BAB 2
TINJAUAN PUSTAKA

2.1. Profil PUSAIR (Bandung)

Sejarah berdirinya Pusat Penelitian dan Pengembangan Sumber Daya Air (PUSAIR) secara bertahap dimulai dari tahun 1936. Saat itu PUSAIR masih bernama *Hidrodynamisch* Laboratorium yang didirikan oleh Departemen *Voor Verkeer In Waterstaat* (V en W). Selama perkembangannya PUSAIR mengalami perubahan perubahan hingga sekarang [2].

Masa-masa perubahan pada PUSAIR:

1. 1936 Departement *voor Verkeer en Waterstaat* (V en W) mendirikan *Hidrodynamisch Laboratorium*
2. 1947 Institut *voor Weg en Waterbouwkundige Onderzoekingen*
3. 1950 Institut Teknik Air dan Tanah
4. 1966 Lembaga Penyelidikan Masalah Air (LPMA)
5. 1974 Direktorat Penyelidikan Masalah Air (DPMA)
6. 1984 Puslitbang Pengairan, berada di bawah Badan Penelitian dan Pengembangan Departemen Pekerjaan Umum
7. 1999 Pusat Penelitian dan Pengembangan Teknologi Sumber Daya Air, berada di bawah Badan Penelitian dan Pengembangan Departemen Permukiman dan Pengembangan Wilayah (Kimbangwil)
8. 2001 Pusat Penelitian dan Pengembangan Sumber Daya Air, berada di bawah Badan Penelitian dan Pengembangan Departemen Permukiman dan Prasarana Wilayah (Kimpraswil)
9. 2004 Pusat Penelitian dan Pengembangan Sumber Daya Air, berada dibawah Badan Penelitian dan Pengembangan Departemen Pekerjaan Umum
10. 2010-2014 Pusat Penelitian dan Pengembangan Sumber Daya Air, berada dibawah Badan Litbang Kementerian Pekerjaan Umum
11. 2015 Pusat Penelitian dan Pengembangan Sumber Daya Air, berada dibawah Badan Litbang Kementerian Pekerjaan Umum dan Perumahan Rakyat

Kemajuan teknologi dalam penelitian dan pengembangan sumber daya air telah membawa perubahan-perubahan yang signifikan untuk mengkaji pengelolaan sumber daya air. Penelitian dan pengembangan sumber daya air tentunya tidak lepas dari data mengenai air yang dikumpulkan secara terus menerus yang dilakukan oleh PUSAIR.

PUSAIR mengembangkan dalam peramalan deteksi banjir, peramalan kekeringan sumber daya air. PUSAIR melakukan pendataan secara terus-menerus terhadap ketinggian muka air pada sungai-sungai yang ada di Indonesia guna untuk kepentingan kewaspadaan akan terjadi banjir maupun penyediaan sumber daya air yang akan datang. Seiring banyaknya data yang diambil PUSAIR memerlukan integrasi, efisiensi pada data yang diambil guna untuk penelitian dan pengembangan yang lebih tepat sasaran.

2.1.1. Logo PUSAIR

Sebuah perusahaan atau instansi mempunyai sebuah tanda pengenal berupa logo perusahaan. PUSAIR memakai logo PU (Pekerja Umum) yang mempunyai sebuah logo sebagai tanda pengenal yaitu berupa baling-baling biru di dalam kotak kuning, gambar logo PUSAIR dapat dilihat pada gambar.



Gambar 2.1 Logo PUSAIR

2.1.2. Visi dan Misi

Visi [2]

Menjadi lembaga terkemuka dalam menghasilkan teknologi dan menyediakan jasa keahlian untuk mendukung tersedianya infrastruktur sumber daya air yang handal.

Misi [2]

1. Meneliti dan mengembangkan teknologi tepat guna bidang sumber daya air (SDA) yang kompetitif dan ramah lingkungan.
2. Menyusun norma, standar, pedoman, manual bidang konstruksi dan bangunan sumber daya air.
3. Menunjang penyelenggaraan penyediaan tenaga ahli pengelola Sumber Daya Air melalui kegiatan diseminasi teknologi.
4. Memberikan advis dan pelayanan teknis bidang sumber daya air.
5. Menyediakan data dan informasi bidang sumber daya air .

2.1.3. Tugas dan Fungsi

Tugas [2]

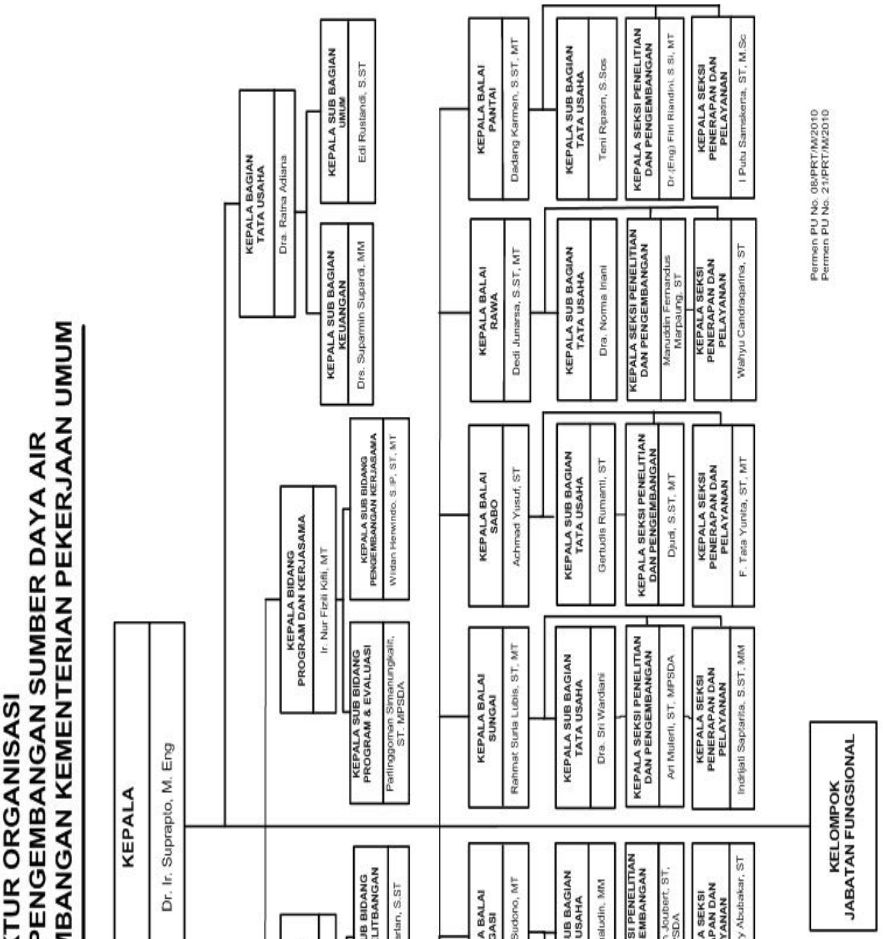
Melaksanakan penelitian, pengembangan, serta penerapan ilmu pengetahuan dan teknologi di bidang sumber daya air.

