BAB 2
TINJAUAN PUSTAKA

Dalam bab ini akan diuraikan konsep dan pengertian mengenai teori-teori yang akan digunakan di dalam penelitian ini baik teori umum maupun teori khusus.

2.1 Teori Umum

Subbab ini berisi teori-teori utama yang dijadikan sebagai panduan utama untuk teori-teori lainnya yang digunakan dalam penelitian skripsi ini.

2.1.1 Interaksi Manusia dan Komputer

Menurut (Shneiderman, Plaisant, Cohen, & Jacobs, 2010:1), Interaksi Manusia dan Komputer adalah disiplin ilmu yang berhubungan dengan perancangan, evaluasi, dan implementasi dari sistem komputer yang interaktif untuk digunakan oleh manusia, dan pembelajaran dari fenomena yang diasosiasikan dengannya.

Dalam perancangan desain antarmuka (*user interface*), menurut (Shneiderman, Plaisant, Cohen, & Jacobs, 2010:88) ada delapan aturan emas yang perlu diperhatikan:

1. Berusaha untuk konsisten

   Menjaga urutan aksi yang konsisten untuk penanganan pada situasi yang serupa. Menjaga konsistensi dalam hal penggunaan terminologi yang identik pada *prompts*, *menu*, dan layar bantu serta konsistensi dalam penggunaan warna, tata letak, huruf kapital, jenis huruf, dan lainnya.
2. Diperuntukkan untuk penggunaan universal

   Mengenali berbagai macam kebutuhan yang disebabkann oleh perbedaan pengguna baik dari pengguna awam sampai ahli, perbedaan usia, kelainan dan perbedaan teknologi. Hal ini bisa dilakukan dengan memberikan penambahan fitur bagi pengguna awam seperti penjelasan dan pemberian fitur sepert jalan pintas (*shortcut*) bagi pengguna ahli.
3. Memberikan umpan balik yang informatif

   Memberikan umpan balik untuk setiap aksi yang dilakukan oleh pengguna. Umpan balik diberikan dalam bentuk yang sederhana untuk aksi yang sering dilakukan dan membutuhkan sedikit aksi serta dalam bentuk yang lebih rumit untuk aksi yang lebih jarang dilakukan dan membutuhkan banyak aksi.
4. Mencegah dan menangani kesalahan penggunaan

Merancang sebuah sistem yang dapat meminimalisir pengguna melakukan kesalahan penggunaan sistem. Sistem juga harus dapat mendeteksi dan menawarkan langkah-langkah penanganan yang sederhana jika terjadi kesalahan penggunaan oleh pengguna.

5. Merancang dialog untuk memberikan penutupan (kondisi akhir)
   Urutan aksi dikelompokkan menjadi kelompok awal, tengah, dan akhir. Umpan balik yang informatif pada setiap penyelesaian dari sebuah kelompok diberikan untuk memberikan tanda kepada pengguna untuk melangkah ke tahapan kelompok berikutnya.
6. Memungkinkan dan memudahkan pengembalian aksi yang sebelumnya
   Merancang sebuah sistem yang memungkinkan pengguna dapat kembali ke aksi sebelumnya jika terjadi kesalahan. Dengan adanya fitur ini, pengguna merasa tenang ketika melakukan kesalahan.
7. Mendukung pusat kendali internal
   Memastikan pengguna sebagai pengendali utama dari sistem dan sistem yang memberikan respons kepada pengguna, bukan sebaliknya.
8. Mengurangi beban ingatan jangka pendek
   Karena manusia memiliki keterbatasan kapasitas ingatan informasi dalam jangka pendek, sistem yang dirancang harus dibuat dalam tampilan yang sesederhana mungkin, pengurangan jumlah transisi window, penyederhanaan beberapa halaman menjadi satu, dan memberikan waktu yang cukup bagi pengguna untuk mempelajari sistem yang ada.

(Shneiderman, Plaisant, Cohen, & Jacobs, 2010:88) juga mengemukakan bahwa ada faktor lain yang perlu diperhatikan dalam perancangan desain antarmuka (*user interface*). Faktor yang dimaksud adalah lima faktor manusia terukur yang digunakan untuk mengevaluasi perancangan desain antarmuka (*user interface*). Kelima faktor tersebut adalah:

1. Waktu Pembelajaran
   Faktor ini memperhatikan berapa jumlah waktu yang dibutuhkan oleh kelompok pengguna tertentu untuk mempelajari penggunaan perintah-perintah yang dibutuhkan untuk menyelesaikan suatu pekerjaan atau tugas.
2. Kecepatan performa
   Faktor ini memperhitungkan jumlah waktu yang diperlukan untuk menyelesaikan suatu pekerjaan atau tugas.

