentukan pola *watermark* pada area kanvas dan bitmap.

3.2.2.3 Proses Masking dan Embossing

Proses *masking* dan *embossing* memanfaatkan *object* bitmap yang telah diproses oleh pembentukan pola *watermark* menjadi nilai koordinat titik-titik pixel. Sehingga dapat ditentukan area penggambaran *background* pola garis pada kanvas dengan metode *masking* dan *embossing*.

Berikut ini adalah alur skema proses *masking* dan *embossing* pada kanvas:

1. Penentuan area koordinat berdasarkan nilai pixel pada *object* bitmap.
2. Proses penggambaran pola garis pada kanvas.
3. Penetapan pengurangan nilai pada penggambaran pola garis karena penentuan deteksi area *string* nama instansi

maupun kode nomor dokumen, yakni minimal setengah dari nilai ketinggian pola garis normal. Sehingga proses *embossing* bisa membentuk pola *string* pada *background* pola garis.

4. Proses deteksi area pola *watermark* berjalan secara bersamaan dengan proses *embossing* teks, sehingga akan berlaku proses penyimpanan nilai koordinat penggambaran sementara dan tidak akan dilakukan proses penggambaran pola garis pada area kanvas.
5. Proses akhir merupakan penggambaran pada *file* digital.

3.2.2.4 Proses Ekspor Dokumen

Proses ekspor dokumen merupakan tahap akhir yang merupakan *output* sistem. Sehingga pola *watermark* yang terbentuk akan diekspor dalam format dokumen digital. Proses ekspor dokumen akan memanfaatkan fitur manajemen *report* pada *Framework Devexpress*. Bagi para pengguna aplikasi, fungsi ini merupakan proses menekan tombol fungsi ekspor yang telah difasilitasi oleh komponen *Xtrareport* pada *Framework Devexpress*.

Berikut ini adalah alur proses ekspor dokumen:

1. Pengguna aplikasi menekan tombol ekspor pada *toolbar Xtrareport* yang tampil di layar.
2. Sistem akan melakukan proses transformasi nomor dokumen berdasarkan *session* yang telah dimiliki oleh sistem.
3. Sistem akan melakukan proses pembentukan pola *watermark*, proses *masking*, serta proses *embossing* pada kanvas.
4. Sistem akan memberikan *output* berupa *file* digital.
5. Disamping itu, pengguna juga bisa melakukan proses *preview* bentuk pola *watermark* pada layar monitor.

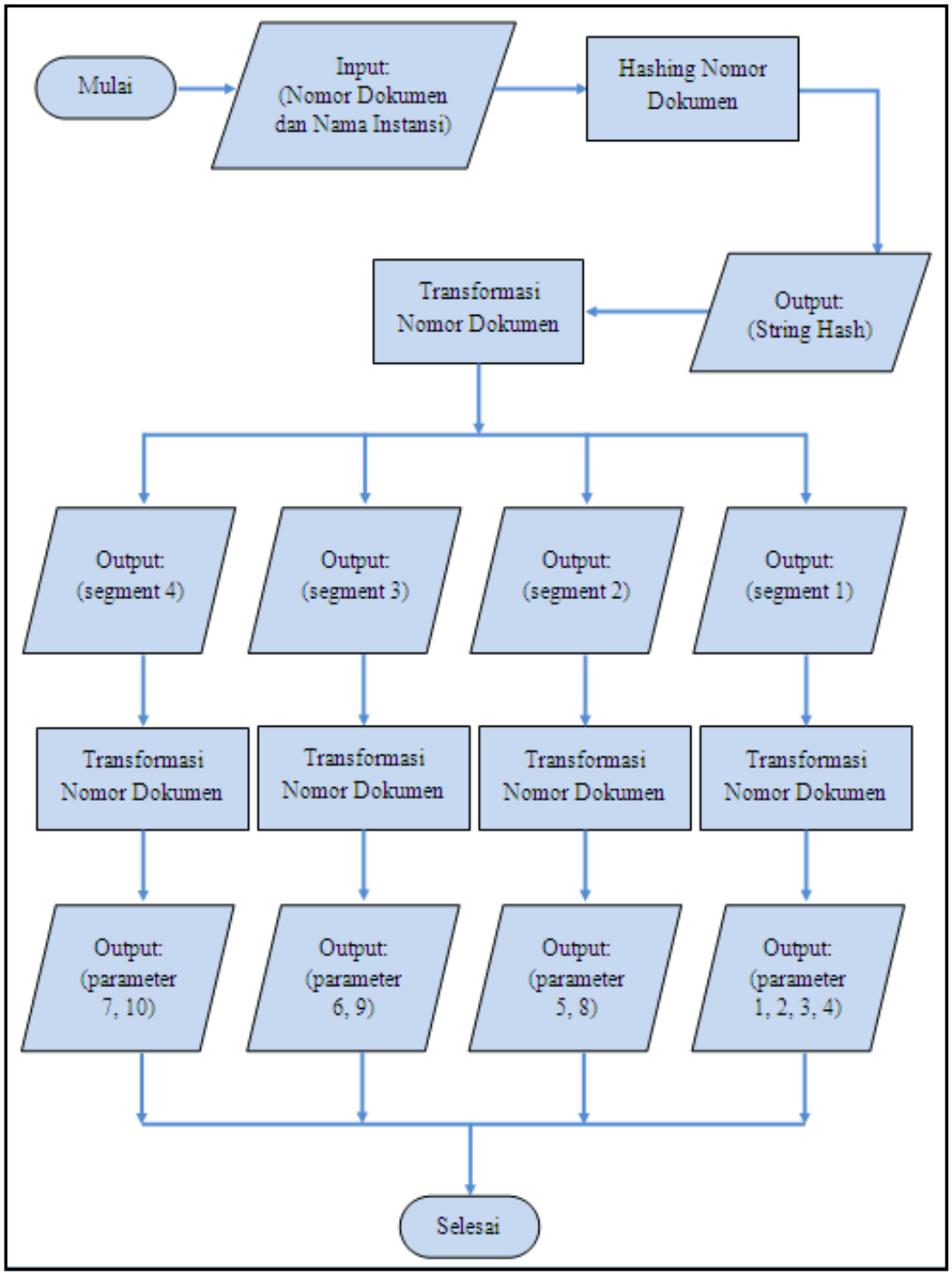
3.2.3 Algoritma dan Diagram Alir

Pada sub bab ini akan dibahas tentang algoritma dan diagram alir yang diimplementasikan pada sistem. Sehingga bisa menjelaskan skema proses-proses yang terjadi pada sistem yang terdiri atas: transformasi nomor dokumen, pembentukan pola *watermark*, proses *masking* dan *embossing*, serta proses ekspor dokumen.

3.2.3.1 Transformasi Nomor Dokumen

Berikut ini adalah penjelasan yang disertai diagram alir (ilustrasi Gambar 3.4) untuk algoritma transformasi nomor dokumen:

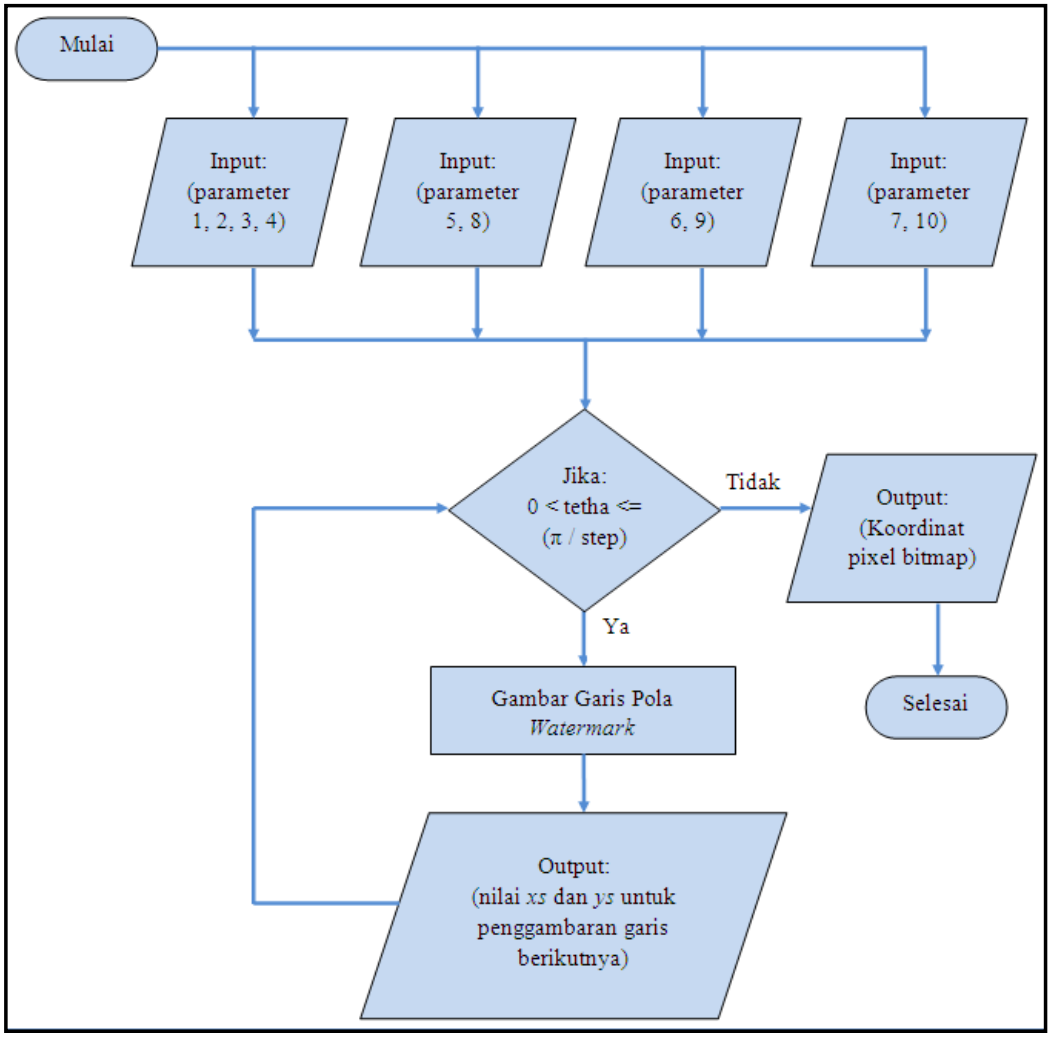
1. Proses *hashing* nomor dokumen dengan algoritma *SHA256Managed* yang terdapat pada *namespace System.Security.Cryptography* pada *.NET Framework*.
2. Penggunaan frase kunci pada algoritma hashing yang ditentukan secara *hardcode*.
3. String *hashing* yang dihasilkan terdiri atas 80-an karakter.
4. String *hashing* yang dihasilkan akan dipisahkan menjadi empat segmen, yang masing-masing terdiri atas 15 karakter.
5. Proses transformasi nomor dokumen menjumlahkan setiap angka (untuk tiap huruf akan ditambahkan dengan nilai yang ditambahkan secara *hardcode*) pada sebuah segmen, sehingga hanya menjadi satu digit nilai *integer*.
6. Segmen pertama akan ditransformasikan menjadi nilai parameter pertama, kedua, ketiga, dan keempat.
7. Segmen kedua akan ditransformasikan menjadi nilai parameter kelima, dan kedelapan.
8. Segmen ketiga akan ditransformasikan menjadi nilai parameter keenam, dan kesembilan.
9. Segmen keempat akan ditransformasikan menjadi nilai parameter ketujuh, dan kesepuluh.



