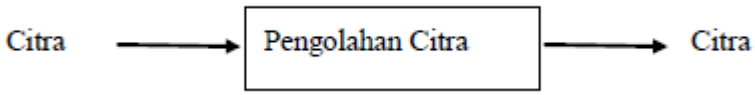
BAB 2
LANDASAN TEORI

2.1 Pengolahan Citra

Secara umum, istilah pengolahan citra digital adalah sebuah pemrosesan gambar berdimensi-dua melalui komputer digital. Pengolahan gambar adalah istilah umum untuk berbagai teknik yang keberadaannya untuk memanipulasi dan memodifikasi gambar dengan berbagai cara. Foto adalah contoh gambar berdimensi dua yang bisa diolah dengan mudah. Setiap foto dalam bentuk gambar digital dapat diolah melalui perangkat-lunak tertentu.

Pengolahan citra adalah salah satu cabang dari ilmu informatika. Pengolahan citra berkutat pada usaha untuk melakukan transformasi suatu gambar menjadi gambar lain dengan menggunakan teknik tertentu. Pengolahan citra merupakan bidang yang bersifat multidisiplin, yang terdiri dari banyak aspek, antara lain fisika, elektronika, matematika, seni dan teknologi komputer. Pengolahan citra/*image processing* memiliki hubungan yang sangat erat dengan disiplin ilmu yang lain. Jika sebuah ilmu disiplin ilmu dinyatakan dengan bentuk proses suatu input menjadi output, maka pengolahan citra memiliki input berupa gambar serta output juga berupa gambar.



Gambar 2. 1 Pengolahan Citra

2.2 Tekstur

Tekstur adalah titik-titik kasar atau halus yang tidak teratur pada suatu permukaan. Titik-titik ini dapat berbeda dalam ukuran ukuran, warna, bentuk atau sifat dan karakternya, seperti misalnya ukuran besar kecil, warna terang gelap, bentuk bulat, persegi atau tak beraturan sama sekali atau lain-lain. Suatu tekstur yang susunannya agak teratur, maka dapat maka dapat disebut sebagai corak (*Pattern*).

Skala, jarak pandang, dan cahaya adalah faktor-faktor penting yang mempengaruhi persepsi kita terhadap tekstur dan permukaan yang ditunjukkannya. Semua material mempunyai tingkat tekstur tertentu. Tetapi, semakin halus skala pola teksturnya, akan semakin halus pula penampilannya.

Skala *relative* suatu tekstur dapat mempengaruhi penampilan dan posisi actual suatu bidang dalam ruang. Tekstur dengan urat-urat yang mempunyai arah tertentu dapat mempertegas panjang atau lebar suatu bidang. Tekstur yang kasar dapat membuat sebuah bidang terlihat seakan-akan lebih dekat, memperkecil skalanya, dan menambah bobot visualnya. Secara umum, tekstur cenderung mengisi secara visual ruang di mana tekstur itu berada.

Tekstur berfungsi untuk dapat memberikan kesan pada persepsi manusia melalui penglihatan visual, seperti misalnya pada suatu bidang rata yang mempunyai perbedaan warna, maka warna yang gelap terlihat sebagai bayangan warna yang terang sehingga timbul kesan seolah-olah bidang tersebut tidak rata [5].

2.3 Analisis Tekstur

Secara umum, analisis tesktur dapat dilakukan dengan tiga pendekatan yaitu pendekatan *struktural, spektral* dan *statistikal*.

Metode analisis *struktural,* dengan metode ini menggunakan deskripsi primitif tekstur dan aturan *sintaktik*. Metode *struktural* banyak digunakan untuk pola-pola *makrostruktur*.

Metode *spektral* berdasarkan pada fungsi *autokorelasi* suatu daerah atau *power distribution* pada domain *transformasi Fourier* dalam mendeteksi periodisitas tekstur.

Metode *statistikal* menggunakan perhitungan statistik distribusi derajat keabuan (*histogram*) dengan mengukur tingkat kekontrasan, granularitas, dan kekasaran suatu daerah dari hubungan ketetanggaan antar piksel di dalam gambar. Paradigma statistik ini penggunaannya tidak terbatas, sehingga sesuai untuk tekstur-tekstur alami yang tidak terstruktur dari sub pola dan himpunan aturan (*mikrostruktur*).

