BAB 2
LANDASAN TEORI

2.1 Sistem Pengendalian

Sistem pengendalian adalah susunan suatu komponen yang dihubungkan sedemikian rupa untuk mengatur suatu kondisi agar mencapai kondisi yang diharapkan. Sistem pengendalian ini secara umum terdiri dari tiga elemen pokok, yaitu input, proses dan output. Output merupakan hal yang dihasilkan oleh kendalian, artinya yang dikendalikan. Sedangkan input adalah yang mempengaruhi kendalian yang mengatur output.

2.2 Pengolahan Citra Digital

Menurut Munir . R. [3], menjelaskan pengolahan citra adalah pemrosesan citra, khususnya menggunakan komputer, menjadi citra yang kualitasnya lebih baik . Meskipun sebuah citra kaya akan informasi, namun sering kali mengalami penurunan mutu, misalnya mengandung cacat atau derau (*noise*), warnanya terlalu kontras, kurang tajam, kabur dan sebagainya. Citra semacam ini sulit untuk diinterpretasikankarena informasi yang disampaikan oleh citra tersebut berkurang. Dengan pengolahan citra, citra yang mengalami gangguan tersebut dimanipulasi menjadi citra lain yang kualitasnya lebih baik sehingga mudah diinterpretasikan oleh manusia maupun mesin.

2.3 *Principal Component Analysis* (PCA)

PCA merupakan algoritma yang di dalamnya terdapat eigenface. “Fitur-fitur suatu citra atau gambar citra menggunakan PCA dipresentasikan sebagai *eigenvector*. *Eigenvector* didefinisikan sebagai kumpulan hubungan karakteristik dari suatu citra untuk mengenali citra tersebut secara spesifik. Setelah matriks kovarian(L) dihitung, langkah selanjutnya adalah mencari *eigenvalue* dan *eigenvector*. *Eigenvalue* yang didapat kemudian diturunkan mulai yang terbesar sampai yang terkecil, *eigenvector* yang bersesuaian dengan *eigenvalue* tersebut juga diurutkan.” [4].

Perhitungan PCA dimulai dengan pembentukan matriks kovarians dari sekumpulan citra latih dalam basis-data. Untuk membuat matriks kovarians dapat dilakukan dengan mengurangkan setiap kolom dari matriks ruang citra dengan reratanya masing-masing sehingga didapatkan suatu matriks transformasi, kemudian mengalikan matriks hasilnya dengan transpos matriks tersebut sendiri. Jika dimisalkan matriks transformasinya adalah X, maka matriks kovariansnya adalah $\Sigma X = XX^T$. Ide utama PCA adalah menemukan vektor dengan nilai terbaik untuk distribusi citra wajah dalam seluruh ruang citra. Vektor-vektor ini mendefinisikan subruang citra wajah atau biasa disebut dengan nama ruang wajah. Vektor ini mendefinisikan subruang dari citra-citra wajah dan subruang tersebut dinamakan ruang wajah. Identifikasi suatu citra uji, membutuhkan proyeksi suatu citra ke dalam ruang wajah untuk menentukan korespondensi kumpulan bobot-bobot. Dengan membandingkan kumpulan bobot-bobot wajah dalam training set, pengujian citra dapat diidentifikasi.

Principal Component Analysis (PCA) dikenal juga dengan metode *Karhunen-Loeve Transformation* (KLT). Dimana metode PCA merupakan teknik standar yang digunakan dalam pengenalan pola statistikal dan pemrosesan sinyal untuk data *reduction* dan ekstraksi fitur. Sebagai pola statistical sering mengandung informasi yang berlebihan, pada pemetaan sebuah vector dapat menyingkirkan redudansi namun sebagian memperbesar instrinsik pola Sebuah gambar wajah dalam 2D dengan ukuran N x N dapet dianggap sebagai vector dimensi . Gambar wajah yang mirip secara keseluruhan konfigurasi tidak akan secara acak didistribusikan dalam ruang gambar yang besar dengan demikian dapat dijelaskan oleh subruang dimensi yang relative lebih rendah. Pemikiran utama dari prinsip komponen untuk menentukan vektor yang paling cocok untuk distriusi citra wajah dalam keseluruhan ruang gambar vector.subruang dari citra wajah, yang kita sebut "*face space*". Masing-masing vektor tersebut adalah panjang , menjelaskan$N \times N$ gambar,dan merupakan kombinasilinear dari gambar wajah aslinya. Karena vektor ini adalah vektor eigen dari matriks kovarians sesuai dengan gambar wajah aslinya.

PCA memberikan transformasi ortogonal yang disebut juga dengan *eigenimage* yang di mana sebuah citra akan direpresentasikan ke dalam bentuk proyeksi linier searah dengan *eigenimage* yang bersesuaian dengan nilai *Eigen* terbesar dari matriks kovarian. Dalam prakteknya, matriks kovarian ini dibangun dari sekumpulan *image training* yang diambil dari berbagai objek.Sebuah citra 2D dengan dimensi kolom dan baris dapat direpresentasikan ke dalam bentuk citra 1D. Dalam penelitian ini ukuran jumlah kolom dan baris pixel citra adalah sama, sehingga nantinya akan terbentuk dimensi .