Gambar 3.4 Diagram Alir Transformasi Nomor Dokumen

3.2.3.2 Pembentukan Pola Watermark

Berikut ini adalah diagram alir (ilustasi Gambar 3.5) yang disertai penjelasan untuk proses pembentukan pola *watermark*:



Gambar 3.5 Diagram Alir Pembentukan Pola Watermark

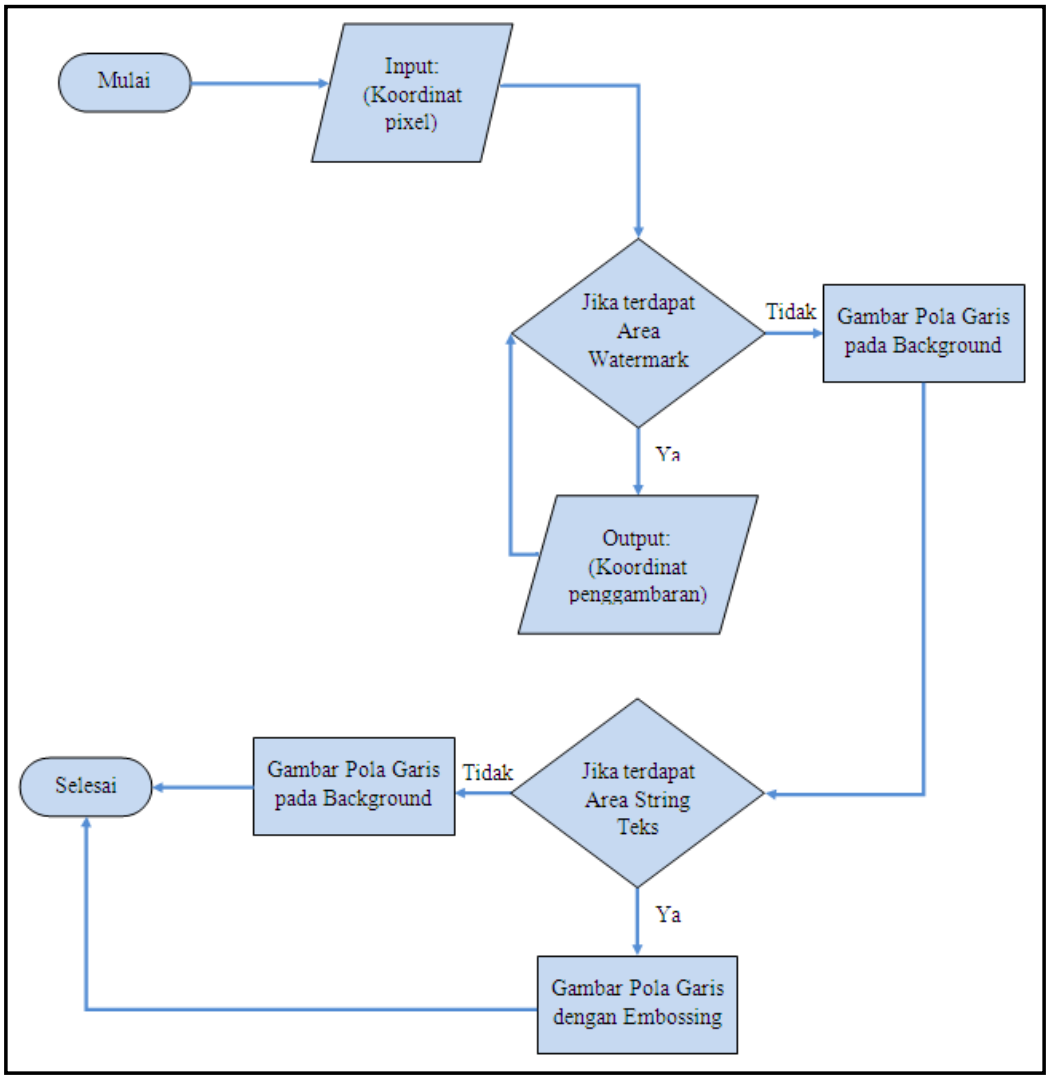
1. Input nilai parameter yang dihasilkan oleh proses transformasi nomor dokumen akan diolah sesuai dengan penentuan pola pembentukan *watermark* pada tiap fungsi.

2. Penggambaran *watermark* berada pada *looping* nilai *tetha*. Nilai variabel *step* akan mempengaruhi banyaknya putaran fase yang digunakan. Fungsi-fungsi parametrik yang digunakan merupakan modifikasi beberapa fungsi dasar kurva *Lissajous, Hypotrochoid* dan *Epitrochoid*.
3. Proses pembentukan pola *watermark* akan disertai proses *drawing* pada *object* bitmap untuk menjadi input proses *masking* dan *embossing*.

3.2.3.3 Proses Masking dan Embossing

Berikut ini adalah penjelasan yang disertai diagram alir (ilustasi Gambar 3.6) untuk proses *masking* dan *embossing*:

1. Deteksi area *object* bitmap berdasarkan nilai warna RGB tiap *pixel*.
2. Proses deteksi *pixel* berada pada area kanvas dengan nilai *padding* sebesar 20.
3. Penentuan area kurva *Lissajous* pada proses perulangan deteksi area *pixel* dengan metode *adaptive step*.
4. Apabila pada langkah pengecekan perulangan pertama atau kedua terdapat area pixel bitmap yang berbeda warna, maka *step* perulangan pada sumbu *x* akan diperkecil menjadi satu.
5. Untuk proses deteksi yang tidak menunjukkan area kurva *Lissajous*, nilai proses perulangan pada sumbu *x* akan dikembalikan pada *step* normal sebesar lima.
6. Apabila tidak terdapat area kurva Lissajous, maka akan dilakukan penggambaran area *background*.
7. Proses penggambaran area *background* merupakan pola kurva sinus pada kanvas.
8. Penentuan area teks suatu instansi berdasarkan deteksi area pixel akan mengurangi nilai koordinat *y*, sehingga menimbulkan efek *emboss* pada area *background*.



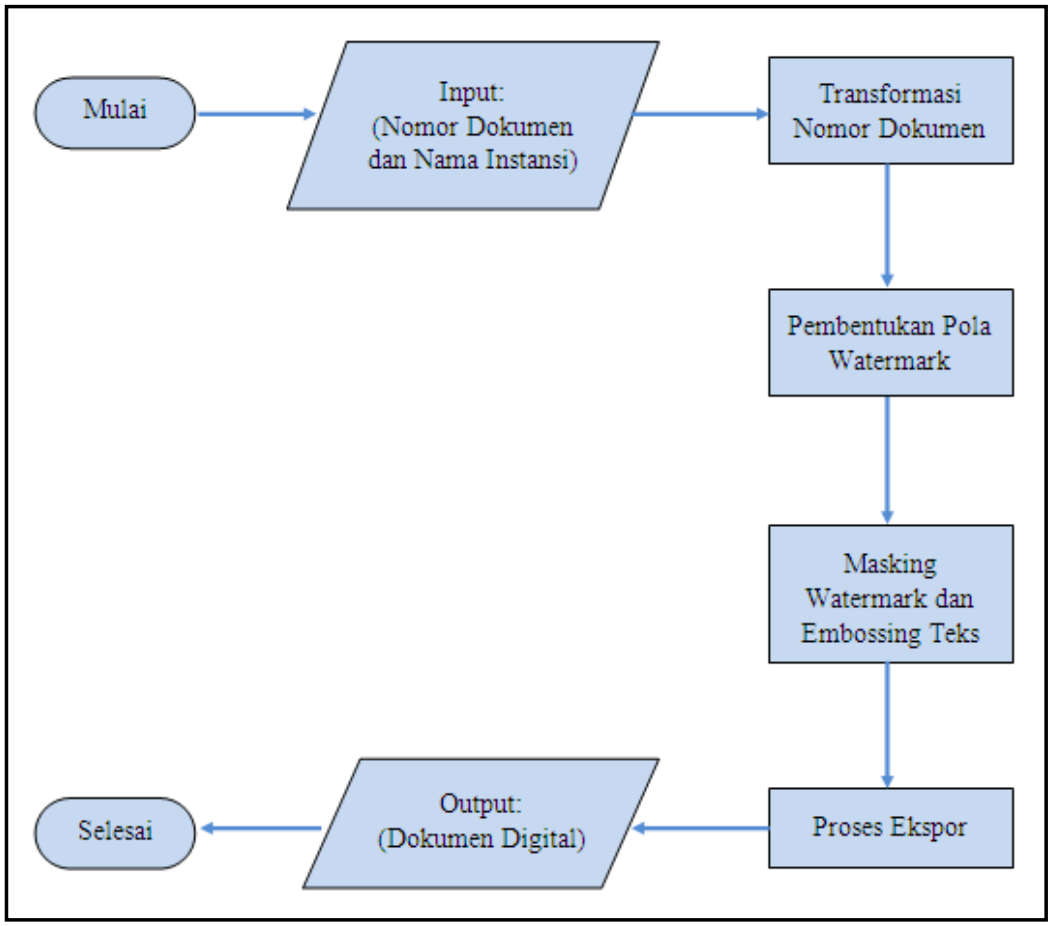
Gambar 3.6 Diagram Alir Proses Masking dan Embossing

3.2.3.4 Proses Ekspor Dokumen

Berikut ini adalah penjelasan yang disertai diagram alir (ilustasi Gambar 3.7) untuk proses ekspor dokumen:

1. Pengguna aplikasi menekan tombol ekspor pada *toolbar Xtrareport* yang tampil di layar.

2. Sistem melakukan proses transformasi nomor dokumen berdasarkan *session* yang telah dimiliki.
3. Sistem akan melakukan proses pembentukan pola *watermark*, proses *masking*, serta proses *embossing* pada kanvas.
4. Sistem akan memberikan output berupa *file* digital.