2.4 Grayscale

Proses awal yang banyak dilakukan dalam *image processing* adalah mengubah citra berwarna menjadi citra *grayscale*, hal ini digunakan untuk menyederhanakan model citra. Citra berwarna terdiri dari 3 layer matrik yaitu *Red layer, Green layer* dan *Blue layer*. Sehingga untuk melakukan proses-proses selanjutnya tetap diperhatikan tiga layer di atas. Bila setiap proses perhitungan dilakukan menggunakan tiga layer, berarti dilakukan tiga perhitungan yang sama. Sehingga konsep itu diubah dengan mengubah 3 layer di atas menjadi 1 layer matrik *grayscale* dan hasilnya adalah citra *grayscale*.

Ada kasus bagaimana representasi nilai RGB (*Red, Green, Blue*) diubah menjadi gambar yang terdiri dari warna putih dan gradiasi warna hitam yang biasa disebut *greyscale*. Untuk mengubah RGB menjadi *greyscale* dapat digunakan rumus sebagai berikut:

$$\text{Greyscale} = 0.299R + 0.587G + 0.114B \qquad (2.1)$$

2.5 Metode LU-Transform

Tekstur merupakan isyarat penting dalam banyak aplikasi visi komputer seperti gambar segmentasi, klasifikasi objek atau bahan, dan sintesis tekstur untuk grafis komputer [6]. Terdapat banyak manfaat dari pemanfaatan metode tekstur deteksi ini dalam sebuah aplikasi, yaitu cepat dalam medeskripsi tekstur sederhana. Meskipun tidak ada definisi formal atau secara matematika, deteksi tekstur sering dianggap sebagai struktur skala kecil dalam gambar, dan banyak deskripsi tentang deteksi tekstur yang berbeda-beda didalam setiap literatur [2].

Tavakoli Targhi dan tim mengusulkan *SVD Transform* [7] dan *Eigen Transform* [8] sebagai metode deteksi tekstur. Deskriptor tekstur ini berasal dari dekomposisi matriks. Ide dasarnya adalah untuk membentuk matriks dari dalam *greyvalues*, baik menghitung nilai tunggal atau *eigen values*, dan membentuk deskripsi sebagai rata-rata nilai tunggal terkecil / nilai *eigen values*. Hal ini menghasilkan satu dimensi deskripsi yang terlihat "kasar" pada bidang gambar.

Prosedur ini diulang untuk semua piksel, atau pada *grid reguler subsampled* [8]. menunjukkan tingkat kesesuaian untuk aplikasi dalam hal deteksi tekstur sebuah objek dan segmentasi sebuah gambar.

Namun, *SVD-transform* dan *Eigen Transform* tidak cukup cepat pada hardware yang ada *off-the-rak*. Hambatannya adalah perhitungan nilai tunggal atau nilai *eigen values* yang cenderung lebih rumit. Dengan kendala menghitung nilai tunggal atau *eigen values*, maka dari sebab itu Tavakoli Targhi dan tim mengusulkan metode baru sebagai perbandingan dari metode *transform* sebelumnya yaitu dengan metode *LU-transform* yang hanya melakukan proses Eliminasi Gauss dalam pencarian nilai *eigen values*. Eliminasi Gauss dalam menentukan *LU-transform* dilakukan proses perhitungan nilai matriks segitiga bawah L (*lower*) dan matriks segitiga atas U (*Upper*). Hal ini juga diketahui bahwa proses Eliminasi Gauss ini jauh lebih cepat daripada harus menemukan nilai-nilai tunggal atau nilai *eigen values* perpikel pada proses *SVD*, dan *Eigen values Transform* pada penelitian sebelumnya [2]. Metode ini sangat baik untuk pelaksanaannya pada *Graphics Processor Unit* (GPU). Untuk ukuran yang lebih kecil dibandingkan metode lain, hal ini membuktikan proses yang lebih cepat, dan memungkinkan pengolahan lainnya akan berlangsung pada CPU.

LU-Transform merupakan beberapa kumpulan perhitungan matrik sederhana mulai dari menentukan matrik A dari sekumpulan matrik pada gambar asli, dengan mengacu pada banyak pikel calon matrik A (w), dan nilai spasi pada stiap pikel (δ), dengan rumus:

$$A(j,i) = \begin{pmatrix} f(j,i) & f(j,i+\delta,) & \cdots & f(j,i+w-\delta,) \\ f(j+\delta,i) & f(j+\delta,i+\delta) & \cdots & f(j+\delta,i+w-\delta) \\ \vdots & \vdots & \ddots & \vdots \\ f(j+h-\delta,i) & f(j+h-\delta,i+\delta) & \cdots & f(i+h-\delta,i+w-\delta) \end{pmatrix}, \quad (2.2)$$

Selanjutnya setelah mendapatkan matrik A lakukan operasi *pivoting matriks*, yaitu sebuah pertukaran baris dengan baris lainya, dilihat dari nilai terbesar yang ada pada baris lainnya. Setelah melakukan pivoting lakukan perhitungan dengan menggunakan LU-Dekomposisi untuk mendapatkan matrik U.