Bentuk umum dari *Principal Component Analysis (PCA)* adalah sebagai berikut :

$$C = \sum_{k=1}^{K} (x_x - \Psi)(x_x - \Psi)^T$$

Tujuan yang ingin dicapai adalah membangun sebuah ruang wajah, dengan setiap komponennya tidak saling berkorelasi. Ini berarti bahwa matriks kovarians dari komponen yang baru haruslah berbentuk diagonal.

2.4 *Face Recognition*

Prinsip dasar pengenalan wajah adalah mengutip informasi unik wajah, kemudian di-*encode* dan dibandingkan dengan hasil *decode* yang sebelumnya dilakukan. Pada metode *eigenface*, *decoding* dilakukan dengan menghitung *eigen vector* kemudian direpresentasikan dalam sebuah matriks. *Eigen vector* juga dinyatakan sebagai karakteristik wajah oleh karena itu metode ini disebut dengan *eigenface*. Setiap wajah direpresentasikan dalam kombinasi linear *eigenface*. Metode *eigenface* adalah bagaimana cara meguraikan informasi yang relevan dari sebuah citra wajah, kemudian mengubahnya ke dalam satu set kode yang paling efisien dan membandingkan kode wajah tersebut dengan database berisi beragam wajah yang telah dikodekan secara serupa.

Eigenfaces PCA digunakan untuk mereduksi dimensi sekumpulan atau ruang gambar sehingga basis atau sistem koordinat yang baru dapat menggambarkan model yang khas dari kumpulan tersebut dengan lebih baik.

Model yang diinginkan merupakan sekumpulan wajah yang dilatihkan. Fitur yang baru tersebut akan dibentuk melalui kombinasi linear. Komponen fitur ruang karakter ini tidak akan saling berkolerasi dan akan memaksimalkan perbedaan yang ada pada variabel aslinya.

2.5 Metode *Eigenface*

Kata Eigenface sebenarnya berasal dar bahasa Jerman "eigenvert" dimana "eigen" yang artinya karakteristik dan "wert" artinya nilai. *Eigenface* adalah salah satu algoritma pengenalan pola wajah yang berdasarkan pada *Principal Component Analysis* (PCA).

Eigenface merupakan sekumpulan *standardize dace ingredient* yang diambil dari analisis statistik dari banyak gambar wajah. Dalam menghasilkan *Eigenface*, sekumpulan citra digital dari wajah manusia diambil pada kondisi pencahayaan yang sama kemudian dinormalisasikan dan diproses pada resolusi yang sama (misal m x n), kemudian citra tadi diperlakukan sebagai vektor dimensi m x n dimana komponennya diambil dari nilai piksel citra [5].

Algoritma *Eigenface* secara keseluruhan cukup sederhana. Image matriks (). Cari nilai rata-rata (Ψ) dan gunakan untuk mengekstraksi *eigenvector*(v) dan *eigenvalue*(λ) dari himpunan matriks. Gunakan nilai eigenvector untuk mendapatkan nilai *Eigenface* dari *image*. Apabila ada sebuah *image* baru atau *test face* () yang ingin dikenali, proses yang sama juga diberlakukan untuk *image* (), untuk mengektraksi *eigenvector* (v) dan *eigenvalue* (λ), kemudian cari nilai *Eigenface* dari image *test face* (). Setelah itu barulah *image* baru () memasuki tahapan pengenalan dengan menggunakan metode *euclidean distance*. Metode *euclidean distance* digunakan untuk mencari jarak dengan data fitur yang telah didapat, dan jarak terkecil adalah hasilnya.

Pengenalan wajah dengan pendekatan *Eigenface* dapat digolongkan dalam suatu metoda *appearance-based*, ini dikarenakan pengenalan wajah Eigenface menggunakan informasi mentah dari *pixel* citra yang kemudian direpresentasikan dalam metode tertentu (misalnya *Principal Component Analysis* (PCA),

transformasi *wavelet*, dll) yang selanjutnya digunakan untuk pelatihan dan klasifikasi indentitas citra. Ide dari metode ini adalah memproyeksikan sebuah wajah yang merupakan sebuah citra dapat dilihat sebagai sebuah vektor.

Metode PCA yang juga dikenal dengan nama *Karhunen-Loeve transformation* (KLT), sudah dikenal sejak 30 tahun lalu dalam dunia pengenalan pola. PCA memberikan transformasi ortogonal yang disebut dengan "*eigenimage"* yang mana sebuah *image* direpresentasikan kedalam bentuk proyeksi linier searah dengan *eigenimage* yang bersesuaian dengan nilai *eigen* terbesar dari matrix kovarian. Secara praktis matrix kovarian ini dibangun dari sekumpulan *image training* yang diambil dari berbagai obyek/kelas.

Pada metode ini setiap citra wajah dianggap sebagai vektor yang panjang, nantinya akan di representasikan sebagai satu titik pada dimensi tinggi. Karena fitur yang akan dihasilkan terletak dalam dimensi tinggi maka perlu dilakukan reduksi dimensi-dimensi fitur yaitu, *Principal Component Analysis* (PCA) [6]. Selain itu model wajah dipelajari melalui proses training dengan menggunakan satu set data pelatihan yang berisi contoh-contoh wajah. Kemudian hasil training ini digunakan untuk pengenalan wajah.