3. Tingkat kesalahan pengguna

   Faktor ini memperhitungkan jumlah kesalahan yang terjadi pada saat pengoperasian aplikasi untuk menyelesaikan sebuah tugas atau pekerjaan.

4. Daya ingat jangka panjang

   Faktor ini memperhatikan bagaimana pengguna mempertahankan pengetahuan mereka dalam hal penggunaan antarmuka pengguna (*user interface*). Biasanya hal ini sering dikaitkan dengan waktu pembelajaran dan frekuensi penggunaan antarmuka pengguna (*user interface*).

5. Kepuasan subjektif

   Faktor ini memperhitungkan tingkat kepuasan pengguna terhadap berbagai aspek antarmuka pengguna (*user interface*).

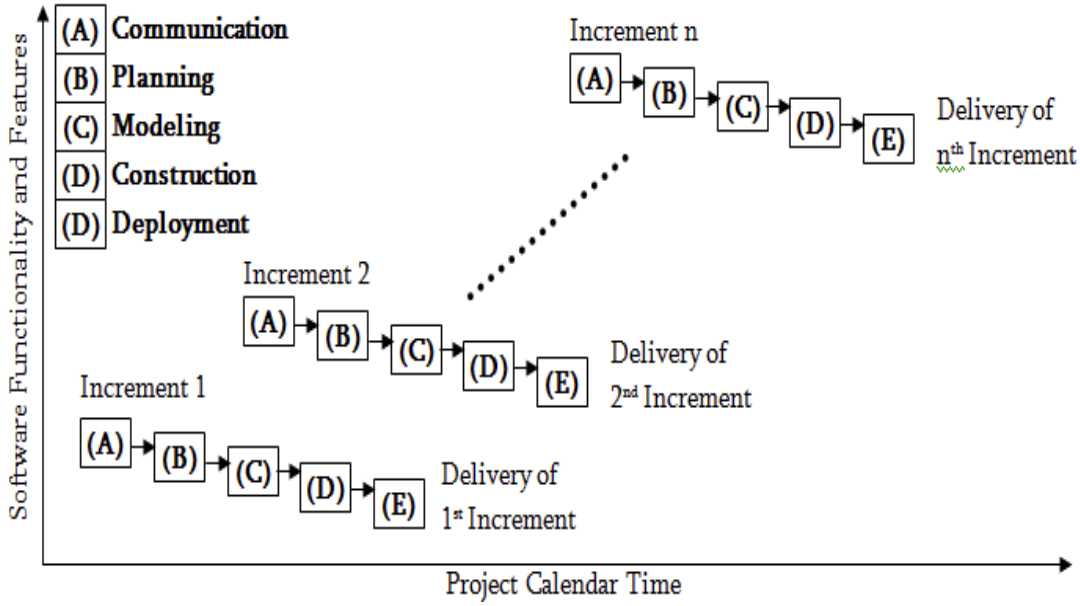
2.1.2 Rekayasa Piranti Lunak

Menurut (Pressman, 2010:4), definisi dari piranti lunak atau *software* adalah:

1. Instruksi-instruksi yang jika dijalankan akan menghasilkan fitur, fungsi, dan performa yang diinginkan.
2. Struktur data yang memungkinkan program untuk memanipulasi informasi dengan baik.
3. Informasi deskriptif baik dalam bentuk cetak dan bentuk *virtual* yang mendeskripsikan operasi dan kegunaan dari program.

2.1.2.1 *Incremental Process Model*

Menurut (Pressman, 2010:41-42), Incremental Process Model mengkombinasikan alur proses yang ada pada process model linear dan paralel seperti pada gambar berikut:



Gambar 2.1 *Incremental Process Model*

(sumber: http://3.bp.blogspot.com/-qh2Fi-p3BLM/UPR2Tc0YfeI/AAAAAAAABDU/cXeSWD78vok/s1600/software_engineering_12.png)

Menurut (Pressman, 2010:31), alur proses linear mengeksekusi masing-masing *framework activities* dari lima *framework activities* secara berurutan. Sedangkan alur proses paralel mengeksekusi satu atau lebih *framework activities* dalam waktu yang bersamaan dengan *activities* yang lainnya. Lima *framework activities* yang dimaksud adalah:

1. *Communication*

   Mengumpulkan semua informasi dan kebutuhan yang diinginkan oleh *user* untuk dianalisa pada tahapan berikutnya.