Gambar 3.7 Diagram Alir Proses Ekspor Dokumen

3.2.4 Perancangan Komponen

Implementasi kode program pada sistem ini akan melibatkan komponen *Xtrareport* pada *Framework Devexpress*, sehingga perlu diperhatikan beberapa hal sebagai berikut:

1. Pengenalan karakteristik kanvas *Xtrareport*, yang terdiri atas beberapa nilai properti yang bisa dimanfaatkan untuk proses penggambaran pola *watermark*.
    a. Penentuan area kanvas yang mewakili ukuran kertas standar A4, yakni melalui nilai properti: *PageWidth*, *PageHeight*, dan *PaperKind*.
    b. Struktur properti margin bernilai nol, agar tidak membatasi area kanvas saat dilakukan proses ekspor dokumen.
2. Proses menggambar pola *watermark* berada diatas kanvas *Xtrareport*. Kanvas tersebut memiliki 3 *layer*, yakni: *PageHeader*, *Detail*, dan *Pagefooter*.
    a. Penambahan komponen *XrPictureBox* pada area *PageHeader*, sehingga bisa melakukan proses *override event OnPaint* yang dimiliki oleh komponen tersebut.
    b. Penentuan *PageHeader* sebagai area penempatan *XrPictureBox* dikarenakan proses *rendering* area kanvas diawali oleh komponen *PageHeader*, sehingga penggambaran pola *watermark* akan berada pada *layer* paling bawah pada kanvas *Xtrareport*.
    c. *Layer* pada komponen *Detail* dan *PageHeader* akan digambar pada saat terakhir proses *rendering*, sehingga memungkinkan semua teks yang terdapat pada dokumen berada pada layer paling atas (*on top view*) pada kanvas *Xtrareport*.

3. *Object* yang digunakan untuk proses penggambaran adalah *Interface IGraphics*.
    a. *Interface IGraphics* merupakan *Class* untuk proses menggambar yang disediakan oleh *Framework Devexpress*.
    b. *Object IGraphics* bisa digunakan dari *event* komponen *XRControl*. Namun penggunaan yang lebih spesifik sesuai fungsi *control* adalah komponen *XrPictureBox*.
    c. *Object Igraphics* hanya memiliki sedikit fungsi untuk proses menggambar. Namun, proses penggambaran pola *watermark* hanya membutuhkan fungsi *DrawLine* dan *DrawString*.
    d. Fungsi *DrawLine* digunakan untuk proses penggambaran pola *watermark*, selain itu penggunaan fungsi ini ditujukan untuk mempermudah proses pewarnaan dengan metode gradasi.
    e. Fungsi *DrawString* akan digunakan untuk proses *overwrite* semua teks yang berada pada *layer PageHeader*, sehingga semua teks pada area tersebut akan berada pada *layer* paling atas (*on top view*) pada kanvas *Xtrareport*.
4. Penentuan formula untuk fungsi-fungsi parametrik yang menghasilkan pola-pola *watermark*.
    a. Fungsi-fungsi parametrik yang digunakan dalam proses implementasi akan diturunkan dari *interface ILissajous*, sehingga lebih mudah untuk digunakan pada masa mendatang.
    b. Formula trigonometri yang berdasarkan konsep persamaan parametrik akan dimodifikasi sedemikian rupa, sehingga bisa memenuhi standar nilai estetika.

3.3 Implementasi Sistem

Pada sub bab berikut ini akan dijelaskan tentang proses implementasi pada sistem, yang meliputi tentang: lingkungan implementasi, struktur komponen program, dan implementasi kode program.

3.3.1 Lingkungan Implementasi

Sistem diimplementasikan pada spesifikasi lingkungan yang dapat diuraikan sebagai berikut:

- Perangkat keras
  Sistem diimplementasikan pada perangkat *personal computer*, dengan tipe prosesor Intel Pentium IV Core™ 2 Duo T5450 @ 1.66 Ghz dan kapasitas memori adalah sebesar 2.5 Gb.
- Perangkat lunak
  Sistem diimplementasikan diatas platform *Windows XP Service Pack 3*, dengan menggunakan IDE Visual Studio 2008. Disamping itu juga terdapat adanya komponen *Framework Devexpress* sebagai *library* untuk pengembangan sistem.

3.3.2 Struktur Komponen Program

Implementasi komponen program akan dibuat berdasarkan struktur yang diuraikan pada tahap perancangan komponen. Berikut ini adalah detil struktur yang berdasarkan skema masing-masing Class, yakni:

1. **Class Docid**
    a. Merupakan struktur *Class* yang akan melakukan proses transformasi nomor dokumen.
    b. Terdiri atas struktur konstruktor *Docid* dan fungsi *Segments* dan *Params*.
    c. Input berupa nomor dokumen dan menghasilkan output berupa nilai-nilai parameter untuk pembentukan pola *watermark*.
2. **Class Wmcon**
    a. Merupakan struktur *Class* yang mendefinisikan semua konfigurasi sistem pembentukan pola *watermark*.
    b. Terdiri atas beberapa struktur fungsi *static* yang mengatur konfigurasi pola *watermark* pada area kanvas, yakni: struktur *CanvasSize*, definisi ketebalan *Class Pen*, definisi tetapan sudut dalam radian, dan definisi untuk letak area *preview watermark* (UpperLeft, UpperRight, LowerLeft, LowerRight, dan Middle).
3. **Class ILissajous**
    a. Merupakan definisi *Class Interface* yang hanya menyediakan struktur fungsi tanpa implementasi.
    b. Terdiri atas struktur fungsi yang mengatur penentuan titik berdasarkan fungsi-fungsi parametrik (GetPoints) dan proses penggambaran pola pada area kanvas (LissajousPatterns).
4. **Class Waterman**
    a. Merupakan struktur *Class* yang mengolah semua proses menggambar pola *watermark* pada area kanvas *Xtrareport*.
    b. Terdiri atas struktur fungsi-fungsi untuk proses penggambaran pola (implementasi dari *interface ILissajous*) serta proses *masking* pola *watermark* dan *embossing* teks.

c. Input berupa *object XrReport*, *object Igraphics*, dan nomor dokumen dan menghasilkan *output* berupa pola-pola *watermark* yang digambar diatas kanvas *object XrReport*.

3.3.3 Implementasi Kode Program

Pada bagian berikut ini akan digambarkan struktur implementasi kode program pada tiap *class* yang dilengkapi penjelasan tentang proses-proses yang terjadi didalamnya.

3.3.3.1 Class Docid

Berikut ini adalah ilustrasi (pada Gambar 3.8) dan penjelasan potongan kode program tentang konstruktor *class Docid*.

```
/// <summary>
/// Konstruktor Class Docid
/// </summary>
/// <param name="docid">nomor dokumen</param>
public Docid(String docid)
{
    int lensegment = 4;
    this.segments = new String[lensegment];
    this.Strdocid = this.ComputeHash( docid );

    int lenhash = this.Strdocid.Length / lensegment;
    for (int i = 0; i < lensegment; i++)
    {
        this.segments[i] =
            this.Strdocid.Substring(i * lenhash, lenhash);
    }
}
```

Gambar 3.8 Konstruktor Class Docid

1. Konstruktor *Docid* berperan untuk proses inisialisasi nomor dokumen sehingga menjadi *string hash* yang akan diolah menjadi nilai-nilai parameter.
2. Konstruktor *Docid* menerima parameter dengan tipe data *string* (*String docid*).
3. Nilai parameter (*docid*) diproses oleh fungsi *ComputeHash*, sehingga diperoleh nilai *string hash*.
4. *String hash* dikelompokkan dalam empat bagian segmen. Segmen tersebut tersimpan pada *array segments* yang terdiri atas 4 elemen (*lensegment*).
5. *String hash* dikelompokkan dengan memakai perulangan sesuai dengan besarnya nilai *array segments*, sehingga tiap segmen akan memiliki bagian dari *string hash* dengan memanfaatkan fungsi *Substring*.

Berikut ini adalah penjelasan ilustrasi potongan kode program berdasarkan Gambar 3.9 tentang fungsi *ComputeHash*:

1. Fungsi *ComputeHash* berperan dalam proses mengubah nomor dokumen menjadi kombinasi karakter yang berdasarkan algoritma *SHA256Managed*.
2. Fungsi *ComputeHash* menerima parameter dengan tipe data *string* (*String Plaintext*).
3. Nomor dokumen diproses menjadi *bytes* dengan memakai fungsi *Encoding.UTF8.GetBytes*.
4. *SaltBytes* digunakan sebagai *key* (*noise*) yang akan digabungkan pada saat proses *hashing* oleh fungsi *hash.ComputeHash*.
5. Nomor dokumen dan *SaltBytes* yang telah *dihash* akan disimpan dalam *array hashWithSaltBytes*.
6. *Array hashWithSaltBytes* dikonversi dalam tipe *string* untuk dikembalikan sebagai *return value* fungsi *ComputeHash*.

```
/// <summary>
/// Fungsi untuk mengembalikan nilai hash terhadap nomor
dokumen
/// </summary>
/// <param name="plainText"></param>
/// <returns>string hash</returns>
private string ComputeHash(string plainText)
{
    /// Definisikan Byte of Array yang akan menyimpan
    /// Byte-byte noise(AbCdEfVwXyZ) Proses Enkripsi
    byte[] saltBytes =
                Encoding.UTF8.GetBytes("AbCdEfGVwXyZ");

    /// Konversi plainText dalam bentuk Byte of Array.
    byte[] plainTextBytes =
                Encoding.UTF8.GetBytes(plainText);

    /// Definisi array, berisi Byte plainText and Salt
    byte[] plainTextWithSaltBytes =
            new byte[plainTextBytes.Length +
                                    saltBytes.Length];

    /// Mengisi Byte hasil dengan nilai plainText
    for (int i = 0; i < plainTextBytes.Length; i++)
         plainTextWithSaltBytes[i] = plainTextBytes[i];

    /// Menambah Byte hasil dengan nilai saltByte
    for (int i = 0; i < saltBytes.Length; i++)
         plainTextWithSaltBytes[plainTextBytes.Length + i]
                                        = saltBytes[i];

    /// Definisi hash dengan Algoritma SHA256Managed
    HashAlgorithm hash = new SHA256Managed(); ;

    /// Hasilkan nilai hash
    byte[] hashBytes =
         hash.ComputeHash(plainTextWithSaltBytes);

    /// Definisi array untuk mengisi Byte hash
    byte[] hashWithSaltBytes =
         new byte[hashBytes.Length + saltBytes.Length];

    /// Masukkan hash Byte dalam array hasil
    for (int i = 0; i < hashBytes.Length; i++)
        hashWithSaltBytes[i] = hashBytes[i];
```

```
    /// Tambahkan saltByte pada array hasil
    for (int i = 0; i < saltBytes.Length; i++)
        hashWithSaltBytes[hashBytes.Length + i] =
                                            saltBytes[i];

    // Konversi nilai hasil menjadi base64-encoded string
    return Convert.ToBase64String(hashWithSaltBytes);
}
```

Gambar 3.9 Fungsi ComputeHash

Nomor dokumen dikelompokkan dalam empat segmen (bagian) *string*, sehingga dapat ditentukan nilai-nilai parameter. Nilai-nilai parameter tersebut tercakup dalam struktur *property class Docid*.