Matrik U tersebut berfungsi untuk memberikan nilai *eigen values* pada gambar yang digunakan dalam proses normalisasi.

$$\Omega(l,w) = \sum_{k=1}^{w} ||u_{kk||}1 \leq l \leq w \qquad (2.3)$$

Nilai *absolute* dari nilai *eigen values* pada matrik U dimulai dari nilai (l) yang telah ditentukan diawal dengan rumus sebagai berikut:

$$||\text{Ukk}|| = \sqrt{\sum UKK^2} \qquad (2.4)$$

Nilai *absolute* dari nilai *eigen values* digunakan untuk normalisasi gambar pada matrik U, dimana:

$$f(i,j) \begin{cases} \text{Gambar asli, } f(i,j) < ||\text{Ukk}|| \\ 0, \; f(i,j) > ||\text{Ukk}|| \end{cases} \qquad (2.5)$$

LU-Transform memiliki beberapa sifat yang berguna untuk: Pertama, menyediakan *output* yang rendah-dimensi yang mudah untuk menyimpan dan memudahkan dalam melakukan perhitungan. Kedua mudah untuk menerapkan bahkan secara paralel dan dengan kebutuhan beberapa parameter untuk menyempurnakan hasilnya. Sifat-sifat pada metode ini membuatnya sangat cocok untuk *real-time segmentasi* dan deteksi obyek.

2.6 Eliminasi Gauss

Matrik menjadi skema yang efisien ketika semua koefisien sistem linear Ax=b berada dalam deret berorde Nx(N+1). Koefisien-koefisien b disimpan dalam kolom N+1 dari deret (yaitu ai, N+1=bi). Tiap baris memuat semua koefisien yang diperlukan untuk menyatakan satu persamaan dalam sistem linear.

Sistem Ax=b, dapat diselesaikan dengan melakukan OBE (operasi-operasi baris elementer) pada matriks lengkap [A,b]. Variabel-variabel xk adalah

pemegang posisi untuk koefisien-koefisien dan dapat dihilangkan sampai akhir perhitungan.

Operasi berikut merupakan operasi baris elementar yang dapat diterapkan pada matriks lengkap dan menghasilkan sistem yang setara, meliputi:

Tumpuan (pivoting) adalah salah satu bentuk penyelesaian eliminasi Gauss dengan menentukan bilangan akk pada posisi (k,k) untuk mengeliminasi xk dalam baris k+1,k+2,…,N. Jika akk=0, maka baris k tidak dapat dipakai untuk menghilangkan elemen-elemen pada kolom k, dan baris k harus ditukar dengan baris lainnya di bawah diagonal untuk memperoleh elemen tumpuan yang tidak nol. Jika ini tidak dapat dilakukan maka sistem persamaan tidak mempunyai selesaian tunggal.

Metode eliminasi gauss memerlukan dua tahap di dalam menyelesaikan suatu sistem persamaan linear. Pertama, tahap eliminasi maju (*forward elimination*) bertujuan mengubah matriks koefisien menjadi matriks segitiga atas. Kedua, adalah subtitusi balik (*back subtitution*) [10].

2.7 Faktorisasi dengan eliminasi gauss

Contoh perhitungan matrik eliminasi gauss sama saja dengan perhitungan LU-Dekomposisi, semua pada intinya mencari nilai matrik segitiga atas L, dan nilai matrik segitga bawah U, sebagai contoh perhitungan dengan faktorisasi gauss, dapat dilihat pada perhitungan dibawah ini, dengan menggunakan rumus 2.6.

$$\boxed{R_{21} - \left(\frac{R21}{R11}\right) R_{11}} \qquad (2.6)$$

Misalkan matriks 3x3 difaktorkan L dan U:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

1. Nyatakan A sebagai A = I A

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

2. Eliminasi matriks A di ruas kanan menjadi matriks segitiga atas U. Tempatkan faktor pengali elemen$_{ij}$ pada posisi i$_{il}$ di matriks I

3. Setelah seluruh proses gauss eliminasi selesai, matriks I menjadi matriks L, dan matriks A di ruas kanan menjadi matriks U.