Fungsi [2]

1. Penyusunan kebijakan teknis, rencana dan strategi penelitian, pengembangan dan penerapan ilmu pengetahuan dan teknologi, serta penyelidikan dan pengkajian di bidang sumber daya air.
2. Pelaksanaan penelitian, pengembangan, penerapan, dan pelayanan ilmu pengetahuan dan teknologi, serta penyelidikan dan pengkajian di bidang sumber daya air.
3. Penyiapan, perumusan, dan evaluasi standar, pedoman, serta manual di bidang sumber daya air.
4. Pemantauan, evaluasi, dan pelaporan tugas penelitian, pengembangan dan penerapan ilmu pengetahuan dan teknologi, serta penyelidikan dan pengkajian di bidang sumber daya air.

2.1.4. Struktur Organisasi

Struktur organisasi di PUSAIR dapat dilihat pada Gambar 2.2 [2].



Gambar 2.2 Struktur Organisasi

2.2. Metode Digitasi

Digitasi secara umum dapat didefinisikan sebagai proses konversi data analog ke dalam format digital. Objek-objek tertentu seperti jalan, rumah, sawah dan lain-lain yang sebelumnya dalam format raster maka menjadi objek-objek vektor.

Proses digitasi secara umum dibagi dalam dua macam:

1. Digitasi menggunakan *digitizer (*zaman dulu tetapi kini hampir tidak lagi*)* Dalam proses digitasi ini memerlukan sebuah meja digitasi atau digitizer.



Gambar 2.3 Contoh Alat *Digittizer*

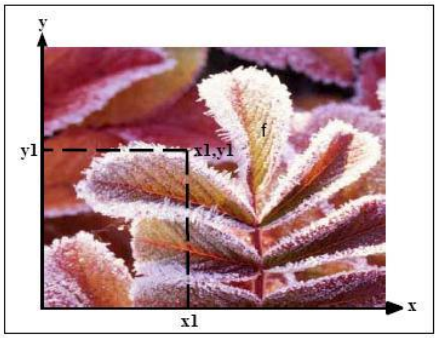
Sumber : http://cdgis.en.alibaba.com/product/297591458-209914896/CD_Series_Tablet_Digitizer_A1.html

2. Digitasi *onscreen* di layar monitor
   Digitasi *onscreen* paling sering dilakukan karena lebih mudah dilakukan, tidak memerlukan tambahan peralatan lainnya, dan lebih mudah untuk dikoreksi apabila terjadi kesalahan.
   Digitasi onscreen biasanya dilakukan pada/dibantu oleh suatu base-layer yang punya referensi spasial, misalnya citra satelit.

2.3. Citra Digital

Suatu citra dapat didefinisikan sebagai fungsi *f(x,y)* berukuran M baris dan N kolom, dengan x dan y adalah koordinat spasial, dan amplitudo f di titik koordinat *(x,y)* dinamakan intensitas atau tingkat keabuan dari citra pada titik tersebut.

Apabila nilai x, y, dan nilai amplitudo f secara keseluruhan berhingga *(finite)* dan bernilai diskrit maka dapat dikatakan bahwa citra tersebut adalah citra digital [3]. Representasi koordinat *pixel* dalam sebuah citra dapat dilihat pada gambar 2.3.



Gambar 2.4 Citra Digital

Citra di digital dapat ditulis dalam bentuk matriks sebagai berikut.

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

2.4. Jenis-jenis Citra Digital

Nilai suatu *pixel* memiliki nilai dalam rentang tertentu, dari nilai minimum sampai nilai maksimum. Jangkauan yang digunakan berbeda-beda tergantung dari jenis warnanya. Namun secara umum jangkauannya adalah 0 – 255 [3]. Berikut adalah jenis-jenis citra berdasarkan nilai *pixel*nya.

1. Citra Biner

   Citra biner adalah citra digital yang hanya memiliki dua kemungkinan nilai *pixel* yaitu hitam dan putih. Citra biner juga disebut dengan citra

B&W (*black and white*) atau citra monokrom. Hanya dibutuhkan 1 bit untuk mewakili nilai setiap *pixel* dari citra biner.

Citra biner sering kali muncul sebagai hasil dari proses prngolahan seperti segmentasi, pengembangan, morfologi, ataupun *dithering*.

Untuk mendapatkan citra biner harus melalui proses *thresholding*, secara matematis dapat dituliskan sebagai berikut.

$$g(x,y) = \begin{cases} 1\ jika\ f(x,y) \geq T \\ 0\ jika\ f(x,y) < T \end{cases} \quad \textbf{(2-2)}$$

Dengan g(x,y) adalah citra biner dari citra *grayscale* f(x,y), dan T menyatakan nilai *threshold* [3]. Nilai T ditentukan dengan menggukan metode *thresholding global*.