Berikut ini adalah ilustrasi potongan kode program (sesuai Gambar 3.10) yang mengubah tiap bagian segmen menjadi nilai parameter dengan tipe *integer*:

```
/// <summary>
/// Fungsi untuk mengolah Segment menjadi nomor parameter
/// </summary>
/// <param name="str">string yang akan diubah menjadi
nomor</param>
/// <param name="value_to_add">nomor untuk penambahan
nilai</param>
/// <returns>nilai dengan tipe int</returns>
private int GetSegmentNumber(String str, int value_to_add)
{
    int param = 0;
    while (str.Length > 1)
    {
        param = 0;
        foreach (char c in str.ToCharArray())
        {
            if (Char.IsNumber(c))
                param += Int32.Parse(c.ToString());
            else
                param += value_to_add;
```

```
            }
            str = param.ToString();
        }
        return param;
    }
```

Gambar 3.10 Fungsi GetSegmentNumber

Berikut ini adalah penjelasan ilustrasi potongan kode program berdasarkan Gambar 3.10 tentang fungsi *GetSegmentNumber*:

1. Fungsi *GetSegmentNumber* berperan dalam proses transformasi potongan *string hash* menjadi nilai *integer* yang hanya terdiri atas satu digit.
2. Fungsi *GetSegmentNumber* menerima parameter berupa *string* (*str*) dan nilai (*value_to_add*) yang bertipe *integer*.
3. Proses akan diawali oleh perulangan yang berdasarkan nilai *length str*.
4. Variabel *param* akan dijumlahkan dengan setiap karakter yang bertipe numerik. Apabila karakter pada string *str* bertipe huruf, maka variable *param* akan dijumlahkan dengan nilai yang terdapat pada *value_to_add*.
5. Proses perulangan akan berhenti apabila nilai *integer* pada variabel param hanya terdiri atas satu digit.
6. Fungsi *GetSegmentNumber* memberikan nilai *integer* variabel *param* sebagai *return value*.

3.3.3.2 Class WmCon

Berikut ini adalah ilustrasi (pada Gambar 3.11) dan penjelasan potongan kode program tentang *class WmCon*:

```
public class Wmcon
{
   public static Size CanvasSize = new Size(2525, 3520);
   public static SizeF CanvasSizeF =
                                 new SizeF(2525f, 3520f);

   public static float PenSize = 0.5f;
   public static int PixelPadding = 20;

   public static Double MaxAngleLiss = 2 * Math.PI;
   public static Double MaxAngleLine = 5 * Math.PI;

   /// untuk definisi: 1/20 derajat
   public static Double RadianAngle = Math.PI / 3600;
   /// untuk definisi: 45   derajat
   public static Double BigPieAngle = Math.PI / 4;

   public Wmcon() { }

   /// Beberapa struktur property
}
```

Gambar 3.11 Struktur Class WmCon

1. *Class WmCon* merupakan struktur yang berisi tentang konfigurasi nilai (dengan *modifier public static*) yang digunakan pada implementasi *Class Waterman*.
2. Variabel *CanvasSize* menyatakan nilai area kanvas dengan standar kertas A4.
3. Variabel *PenSize* merupakan ketebalan pena untuk menggambar kurva *Lissajous*.

4. Variabel *PixelPadding* menyatakan nilai *padding* yang digunakan untuk proses deteksi area kurva dan teks pada *object bitmap*.
5. Variabel *MaxAngleLiss* dan *MaxAngleLine* menyatakan besarnya periode putaran berdasarkan standar *radian*.
6. Variabel *RadianAngle* dan *BigPieAngle* menyatakan besarnya nilai pergeseran sudut dalam standar *radian*.

3.3.3.3 Class ILissajous

Class ini merupakan sebuah *interface*, sehingga struktur *class ILissajous* hanya berisi definisi prosedur yang akan diimplementasikan oleh *class Waterman*. Pada ilustrasi Gambar 3.12 terdapat 2 pokok prosedur yang terdapat pada *interface ILissajous*, yakni: *DrawILissajous* yang akan menggambarkan kurva *Lissajous* pada kanvas serta *LissajousPatternVx* yang merupakan prosedur yang berisi perumusan fungsi-fungsi trigonometri berdasarkan versi pola yang diinginkan (*Vx*).

```
// Definisi interface ILissajous
public interface ILissajous
{
   void DrawILissajous(IGraphics Oigraph,
      Graphics Ographview, Graphics Ographtemp,
      Double tetha, float xs, float ys, float x, float y);

   void LissajousPatternV1();
   void LissajousPatternV2();
   void LissajousPatternV3();
}
```

Gambar 3.12 Struktur Interface ILissajous

3.3.3.4 Class Waterman

Pada sub bab berikut ini akan dibahas tentang potongan kode program untuk proses *masking*, *embossing*, dan penggambaran pola *watermark*.

Berikut ini adalah ilustrasi (pada Gambar 3.13) dan penjelasan potongan kode program tentang fungsi *DrawLines* yang mencakup proses *masking* dan *embossing*:

```
/// <summary>
/// Prosedur untuk membuat garis pada background watermark
/// dari batas atas hingga bawah suatu halaman
/// (Standart kertas A4).
/// Type: void
/// </summary>
private void DrawLines()
{
   // Deklarasi untuk drawing line dan WmText
   float y = 0, yt = 0, xs = 0, ys = 0;
   // Deklarasi untuk proses deteksi Pixel
   int xpix = 0, ypix = 0;

   double tetha =
   Wmcon.MaxAngleLine / Wmcon.CanvasSizeF.Width;

   int    counterheight = 8;
   double counterwidth  = 5.0;
   int PixelBoundHeight =
            Wmcon.CanvasSize.Height - Wmcon.PixelPadding;
   int PixelBoundWidth  =
            Wmcon.CanvasSize.Width - Wmcon.PixelPadding;
   int xtemp = 0, ytemp = 0;

   bool iscurve = false;
   Pen LinePen  = new Pen(Color.SkyBlue, Wmcon.PenSize);

   for (int i = 0; i < Wmcon.CanvasSize.Height;
                                       i += counterheight)
   {
```

```
for (double x = 0;
            x < (double)Wmcon.CanvasSizeF.Width;
            x += counterwidth)
{
   y = (float)(Math.Sin(x * tetha) * 10.0 + i -
                                              10.0);

   xpix = (int)(x);
   ypix = (int)(y);

   yt = 0;
   if ( Wmcon.PixelPadding < ypix &&
        ypix < PixelBoundHeight &&
        Wmcon.PixelPadding < xpix &&
        xpix < PixelBoundWidth )
   {


         for (int stepclip = 1; stepclip <= 2;
                                       stepclip++)
         {
            xtemp = (int)(x +
                        (stepclip * counterwidth));
            ytemp = (int)(Math.Sin(xtemp * tetha) *
                        10.0 + i - 10.0);

            if (this.OBitmapTemp.GetPixel(xpix,
                                          ypix) !=
               Color.FromArgb(255, 255, 255, 255) &&
               this.OBitmapTemp.GetPixel(xpix, ypix)
               != Color.FromArgb(255, 255, 0, 0) )
            {
               counterwidth = 1.0;
            }
            else
            {
               counterwidth = 5.0;
            }
         }

         if (this.OBitmapTemp.GetPixel(xpix, ypix)
               == Color.FromArgb(255, 255, 0, 0))
            yt = -5.5f;

         iscurve = false;
```

```
                if ((this.OBitmapTemp.GetPixel(xpix, ypix)
                   != Color.FromArgb(255, 255, 255, 255) &&
                   this.OBitmapTemp.GetPixel(xpix, ypix) !=
                   Color.FromArgb(255, 255, 0, 0)))
                {
                   iscurve = true;
                }
            }

            if (x > 0)
            {
                if (!iscurve)
                {
                   if (this.OIGraph != null)
                   {
                      this.OIGraph.DrawLine(LinePen,
                            xs, ys, (float)x, (y + yt));
                   }

                this.OGraphView.DrawLine(LinePen, xs, ys,
                                        (float)x, (y + yt));
                }
             }
             xs = (float)x;
             ys = (y + yt);
        }
    }
}
```

Gambar 3.13 Struktur Proses Masking dan Embossing Teks

1. Pada proses ini terdapat beberapa definisi variabel yang digunakan pada titik koordinat kanvas (*x*, *y*, *xs*, *ys*, *yt*, *xtemp*, *ytemp*) dan pada koordinat *pixel* (*xpix*, *ypix*).
2. Variabel *tetha* menyatakan besarnya pergeseran sudut dalam *radian* untuk proses menggambar kurva sinus.
3. Variabel *PixelBoundHeight* dan *PixelBoundWidth* menyatakan area *padding* pada kanvas berdasarkan koordinat *object* bitmap.

4. Variabel *counterwidth* dan *counterheight* menyatakan nilai untuk penambahan step terhadap sumbu *x* dan *y* untuk penggambaran garis kurva sinus.
5. Variabel *iscurve* menyatakan area yang termasuk dalam kurva Lissajous.
6. Variabel *LinePen* menyatakan inisialisasi *object* pena untuk menggambar kurva sinus sebagai *background*.
7. Proses *masking* dan *embossing* terdapat pada lingkup perulangan untuk menggambar garis kurva sinus sepanjang area kanvas (ukuran kertas A4).
8. Deteksi area *object* bitmap berdasarkan nilai warna RGB tiap *pixel* (berdasarkan nilai *xpix* dan *ypix*).
9. Proses deteksi *pixel* dimulai pada area kanvas yang telah diberi nilai *padding* sesuai definisi variabel *PixelBoundHeight* dan *PixelBoundWidth*.
10. Penentuan area kurva *Lissajous* pada proses perulangan deteksi area *pixel* dengan metode *adaptive step* (proses perulangan dengan *counter* variabel *stepclip*).
11. Apabila pada langkah pengecekan perulangan pertama atau kedua terdapat area *pixel* bitmap yang berbeda warna, maka *step* perulangan pada sumbu *x* akan diperkecil menjadi satu.
12. Untuk proses deteksi yang tidak menunjukkan area kurva *Lissajous*, nilai proses perulangan pada sumbu *x* akan dikembalikan pada *step* normal sebesar lima.
13. Apabila tidak terdapat area kurva *Lissajous*, maka akan dilakukan penggambaran area *background* oleh fungsi *OIgraph.DrawLine*.
14. Penentuan area teks suatu instansi berdasarkan deteksi area pixel akan mengurangi nilai koordinat *y* (tersimpan dalam nilai variabel *yt*), sehingga menimbulkan efek *emboss* pada area *background*.