Contoh:

a. Faktorkan matriks A berikut dengan metode Eliminasi Gauss

$$A = \begin{pmatrix} 4 & 3 & -1 \\ -2 & -4 & 5 \\ 1 & 2 & 6 \end{pmatrix}$$

b. Penyelesaian:

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

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

c. Eliminasi matriks A di ruas kanan menjadi matriks segitiga atas U, dan tempatkan faktor pengali elemen$_{ij}$ pada posisi i$_{ij}$ di matriks L.

$$\begin{pmatrix} 4 & 3 & -1 \\ -2 & -4 & 5 \\ 1 & 2 & 6 \end{pmatrix} \sim \begin{matrix} \\ R_{21}-(\frac{R21}{R11})R_{11} \\ R_{31}-(\frac{R31}{R11})R_{11} \end{matrix} \begin{pmatrix} 4 & 3 & -1 \\ 0 & -2{,}5 & 4{,}5 \\ 0 & 1.25 & 6.25 \end{pmatrix}$$

d. Tempatkan faktor pengali elemen$_{21}$ = $\frac{-2}{4}$ = -0,5 ke i$_{21}$ dan faktor pengali elemen$_{31}$ = $\frac{1}{4}$ = 0,25 ke i$_{31}$ di matriks I

$$\begin{pmatrix} 1 & 0 & 0 \\ -0,5 & 1 & 0 \\ 0,25 & 0 & 1 \end{pmatrix}$$

e. Teruskan proses Eliminasi Gauss.

$$\begin{pmatrix} 4 & 3 & -1 \\ 0 & -2.5 & -5.5 \\ 0 & 1.25 & 6.25 \end{pmatrix} \sim \sim \begin{pmatrix} 4 & 3 & -1 \\ 0 & -2.5 & -5.5 \\ 0 & 0 & 3.5 \end{pmatrix}$$

$R_{32}-(\frac{R32}{R22})R_{22}$

f. Tempatkan faktor pengali elemen$_{32}$ = $\frac{1,25}{-2,5}$ = -0,5 ke i$_{32}$ di matriks I

$$\begin{pmatrix} 1 & 0 & 0 \\ -0,5 & 1 & 0 \\ 0,25 & -0,5 & 1 \end{pmatrix}$$

g. Jadi

$$A= \begin{pmatrix} 4 & 3 & -1 \\ 0 & -2.5 & -5.5 \\ 0 & 1.25 & 6.25 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -0.5 & 1 & 0 \\ 0.25 & -0.5 & 1 \end{pmatrix} \begin{pmatrix} 4 & 3 & -1 \\ 0 & -2.5 & -5.5 \\ 0 & 0 & 3.5 \end{pmatrix}$$

Begitulah contoh cara perhitungan eliminasi gauss untuk mendapatkan matrik L dan U yang berguna untuk pembentukan matrik baru dalam proses tekstur deteksi *LU-Transform*. Namun untuk mendapatkan matrik LU ini tidak terbatas hanya dengan cara gauss elimansi seperti pada contoh diatas, karena banyak referensi lain untuk mendapatkan matrik LU tersebut, yang lebih mudah untuk dipelajari, yang hasilnya pasti sama saja akan mendapatkan matrik L yang

merupakan matriks segitiga bawah L (lower) dan matrik U merupakan matriks segitiga atas U (Upper).

2.8 Template Matching Correlation

Template Matching Correlation, merupakan algoritma yang sederhana untuk diterapkan pada pengenalan pola gambar digital dan memiliki tingkat keakuratan yang tinggi.

Template Matching sendiri merupakan proses mencari suatu objek pada keseluruhan objek yang berada dalam suatu gambar.

Template Matching Correlation adalah teknik statistik yang digunakan untuk mencari dua variabel/matrik atau lebih yang sifatnya kuantitatif. Algoritma ini biasa digunakan untuk mencocokkan dua matriks gambar, dengan akurasi yang cukup besar/tepat. Tetapi akurasi yang besar tersebut terbayar dengan kekurangannya, yaitu penghitungan dan penyimpanan data yang cukup besar untuk mencapai hasil optimum. Algoritma ini mencocokkan setiap pixel pada suatu matriks gambar digital dengan gambar lain yang menjadi template acuan (data latih).