Nilai T memegang peranan yang penting dalam proses pengambangan, kualitas hasil citra biner sangat tergantung pada nilai T yang digunakan.



Gambar 2.5 Citra Biner

2. Citra *Grayscale*

Citra *grayscale* merupakan citra digital yang hanya memiliki satu nilai kanal pada setiap *pixel*nya, dengan kata lain nilai bagian RED = GREEN = BLUE. Nilai tersebut digunakan untuk menunjukkan tinggi intensitas. Warna yang dimiliki adalah warna dari hitam, keabuan dan putih. Tingkatan keabuan di sini merupakan warna abu dengan berbagai tingkatan dari hitam hingga mendekati putih. Citra *grayscale* berikut memiliki kedalaman warna 8 bit (256 kombinasi warna keabuan).

Citra *grayscale* bisa didapat dari hasil konversi dari sebuah citra yang mempunyai warna dengan rumus *Luma* [4]:

$$g = 0{,}2126R + 0{,}7152G + 0{,}0722B \quad \textbf{(2-3)}$$



Gambar 2.6 Citra *Grayscale*

3. Citra Berwarna

   Citra yang nilai *pixel*-nya merepresentasikan warna tertentu, banyaknya warna yang mungkin digunakan bergantung pada kedalam *pixel* citra yang bersangkutan. Pada umumnya citra digital disusun oleh susunan kombinasi warna RGB (*Red Green Blue)* . Derajat keabuan merah(fmerah(x,y)), hijau (fhijau(x,y) dan biru(fbiru(x,y)). Dari sebuah kombinasi tersebut dapat menghasilkan warna lain yang dibutuhkan.

Citra warna terbagi menjadi 3 yaitu, citra warna 8 bit, 16 bit, dan 24 bit. Setiap *pixel* dari citra warna diwakili dengan 8,16, atau 24 bit semakin tinggi bit yang ada pada sebuah *pixel* maka kedalaman warna satu citra sangat tinggi.



Gambar 2.7 Citra Berwana

2.5. Tapis *Mean*

Tapis *Mean* sangat sederhana dan mudah diimplementasikan serta berguna untuk mengurangi variasi nilai satu *pixel* (intensitas) dengan *pixel* berikutnya. Tapis *Man* menghitung nilai *pixel* baru dengan nilai rata-rata dari *pixel* tetangga dan *pixel* bersangkutan [3].

$$\begin{bmatrix} \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} \end{bmatrix} \qquad \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

(a) (b)

Gambar 2.8 (a) Tapis mean 2x2 (b) Tapis mean 3x3

2.6. Deteksi Tepi Berdasarkan Turunan Kedua

Tepian dari satu Citra mengantung informasi penting dari Citra bersangkutan. Tepian Citra dapat merepresentasikan objek-objek yang terkandung dalam Citra tersebut, bentuk, dan ukurannya serta terkadang juga informasi tentang teksturnya. Tepian Citra adalah posisi di mana intensitas *pixel* dari citra berubah dari nilai rendah ke nilai tinggi atau sebaliknya. Deteksi tepi umumnya adalah langkah awal melakukan segmentasi citra [3].

Setiap operator deteksi tepi yang telah dijelaskan sebelumnya, akan menampilkan area yang terdeteksi sebagai tepian, tepian yang dihasilkan masih berupa garis yang tebal. Idealnya, satu operator deteksi harus menampilkan pusat tepian. Jika satu deteksi tepi menghasilkan tepian yang lebar maka akan sulit untuk menentukan pusat dari tepian tersebut. Oleh karena itu perlu dilakukan proses *thining* untuk mengurangi lebar tepian menjadi 1 *pixel*. Deteksi tepi dari turunan kedua dapat menghasilkan tepian citra yang lebih baik karena menghasilkan tepian yang lebih tipis. Salah satu operator dari turunan kedua adalah *Laplacian of Gaussian* (LOG) [3].

2.7. Laplacian of Gaussian (LOG)

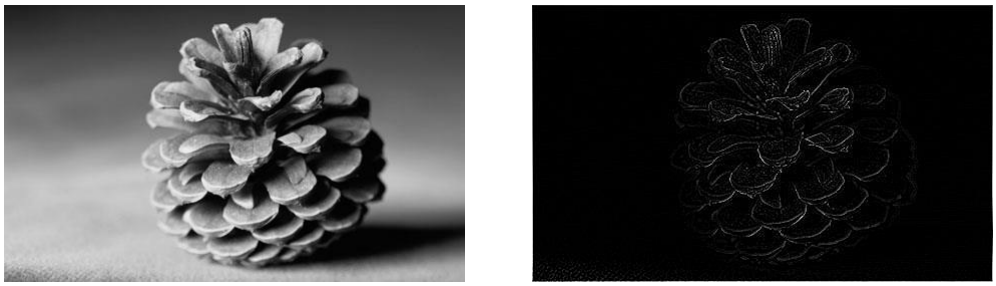
Laplacian of Gaussian adalah salah satu operator deteksi tepi yang dikembangkan dari turunan kedua. Operator *Laplacian of Gaussian* sangat berbeda dengan operator yang lainnya, karena operator Laplacian berbentuk *Enny directional* (tidak horizontal tidak vertikal). Operator ini akan menangkan tepian dari semua arah dan menghasilkan tepian yang lebih tajam dari operator yang lainnya. *Laplacian of Gaussian* terbentuk dari proses *Gaussian* yang diikuti operasi *laplace.* Hasilnya tidak terlalu terpengaruh oleh dera karena fungsi *Gaussian* adalah mengurangi derau. *Laplacian Mask* meminimlisir kemungkinan kesalahan deteksi tepi. Fungsi dari *Laplacian of Gaussian* adalah sebagai berikut:

$$LOG(x,y) = \frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} \quad \textbf{(2-4)}$$