Berikut ini merupakan ilustrasi (pada Gambar 3.14) dan penjelasan tentang proses pembentukan pola *watermark* pada kanvas dengan prosedur *DrawILissajous* dan *LissajousPatternV1* (merupakan implementasi prosedur *interface ILissajous*).

```
/// <summary>
/// Struktur proses menggambar pola Watermark
/// </summary>
/// <param name="Oigraph"></param>
/// <param name="Ographview"></param>
/// <param name="Ographtemp"></param>
/// <param name="tetha"></param>
/// <param name="xs"></param>
/// <param name="ys"></param>
/// <param name="x"></param>
/// <param name="y"></param>
public void DrawILissajous(IGraphics Oigraph,
      Graphics Ographview, Graphics Ographtemp,
      Double tetha, float xs, float ys, float x, float y)
{
    int p1 = (this.ODocid.Param11 + 1);
    int p2 = (this.ODocid.Param12 + 1);
    int p3 = (this.ODocid.Param13 + 1);

    int pa1 = Math.Abs(this.ODocid.Param21 - 1);
    int pa2 = Math.Abs(this.ODocid.Param31 - 2);
    int pa3 = Math.Abs(this.ODocid.Param41 - 3);

    if (tetha > 0)
    {
        int s1 = 200 + Convert.ToInt32((pa1 * 6) *
                        Math.Sin(tetha * pa3 + pa2));
        int s2 = 200 + Convert.ToInt32((pa2 * 5) *
                        Math.Sin(tetha * pa1 + pa3));
        int s3 = 200 + Convert.ToInt32((pa3 * 7) *
                        Math.Sin(tetha * pa2 + pa1));

        Pen OPen = new Pen(Color.FromArgb(200,
                            s1, s2, s3), Wmcon.PenSize);

        if (Oigraph != null)
            Oigraph.DrawLine(OPen,
                            xs, ys, (float)x, (float)y);
```

```
        Ographview.DrawLine(OPen,
                            xs, ys, (float)x, (float)y);
        Ographtemp.DrawLine(OPenBitmap,
                            xs, ys, (float)x, (float)y);
    }
}


/// <summary>
/// Struktur Proses mengolah titik-titik untuk pembentukan
pola watermark
/// </summary>
public void LissajousPatternV1()
{
   double x, y;
   float xs = 0, ys = 0;

   int p1 = ((this.ODocid.Param11 + 1) % 5) + 6;
   int p2 = ((this.ODocid.Param12 + 1) % 4) * 10;
   int p3 = ((this.ODocid.Param13 + 1) % 5) + 6;
   int p4 = this.ODocid.Param14 + 15;
   double pf = this.ODocid.Param14 / 3.15;
   if (p4 < 5) pf = -pf;

   for (int layer = 0; layer < 20; layer+=10)
   {
      for (double tetha = 0; tetha < Wmcon.MaxAngleLiss;
                            tetha += (Math.PI / 1800))
      {
         x = this.CX + (650 - layer + (p4 * Math.Cos(p1 *
tetha))) * (Math.Sin(tetha - (3.15 + pf)));
         y = this.CY + (650 - layer + (p4 * Math.Cos(p1 *
tetha))) * (Math.Cos(tetha - (3.15 + pf)));

         this.DrawILissajous(this.OIGraph,
            this.OGraphView, this.OGraphTemp,
            tetha, xs, ys, (float)x, (float)y);
                  xs = (float)x;
                  ys = (float)y;
      }
   }

   for (double tetha = 0; tetha < Wmcon.MaxAngleLiss;
tetha += (Math.PI / 2700))
   {
```

```
      x = this.CX + (600 + (p4 * Math.Cos(p1 * tetha))) *
(Math.Sin(tetha - (3.15 + pf))) - (40 * Math.Sin(300 *
tetha));
      y = this.CY + (600 + (p4 * Math.Cos(p1 * tetha))) *
(Math.Cos(tetha - (3.15 + pf))) + (40 * Math.Cos(300 *
tetha));

      this.DrawILissajous(this.OIGraph, this.OGraphView,
         this.OGraphTemp, tetha, xs, ys,
         (float)x, (float)y);
                xs = (float)x;
                ys = (float)y;
   }

   this.OPenBitmap.Width = 20f;

   for (int layer = 0; layer < 45 + p2; layer += 8)
   {
      for (double tetha = 0; tetha < Wmcon.MaxAngleLiss;
tetha += (Math.PI / 2700))
      {
         x = this.CX + (255 - layer + (20 * Math.Cos(p3 *
tetha))) * (Math.Sin(tetha - (3.15 + pf)));
         y = this.CY + (255 - layer + (20 * Math.Cos(p3 *
tetha))) * (Math.Cos(tetha - (3.15 + pf)));

         this.DrawILissajous(this.OIGraph,
               this.OGraphView, this.OGraphTemp,
               tetha, xs, ys, (float)x, (float)y);

         xs = (float)x;
         ys = (float)y;
      }
   }


   double Rr = 426 + 0.2;
   double Rp = 165 - 0.2;
   double mul = 1.5;
   double s = (Rr) / 0.2;
   for (double tetha = 0; tetha < Wmcon.MaxAngleLiss;
tetha += (Math.PI / 600))
   {
      x = this.CX + (Rr * Math.Cos(mul * tetha)) + (Rp *
Math.Cos(mul * s * tetha - (3.15 + pf)));
```

```
      y = this.CY + (Rr * Math.Sin(mul * tetha)) - (Rp *
Math.Sin(mul * s * tetha - (3.15 + pf)));

      this.DrawILissajous(this.OIGraph,
                this.OGraphView, this.OGraphTemp,
                tetha, xs, ys, (float)x, (float)y);
                 xs = (float)x;
                 ys = (float)y;
    }
}
```

Gambar 3.14 Struktur Penggambaran Pola Watermark

Uraian berikut merupakan penjelasan prosedur *DrawILissajous* yang terdapat pada Gambar 3.14:

1. Prosedur DrawILissajous menerima parameter beberapa *object graphic*, *tetha*, dan nilai koordinat (*xs*, *ys*, *x*, *y*).
2. Variabel *object graphic* memungkinkan proses penggambaran garis kurva *Lissajous* pada kanvas maupun *object* bitmap dengan menggunakan prosedur *DrawLine*.
3. Variabel *ODocid* menyatakan *object Docid* yang digunakan untuk pemilahan nilai-nilai parameter.
4. Nilai-nilai parameter digunakan untuk mengolah pilihan warna pada proses penggambaran garis kurva *Lissajous* oleh variabel *OPen*, sehingga menghasilkan variasi warna pada kurva.
5. Nilai warna RGB akan dihasilkan oleh fungsi sinus, sehingga diperlukan konversi menjadi *integer* dengan fungsi *Convert.ToInt32*.

Uraian berikut ini merupakan penjelasan terhadap prosedur *LissajousPatternV1* yang terdapat pada Gambar 3.14:

1. Prosedur *LissajousPatternV1* menggunakan variabel global *ODocid* untuk menerima nilai-nilai parameter, sehingga dapat digunakan pada proses penentuan titik-titik kurva berdasarkan rumus-rumus trigonometri.
2. Variabel *xs*, *ys*, *x*, dan *y* menyatakan nilai koordinat pada area kanvas.
3. Variabel *CX* dan *CY* menyatakan nilai koordinat yang menjadi acuan sumbu pusat penggambaran bentuk kurva.
4. Beberapa struktur kode pada *LissajousPatternV1* merupakan proses manipulasi fungsi parametrik berdasarkan rumus dasar yang telah dibahas pada sub bab garis besar sistem.
5. Pemilihan nilai parameter yang digunakan pada fungsi parametrik sangat mempengaruhi hasil pola bentuk gambar *watermark*.
6. Variabel *OpenBitmap* digunakan untuk mengubah ketebalan bentuk pena yang menggambar pada *object* bitmap, sehingga proses *masking* dapat membentuk area kurva *Lissajous*.
7. Proses perulangan yang terdapat pada tiap struktur penentuan koordinat kurva ditentukan oleh konsep rumus dasar yang digunakan, sehingga dapat menghasilkan pola yang sesuai.

BAB IV
UJI COBA DAN EVALUASI

Pada bab ini akan dibahas uji coba dan evaluasi sistem *Generator Watermark*. Skema uji coba mencakup beberapa skenario yang telah disyaratkan pada tujuan pembuatan sistem yang tercakup dalam pokok bahasan bab pertama.

Skenario uji coba sistem meliputi aspek variasi bentuk pola, warna, dan estetika *watermark*.

4.1 Lingkungan Uji Coba

Uji coba dilakukan dengan menggunakan spesifikasi perangkat keras (*hardware*) dan perangkat lunak (*software*) sebagai berikut.

4.1.1 Komputer Host

- Tipe : DELL INSPIRON 1420
- Processor : Intel Pentium IV Core™ 2 Duo 1.66 Ghz
- RAM : 2.5 Gb

4.1.2 Printer

Spesifikasi printer yang digunakan pada saat uji coba adalah sebagai berikut:

CANON PIXMA iP1980

Gambar 4.1 Canon Pixma iP1980

- Metode Pencetakan: *Inkjet Printing, Serial*
- Kecepatan Pencetakan: untuk tipe hitam-putih 21ppm dan untuk tipe warna 17 ppm.
- Resolusi: 4800 x 1200 dpi
- Tipe Borderless: kertas ukuran 4”x6” / 8”x10” / A4
- Sistem Operasi: Windows 2000 SP4 / XP SP2 / Vista
- Interface: USB
- Dimensi: 442 x 237 x 152
- Berat: 3.3 kg

4.1.3 Software

Berikut ini adalah beberapa *tools* perangkat lunak yang digunakan untuk proses uji coba.

1. Sistem Operasi Windows XP SP3
2. Visual Studio 2008 dengan menggunakan Framework Devexpress versi 8.2
3. Internet Information Services (IIS) 5.1
4. Browser Mozilla Firefox versi 3.0.1
5. Adobe Acrobat Professional versi 7.0

4.2 Uji Coba

Sub bab berikut menjelaskan tentang proses uji coba sistem *Generator Watermark* yang terdiri atas uji coba variasi dan faktor estetika pola *watermark*.

4.2.1 Uji Coba Variasi

Uji coba variasi meliputi skenario pengujian terhadap variasi bentuk dan variasi warna. Uji coba variasi bentuk memaparkan proses pengolahan nilai-nilai parameter dalam rumus persamaan parametrik yang telah diberikan pada sub bab garis besar sistem, sehingga dapat membentuk pola bentuk dasar *watermark* yang unik dalam implementasi sistem. Sedangkan uji coba variasi warna akan memberikan penjelasan tentang konsep pengolahan pilihan warna dalam penggambaran pola *watermark*.