Proses identifikasi image wajah adalah sebagai berikut:

1. Pengambilan Citra Wajah

Pengambilan citra wajah adalah proses pengambilan wajah yang akan digunakan sebagai sistem keamanan rumah. Pengambilan citra wajah menggunakan *Web Camera* atau yang sebagaimana biasa disingkat sebagai *webcam*. *Webcam* merupakan kamera kecil yang terdapat di komputer. Alat ini digunakan untuk mengambil gambar untuk digunakan sebagai sistem keamanan rumah dalam aplikasi ini.

Pengenalan wajah adalah suatu pengenalan pola yang khusus untuk kasus wajah. Ataupun sebagai pengklasifikasian suatu wajah apakah dikenali (*know*)

atau tidak dikenali (*unkwon*), dimana setelah dibandingkan kemudian disimpan secara tersendiri.

Pada sistem dibagi menjadi dua, yaitu sistem *feature based* dan *image based.* Sistem yang pertama digunakan fitur yang diekstraksi dari komponen citra wajah (mata, hidung, mulut, dll) kemudian hubungan antara fitur-fitur tersebut dimodelkan secara geometris. Sedangkan sistem kedua menggunakan informasi mentah dari piksel citra (*webcam*) kemudian dipresentasikan dalam metode tertentu seperti *Principal Component Analysis* (PCA), transformasi *wavelet* yang kemudian digunakan untuk klasifikasi identitas citra.

Pada metode ini, model wajah dipelajari melalui proses training dengan menggunakan satu set data pelatihan yang berisi contoh-contoh wajah. Kemudian hasil training ini digunakan untuk pengenalan wajah. Yang termasuk dengan metode *Eigenface.*

Algoritma *Eigenface* memanfaatkan *Principal Component Analysis* (PCA) untuk mereduksi dimensinya guna menemukan vektor-vektor yang mempunyai nilai terbaik untuk distribusi citra wajah didalam ruang citra masukan.

PCA menemukan vektor dengan nilai terbaik untuk distribusi citra wajah dalam seluruh ruang citra. Vektor-vektor ini mendefenisikan subruang citra wajah atau biasa disebut dengan ruang wajah.

2. Input Citra

Input citra merupakan proses memasukkan gambar yang diambil dari webcam. Aplikasi ini membutuhkan input pengambilan gambar wajah sebanyak 10 kali. Jika gambar pemilik rumah atau orang yang sudah dikenal sudah terdapat didatabase, maka gambar wajah yang diinput akan disesuaikan dengan gambar yang sudah ada didatabase, dan jika sesuai maka tidak akan ada pemberitahuan kepada pemilik rumah. Jika tidak sesuai, maka sistem akan memberitahukan kepada pemilik rumah melalui *sms gateway* bahwa ada orang tidak dikenal mencoba memasuki rumah tersebut.

3. *Dilation*

Jika gambar wajah belum terdapat didatabase, maka proses akan berlanjut ke *Dilation*. Proses ini merupakan proses normalisasi *image* (gambar). Pada proses ini ukuran dari gambar wajah yang diinput akan diseragamkan dengan menggunakan ukuran 90 x 120 piksel.

4. *Grayscale*

Tahapan berikutnya adalah pengubahan warna gambar menjadi *grayscale*. Tahapan ini termasuk ke dalam tahapan normalisasi image juga.Pada tahapan ini, gambar yang telah diseragamkan ukurannnya akan diturunkan kualitas warnanya menjadi tipe grayscale. Tujuan dari penurunan kualitas warna menjadi tipe grayscale adalah untuk memudahkan pengenalan pola wajah.

5. *Eigenface Feature Extraction*

Pada metode ini setiap citra wajah dianggap sebagai vektor yang panjang, nantinya akan di representasikan sebagai satu titik pada dimensi tinggi. Karena fitur yang akan dihasilkan terletak dalam dimensi tinggi maka perlu dilakukan reduksi dimensi-dimensi fitur yaitu, *Principal Component Analysis* (PCA) [6],

a. Kalkulasi *Eigenface*

Tujuan kalkulasi *Eigenface* adalah untuk menentukan nilai *Eigenface* dari *image* wajah.

b. Hitung *Euclidean Distance*

Setelah nilai dari kalkulasi *Eigenface* diperoleh, maka tahapan selanjutnya adalah menghitung *Euclidean Distance* untuk mencari selisih terkecil antara titik yang satu dengantitik lain. Setelah proses eigenface selesai, maka pola gambar wajah akan tersimpan di database.

6. *Matching*

Pada tahapan ini, aplikasi akan menyesuaikan pola wajah untuk memonitoring sistem keamanan rumah.

2.6 *Microsoft Visual C#*

Microsoft Visual C-Sharp atau yang lebih dikenal dengan *Visual C#* adalah sebuah bahasa yang tidak diragukan lagi dalam proses pengembangan

aplikasi berbasis *.NET Framework*, dimana *C#* bebas dari masalah kompabilitas dilengkapi dengan berbagai fitur yang sebagian besar merupakan fitur baru, menarik, dan tentu saja menjanjikan.