Algoritma *Template Matching Correlation* tersebut akan dijabarkan sebagai berikut :

$$r = \frac{\sum_{k=1}^{n}(x_{ik} - x_i) * (x_{jk} - x_j)}{\sqrt{[\sum_{k=1}^{n}(x_{ik} - x_i)^{\,2} * \sum_{k=1}^{n}(x_{jk} - x_j)^{\,2}]}} \qquad (2.7)$$

Dimana, xi dan xj merupakan rata-rata dari matriks i dan j yang dapat dihitung dengan:

$$x_i = \frac{1}{n}\sum_{k=1}^{n} x_{ik}$$

$$x_j = \frac{1}{n}\sum_{k=1}^{n} x_{jk} \qquad (2.8)$$

Keterangan :

r = nilai korelasi antara dua buah matriks

xik = nilai pixel ke-k pada matriks i

xjk = nilai pixel ke-k pada matriks j

xi = rata-rata nilai pixel matriks i

xj = rata-rata nilai pixel matriks j

n = jumlah pixel pada suatu matriks

Nilai korelasi berkisar antara -1 hingga 1, semakin mendekati nilai 1 maka tingkat kemiripan semakin besar. Decisions merupakan sebuah proses penentuan klasifikasi gambar yang telah diuji dengan menggunakan algoritma *Template Matcing Correlation.*

2.9 Kelinci

2.9.1 Definisi Kelinci

Kelinci adalah binatang mamalia yang merupakan kerabat dari *Leporidae* yang terkadang disamakan dengan arnab. Kelinci memiliki punggung yang melengkung dan ekornya pendek. Kelinci merupakan hewan peliharaan favorit keluarga karena kelinci hewan yang lucu jinak serta memiliki bulu yang lembut. Selain itu kelinci merupakan hewan penghasil daging *alternatif* selain daging kambing, domba, ayam, sapi yang selama ini menjadi konsumsi utama masyarakat di indonesia.

Namun dalam pemeliharaanya mungkin masih banyak bagi para pencinta kelinci yang sulit untuk mengetahui jenis-jenis penyakit apa yang sering menjangkit pada kelinci tersebut. Sehingga mereka telat dalam penanganannya dan akhirnya menyebabkan kelinci kesayanganya tersebut mati mendadak.

Berdasarkan penjelasan tersebut, dapat disimpulkan bahwa terdapat beberapa jenis penyakit kelinci yang dapat terlihat oleh visual mata peternak tersebut namun mereka masih belum mengetahui jenis penyakit apa dan cara penanganannya seperti apa. Terlebih lagi saat ini masih sulit untuk berkonsultasi dengan dokter hewan, sehingga ini juga merupakan masalah yang masih sulit untuk diselesaikan.

2.9.2 Jenis-Jenis Penyakit kulit Kelinci

2.9.2.1 Scabies

Scabiesis adalah penyakit kulit yang disebabkan tungau (sejenis kutu) *scabies/sarcoptes*. Penyakit ini sering menyerang anjing, kucing, kelinci dan dapat juga menular ke manusia. Pada kelinci biasanya scabies ini disebabkan oleh Penyakit kulit yang disebabkan oleh ektoparasit *Sarcoptes scabiei* dan *Notoedres cati*.

Tanda-tanda awal terkena penyakit ini biasanya berupa rontok dan gatal disekitar telinga. Dipinggiran daun telinga terlihat ada kerak berwarna putih. Penyakit dapat menyebar dengan cepat ke daerah sekitar wajah, leher, hidung dan kelopak mata. Kadang-kadang tungau juga dapat menyebar hingga ke daerah perut dan telapak kaki. Rasa gatal yang timbul menyebabkan kelinci sering menggaruk-garuk. Infeksi kronis/lama dapat menyebabkan penebalan dan keriput pada kulit ditutupi oleh kerak-kerak berwarna abu-abu kekuningan. Infeksi yang parah mengakibatkan luka dan berkembang menjadi infeksi sekunder.



Gambar 2. 2 Penyakit scabies pada kelinci

2.9.2.2 Sore Hocks

Sore hocks bisa sangat menyakitkan bagi kelinci. Sore hocks tidak terbatas hanya breeds besar, namun dapat terjadi pada setiap jenis kelinci.

Sore hocks umumnya muncul pada kaki belakang. Biasanya, bulu akan hilang dari kaki dekat persendian, area kaki belakang dan di bagian bawah. Juga akan ada kemerahan di daerah itu memanjang sampai ke kaki. Jika tidak diobati

infeksi akan berkembang dan luka terbuka dapat terlihat. infeksi yang tidak diobati dapat mengakibatkan infeksi bakteri, dan akhirnya kematian kelinci.