2. *Planning*

   Merencanakan berbagai modul yang akan dikerjakan sumber daya yang diperlukan, serta jangka waktu yang diperlukan.

3. *Modeling*

   Membuat analisis mengenai struktur data yang akan digunakan, modul algoritma, rancangan perangkat lunak, dan desain rancangan layar yang sesuai.

4. *Construction*

Melakukan pembuatan program berdasarkan hasil analisis sebelumnya. Pada tahapan ini juga dilakukan proses pengujian (*testing*) untuk memperbaiki kesalahan yang terjadi pada program.

5. *Deployment*

   Piranti lunak yang telah dihasilkan akan diuji coba dan dievaluasi oleh pengguna. Hasil evaluasi yang ada akan digunakan sebagai *response* atau timbal balik terhadap tahapan *increment* berikutnya.

Menurut (Pressman, 2010: 41-42), tahapan pertama yang dilakukan dalam *incremental model* adalah *core product* (mengandung kebutuhan dasar yang dibutuhkan akan tetapi fitur-fitur pendukung belum disertakan). *Core product* yang telah dihasilkan, akan dievaluasi dan akan dikembangkan berdasarkan evaluasi sebelumnya pada *increment* berikutnya. Hal ini dilakukan untuk menghasilkan produk yang memenuhi kebutuhan *user* dan memberikan tambahan fitur beserta fungsionalitas yang diperlukan.

Incremental Process Model memiliki kelebihan untuk digunakan dalam kondisi sebagai berikut:

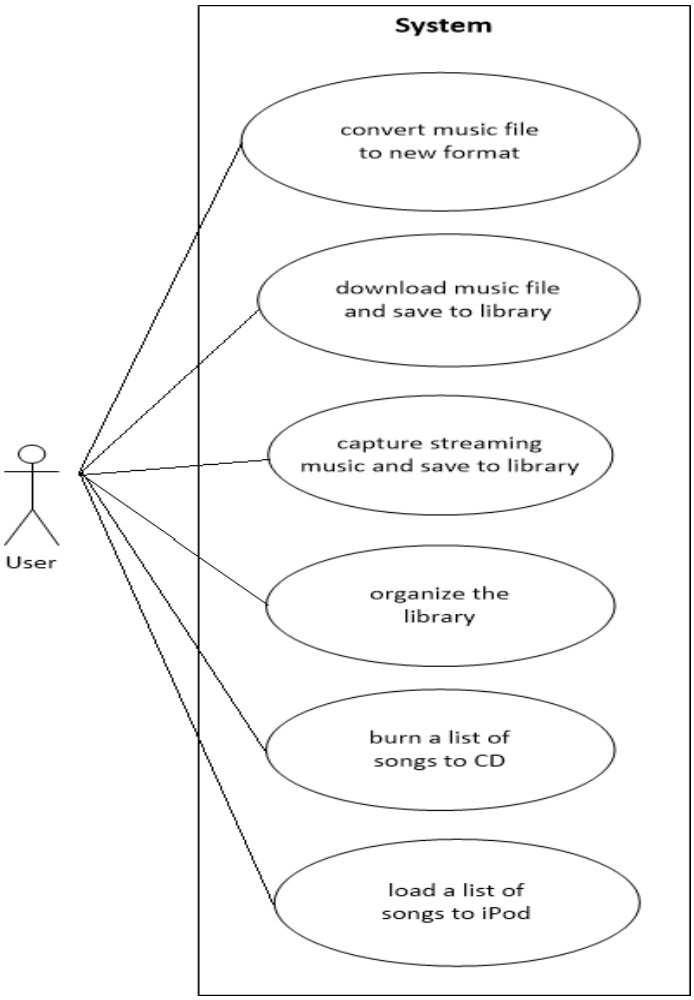
1. Dapat diterapkan dalam suatu project yang mempunyai tenggat waktu yang telah ditetapkan sebelumnya.
2. Memungkinkan untuk dikerjakan oleh jumlah orang yang sedikit (memungkinkan untuk dikerjakan oleh 1 orang) dalam pengerjaan setiap *increment*.
3. Sebagai tambahan, model ini dapat direncanakan untuk mengatas resiko teknis.