Visual C# dibuat berdasarkan pemrograman *C#* yang merupakan bahasa pemrograman berorientasi objek dan mempunyai banyak kesamaan dengan C++, Java, dan VB. C# pada faktanya merupakan kombinasi antara efisiensi pemrograman C++, kesederhanaan pemrograman Java, dan penyederhanaan dari pemrograman Visual Basic [7].

Seperti pemrograman Java, C# juga tidak memperbolehkan *multiple inheritance* atau pengunaan *pointer* (pada *safe/managed code*), tetapi C# menyediakan *garbage memory colletion* pada saat *runtime* dan pada saat pengecekan akses memori. Meskipun bertentangan dengan pemrograman Java, C# tetap mempertahankan operasi unik yang terdapat pada bahasa pemrograman C++ seperti *overloading, enumerations, pre-processor directive, pointer* (pada *unmanaged/unsafe code*), dan fungsi *pointer.* Seperti halnya *Visual Basic*, bahasa pemrograman *C#* juga dilengkapi dengan properties.

Sebagai tambahan, bahasa pemrograman C# juga datang dengan beberapa fitur baru dan sangat menarik seperti *reflections, attributes, marshalling, remote, threads, streams* data *acces* dengan *ADO.NET*, dan masih banyak lagi.

2.7 *Open CV Library*

OpenCV (*Open Computer Vision*) adalah sebuah API (*Application Programming Interface*) *Library* yang sudah sangat familiar pada Pengolahan Citra *Computer Vision*. *Computer Vision* itu sendiri adalah salah satu cabang dari Bidang Ilmu Pengolahan Citra (*Image Processing*) yang memungkinkan komputer dapat melihat seperti manusia. Dengan *vision* tersebut komputer dapat mengambil keputusan, melakukan aksi, dan mengenali terhadap suatu objek. Beberapa pengimplementasian dari *Computer Vision* adalah *Face Recognition, Face Detection, Face/Object Tracking, Road Tracking, dll*. *OpenCV* adalah library *Open Source* untuk *Computer Vision* untuk C/C++, OpenCV didesain

untuk aplikasi real-time, memiliki fungsi-fungsi akuisisi yang baik untuk *image/video* [8].

OpenCV juga menyediakan *interface* ke *Integrated Performance Primitives* (IPP) Intel sehingga jika anda bisa mengoptimasi aplikasi *Vision* anda jika menggunakan prosesor Intel.

Feature yang dimiliki *OpenCV* antara lain :

- Manipulation data citra (alokasi, *copying, setting*, konversi).
- Citra dan video I/O (file dan kamera *based input, image/video file output*).
- Manipulasi Matriks dan Vektor beserta rutin-rutin aljabar linear (*products, solvers, eigenvalues, SVD*).
- Data struktur dinamis (*lists, queues, sets, trees, graphs*).
- Pemroses Citra fundamental (*filtering, edge detection, corner detection, sampling and interpolation, color conversion, morphological operations, histograms, image pyramids*).
- Analisis struktur(*connected components, contour processing, distance transform, various moments, template matching, Hough transform, polygonal approximation, line fitting, ellipse fitting, Delaunay triangulation*).
- Kalibrasi kamera (*calibration patterns, estimasi fundamental matrix, estimasi homography, stereo correspondence*).
- Analisis gerakan (*optical flow, segmentation, tracking*).
- Pengenalan obyek (*eigen-methods, HMM*).
- *Graphical User Interface* (*display image/video*, penanganan *keyboard* dan *mouse handling, scroll-bars*).
- Pelabelan citra (*line, conic, polygon, text drawing*)

Selain itu OpenCV juga dilengkapi dengan *Machine Learning library* yang memiliki algoritma berikut :

- *Naive Bayes classifier*
- *k-nearest neighbor algorithm*
- *Support Vector Machine*

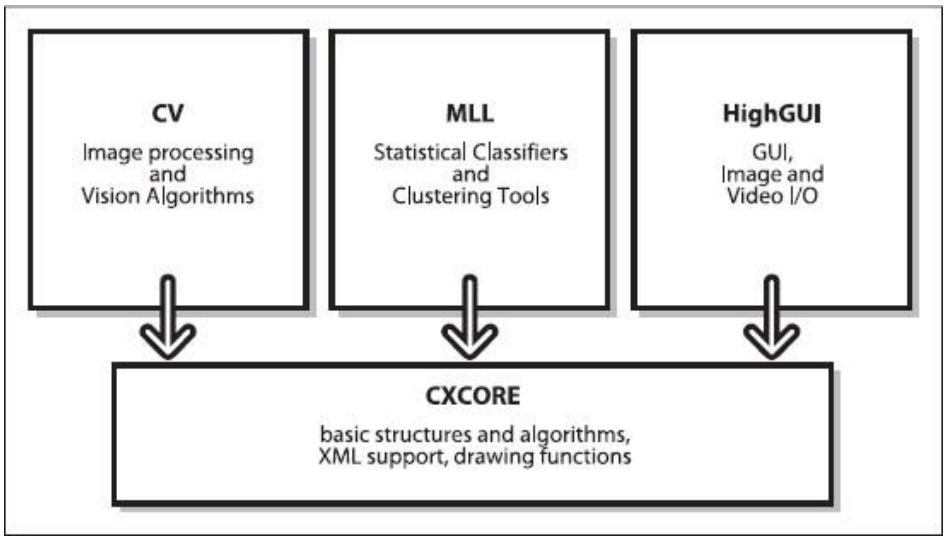
- *Decision Trees*
- *Boosting*
- *Random forest*
- *Expectation Maximization*
- *Neural Networks*

Library ini bisa di-compile dengan Visual C++ 6.0, Visual C++.Net 2003/2005/2008 baik Standard, Professional maupun express edition.