Tabel 4.1 menyajikan skema uji coba variasi dengan 50 nilai nomor dokumen sebagai input pada sistem. Input tersebut diolah oleh sistem untuk menghasilkan *string hash* (terlampir pada Tabel 4.2) sehingga diperoleh nilai-nilai parameter.

Tabel 4.1 Uji Coba Variasi dengan Variasi Nomor Dokumen

No	**Nomor Dokumen**	**Nilai Parameter**
1.	10250/A4.5/KP/2008	2 - 2 - 4 - 5 - 8 - 5 - 5 - 4 - 8 - 5
2.	10251/A4.5/KP/2008	8 - 1 - 3 - 5 - 6 - 1 - 4 - 5 - 7 - 3
3.	10252/A4.5/KP/2008	1 - 3 - 6 - 5 - 8 - 4 - 4 - 5 - 8 - 9
4.	10253/A4.5/KP/2008	9 - 9 - 2 - 5 - 4 - 9 - 6 - 2 - 5 - 4
5.	10254/A4.5/KP/2008	9 - 8 - 2 - 4 - 2 - 1 - 4 - 3 - 4 - 5
6.	10255/A4.5/KP/2008	4 - 5 - 8 - 5 - 4 - 4 - 3 - 1 - 7 - 3
7.	10350/A4.5/KP/2008	8 - 1 - 5 - 4 - 6 - 4 - 5 - 1 - 5 - 4

8.	10400/A4.5/KP/2008	2 - 3 - 6 - 4 - 4 - 6 - 8 - 6 - 5 - 5
9.	10450/A4.5/KP/2008	9 - 3 - 6 - 5 - 1 - 7 - 4 - 5 - 9 - 6
10.	10500/A4.5/KP/2008	1 - 2 - 4 - 5 - 6 - 2 - 4 - 2 - 8 - 8
11.	10550/A4.5/KP/2008	2 - 2 - 4 - 5 - 8 - 2 - 4 - 9 - 9 - 7
12.	10600/A4.5/KP/2008	3 - 3 - 7 - 4 - 7 - 3 - 6 - 9 - 4 - 6
13.	10700/A4.5/KP/2008	2 - 9 - 2 - 4 - 4 - 3 - 5 - 7 - 7 - 3
14.	10800/A4.5/KP/2008	1 - 2 - 5 - 4 - 5 - 2 - 4 - 2 - 7 - 1
15.	10900/A4.5/KP/2008	1 - 4 - 7 - 5 - 6 - 1 - 2 - 5 - 7 - 1
16.	11256/A4.5/KP/2008	2 - 8 - 9 - 8 - 5 - 2 - 8 - 2 - 7 - 6
17.	11257/A4.5/KP/2008	3 - 4 - 7 - 5 - 6 - 4 - 4 - 9 - 9 - 5
18.	11258/A4.5/KP/2008	3 - 2 - 6 - 4 - 2 - 3 - 4 - 7 - 5 - 1
19.	11259/A4.5/KP/2008	8 - 8 - 2 - 4 - 6 - 4 - 8 - 7 - 5 - 9
20.	11260/A4.5/KP/2008	4 - 2 - 4 - 4 - 7 - 5 - 4 - 9 - 9 - 5
21.	12345/A4.5/KP/2008	2 - 3 - 6 - 4 - 8 - 2 - 4 - 5 - 9 - 4
22.	12346/A4.5/KP/2008	1 - 3 - 7 - 4 - 4 - 3 - 5 - 5 - 6 - 5
23.	12347/A4.5/KP/2008	2 - 3 - 7 - 4 - 9 - 9 - 4 - 4 - 2 - 3
24.	12348/A4.5/KP/2008	2 - 2 - 4 - 5 - 5 - 2 - 3 - 8 - 7 - 3
25.	12349/A4.5/KP/2008	2 - 4 - 7 - 4 - 8 - 7 - 8 - 5 - 4 - 5
26.	24682/A4.5/KP/2008	7 - 8 - 1 - 8 - 4 - 2 - 1 - 8 - 4 - 6
27.	24684/A4.5/KP/2008	9 - 1 - 5 - 5 - 1 - 2 - 3 - 7 - 8 - 2
28.	24686/A4.5/KP/2008	7 - 9 - 2 - 4 - 2 - 5 - 6 - 7 - 4 - 4
29.	24688/A4.5/KP/2008	4 - 6 - 9 - 5 - 5 - 1 - 6 - 6 - 3 - 2
30.	24690/A4.5/KP/2008	4 - 3 - 6 - 4 - 6 - 4 - 4 - 2 - 7 - 2
31.	36822/A4.5/KP/2008	5 - 4 - 7 - 5 - 1 - 1 - 8 - 6 - 6 - 7
32.	36824/A4.5/KP/2008	1 - 3 - 6 - 5 - 9 - 1 - 5 - 9 - 4 - 1
33.	36826/A4.5/KP/2008	3 - 4 - 7 - 5 - 8 - 9 - 5 - 1 - 7 - 4
34.	36828/A4.5/KP/2008	9 - 1 - 3 - 5 - 7 - 9 - 4 - 8 - 6 - 5
35.	36830/A4.5/KP/2008	3 - 5 - 7 - 5 - 3 - 3 - 4 - 2 - 3 - 1
36.	46822/A4.5/KP/2008	9 - 3 - 7 - 5 - 1 - 3 - 4 - 4 - 1 - 2
37.	46824/A4.5/KP/2008	9 - 1 - 5 - 4 - 9 - 4 - 5 - 9 - 3 - 7
38.	46826/A4.5/KP/2008	1 - 1 - 3 - 1 - 6 - 2 - 8 - 9 - 3 - 5
39.	46828/A4.5/KP/2008	8 - 8 - 1 - 4 - 6 - 6 - 1 - 7 - 7 - 1
40.	46830/A4.5/KP/2008	9 - 2 - 4 - 4 - 8 - 3 - 5 - 5 - 8 - 3

41.	03785/A4.5/KP/2008	3 - 3 - 6 - 4 - 9 - 1 - 8 - 3 - 9 - 1
42.	14896/A4.5/KP/2008	9 - 9 - 3 - 4 - 8 - 1 - 4 - 2 - 3 - 2
43.	25907/A4.5/KP/2008	7 - 7 - 8 - 4 - 1 - 8 - 2 - 7 - 6 - 3
44.	36018/A4.5/KP/2008	3 - 5 - 8 - 5 - 2 - 2 - 4 - 6 - 4 - 9
45.	47129/A4.5/KP/2008	2 - 1 - 3 - 4 - 1 - 3 - 2 - 2 - 2 - 7
46.	58230/A4.5/KP/2008	8 - 3 - 6 - 4 - 4 - 6 - 1 - 8 - 7 - 7
47.	69341/A4.5/KP/2008	3 - 2 - 4 - 4 - 5 - 4 - 4 - 4 - 5 - 6
48.	70452/A4.5/KP/2008	8 - 2 - 5 - 5 - 8 - 1 - 2 - 9 - 3 - 1
49.	81563/A4.5/KP/2008	5 - 4 - 7 - 5 - 3 - 2 - 2 - 3 - 7 - 5
50.	92674/A4.5/KP/2008	2 - 1 - 3 - 4 - 9 - 3 - 5 - 3 - 2 - 1

Tabel 4.2 Nomor Dokumen dan Nilai String Hash

No	**Nomor Dokumen**	**Nilai String Hash**
1.	10250/A4.5/KP/2008	5Fc0YTlUT2VQUjljxVICeZJQNl84MdUwRjLNNacw7C8YQf000y74X2xUa2gwTxFtXUFLchhLOTnZSXyj
2.	10251/A4.5/KP/2008	KFc9YZ1UDmXKUtBjwlJzeRZQMF/QMXIwBTIZNZgx3S/8Qak0oi6RX/1UkmjuT2FtI0EUckFLbTk2V9qO
3.	10252/A4.5/KP/2008	vlcAYWtUeWWJUlhj5FKyeVFQIl84MRwwXDLzNXsylS8bQYU06C7sX8FU6GjjT3ltj0E/ctpLXjmrzqw/
4.	10253/A4.5/KP/2008	BleCYT9UxGWHUhdjdVJUeZtQvl+AMWwwIDKUNT0zhC9YQdk0Jy5rX29UumhUT0lt/UEAcjZLLDnFQGua
5.	10254/A4.5/KP/2008	iFfDYc9UPWUhUqNjL1LjeUtQm1+uMSQwPzJVNRc04S8HQXA0GC54XwpUQGh8T9VteEGVcj5L4Dk514Lf
6.	10255/A4.5/KP/2008	JVdaYWBUNGUZUvtjW1KgefJQyV+wMfAwxzKANaM1Ri+lQZI0CC5EX9ZUSmi8T3ZtW0ErcsNLyTnTbkjN

7.	10350/A4.5/KP/ 2008	B1dxYfdUrGXnUvpj4lJveehQZ18IMb EwUDMpNbAwtC8zQTE0FC5QX7dU UWjKT49tP0FWcgFLMTlVbkOm
8.	10400/A4.5/KP/ 2008	olcAYYRUDmVgUq9jaFL+eW5QZl9 8MRwwgjSXMBQwEy/8QUY09S6tX 25UW2grT0pt/kG5citL+Tktg3yG
9.	10450/A4.5/KP/ 2008	0VcPYTNUsmUWUmhjLVI/eR9Qpl9 YMZYw4TQ6NTEwcS/jQcA02S6yX2 xUgGjkT51tTkHTcghLAzk+DNck
10.	10500/A4.5/KP/ 2008	1FcIYRJUs2VOUkFjJVJjeY9Qs18IM VswnDXbMNEwvS+xQVk0bC6wX0V UlGh2T3BtAkFLcgVLpjmmQu6r
11.	10550/A4.5/KP/ 2008	MleUYRBUn2WTUttjo1IJeTZQFF8g MeEwATXzNdswSi8iQWA0KS4mX5l UdWgIT/5ts0FOcqZLgDnTaYS2
12.	10600/A4.5/KP/ 2008	oFfKYeNUDWXnUlBjYFKOeS1Q4l8r McYwxjZ4MPIw3i/AQTA0eS5+X3N ULmgsTydttEE6cq5LADnRw6Jy
13.	10700/A4.5/KP/ 2008	3Fe/YVtU2mUuUgNjelIqeblQYl/BM WswpzeWMP8wyC/KQWo0Di6PX4l U32i/T1JtGkF1crVL/zlqcIJS
14.	10800/A4.5/KP/ 2008	fFddYWlUjGUsUpdjZlIOee1Q1V/WM QswazjqMPYwBi/UQVY0ji5+XydUK WirTyNtHUHzcp9LLjlJ0v/G
15.	10900/A4.5/KP/ 2008	elemYQJUHmUYUmlj+VLOeWRQ11 8GMTcwBDkRMNUwVi+2QYM02C7 FX0FU3mjGT4VtoUE2cm5LBDlx/Jul
16.	11256/A4.5/KP/ 2008	yVeuYTVU+2W0UkRjXFJZefxQbF+ GMeMxsjLqNWY2Yi+EQRw0wi7dX w9UJWjkT+Vt2UG5cq9LaTkq2Jhu
17.	11257/A4.5/KP/ 2008	yFcYYYZUnWVEUopjzVK/eZhQv19 PMfcxejJ5Na037i+RQZM0Bi6QX4hU Omj9Txht3UHacitL1DnLSBOQ