Umumnya penyebabnya adalah kondisi yang tidak sehat dalam tempat hidup kelinci. Akumulasi kotoran di kandang, atau alas yang basah atau lembab. Untuk kelinci yang lebih besar, masalah dapat dikaitkan dengan dasar kandang kawat yang menyebabkan luka karena berat kelinci. Mungkin juga akibat kuku yang panjang dan kecenderungan turun temurun.



Gambar 2. 3 Penyakit sore hocks pada kelinci

2.10 Object Oriented Programming

Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Metode berorientasi objek didasarkan pada penerapann prinsip-prinsip pengelolaan kompleksitas. Metode berorientasi objek meliputi rangkaian aktifitas analisis beorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek.

Pada saat ini, metode berorientais objek banyak dipilih karena metodologi lama banyak menimbulkan masalah seperti adanya kesulitan pada saat mentranformasi hasil dari satu tahap pengembangan ke tahap berikutnya, misalnya pada metode pendekatan terstruktur, jenis aplikasi yang dikembangkan saat ini berbeda dengan masa lalu. Aplikasi yang dikembangkan saat ini sangat beragam (aplikasi bisnis, real-time, utility dan sebagainya) dengan platform yang

berbeda-beda, sehingga menimbulkan tuntutan kebutuhan metodologi pengembangan yang dapat mengakomodasi ke semua jenis aplikasi.

Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut:

a. Meningkatkan Produktivitas, Karena kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk masalah lainnya yang melibatkan objek tersebut (reuseable).
b. Kecepata Pengembangan, sebuah sistem yang dibangun dengan baik dan benar pada saat analisis dan perancangan akan mennyababkan berkurangnya kesalahan pada saat pengkodean.
c. Kemudahan Pemeliharaan, dengan model objek, pola-pola yang cendrung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering diubah-ubah. Adanya Konsistensi, dengan sifat pewarisan dan penggunaan notasi yang sama pada saat analisis, perancangan maupun pengkodean.
d. Meningkatkan Kualitas Perangkat Lunak, adanya pendekatan pengembangan lebih dekat dengan dunia nyata dan adanya konsistensi pada saat pengambangannya, perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

2.10.1 Konsep Dasar

Berikut adalah konsep dasar pemrograman berorientasi objek :

1. Kelas Abstrak (Class Abstraksi)

   Kelas merupakan deskripsi abstrak informasi dan tingkah laku dari sekumpulan data. Kelas dapat diilustrasikan sebagai suatu cetak biru(blueprint) atau prototipe yang digunakan untuk menciptakan objek. Kelas merupakan tipe data bagi objek yang mengenkapsulasi data dan operasi pada data dalam suatu unit tunggal. Kelas mendefinisikan suatu struktur yang terdiri atas data kelas (data field), prosedur atau fungsi (method), dan sifat kelas (property).

2. Enkapsulasi (encapsulation)

   Istilah enkapsulasi sebenarnya adalah kombinasi data dan fungsionalitas dalam sebuah unit tunggal sebagai bentuk untuk menyembunyikan detail informasi. Proses enkapsulasi memudahkan kita untuk menggunakan sebuah objek dari suatu kelas karena kita tidak perlu mengetahui segala hal secara rinci. Enkapsulasi menekankan pada antarmuka suatu kelas, atau dengan kata lain bagaimana menggunakan objek kelas tertentu. Contoh: kelas mobil menyediakan antarmuka fungsi untuk menjalankan mobil tersebut, tanpa kita perlu tahu komposisi bahan bakar, udara dan kalor yang diperlukan untuk proses tersebut.

3. Pewarisan (Inheritance)

   Kita dapat mendefinisikan suatu kelas baru dengan mewarisi sifat dari kelas lain yang sudah ada. Penurunan sifat ini bisa dilakukan secara bertingkattingkat, sehingga semakin ke bawah kelas tersebut menjadi semakin spesifik. Sub kelas memungkinkan kita untuk melakukan spesifikasi detail dan perilaku khusus dari kelas supernya. Dengan konsep pewarisan, seorang programmer dapat menggunakan kode yang telah ditulisnya pada kelas super berulang kali pada kelas-kelas turunannya tanpa harus menulis ulang semua kode-kode itu.