OpenCV sendiri terdiri dari 5 library, yaitu :

1. CV : untuk algoritma *Image processing dan Vision.*
2. ML : untuk *machine learning library*
3. Highgui : untuk GUI, *Image* dan *Video* I/O.
4. CXCORE : untuk struktur data, *support XML* dan fungsi-fungsi grafis.

Struktur dan Konten OpenCV :



Gambar 2.1 Struktur dan Konten OpenCV

2.8 *Object Oriented Analysis and Design*

Analisis dan Desain Berorientasi Objek (*Object Oriented Analysis and Design*) adalah cara baru dalam memikirkan suatu masalah dengan menggunakan model yang dibuat menurut konsep sekitar dunia nyata. Dasar pembuatan adalah objek, yang merupakan kombinasi antara struktur data dan perilaku dalam satu entitas.

Pengertian "berorientasi objek" berarti bahwa kita mengorganisasi perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data dan perilakunya.

Konsep OOAD mencakup analisis dan desain sebuah sistem dengan pendekatan objek, yaiut analisis berorientasi objek (OOA) dan desain berorientasi objek (OOD). OOA adalah metode analisis yang memerika requirement (syarat/keperluan) yang harus dipenuhi sebuah sistem) dari sudut pandang kelas-kelas dan objek-objek yang ditemui dalam ruang lingkup perusahaan. Sedangkan OOD adalah metode untuk mengarahkan arsitektur software yang didasarkan pada manipulasi objek-objek sistem atau subsistem.

2.8.1 OOA *(Object Oriented Analysis)*

OOA mempelajari permasalahan dengan menspesifikasikannya atau mengobservasi permasalahn tersebut dengan menggunakan metode berorientasi objek. Biasanya analisa sistem dimulai dengan adanya dokumen permintaan (requirement) yang diperoleh dari semua pihak yang berkepentingan. (Misal: klien,developer, pakar, dan lain-lain).

Dokumen permintaan memiliki 2 fungsi yaitu : memformulasikan kebutuhan klien dan membuat suatu daftar tugas. Analisis berorientasi obyek (OOA) melihat pada domain masalah, dengan tujuan untuk memproduksi sebuah model konseptual informasi yang ada di daerah yang sedang dianalisis. Model analisis tidak mempertimbangkan kendala-kendala pelaksanaan apapun yang mungkin ada, seperti konkurensi, distribusi, ketekunan, atau bagaimana sistem

harus dibangun. Kendala pelaksanaan ditangani selama desain berorientasi objek (OOD).

Sumber-sumber untuk analisis dapat persyaratan tertulis pernyataan, dokumen visi yang formal, wawancara dengan stakeholder atau pihak yang berkepentingan lainnya. Sebuah sistem dapat dibagi menjadi beberapa domain, yang mewakili bisnis yang berbeda, teknologi, atau bidang yang diminati, masing-masing dianalisis secara terpisah.

Hasil analisis berorientasi objek adalah deskripsi dari apa sistem secara fungsional diperlukan untuk melakukan, dalam bentuk sebuah model konseptual. Itu biasanya akan disajikan sebagai seperangkat menggunakan kasus, satu atau lebih UML diagram kelas, dan sejumlah diagram interaksi. Tujuan dari analisis berorientasi objek adalah untuk mengembangkan model yang menggambarkan perangkat lunak komputer karena bekerja untuk memenuhi seperangkat persyaratan yang ditentukan pelanggan.

UML (*Unified Modeling Language*) adalah sebuah bahasa yang berdasarkan grafik/gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan software berbasis OO (*Object-Oriented*). UML sendiri juga memberikan standar penulisan sebuah sistem blue print, yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema database, dan komponen-komponen yang diperlukan dalam sistem software. *Unified Model Language* (UML) adalah bahasa universal untuk :

- memvisualisasikan grafis model yang tepat.
- menetapkan model yang tepat, lengkap, dan tidak ambigu untuk mengampil semua keputusan penting dalam analisis, desain dan implementasi.
- membangun model yang dapat dihubungkan langsung dengan bahasa pemrograman.

- mendokumentasikan semua informasi yang dikumpulkan oleh tim sehingga memungkinkan untuk berbagi informasi.