18.	11258/A4.5/KP/ 2008	TFdQYcBUfGVwUnhjN1IjeRhQr1+R MToxNzLFNdM4Ui8AQVU0RC5tXw lUcWifT6ptL0GncuVLTjl/p7ir
19.	11259/A4.5/KP/ 2008	cleKYatUd2XsUudjE1LfeU5Q219kMb AxuTJsNWI5ny9CQdg0By60XzBUiW iOTxtt+kGdcj5LEzlt8vme
20.	11260/A4.5/KP/ 2008	5Vf1YVVUC2UiUn9jolJkeTxQOl8gM VgxCTJLNm0wYC/jQZ80VS6cX/lUO Wg/T+5tNUHEcjhLqzmSy1pk
21.	12345/A4.5/KP/ 2008	rVezYQlUSGXgUnljt1JoeVdQIV8lM YkyhjN4NBo1RS9GQUI0Ki4/X39Ua mgOT6xtY0FCciFLtTnt8nZ/
22.	12346/A4.5/KP/ 2008	vlelYbBUMWVgUtNjGlLoedBQel/bM TIyWTOZNLY2by80Qaw08C5UX6tU h2j8TxFt3kH+cvVLXjlFRcfZ
23.	12347/A4.5/KP/ 2008	ildZYfRURGXrUo1jp1K5eZ9QjV8cM RQyIjMtNGw3ay9QQQc0vC4cX6VU G2hzT1ttHUGFcptLiTlWl/sY
24.	12348/A4.5/KP/ 2008	vVeWYQRUv2VTUi9jJlILeV1QUl+M MQEycDM8NBs4+S8RQfU0tC6tXzp Us2jFT1Ft2UEscvtLnTmHhEsZ
25.	12349/A4.5/KP/ 2008	xlcyYYBUEWXpUvFj/1KqeWtQ6l9b MaIy0TOBNEE5My/ZQRM08C7AX+l U3GiQT1ttM0E/cg5LhTnvE+aF
26.	24682/A4.5/KP/ 2008	aVc8YUpUTmX9UipjBFJ5eShQGV/G Mns0fzahOIsyfy8wQfU0Pi5NX4JUM Wh5T6ZtO0G8cppLwDnUQYX+
27.	24684/A4.5/KP/ 2008	HlfZYbpUsmWPUvJj41LDeexQdF9A MjM0+Tb5OAo0Ey+VQXo0vi7uX59 UXWh5T2tt5EGncqhLjzmo9FM9
28.	24686/A4.5/KP/ 2008	iFchYc1UwmU5UkRjYVLOeRlQAl/r Mto0YDbnOA42py92QW00ry6UX6F USGgzT3Bts0HqckpL4znU+vvF

29.	24688/A4.5/KP/2008	X1cwYexU7GWPUmxjCVK9eT1QgV+FMiQ0bzbfOI84cC9UQdY0BC64X2VU8mhTT75teEEEcpFLWzn8MQdQ
30.	24690/A4.5/KP/2008	YFeJYSZUEmUcUsxjdlJ/eZBQn19oMrM0sjaWORMwoi8GQYo0EC6UXzFUXWjcTxlt0kEhcgJLITnrKR3l
31.	36822/A4.5/KP/2008	uVd5YWBUqWVQUrxjwlICeW1Q119wMy82+zhJMpEy2C9EQSc0uy5MXx9UiWgqT15tmUEvcoJL4TnCVAo8
32.	36824/A4.5/KP/2008	nle+YXpUVmUEUhdjS1JDefVQpF/MM7E2Pzh7Moc0fC9sQSU0Ky4oX1hUMmjnT9JtiUECciZLpzmpvHB1
33.	36826/A4.5/KP/2008	bFc3YXpUJGXBUvRjQVKeecVQl18cM6427Di4MgU2sS98QT40FC5uX5ZU2WjTT5dtmUFVcg1LkjmdgrXO
34.	36828/A4.5/KP/2008	H1dPYR1UwWWPUupj1lIXeTVQy18KM082vjgQMnQ4uS9UQRM0wi4mX+1UmGjKT7Ntd0HEchBLwTnqbkm1
35.	36830/A4.5/KP/2008	N1efYZ5UWmVOUt5jrVIIeRNQ5V+uM3o2PziwM9kwhi8yQcA0Py6UX+dU+Gh7TxRtYEF7ckZL2TmJ6u+5
36.	46822/A4.5/KP/2008	D1daYclUDGXAUp9jAFJveQNQ7l/VNP02FThHMqYyZi/PQe00yi5GX4xU1GgZT9BtDEFxckdLSTlUS0Ok
37.	46824/A4.5/KP/2008	dFcnYfhUgWWlUoBj9lJ2ealQ6F9tNJo2FjhPMtE0QS/0QVM0Di7bX3pULmhMT8dtAEHDcsFLejmbf8LS
38.	46826/A4.5/KP/2008	lVemYZhUaGU2UmtjIlIceYdQ1V+/NJI21jgjMrY2bi8pQU40Ni59X4pU2GjSTyVtnUFecoxL5DkkD1/h
39.	46828/A4.5/KP/2008	EldrYX1UrmXqUgNjRlLneTtQRV9lNM02OzhgMng40C/aQWE0ZC7LX6dUwWhZT5ttVUFdchRL6zlS8PFN

40.	46830/A4.5/KP/ 2008	0Fe2YXpU3WUoUj1juFLlea1Qj19BN FE2cjiHMyMwKy8VQSU0ky6CX+R UaGjZT91t40GSco1LEzkWh14h
41.	03785/A4.5/KP/ 2008	nVcXYRRUQmWjUvZj+VJBefRQGF/ rML8zDDdrOPM10S8dQe40Ey7DX0h UMWjaT3Ntn0H5cmBLoTlfy6u/
42.	14896/A4.5/KP/ 2008	J1fEYfZUm2UfUrBjRVJ5eTBQS1+1 MeQ0TDg3Oek2zy/uQa80iS4lXwRUv 2jRT3ltn0FxcoRLYzkPnvEK
43.	25907/A4.5/KP/ 2008	NVcaYc9Uq2W5UiBjx1LQeQRQ2198 Mpo1gzmeMMk3XC9vQf40PC4UX5J U/WiwTw5t8kFgcghLnDm2THgO
44.	36018/A4.5/KP/ 2008	yVcaYXJUNWWOUutjl1KyeSxQQl/O M2k2YDD3MX84MC9/QQs0wi7jX6B UjmhUTyZtREEscutLEDk7aGdP
45.	47129/A4.5/KP/ 2008	CVcpYRZUl2UkUq1jyFKJeRlQ1F8W NNY36DHZMtQ5oC9qQXI0EC5jX15 U0miVTzBtu0GmcjRL8jnaLgQE
46.	58230/A4.5/KP/ 2008	01c/YVNUVmXrUlRjlVK6eZ5Q2F/u NXw4vTLjMy4wlC/fQZI01i5BX1NU8 GhdTyRt3UHRct5LGDlA9isc
47.	69341/A4.5/KP/ 2008	DlfGYRVU/2VbUvZjc1IkeQhQl1/7Nu U5PzP7NN8x3y9sQVg0fS50X4hU5Wi sT9xt70FJcu9LXDmxU2VC
48.	70452/A4.5/KP/ 2008	pldiYWpUdmU+UhFjb1LlefZQGF/+N 6AwhjSXNQYy0S+BQdQ0+C4xX3B U3GjhT+RtyUExcvNL7DndQMmI
49.	81563/A4.5/KP/ 2008	FFcUYe5UfGXWUphjn1J0eaZQW198 OOQx0DWWNugzly/iQSo08i5pX4JU MWibT7FtJ0GtcjpL9DnYFCq0
50.	92674/A4.5/KP/ 2008	IlcUYRpUI2XfUh9jRVITeUVQiF8AO aQyTzZ5N4U01S/DQeo0QS7iX8JU4 mgzTwNtlEH/cm1LwjknDur/

Berdasarkan hasil uji coba variasi pola *watermark* pada Tabel 4.1, maka sistem *Generator Watermark* telah memberikan hasil yang cukup memuaskan dengan memberikan variasi bentuk pola *watermark* yang unik antara dokumen yang satu dengan lainnya yang terdiri atas variasi bentuk dan warna.

4.2.2 Uji Coba Estetika

Uji coba estetika dilakukan untuk melihat apakah penggabungan kombinasi beberapa fungsi-fungsi dasar tersebut diatas dapat memberikan bentuk pola *watermark* yang memenuhi aspek estetika. Oleh karena itu, proses pengujian pada sub bab berikut akan menerima *input* (data masukan) berupa nomor dokumen dengan beberapa kriteria. Sehingga dapat diketahui validitas faktor estetika terhadap kombinasi fungsi-fungsi tersebut. Fungsi-fungsi tersebut diatas telah dimodifikasi sedemikian rupa serta dipadukan dengan beberapa pola dasar lainnya, sehingga dapat memperkaya variasi pola bentuk dan warna *watermark*.

Tabel 4.3 menyajikan skema uji coba estetika dengan 25 nilai nomor dokumen sebagai *input* pada sistem. Pada tabel tersebut terdapat kolom penilaian faktor estetika yang menyajikan rentang nilai (1 – 5) terhadap bentuk pola *watermark* yang berdasarkan pada tingkat kompleksitas variasi bentuk, keindahan perpaduan antar bentuk pola, serta keindahan variasi corak warna pola.