2.1.3 *Unified Modeling Language (UML)*

Menurut (Pressman, 2010:841), *Unified Modeling Language* (*UML*) adalah bahasa standar yang digunakan untuk membuat *blueprint* dari sebuah piranti lunak. *UML* dibuat oleh arsitek piranti lunak untuk membantu pengembang piranti lunak dalam mengembangkan piranti lunak.

2.1.3.1 *Use Case Diagram*

Use Case Diagram adalah gambaran umum dari semua *use case* dan bagaimana mereka saling berelasi. *Use Case* mendeskripsikan bagaimana *user* berinteraksi dengan system dengan cara mendefinisikan langkah-langkah yang diperlukan untuk mencapai tujuan tertentu. Berikut ini merupakan contoh *Use Case Diagram* untuk aplikasi digital *music player*.

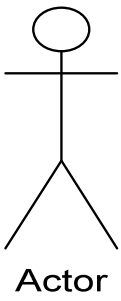


Gambar 2.2 *Use Case Diagram Digital Music Player*

Dalam penggunaannya, *Use Case Diagram* memiliki beberapa istilah sebagai berikut:

1. *Actor*

   Actor digunakan untuk menggambarkan siapapun dan apapun yang berinteraksi dengan sistem untuk bertukar informasi (Whitten, 2007:246).



Gambar 2.3 *Actor*

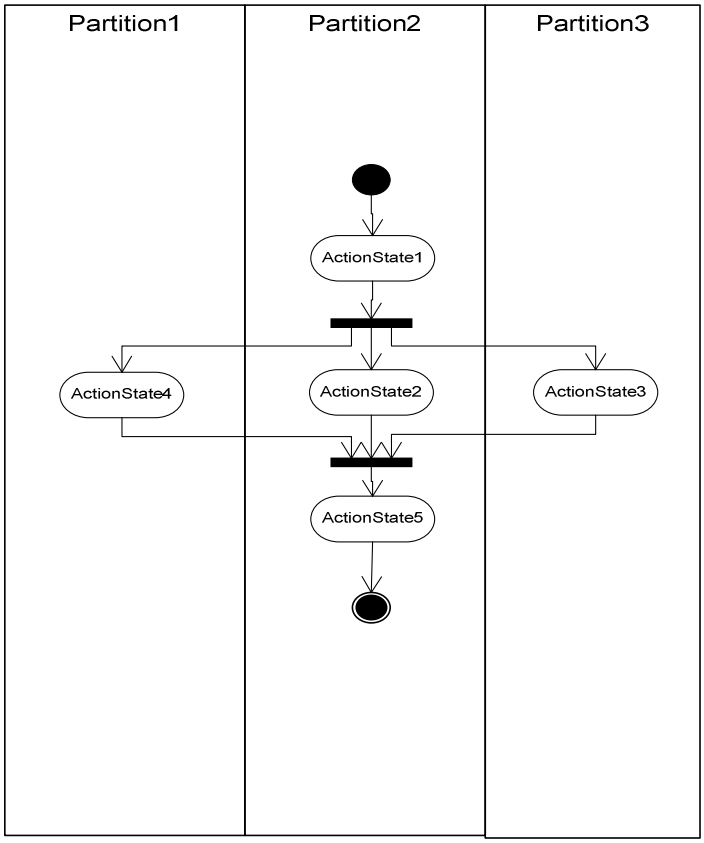
2. *Use Case*

   Use Case digunakan untuk menggambarkan fungsionalitas dan kebutuhan sistem dari sudut pandang *Actor* (Whitten, 2007:246). Sebuah *use case* juga menggambarkan urutan aktivitas dan interaksi *user* dalam mencapai tujuan.

Gambar 2.4 *Use Case*

2.1.3.2 *Activity Diagram*

Activity diagram menggambarkan pergerakan dinamis dari semua sistem atau bagian dari sebuah sistem yang melalui proses kontrol dari berbagai tindakan yang dilakukan oleh sistem. *Activity Diagram* memiliki kemiripan dengan *flowchart*, namun *Activity Diagram* dapat menunjukkan pergerakan yang terjadi secara bersamaan (Pressman, 2010:853-855). Gambar berikut ini merupakan contoh dari *Activity Diagram*.