2.8.2 OOD *(Object Oriented Design)*

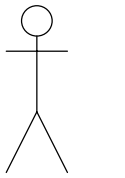
OOD mengubah model konseptual yang dihasilkan dalam analisis berorientasi objek memperhitungkan kendala yang dipaksakan oleh arsitektur yang dipilih dan setiap non-fungsional – teknologi atau lingkungan – kendala, seperti transaksi throughput, response time, run – waktu platform, lingkungan pengembangan, atau bahasa pemrograman. Jenis-jenis diagram UML :

2.8.2.1 *Use Case Diagram*

Use case diagram menggambarkan kebutuhan sistem dari sudut pandang *user*. Digunakan untuk menggambarkan hubungan antara internal sistem dan eksternal sistem atau hubungan antara *use case* dan aktor.

1. *Actor*

Actor adalah sesuatu (entitas) yang berhubungan dengan sistem dan berpartisipasi dalam *use case*. *Actor* menggambarkan orang, sistem atau entitas eksternal yang secara khusus membangkitkan sistem dengan *input* atau masukan kejadian-kejadian, atau menerima sesuatu dari sistem. *Actor* dilukiskan dengan peran yang mereka mainkan dalam *use case*, seperti Staff, Kurir dan lain-lain. Lambang *actor* dapat dilihat pada gambar 2.2.

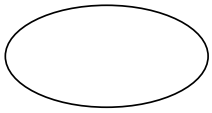


Gambar 2.2. Bentuk *Actor* dalam UML

Dalam *use case* diagram terdapat satu aktor pemulai atau *initiator* actor yang membangkitkan rangsangan awal terhadap sistem, dan mungkin sejumlah aktor lain yang berpartisipasi atau *participating* actor. Akan sangat berguna untuk mengetahui siapa aktor pemulai tersebut.

2. *Use case*

Use case yang dibuat berdasar keperluan aktor merupakan gambaran dari "apa" yang dikerjakan oleh sistem, bukan "bagaimana" sistem mengerjakannya. *Use case* diberi nama yang menyatakan apa hal yang dicapai dari interaksinya dengan aktor. *Use case* dinotasikan seperti pada gambar 2.3.

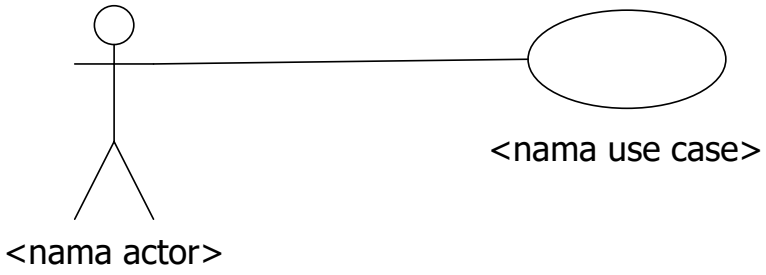


Gambar 2.3. Bentuk *Use Case* dalam UML

3. *Relationship*

Relasi (*relationship*) digambarkan sebagai bentuk garis antara dua simbol dalam *use case* diagram. Relasi antara *actor* dan *use case* disebut juga dengan asosiasi (*association*). Asosiasi ini digunakan untuk menggambarkan bagaimana hubungan antara keduanya.

Relasi-relasi yang terjadi pada *use case* diagram bisa antara *actor* dengan *use case* atau *use case* dengan *use case*. Relationship dinotasikan seperti pada gambar 2.4.



Gambar 2.4. Bentuk *Relationship* dalam UML

Relasi antara *use case* dengan *use case* :

a. *Include*, pemanggilan *use case* oleh *use case* lain atau untuk menggambarkan suatu *use case* termasuk di dalam *use case* lain (diharuskan). Contohnya adalah pemanggilan sebuah fungsi program. Digambarkan dengan garis lurus berpanah dengan tulisan <<*include*>>.

b. *Extend*, digunakan ketika hendak menggambarkan variasi pada kondisi perilaku normal dan menggunakan lebih banyak kontrol *form* dan mendeklarasikan *ekstension* pada *use case* utama. Atau dengan kata lain adalah perluasan dari *use case* lain jika syarat atau kondisi terpenuhi. Digambarkan dengan garis berpanah dengan tulisan <<*extend*>>.

c. *Generalization/Inheritance*, dibuat ketika ada sebuah kejadian yang lain sendiri atau perlakuan khusus dan merupakan pola berhubungan *base-parent use case*. Digambarkan dengan garis berpanah tertutup dari *base use case* ke parent *use case*.

2.8.2.2 *Activity Diagram*

Diagram aktivitas menggambarkan proses bisnis dan urutan aktivitas-aktivitas yang mendukung penggambaran tindakan sistem baik yang bersifat kondisional maupun paralel. Tindakan kondisional dilukiskan dengan cabang *(branch)* dan penyatuan *(merge)*.

Sebuah *branch* memiliki sebuah *transition* masuk atau yang disebut dengan *incoming transition* dan beberapa *transition* keluar atau yang disebut dengan *outgoing transition* dari *branch* yang berupa keputusan-keputusan. Hanya satu dari *outgoing transition* yang dapat diambil, maka keputusan-keputusan tersebut harus bersifat *mutually exclusive*. [*else*] digunakan sebagai keterangan singkat yang menunjukkan bahwa *transition* "*else*" tersebut harus digunakan jika semua keputusan yang ada pada *branch* salah.