Dasar penilaian faktor estetika pada kolom Tabel 4.3 dilakukan berdasarkan penilaian oleh lima orang *user* dengan memberikan nilai yang terbagi dalam kategori sebagai berikut:

1: Cukup **2: Biasa** **3: Bagus**

4: Bagus Sekali **5: Sangat bagus**

Tabel 4.3 Uji Coba Estetika dengan Variasi Nomor Dokumen

No	Nomor Dokumen	Penilaian Faktor Estetika				
		I	II	III	IV	V
1.	10250/A4.5/KP/2008	5	4	3	3	4
2.	10251/A4.5/KP/2008	3	4	4	3	4
3.	10252/A4.5/KP/2008	3	4	5	4	3
4.	10253/A4.5/KP/2008	5	3	4	5	3
5.	10254/A4.5/KP/2008	5	3	4	3	3
6.	10255/A4.5/KP/2008	5	5	5	5	2
7.	10350/A4.5/KP/2008	5	3	4	5	2
8.	10400/A4.5/KP/2008	3	4	4	4	4
9.	10450/A4.5/KP/2008	5	3	4	3	3
10.	10500/A4.5/KP/2008	4	3	3	4	3
11.	10550/A4.5/KP/2008	5	4	4	4	3
12.	10600/A4.5/KP/2008	5	4	5	4	3
13.	10700/A4.5/KP/2008	4	4	5	3	3
14.	10800/A4.5/KP/2008	3	3	5	4	4
15.	10900/A4.5/KP/2008	5	4	3	4	4
16.	11256/A4.5/KP/2008	3	4	4	3	3
17.	11257/A4.5/KP/2008	4	4	5	5	3
18.	11258/A4.5/KP/2008	3	3	4	3	3
19.	11259/A4.5/KP/2008	3	2	4	3	4
20.	11260/A4.5/KP/2008	5	4	5	5	2
21.	12345/A4.5/KP/2008	4	4	5	4	4
22.	12346/A4.5/KP/2008	3	4	5	4	4
23.	12347/A4.5/KP/2008	4	4	5	5	5
24.	12348/A4.5/KP/2008	5	4	3	3	3
25.	12349/A4.5/KP/2008	3	4	4	5	5

Berdasarkan hasil penilaian bentuk dan warna pola *watermark* pada Tabel 4.3, maka sistem *Generator Watermark* telah

memberikan hasil yang cukup memuaskan dengan variasi bentuk dan corak warna yang memenuhi aspek estetika.

Semua gambar hasil uji coba variasi dan estetika pada Tabel 4.1 dan 4.3 tersebut diatas terdapat pada bagian lampiran.

4.3 Evaluasi

Sebagaimana dijelaskan pada sub bab 4.2 tentang semua hasil uji coba terhadap sistem *Generator Watermark* berdasarkan nomor dokumen, telah diperoleh hasil yang memenuhi tujuan pembuatan Tugas Akhir ini yang sesuai dengan rumusan permasalahan yang diuraikan pada bab pertama, yakni:

1. Bagaimana mendapatkan persamaan-persamaan parametrik yang menghasilkan gambar atau bentuk *watermark* yang indah dengan variasi yang cukup.
2. Bagaimana menentukan variasi parameter untuk mengubah bentuk dan warna gambar *watermark*.
3. Bagaimana melakukan proses transformasi nomor dokumen menjadi nilai-nilai parameter.
4. Bagaimana proses implementasi generator gambar *watermark* diatas kanvas *Xtrareport* berdasarkan *framework Devexpress* pada ASP.NET 2008.

Nomor dokumen sebagai nilai input merupakan dasar pembuatan *string hash* untuk menghasilkan nilai-nilai parameter dengan variasi nilai yang cukup kompleks, sehingga memungkinkan pembentukan pola gambar *watermark* yang unik. Faktor estetika yang ingin dicapai dalam pembuatan Tugas Akhir ini dapat diperoleh melalui penentuan nilai batas yang diberikan pada fungsi parametrik.

Hal yang perlu diperhatikan mengenai integrasi komponen program dalam suatu sistem adalah penggunaan versi *framework Devexpress*. Hal ini dikarenakan proses implementasi komponen program dibangun diatas framework *Devexpress* versi 8.2 (*trial*) dan belum dilakukan proses uji coba apabila integrasi dilakukan pada versi *framework Devexpress* yang lain.

Hal lain yang patut dicermati adalah mengenai hasil pencetakan dokumen digital pada kertas dengan ukuran A4. Hal ini terkait dengan pilihan pencetakan *borderless printing*. Disamping itu, hasil pencetakan yang maksimal tergantung pada kualitas printer. Printer *Canon Pixma iP1980* cukup berkualitas untuk pencetakan dokumen digital hasil uji coba sistem.

[Halaman ini sengaja dikosongkan]

BAB V
PENUTUP

Bab ini membahas kesimpulan yang dapat diambil dari tujuan pembuatan perangkat lunak, serta hasil uji coba yang telah dilakukan. Selain itu terdapat beberapa saran untuk pengembangan lebih lanjut.

5.1 Kesimpulan

Dari hasil pengamatan selama perancangan, implementasi, dan proses uji coba perangkat lunak yang telah dilakukan, maka dapat diambil kesimpulan sebagai berikut:

a. Penggunaan konsep dasar kurva *Lissajous* dengan memberlakukan beberapa modifikasi fungsi dan batasan-batasan nilai parameter pada fungsi parametrik dapat menghasilkan pola gambar yang unik dengan tetap memenuhi aspek estetika.
b. Penentuan batasan-batasan nilai parameter untuk pembentukan pola *watermark* berdasarkan pada hasil uji coba dan penggabungan beberapa pola dasar, sehingga dapat memenuhi pembentukan pola gambar yang unik.
c. Pola bentuk dan warna *watermark* yang diciptakan oleh sistem tergantung pada nilai-nilai parameter yang dihasilkan oleh *string hash* yang berasal dari nomor dokumen.
d. Komponen program *Generator Watermark* yang dibangun pada platform *.NET framework* dengan memanfaatkan komponen *XtraReport* pada *framework Devexpress* versi 8.2 mampu memenuhi kebutuhan proses implementasi sistem.

e. Hasil pencetakan pola gambar *watermark* pada dokumen digital memberikan hasil yang cukup baik terhadap aspek estetika dan memiliki tingkat kompleksitas dan keunikan pola gambar yang memenuhi tujuan pengerjaan Tugas Akhir ini.

5.2 Saran

Dalam pembuatan Tugas Akhir ini, terdapat beberapa kemungkinan pengembangan aplikasi yang dilakukan, yaitu:

a. Pengembangan bentuk pola *watermark* yang lebih kompleks untuk memenuhi kebutuhan pengguna pada masa mendatang.
b. Pengembangan struktur komponen program yang lebih baik, sehingga dapat dengan mudah diintegrasikan dengan sistem lain.
c. Pembuatan aplikasi untuk proses verifikasi pola *watermark* secara otomatis, sehingga mampu melakukan verifikasi pola *watermark* pada dokumen yang telah dicetak (*hardcopy*).

DAFTAR PUSTAKA

[1] Andriyano, Devid. 2002. **Watermark Beramplitudo tinggi pada Sinyal Suara**. <URL: http://digilib.itb.ac.id/gdl.php?mod=browse&op=read&id=jbptitbpp-gdl-s2-2002-devidandri-1849&q=Human>. Diakses tanggal 13 Desember 2008.

[2] Anton, Howard. 2005. **Calculus: Early Transcendentals Single Variable, 8th Edition**. <URL: http://higheredbcs.wiley.com/legacy/college/anton/0471482382/add_mat/parametric_equations.pdf>. Diakses tanggal 13 Desember 2008.

[3] Goldstein, Andrea, dkk. 2002. **Electronic Commerce for Development**. Paris: OECD Publishing.

[4] Herzog, David Alan. 2005. **Trigonometry**. Hoboken, NJ: Wiley Publishing, Inc.

[5] Khoshafian, Setrag, dkk. 1996. **Multimedia and Imaging Databases**. San Fransisco: Morgan Kaufmann.

[6] Kurniawan, Budi. 2002. **System.Drawing with C#**. <URL: http://www.ondotnet.com/pub/a/dotnet/2002/05/20/drawing.html?page=1>. Diakses pada 13 Desember 2008.

[7] Kurniawan, Budi. 2002. **System.Drawing with C#**. <URL: http://www.ondotnet.com/pub/a/dotnet/2002/05/20/drawing.html?page=2>. Diakses pada 13 Desember 2008.

[8] Kurniawan, Budi. 2002. **System.Drawing with C#**. <URL: http://www.ondotnet.com/pub/a/dotnet/2002/05/20/drawing.html?page=3>. Diakses pada 13 Desember 2008.

[9] Lawrence, J.D. 1972. **A Catalog of Special Plane Curves**. New York: Dover.

[10] Prayudi, Yudi. 2006. **Metode Watermarking Ganda Sebagai Teknik Pengaman Pada Citra Digital**. <URL: http://digilib.itb.ac.id/gdl.php?mod=browse&op=read&id=oai:digilib.its.ac.id:ITS-Master-3100002015912&q=watermark>. Diakses tanggal 13 Desember 2008.

[11] Saltzer, Jerry, dkk. 1994. **Library 2000 - Project of the M. I. T. Laboratory for Computer Science**. <URL: http://ltt-www.lcs.mit.edu/Public/Architecture/jeremy-draft-2.html>. Diakses tanggal 13 Desember 2008.

[12] Seitz, Juergen. 2005. **Digital Watermarking for Digital Media**. Information Science Publishing.

[13] Support, Devexpress. 2000-2008. **Online Documentation**. <URL: http://www.devexpress.com/Support/OnlineHelp.xml>. Diakses tanggal 1 Agustus 2008.

[14] Support, Devexpress. 2000-2008. **Online Documentation of Xtrareports**. <URL: http://www.devexpress.com/Help/?document=XtraReports>. Diakses tanggal 1 Agustus 2008.

BIODATA PENULIS

Wachid Asari. Lahir di Surabaya, Jawa Timur, tanggal 5 November 1985. Penulis menyelesaikan jenjang pendidikan Sekolah Dasar (tahun 1998) sampai dengan Sekolah Menengah Atas di Surabaya, Jawa Timur pada bulan Juni 2004. Selanjutnya, pada Agustus 2004 penulis memulai pendidikan S-1 Teknik Informatika di Institut Teknologi Sepuluh Nopember Surabaya.

Selama proses pembelajaran kampus Teknik Informatika ITS, penulis aktif sebagai pengurus himpunan mahasiswa Teknik Informatika ITS periode tahun 2005, serta merupakan salah seorang administrator Laboratorium Pemrograman Teknik Informatika pada periode 2006 – 2008. Selama periode tersebut, penulis juga terlibat aktif sebagai asisten praktikum Pemrograman Terstruktur, koordinator praktikum Struktur Data, asisten mata kuliah Pemrograman API, serta asisten Pemrograman web di lingkungan PIKTI ITS. Disamping itu, penulis aktif sebagai *freelance* website *developer* dan telah terlibat dalam beberapa *project software development* yang berbasis pada aplikasi berbasis web (periode tahun 2006 hingga saat ini).

Ketertarikan penulis pada dunia teknologi informasi meliputi pengembangan perangkat lunak terlebih pada aplikasi berbasis web, walaupun sering kali menggunakan aplikasi berbasis *desktop* dengan menggunakan platform *.NET framework*. Penulis sangat menggemari pengembangan aplikasi berbasis web dengan standar web 2.0, sehingga memungkinkan implementasi aplikasi web yang interaktif bagi para pengguna.

[Halaman ini sengaja dikosongkan]

LAMPIRAN

Gambar Hasil Uji Coba Pola Watermark

[Halaman ini sengaja dikosongkan]