Gambar 2.5 Contoh *Activity Diagram*

Dalam penggunaannya, *Activity Diagram* memiliki beberapa istilah sebagai berikut:

1. *Initial Node*

   Initial node digunakan untuk menggambarkan titik awal proses dalam *activity diagram*. *Initial node* digambarkan sebagai lingkaran hitam (Pressman, 2010:853).

Gambar 2.6 *Initial node*

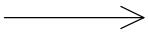
2. *Action Node*

   Action node digunakan untuk menggambarkan proses yang dilakukan oleh sistem dalam *activity diagram*. *Action node* digambarkan sebagai *rounded rectangle* (Pressman, 2010:853).

Gambar 2.7 *Action node*

3. *Control Flow*

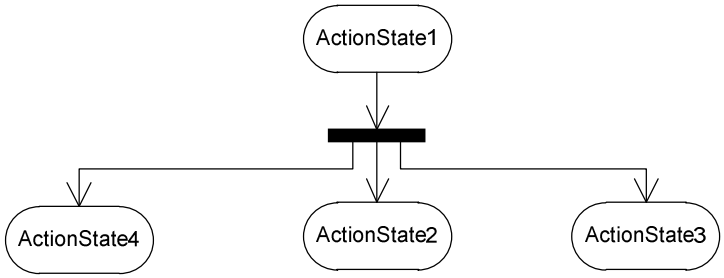
   Control flow digunakaan untuk menggambarkan alur dari suatu elemen ke elemen lainnya dalam *activity diagram*. *Control flow* digambarkan sebagai garis panah (Pressman, 2010:853).



Gambar 2.8 *Control flow*

4. *Fork*

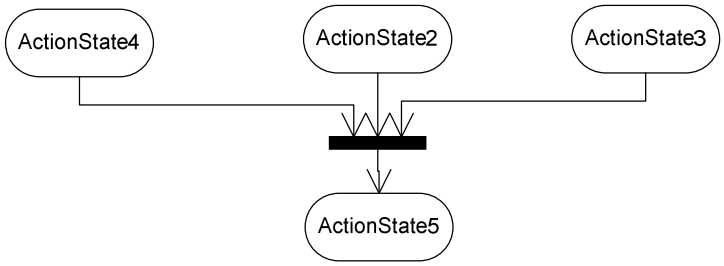
   Fork digunakan untuk menggambarkan pemisahan suatu proses menjadi dua atau lebih proses yang konkuren. *Fork* digambarkan sebagai persegi panjang hitam horizontal dengan satu panah *input* dan dua atau lebih panah *output* (Pressman, 2010:853).



Gambar 2.9 *Fork*

5. *Join*

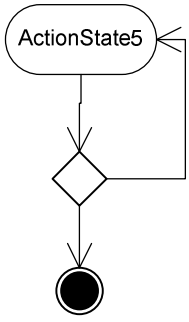
   Join digunakan untuk menggambarkan sinkronisasi proses yang konkuren. *Join* digambarkan sebagai persegi panjang hitam horizontal dengan banyak panah *input* dan satu panah *output* (Pressman, 2010:854).



Gambar 2.10 *Join*

6. *Decision*

   Decision digunakan untuk menggambarkan kondisi seleksi dalam *control flow*. *Decision* digambarkan sebagai wajik dengan satu panah *input* dan dua atau lebih panah *output*. Setiap panah *output* akan diberi keterangan (Pressman, 2010:854-856).