Fungsi di atas merupakan fungsi untuk membentuk tapis dari *Laplacian of Gaussian.* Salah satu contoh operator LOG dalam matris 3x3 adalah sebagai berikut:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} atau \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Contoh penggunaan operator *Laplacian of Gaussian* dengan menggunakan citra *grayscale* adalah sebagai berikut:



(a) (b)

Gambar 2.9 Deteksi tepi *Laplacian of Gaussian* (a) Citra awal (b) Citra hasil

2.8. Penandaan Komponen Terhubung (*Connected Component Labeling*)

Penandaan komponen terhubung memeriksa satu citra dan mengelompokkan setiap *pixel* ke dalam suatu komponen terhubung menurut natura keterhubungan (4,8, atau *m-connectivity*). Setiap komponen terhubung yang saling tidak terhubung (diskon) pada suatu citra akan diberi tanda (label) berbeda. Memisahkan dan memberikan tanda pada tiap komponen terhubung maupun tidak terhubung pada suatu citra memegang peranan sentral pada beberapa aplikasi analisis citra secara otomatis [3].

Penandaan komponen terhubung dilakukan dengan memeriksa suatu citra, *pixel* per *pixel* (dari kiri ke kanan dan atas ke bawah) untuk mengidentifikasi area *pixel* terhubung yaitu suatu area dari *pixel* perbatasan yang memiliki nilai intensitas sama atau nilai intensitas berada dalam suatu himpunan V (pada citra biner V ={1}, pada citra keabuan himpunan V disesuaikan dengan kebutuhan). Penandaan komponen terhubung dapat dilakukan pada citra biner maupun citra keabuan.

Algoritma dari Connected Components Labeling adalah dengan langkah pertama yaitu periksa (*Scan*) citra dengan bergerak sepanjang baris sampai menemukan *pixel* p (nilai p berada dalam himpunan V). Bila p sudah ditemukan maka periksa nilai *pixel* tetangga dari p, yaitu *pixel* di atas, di kiri p dan juga *pixel* diagonal atas dari p, sehingga ada 4 *pixel* tetangga p yang diperiksa yaitu bawah, kanan, diagonal bawah kiri dan diagonal bawah kanan dengan ketentuan berikut.

- Bila keempat *pixel* tetanggaan bernilai 0 maka berilah tanda baru pada p.
- Bila hanya salah satu *pixel* tetangga bernilai 1 maka berilah tanda dari *pixel* tetangga tersebut pada p.
- Bila dua atau lebih *pixel* tetangga tersebut pada p, kemudian buat catatan bahwa semua tanda dari *pixel* tetangga bernilai 1 tersebut ekuivalen.

Langkah terakhir dari proses penandaan adalah melakukan pemeriksaan (*scanning*) kembali pada citra dan ganti setiap tanda dengan tanda dari kelas ekuivalen.

2.9. Grafika komputer

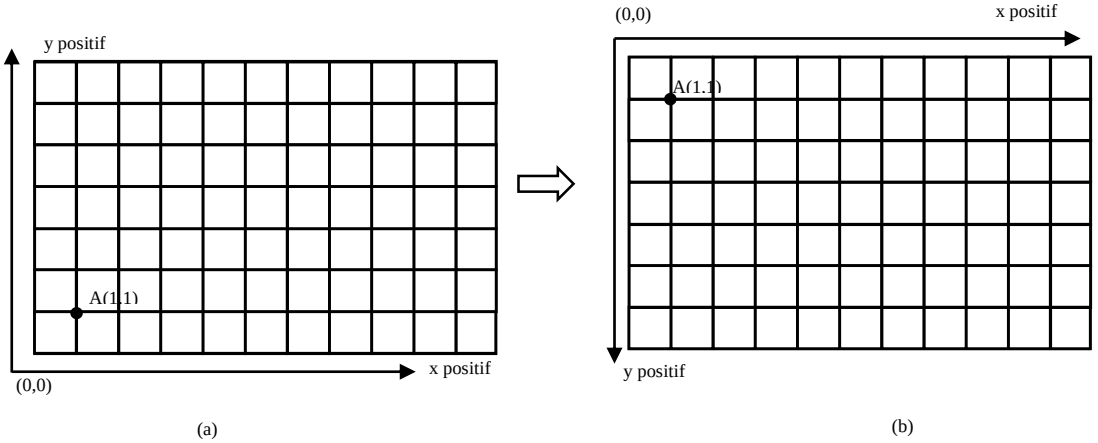
Grafika Komputer adalah suatu bidang komputer yang mempelajari cara-cara untuk meningkatkan dan memudahkan komunikasi antara manusia dengan komputer dengan jalan membangkitkan , menyimpan dan memanipulasi gambar suatu objek. Grafika komputer memungkinkan kita untuk berkomunikasi lewat gambar, bagan dan diagram–diagram. Hal ini disebabkan grafika berhadapan dengan teknik dan teori sintesis gambar komputer.

2.10. Sistem Koordinat *Cartesian*

Pada Koordinat *Cartesian* arah vertikal digunakan sebagai sumbu y dengan y positif menuju ke atas dan arah ke bawah sebagai y negatif, arah mendatar digunakan sebagai sumbu x dengan x positif ke arah kanan dan x negatif ke arah kiri. Tetapi layar monitor menggunakan orientasi yang berbeda. Monitor

menggunakan arah ke bawah sebagai sumbu y positif, disamping itu monitor juga tidak mengenal nilai negatif.

Setiap titik yang digambar dengan teknik *poin-plotting* lokasinya ditentukan berdasarkan sistem koordinat *cartesian*. Di dalam koordinat *cartesian* setiap titik ditentukan lokasinya melalui pasangan nilai x dan y. Nilai koordinat x bertambah positif dari kiri ke kanan dan nilai y bertambah positif dari bawah ke atas. Pada layar komputer sumbu x bertambah positif ke kanan dan sumbu y bertambah positif ke bawah seperti diperlihatkan pada Gambar 2.10 [5].