1. Initial, merupakan titik awal untuk memulai aktivitas. Dapat dilihat pada gambar 2.5.



Gambar 2.5. Simbol *Initial*

2. Final, merupakan titik akhir untuk mengakhiri aktivitas. Dapat dilihat pada gambar 2.6.



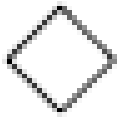
Gambar 2.6. Simbol *Final*

3. Activity, menandakan aktivitas. Dapat dilihat pada gambar 2.7.



Gambar 2.7. Simbol *Activity*

4. Decision, merupakan pilihan untuk mengambil keputusan/kondisi if-else. Dapat dilihat pada gambar 2.8.



Gambar 2.8. Simbol *Decision*

5. Fork/Join, digunakan untuk menunjukan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan menjadi satu. Dapat dilihat pada gambar 2.9.



Gambar 2.9. Simbol *Fork/Join*

2.8.2.3 *Class Diagram*

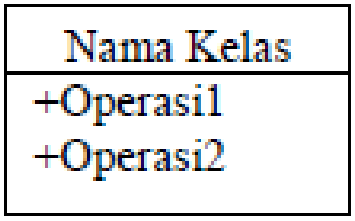
Class diagram merupakan bagian yang paling penting dalam analisa dan perancangan berorientasi obyek. Dalam UML diagram kelas digunakan untuk memodelkan *static structure* dari sistem informasi.

Kelas merupakan himpunan dari obyek yang sejenis yang mempunyai atribut dan perilaku *(behaviors/method)* yang sama. Atribut adalah sebuah nilai data karakteristik yang dimiliki oleh obyek sebuah kelas sedangkan *method* adalah

perilaku atau operasi yang dikenakan oleh suatu kelas. Pada gambar kelas terdapat tiga bagiannya.

1. *Class*

   Class adalah blok - blok pembangun pada pemrograman berorientasi objek. Sebuah class digambarkan sebagai sebuah kotak yang terbagi atas 3 bagian. Bagian atas adalah bagian nama dari class. Bagian tengah mendefinisikan property/atribut class. Bagian akhir mendefinisikan method-method dari sebuah class. Dapat dilihat pada gambar 2.10.



Gambar 2.10. Bentuk Class

2. *Assosiation*

   Sebuah asosiasi merupakan sebuah *relationship* paling umum antara 2 class, dan dilambangkan oleh sebuah garis yang menghubungkan antara 2 class. Garis ini bisa melambangkan tipe-tipe *relationship* dan juga dapat menampilkan hukum-hukum *Compositionmultiplisitas* pada sebuah relationship (Contoh: *One-to-one, one-to-many, many-to-many*). Dapat dilihat pada gambar 2.11.



Gambar 2.11. Bentuk *Associasion*

3. *Dependency*

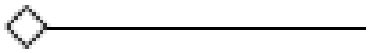
   Kadangkala sebuah *class* menggunakan *class* yang lain. Hal ini disebut *dependency*. Umumnya penggunaan dependency digunakan untuk menunjukkan operasi pada suatu *class* yang menggunakan *class* yang lain. Sebuah *dependency* dilambangkan sebagai sebuah panah bertitik-titik. Dapat dilihat pada gambar 2.12.



Gambar 2.12. Simbol *Dependency*

4. *Aggregation*

   Aggregation mengindikasikan keseluruhan bagian relationship dan biasanya disebut sebagai relasi "mempunyai sebuah" atau "bagian dari". Sebuah aggregation digambarkan sebagai sebuah garis dengan sebuah jajaran genjang yang tidak berisi/tidak solid. Dapat dilihat pada gambar 2.13.



Gambar 2.13. Simbol *Aggregation*

5. *Generalization*

   Sebuah relasi generalization sepadan dengan sebuah relasi *inheritance* pada konsep berorientasi objek. Sebuah *generalization* dilambangkan dengan sebuah panah dengan kepala panah yang tidak solid yang mengarah ke kelas "parent"- nya/induknya. Dapat dilihat pada gambar 2.14.



Gambar 2.14. Simbol *Generalization*

2.8.2.4 *Sequence Diagram*

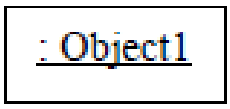
Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang disusun dalam suatu urutan waktu.

Secara khusus, diagram ini berasosiasi dengan *use case*. Sequence diagram menggambarkan *behavior internal* sebuah sistem. Dan lebih menekankan pada penyampaian *message* dengan parameter waktu.

1. *Object*

   Object atau biasa juga disebut partisipan merupakan instance dari sebuah class dan dituliskan tersusun secara *horizontal*. Digambarkan sebagai

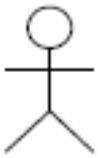
sebuah *class* (kotak) dengan nama objek didalamnya yang diawali dengan sebuah titik koma. Dapat dilihat pada gambar 2.15.



Gambar 2.15. Simbol *Object*

2. *Actor*

   Actor juga dapat berkomunikasi dengan *object*, maka *actor* juga dapat diurutkan sebagai kolom. Dapat dilihat pada gambar 2.16.



Gambar 2.16. Simbol *Actor*

3. *Lifeline*

   Lifeline mengindikasikan keberadaan sebuah *object* dalam basis waktu. Notasi untuk *Lifeline* adalah garis putus-putus vertikal yang ditarik dari sebuah objek sepeti pada gambar 2.17.