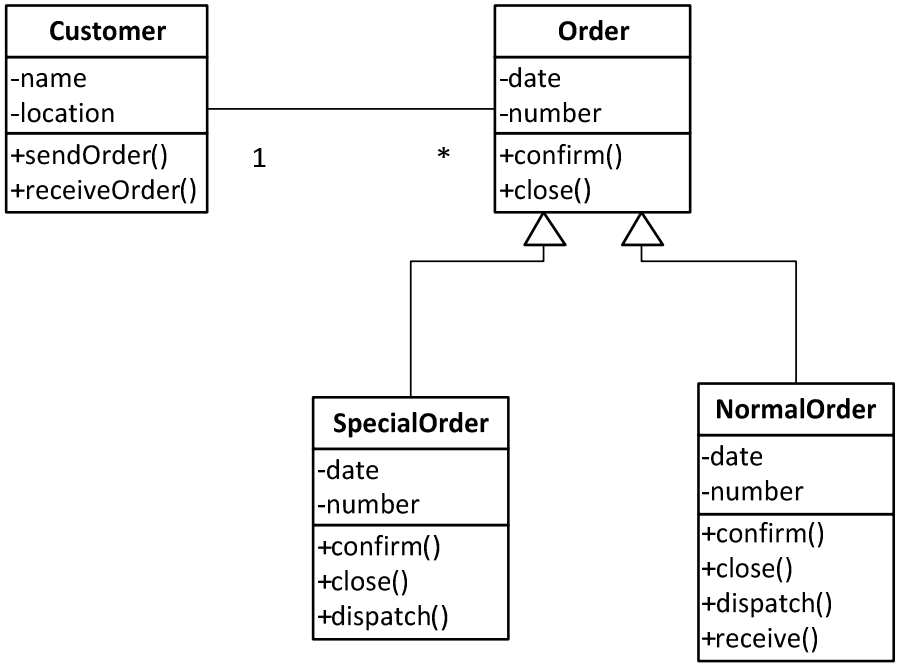


Gambar 2.11 *Decision*

2.1.3.3 Class Diagram

Class Diagram adalah *diagram* yang memberikan gambaran statis atau struktural dari sebuah sistem. *Class diagram* tidak menunjukkan sifat dinamis dari komunikasi antar objek dari class-class yang ada pada *diagram*. Elemen utama dari

class diagram adalah kotak yang merepresentasikan *class* dan *interface.* Sebuah *class diagram* juga dapat menjelaskan mengenai relasi antar *class* (Pressman, 2010:842-843). Gambar berikut ini merupakan contoh dari *Class Diagram*.

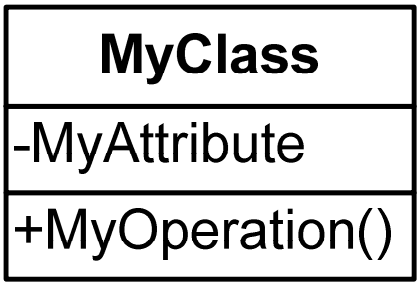


Gambar 2.12 Contoh *Class Diagram*

Dalam penggunaannya, *Class Diagram* memiliki beberapa istilah sebagai berikut:

1. *Class*

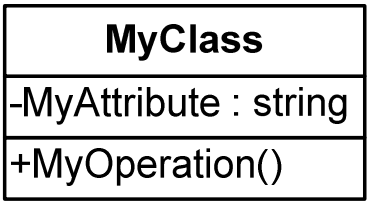
   Class merupakan sebuah *template*, cetakan atau *prototype* yang menggambarkan sebuah objek. Sebuah *Class* dalam *Class Diagram* terdiri atas 3 bagian, yaitu nama *class*, *attribute*, dan *operation*. *Attribute* merujuk kepada suatu data yang dimiliki dan diketahui oleh objek dari *class* terkait. *Attribute* umumnya diimplementasikan sebagai variabel dalam *Class*. *Operation* menunjuk ke sesuatu yang dapat dilakukan oleh *objek* dari *class* terkait. *Operation* umumnya diimplementasikan sebagai *method* dalam *Class* (Pressman, 2010:842).



Gambar 2.13 *Class*

2. *Type*

   Type menggambarkan jenis atau tipe dari suatu *attribute* atau *operation*. *Type* ditulis dengan mengikuti nama *attribute* atau *operation* dan dipisahkan dengan tanda titik dua (Pressman, 2010:842-843).



Gambar 2.14 *Type* dalam *Class Diagram*

3. *Visibility*

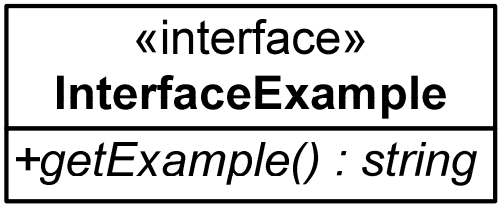
   Visibility digunakan untuk mendeskripsikan hak akses *dari* suatu *attribute* atau *operation* dalam suatu *Class* (Pressman, 2010:842-843). Berikut ini adalah berbagai tipe dari *visibility*:

Tabel 2.1 Tabel *Visibility*

Visibility	**Simbol**	**Deskripsi**
Private	-	Hanya dapat diakses oleh *Class* yang memiliki.
Public	+	Dapat diakses oleh *Class* lainnya.
Protected	#	Hanya dapat diakses oleh *Class* yang memiliki serta turunan(*inherit*) dari *Class* tersebut.