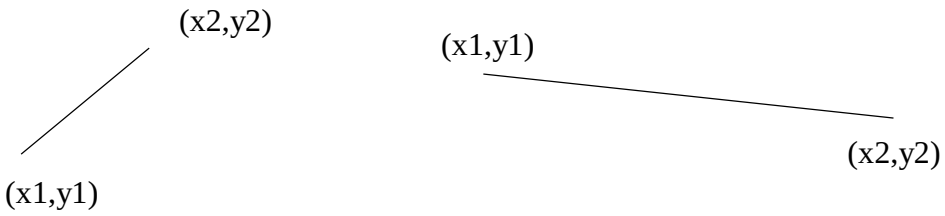


Gambar 2.10 (a) *Cartesian* (b) Layar Komputer

Pada Gambar 2.10 a sebagai titik *pixel*. *Pixel* merupakan elemen penyusun terkecil dari suatu objek dalam grafika komputer. Titik didefinisikan pada suatu posisi *(x,y)* dalam koordinat *cartesian*. Masing-masing *pixel* memiliki atribut warna dan koordinat *(x,y)*.

2.11. Garis

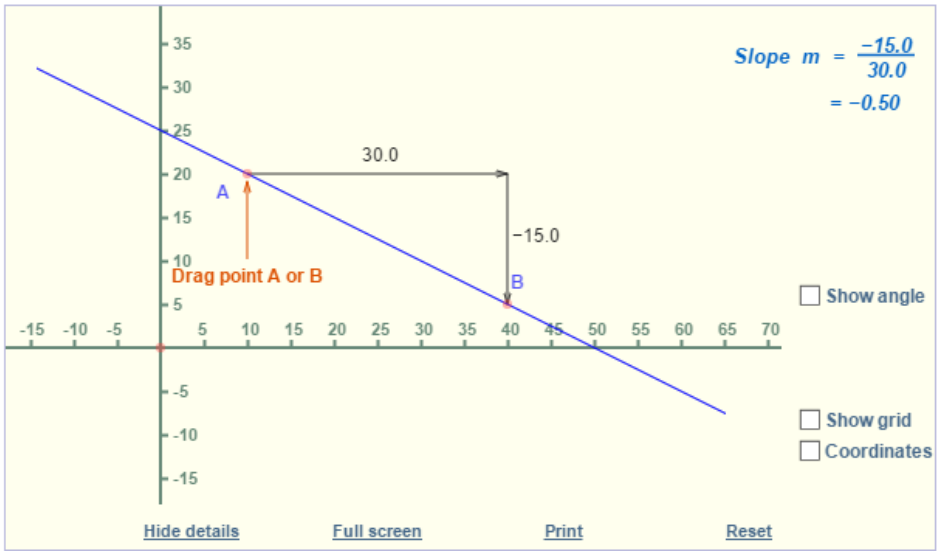
Garis merupakan salah satu bentuk dasar dari gambar. Sebuah garis, dalam grafika disebut sebagai *segmen*, dinyatakan dengan empat nilai yaitu x1,y1, x2 dan y2. x1 dan y1 menyatakan koordinat awal dari sebuah garis sedangkan x2 dan y2 menyatakan koordinat akhir dari sebuah garis. Gambar 2.11 Memberikan ilustrasi mengenai bentuk garis [5].



Gambar 2.11 Garis

Setiap garis mempunyai nilai kecenderungan (*gradien/slope*) yang disimbolkan dengan huruf *m* dan dapat dihitung dengan rumus (2-5) [6].

$$m = \frac{A_y - B_y}{A_x - B_y} \qquad \textbf{(2-5)}$$

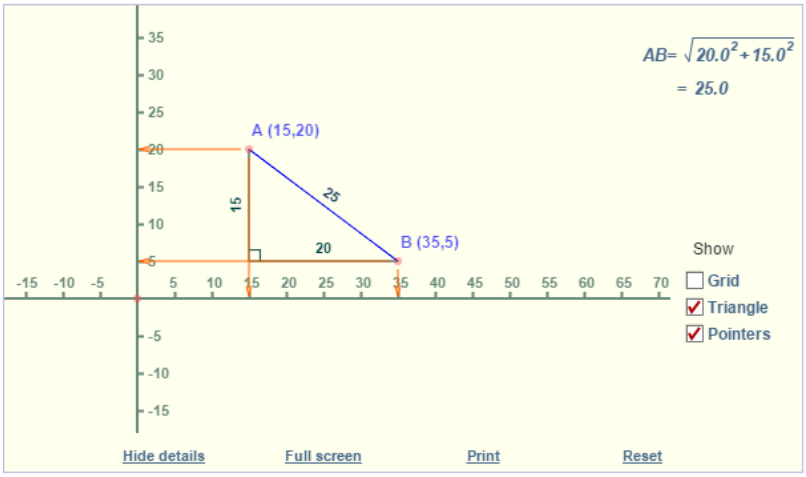


Gambar 2.12 *Slope* pada garis

Sumber : http://www.mathopenref.com/coordslope.html

2.11.1. Jarak Garis

Untuk menentukan jarak garis yang dibentuk oleh kedua titik (x,y) dengan menggunakan rumus *Phytagoras*, yaitu sisi miring^2 = alas^2 + tinggi^2. Contoh dapat dilihat pada Gambar 2.13.