Gambar 2.17. Simbol *Lifeline*

4. *Activation*

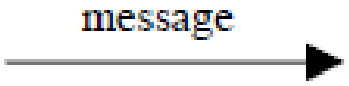
   Activation dinotasikan sebagai sebuah kotak segi empat yang digambar pada sebuah *lifeline*. *Activation* mengindikasikan sebuah objek yang akan melakukan sebuah aksi. Dapat dilihat pada gambar 2.18.



Gambar 2.18. Simbol *Activation*

5. *Message*

   Message digambarkan dengan anak panah *horizontal* antara *Activation.* *Message* mengindikasikan komunikasi antara *object-object*. Dapat dilihat pada gambar 2.19.



Gambar 2.19. Simbol *Message*

2.9 Modem *Wavecom Fastrack* M1306B

Modem adalah sebuah alat yang dapat membuat komputer terkoneksi dengan *internet* melalui *line* telepon standar. *Modem* banyak digunakan komputer rumah dan jaringan sederhana untuk dapat berkomunikasi dengan jutaan komputer lain dalam lalu lintas *internet.* Kata *Modem* itu sendiri merupakan kependekan dari *Modulator Demodulator*. Ini berarti *Modem* bekerja dengan cara mengubah informasi digital dari komputer pengirim ke dalam bentuk sinyal *analog* yang ditransmisikan melaluli *line* telepon. Selanjutnya *Modem* pada komputer penerima akan mengubah ulang sinyal *analog* ke sinyal *digital*. GSM *Modul* atau *Modem GSM* adalah jenis khusus dari *modem* yang menerima kartu *SIM*, dan mengoperasikan selama berlangganan ke operator *mobile*, seperti ponsel. *Modem GSM* dihubungkan dengan suatu *interface* yang memungkinkan aplikasi seperti *SMS* untuk mengirim dan menerima pesan melalui *Modem* Beberapa fungsi kegunaan *modem* ini di masyarakat adalah antara lain:

- SMS *Broadcast application.*

- SMS *Quis application.*
- SMS *Polling.*
- SMS *auto-reply.*
- Aplikasi *Server* Pulsa.
- *Telemetri.*
- *Payment Point* Data.

Pada penelitian tugas akhir ini, digunakan *Modem* GSM *Serial Wavecom Fastrack M1306B*. Untuk *Modem* seri ini memiliki dua *type* konektor yaitu *serial* dan *USB.*



Gambar 2.20. Modem GSM *Fastrack* M1306B



Gambar 2.21. *Port* Modem GSM *Fastrack* M1306B

2.10 *Webcam* (Web Kamera)

Webcam adalah sebuah kamera *video digital* kecil yang dihubungkan ke komputer biasanya melalui *port USB* atau pun *port com*. pada umumnya *WebCam* tidak membutuhkan kaset atau tempat penyimpanan data, data hasil perekaman yang didapat langsung ditransfer ke komputer. Istilah "*webcam"* mengarah pada

jenis kamera yang digunakan untuk kebutuhan layanan berbasis *web*. Defenisi lain tentang *WebCam* adalah sebuah periferal berupa kamera sebagai pengambil citra/gambar dan mikropon (optional) sebagai pengambil suara/*audio* yang dikendalikan oleh sebuah komputer atau oleh jaringan komputer.



Gambar 2.22 Webcam

Webcam bekerja seperti halnya sebuah kamera digital hanya saja *webcam* ini di desain untuk computer jadi tidak bisa dibawa ke mana-mana dan jauh lebih simple di banding kamera-kamera pada umumnya. Sebuah *web camera* biasanya dilengkapi dengan *software, software* ini mengambil gambar-gambar dari kamera *digital* secara terus menerus ataupun dalam interval waktu tertentu dan menyiarkannya melalui koneksi *internet*. Ada beberapa metode penyiaran, metode yang paling umum adalah *software* mengubah gambar ke dalam bentuk file *JPG*.

2.11 Pengujian *Black Box*

Pengujian black box merupakan pendekatan komplementer dari teknik white box, karena pengujian black box diharapkan mampu mengungkap kelas kesalahan yang lebih luas dibandingkan teknik white box. Pengujian black box berfokus pada pengujian persyaratan fungsional perangkat lunak, untuk mendapatkan serangkaian kondisi input yang sesuai dengan persyaratan fungsional suatu program.

Pengujian black box adalah pengujian aspek fundamental sistem tanpa memperhatikan struktur logika internal perangkat lunak. Metode ini digunakan untuk mengetahui apakah perangkat lunak berfungsi dengan benar. Pengujian black box merupakan metode perancangan data uji yang didasarkan pada

spesifikasi perangkat lunak. Data uji dibangkitkan, dieksekusi pada perangkat lunak dan kemudian keluaran dari perangkat lunak dicek apakah telah sesuai dengan yang diharapkan. Pengujian black box berusaha menemukan kesalahan dalam kategori:

1. fungsi-fungsi yang tidak benar atau hilang
2. kesalahan *interface*
3. kesalahan dalam struktur data atau akses database eksternal
4. kesalahan kinerja
5. inisialisasi dan kesalahan terminasi.