4. *Interface*

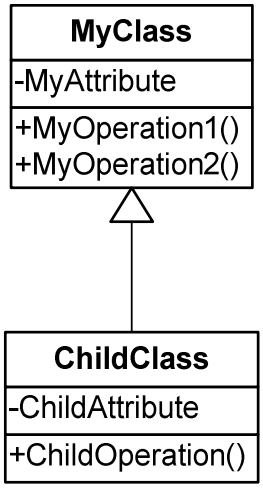
   Interface merupakan elemen dari *Class Diagram* yang mendefinisikan sejumlah *operation* untuk diimplementasikan ke dalam *Class* atau elemen lainnya. *Interface* dalam *Class Diagram* terdiri atas nama *interface* (dengan tambahan kata "<<interface>>") dan *operation* (Pressman, 2010:843).



Gambar 2.15 *Interface*

5. *Generalizaton*

   Generalization digunakan untuk mendeskripsikan relasi berupa turunan (*inheritance*) dalam *Class Diagram*. *Inheritance* pada *Class Diagram* melibatkan minimal 2 *Class*, yaitu *superclass* dan *subclass*. *Generalization* digambarkan sebagai sebuah panah dengan garis lurus dan mata panah segitiga pada *superclass* (Pressman, 2010:843).



Gambar 2.16 *Generalization*

6. *Association*

   Association digunakan untuk mendeskripsikan relasi struktural antar *Class*. *Association* umumnya digambarkan sebagai garis penuh dan dapat diberi label untuk mengindikasikan peran dari *Class* tersebut. Garis penuh dengan panah pada salah satu ujungnya dalam *Class Diagram* mengindikasikan *one-way navigability association*. *Assocation* juga menyatakan adanya ketergantungan (*dependency*) antara *Class* yang berhubungan. *Multiplicity* digunakan untuk

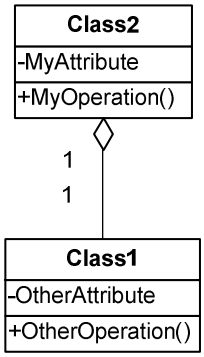
menyatakan jumlah objek dari suatu *Class* yang memiliki hubungan *association* dengan *Class* lainnya. *Multiplicity* dinyatakan dengan satu atau jarak bilangan bulat positif di tiap ujung garis *association* (Pressman, 2010:844–845). Berikut ini adalah berbagai tipe dari *multiplicity*:

Tabel 2.2 Tabel *Multiplicity*

Multiplicity	**Simbol**	**Deskripsi**
Zero or one	0…1	*Association* melibatkan satu atau tidak ada objek
Exactly one	1	*Association* melibatkan tepat satu objek
Many	*	*Association* melibatkan banyak objek
One or many	1…*	*Association* melibatkan satu atau banyak objek
Zero or many	0…*	*Association* melibatkan banyak objek atau tidak ada sama sekali

7. *Aggregation*

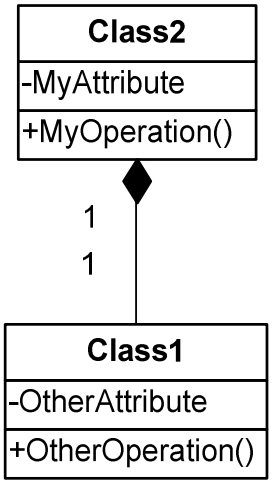
Aggregation digunakan untuk mendeskripsikan relasi kepemilikan antar *Class*. Kepemilikan dalam *aggregation* mendefinisikan suatu *Class* sebagai bagian dari *Class* lainnya. Tiap *Class* yang memiliki hubungan *Aggregation* tetap bersifat *independent* (tidak memiliki ketergantungan). *Aggregation* digambarkan sebagai garis penuh dengan salah satu ujungnya berupa bentuk wajik (Pressman, 2010:845).