Gambar 2.13 Jarak garis dari dua titik

Sumber : http://www.mathopenref.com/coorddist.html

untuk menentukan jarak garis tersebut menggunakan rumus sebagai berikut:

$$ab^2 = \Delta y^2 + \Delta x^2 \qquad \textbf{(2-6)}$$

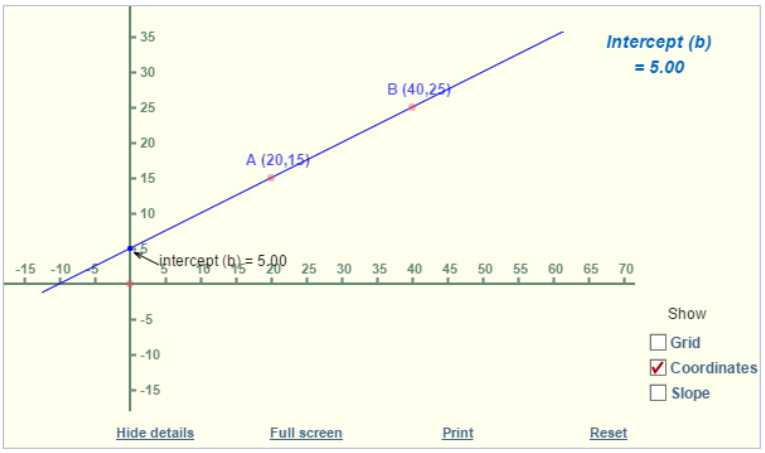
Dengan penyederhanaan sehingga menjadi :

$$ab = \sqrt{\Delta y^2 + \Delta x^2} \qquad \textbf{(2-7)}$$

2.11.2. Perpotongan Dua Gradien

Perpotongan dua gradien dibutuhkan pada saat garis berpotongan dan garis tersebut perpotongan dapat dicari menggunakan rumus [7]:

$$b = y - mx \qquad \textbf{(2-8)}$$

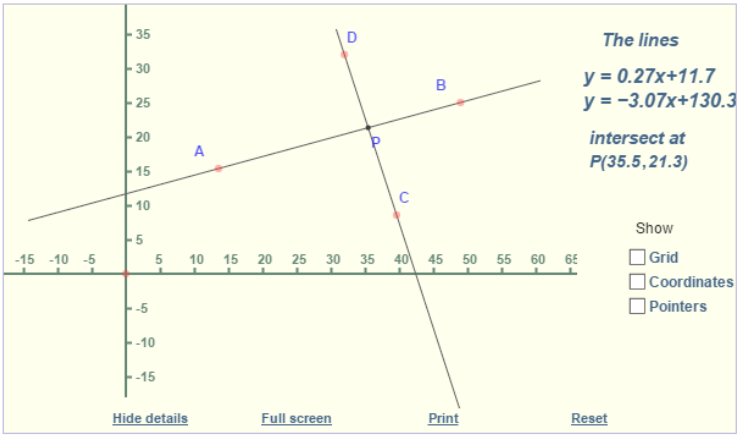


Gambar 2.14 Perpotongan garis

Sumber : http://www.mathopenref.com/coordintercept.html

Pada dua garis gradien menggunakan rumus kedua kemiringan dari garis tersebut dan dapat dicari dengan rumus [8]:

$$y - mx = y - mx \qquad \textbf{(2-9)}$$



Gambar 2.15 Perpotongan dua garis gradien

Sumber : http://www.mathopenref.com/coordintersection.html

2.11.3. Rasio

Rasio didefinisikan sebagai cara membandingkan atau menunjukkan hubungan antara dua angka dari jenis yang sama. Rasio biasanya dinyatakan dengan

memisahkan dua angka dengan titik dua (:). Juga tanda bagi (/) digunakan untuk mengekspresikan rasio. Untuk rasio untuk memiliki makna, baik nomor harus > 0. Rasio jumlah dinyatakan sebagai pecahan atau desimal, dan beberapa sebagai persentase [9]. Dinyatakan dengan rumus :

$$a : b \implies \frac{a}{b} \qquad \textbf{(2-10)}$$

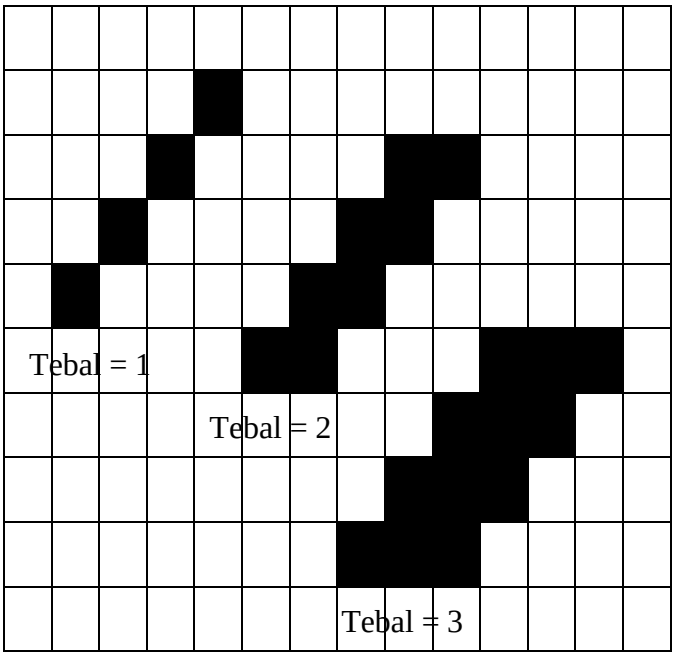
2.11.4. Atribut Garis

Atribut garis menentukan bagaimana sebuah garis atau lingkaran ditampilkan di layar. Beberapa atribut yang cukup penting bagi sebuah garis atau lingkaran adalah [5]:

1. Tebal Garis
2. Tipe Garis
3. Warna

2.11.4.1. Tebal Garis

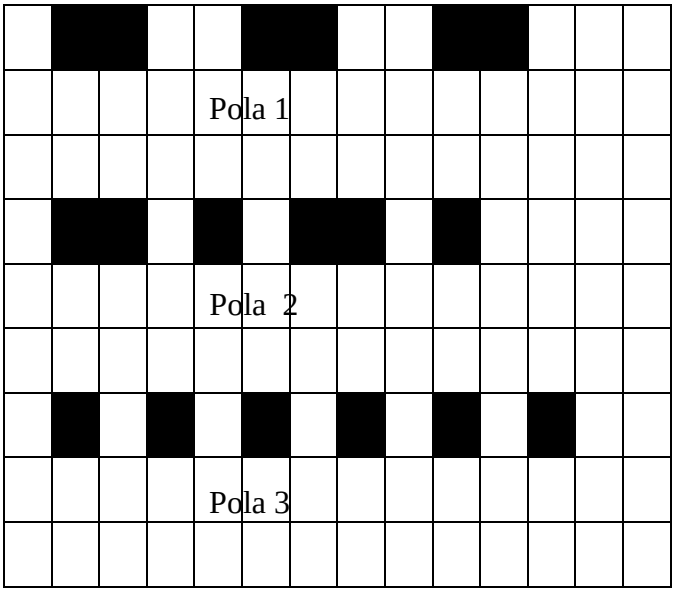
Tebal garis menyatakan berapa banyak *pixel* yang akan digunakan untuk menggambar satu titik dari garis. Gambar 2.16 Menunjukkan pengaruh tebal garis terhadap tampilan garis.



Gambar 2.16 Tebal garis dan *pixel* yang digunakan

2.11.4.2. Tipe Garis

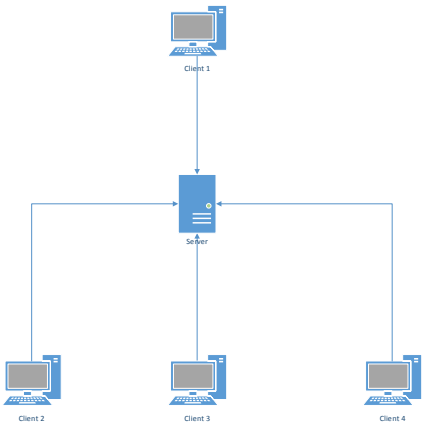
Tipe garis meyatakan pola *pixel* yang digunakan untuk menampilkan garis. Pola garis dapat dinyatakan sebagai sebuah rangkaian angka 0 dan 1, dengan 0 (nol) menyatakan tidak menggambar *pixel* dan 1 (satu) menyatakan menggambar *pixel*. Gambar 2.17 menunjukkan pengaruh dari pola garis terhadap tampilan sebuah garis.



Gambar 2.17 Pola garis dan *pixel* yang digunakan

2.12. *Client – Server*

server adalah komputer *database* yang berada di pusat, di mana informasinya dapat digunakan bersama-sama oleh beberapa *user* yang menjalankan aplikasi di dalam komputer lokalnya yang disebut dengan *client* [10].



Gambar 2.18 Jaringan *Client – Server*

2.13. Pemodelan Analisis

Model analisis merupakan sebuah representasi dari sebuah sistem yang dibangun, ada dua jenis pemodelan yang saat ini mendominasi dalam pembangunan sebuah sistem yaitu pemodelan menggunakan analisis terstruktur dan analisis berorientasi objek.

Ada enam kriteria yang dikandung oleh analisis terstruktur ini yaitu [11]:

1. Titik awal yang jelas, baik bagi analis yang membuat spesifikasi maupun bagi siapapun yang membacanya,
2. Tempat yang jelas bagi setiap informasi kebutuhan yang relevan serta hubungan yang jelas antar komponen dari spesifikasi sistem,
3. Dihindarinya pengulangan,
4. Konsistensi antar komponen,
5. Dihindarinya ambigu, dan
6. Akhir tertentu yang menyatakan bahwa spesifikasi berakhir.

Ada empat kegiatan yang dilakukan oleh analisis terstruktur ini, yaitu:

1. Perancangan arsitektur

   Perancangan perangkat lunak yang mengacu pada sebuah model analisis yang sesuai (DFD).

2. Perancangan data

   Perancangan aliran data dari sebuah proses yang berjalan pada sebuah perangkat lunak yang dimodelkan dengan ERD.

3. Perancangan antarmuka

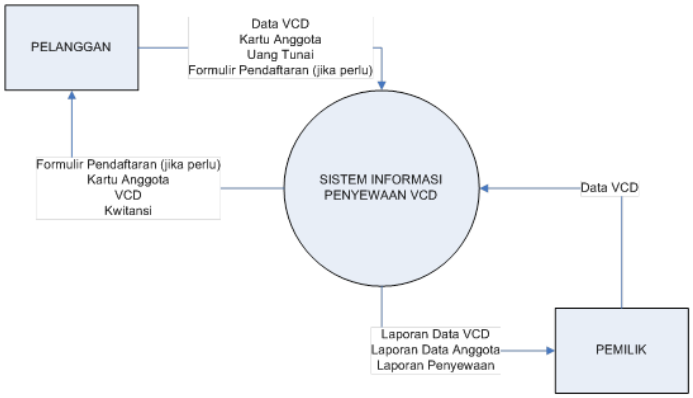
   Merepresentasikan antarmuka yang akan didesain untuk antarmuka prangkat lunak, antarmuka ini juga meliputi setiap fungsi dari tombol-tomboh pada sebuah perangkat lunak.

4. Perancangan prosedural

   Perancangan detil dari setiap fungsi pada perangkat lunak.