4. Polimorfisme (polymorphism)

   Polimorfisme merupakan kemampuan objekobjek yang berbeda kelas namun terkait dalam pewarisan untuk merespon secara berbeda terhadap suatu pesan yang sama.

   Polimorfisme juga dapat dikatakan kemampuan sebuah objek untuk memutuskan *method* mana yang akan diterapkan padanya, tergantung letak objek tersebut pada jenjang pewarisan.

2.11 UML (Unified Modelling Language)

Unified Modelling Language (UML) adalah sekumpulan spesifikasi yang dikeluarkan oleh OMG. UML terbaru adalah UML 2.3 yang terdiri dari 4 macam spesifikasi, yaitu : Diagram *Interchange Spesification*, UML *Infrastrukture*, UML

Superstrukture, dan *Object Constraint Language* (OCL). Pada UML 2.3 terdiri 13 macam diagram yang dikelompokan pada 3 kategori, yaitu:

a. *Behavior Diagram*, yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem. Pada *Behavior Diagram* dibagi menjadi 3 bagian :

   1. *Use Case Diagram*

      Use case diagram merupakan pemodelan untuk kelakuan (behavior) sistem informasi yang akan dibuat,. Use case mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat

   2. *Activity Diagram*

      Activity diagram menggambarkan *workflow* atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak.

   3. *State Machine Diagram*

      State machine diagram digunakan untuk menggambarkan perubahan status atau transisi status dari sebuah mesin atau sistem atau objek.

b. *Interactions Diagram*, yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi antar subsistem pada suatu sistem. *Pada Interactions Diagram* dibagi menjadi 4 bagian:

   1. *Sequence Diagram*

      Sequence diagram menggambarkan kelakuan objek pada *use case dengan* mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek.

   2. *Communication Diagram*

      Communication diagram menggambarkan interaksi antar bojek/bagian dalam bentuk urutan pengiriman pesan. Diagram komunikasi merepresentasikan informasi yang diperoleh dari diagram kelas, diagram sequence, dan diagram use case untuk mendeskripsikan gabungan antara struktur statis dan tingkah laku dinamis dari suatu sistem.

3. *Timing Diagram*

   Timing diagram merupakan diagram yang fokus pada penggambaran terkait batas waktu.

4. *Interaction Overview Diagram*

   Interaction overview diagram mirip dengan diagram aktivitas yang berfungsi untuk menggarbarkan sekumpulan urutan aktivitas, diagram ini adlaah bentuk aktivias diagram yang setiap titik merepresentasikan diagram interaksi.

c. Struktur Diagram, yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan. Pada Struktur Diagram dibagi menjadi 6 bagian:

   1. Diagram kelas

      Diagram kelas menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut attribut dan metode atau operasi.

   2. Diagram Objek

      Digram objek menggambarkan struktur sistem dari segi penamaan objek dan jalannya objek dalam sistem.

   3. Diagram Komponen

      Diagram komponen dibuat untuk menunjukan organisasi dan ketergantungan diantara kumpulan komponen dalam sebuah sistem.

   4. *Composite Structure Diagram*

      Composite Structure Diagram bru muali ada pada UML versi 2.0. diagram ini dapat digunakan untuk menggambarkan struktur dari bagian-bagian yang saling terhubung maupun mendeskripsikan struktur pada saat berjalan (runtime).

   5. *Package Diagram*

      Package diagram menyediakan cara mengumpulkan elemen-elemen yang saling terkait dalam diagram UML. Hampir semua diagram dalam UML dapat dikelompokan menggunakan *Package Diagram*.

6. *Deployment Diagram*

   Deployment diagram menunjukan konfigurasi komponen dalam proses eksekusi aplikasi.

2.12 K-Fold Cross Validation

Validasi silang dan *bootstrapping* merupakan metode dalam memperkirakan generalisasi *error* berdasarkan resampling [12]. Metode *k-fold cross validation* membagi sebuah himpunan contoh secara acak menjadi k subset yang saling bebas. Dilakukan pengulangan sebanyak k-kali untuk pelatihan dan pengujian. Pada setiap ulangan, disisakan satu subset untuk pengujian dan subset lainnya untuk pelatihan. Tingkat akurasi dihitung dengan membagi jumlah keseluruhan klasifikasi yang benar dengan jumlah semua instance pada data awal.

Cross Validation merupakan salah satu teknik untuk menilai/memvalidasi keakuratan sebuah model yang dibangun berdasarkan dataset tertentu . Pembuatan model biasanya bertujuan untuk melakukan prediksi maupun klasifikasi terhadap suatu data baru yang boleh jadi belum pernah muncul di dalam dataset. Data yang digunakan dalam proses pembangunan model disebut data latih/training, sedangkan data yang akan digunakan untuk memvalidasi model disebut sebagai data test.