2.13.1. Diagram Konteks

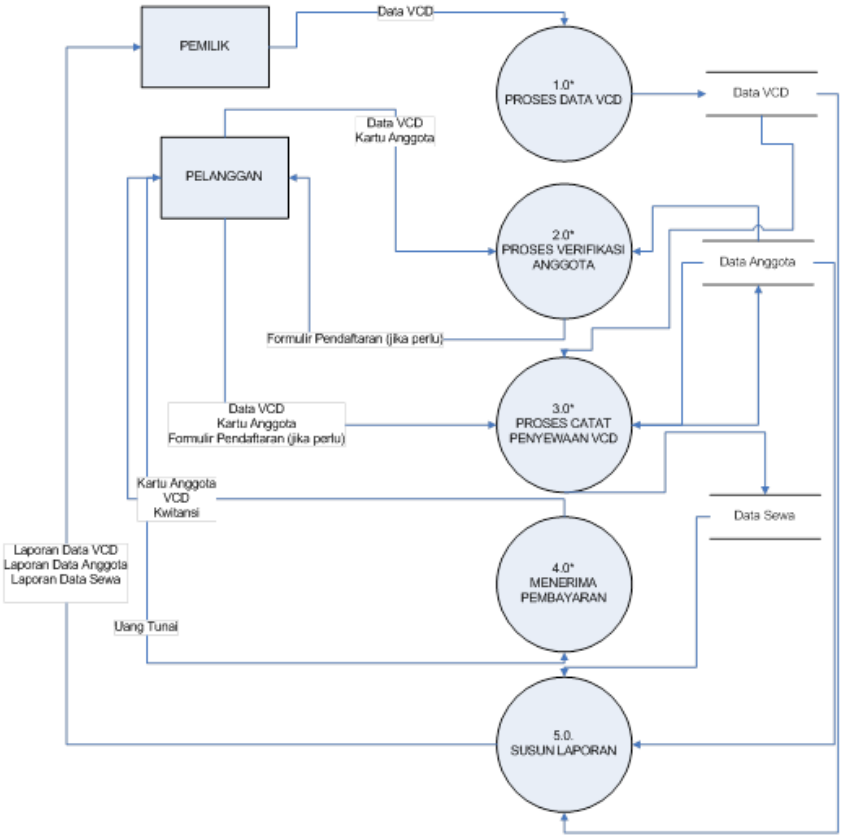
Diagram konteks menggambarkan hubungan antar sistem dengan entitas paling luar. Diagram konteks berfungsi menggambarkan alur *input* data menjadi *output* data dari sebuah proses. Contoh diagram konteks dapat dilihap pada Gambar 2.19:



Gambar 2.19 Contoh Diagram Konteks

2.13.2. Data Flow Diagram (DFD)

DFD adalah suatu model logika data atau proses untuk menggambarkan sebuah aliran data dari asal data ke tujuan data. DFD merupakan rincian dari sebuah diagram konteks yang dirincikan pada setiap level proses. DFD sering digambarkan pada sebuah sistem yang telah ada maupun sistem yang baru akan dikembangkan. Contoh DFD dapat dilihat pada Gambar 2.20 sebagai rincian dari diagram konteks Gambar 2.19:

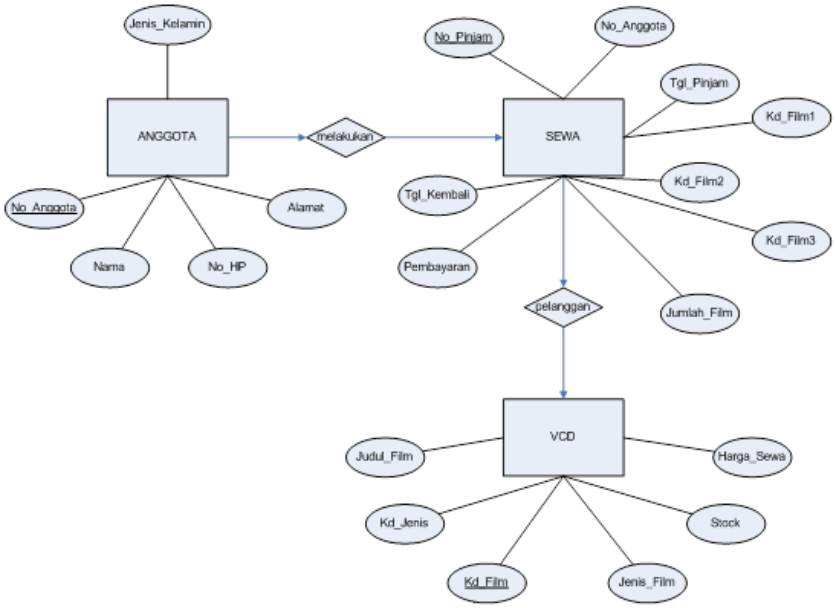


Gambar 2.20 Contoh DFD

2.13.3. Entity-Relationship Diagram (ERD)

ERD adalah sebuah model konseptual yang merepresentasikan sebuah hubungan antara entitas data pada sebuah penyimpanan yang ada pada sebuah sistem yang akan di bangun maupun pada sistem yang sudah ada. Data dimodelkan bertujuan agar mengetahui ke mana sebuah data dan hubungannya bagaimana ini akan memudahkan pada saat pembuatan basis data dan pada pengambilan data

karena dapat melihat hubungan dari sebuah model yang sudah ada. Contoh model ERD dapat dilihat dari Gambar 2.21:



Gambar 2.21 Contoh ERD

2.14. *Aforge.NET Framework*

AForge.NET merupakan libraty *Open-source* C # yang dirancang untuk pengembang dan peneliti di bidang Computer Vision dan Artificial Intelligence - pengolahan gambar, jaringan saraf, algoritma genetika, logika fuzzy, mesin pembelajaran, robotika.

Kerangka ini terdiri dari library dan aplikasi sampel, yang menunjukkan karakteristiknya:

AForge.Imaging - *library* dengan rutinitas pengolahan citra dan filter.

AForge.Vision- *Computer Vision library.*

AForge.Video – library untuk pemrosesan video.

AForge.Neuro – library perhitungan jaringan saraf tiruan.

AForge.Genetic – *evolution computations libarary.*

AForge.Fuzzy – *libarary* perhitungan *fuzzy*.

AForge.Robotics – *Library* yang mendukung robotik

AForge.MachineLearning – *Machines learning library*