Salah satu metode *cross-validation* yang populer adalah *K-Fold Cross Validation*. Dalam teknik ini dataset dibagi menjadi sejumlah K-buah partisi secara acak. Kemudian dilakukan sejumlah K-kali eksperimen, dimana masing-masing eksperimen menggunakan data partisi ke-K sebagai data testing dan memanfaatkan sisa partisi lainnya sebagai data training. Sebagai gambaran, jika kita melakukan 5-*Fold Cross-Validation* maka desain data eksperimennya sebagai berikut:

Tabel 2. 1 Jumlah dataset

Data Set				
K1	K2	K3	K4	K5

Tabel 2. 2 Data eksperimen

Eksperimen ke	Data lattih	Data tes
1	K2, K3, K4, K5	K1
2	K1, K3, K4, K5	K2
3	K1, K2, K4, K5	K3
4	K1, K2, K3, K5	K4
5	K1, K2, K3, K4	K5

Untuk mendapatkan nilai akurasi ataupun ukuran penilaian lainnya dari hasil eksperimen yang kita lakukan, dapat diambil nilai rataan dari seluruh eksperimen tersebut.

2.13 C#

C# merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft sebagai bagian dari inisiatif kerangka *.NET Framework*. Bahasa pemrograman ini dibuat berbasiskan bahasa C++ yang telah dipengaruhi oleh aspek-aspek ataupun fitur bahasa yang terdapat pada bahasa-bahasa pemrograman lainnya seperti *Java, Delphi, Visual Basic*, dan lain-lain, dengan beberapa penyederhanaan.

Sifat-sifat bahasa pemograman C#:

1. *Simple*

   C# bersifat sederhana, karena bahasa ini didasarkan kepada bahasa C dan C++. Jika anda familiar dengan C dan C++ atau bahkan Java, anda akan menemukan aspek-aspek yang begitu familiar, seperti statements, expression, operators, dan beberapa fungsi yang diadopsi langsung dari C dan C++, tetapi dengan berbagai perbaikan yang membuat bahasanya menjadi lebih sederhana.

2. Object Oriented Language

C# memenuhi syarat-syarat sebagai sebuah bahasa pemrograman yang bersifat *Object Oriented*, yaitu *encapsulation, inheritance* dan *polymorphism.*

3. Powerfull dan Fleksibel

C# bisa digunakan untuk membuat berbagai macam aplikasi, seperti aplikasi pengolah kata, grapik, *spreadsheets*, atau bahkan membuat kompiler untuk sebuah bahasa permrograman.

4. Efficient

C# tidak memiliki terlalu banyak keyword, sehingga dapat mengurangi kerumitan.

5. Modular

Kode C# ditulis dengan pembagian masing *Class-Class* (classes) yang terdiri dari beberapa routines yang disebut sebagai member methods. *Class-Class* dan metode-metode ini dapat digunakan kembali oleh program atau aplikasi lain. Hanya dengan memberikan informasi yang dibutuhkan oleh *Class* dan metode yang dimaksud, maka kita akan dapat membuat suata kode yang dapat digunakan oleh satu atau beberapa aplikasi dan program (reusable code).

2.14 Visual Studio 2012

Microsoft Visual Studio merupakan sebuah perangkat lunak lengkap (suite) yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya, dalam bentuk aplikasi *console,* aplikasi *Windows*, ataupun aplikasi *Web*. Visual Studio mencakup kompiler, SDK, *Integrated Development Environment* (IDE), dan dokumentasi (umumnya berupa MSDN Library). Kompiler yang dimasukkan ke dalam paket Visual Studio antara lain Visual C++, Visual C#, Visual Basic, Visual Basic .NET, Visual InterDev, Visual J++, Visual J#, Visual FoxPro, dan Visual SourceSafe.

Microsoft Visual Studio dapat digunakan untuk mengembangkan aplikasi dalam native code (dalam bentuk bahasa mesin yang berjalan di atas Windows) ataupun managed code (dalam bentuk *Microsoft Intermediate Language* di atas .NET *Framework*). Selain itu, Visual Studio juga dapat digunakan untuk mengembangkan aplikasi *Silverlight*, aplikasi *Windows Mobile* (yang berjalan di atas .NET *Compact Framework*).