Gambar 2.17 *Aggregation*

8. *Composition*

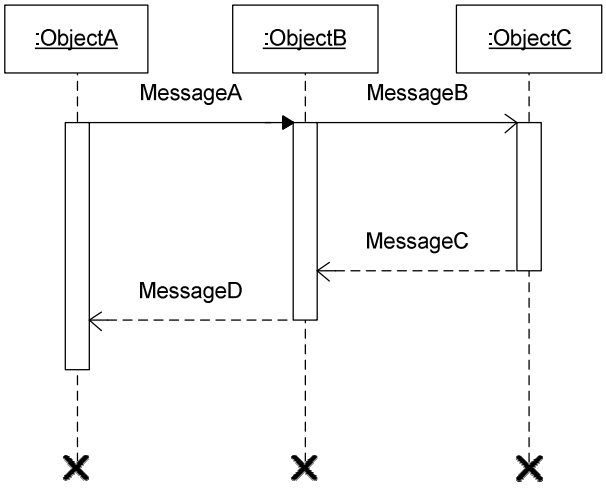
   Composition memiliki kesamaan dengan *aggregation*. Perbedaannya adalah adanya kepemilikan atau ketergantungan yang kuat (*dependen*t) antar *Class* dalam *composition*. *Composition* digambarkan sebagai garis penuh dengan salah satu ujungnya berupa bentuk wajik berwarna hitam (Pressman, 2010:845).



Gambar 2.18 *Composition*

2.1.3.4 *Sequence Diagram*

Sequence Diagram adalah diagram yang digunakan untuk menunjukkan komunikasi dinamis antar objek selama proses eksekusi berlangsung. *Sequence Diagram* juga menggambarkan adanya berbagai pesan yang dikirim antar objek untuk menyelesaikan suatu proses (Pressman, 2010:848). Berikut ini adalah contoh dari *Sequence Diagram*:



Gambar 2.19 Contoh *Sequence Diagram*

Dalam penggunaannya, *Sequence Diagram* memiliki beberapa istilah sebagai berikut:

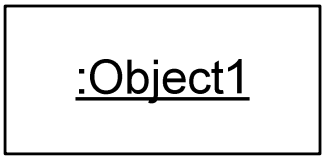
1. *Lifelines*

   Lifelines digunakan untuk menggambarkan jangka waktu dari suatu *intance* objek atau *role*. *Lifelines* digambarkan sebagai garis putus-putus di bawah objek (Pressman, 2010:849).

Gambar 2.20 *Lifelines*

2. *Instance Role*

   Instance role digunakan untuk menggambarkan *instance* yang terlibat dalam suatu proses. *Instance* umumnya menunjuk ke suatu objek. Nama dari *instance role* dapat diawali tanda titik dua jika mengacu pada sebuah objek (Pressman, 2010:849).



Gambar 2.21 *Instance*

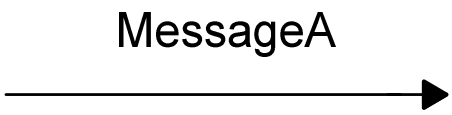
3. *Activation bar*

   Activation bar digunakan untuk menggambarkan lama waktu eksekusi suatu operasi. *Activation bar* digambarkan sebagai persegi panjang putih pada *lifelines* (Pressman, 2010:849).

Gambar 2.22 *Activation bar*

4. *Messages* (*call*)

   Messages (*call*) digunakan untuk menggambarkan pesan yang dikirim dari *instance* pengirim ke penerima. *Messages* (*call*) digambarkan sebagai sebuah garis panah penuhdengan nama *operation* yang dieksekusi. *Messages* (*call*) juga dapat berisi parameter, jenis serta *return type* dari *operation* yang dieksekusi. Jika *instance* pengirim dan penerima sama, maka panah dapat menunjuk kembali ke *instance* yang sama (Pressman, 2010, 849-851).



Gambar 2.23 *Messages (call)*

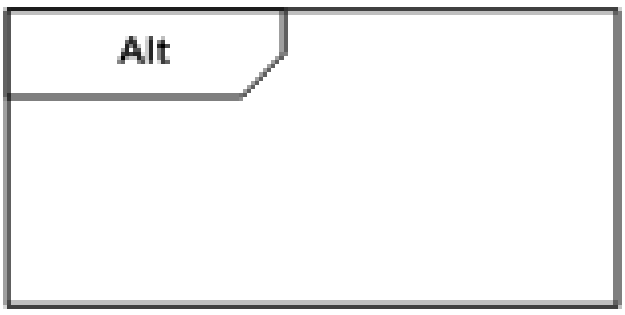
5. *Messages* (*return*)

   Messages (*return*) digunakan untuk menggambarkan pesan balasan (*reply*) dari *instance* penerima ke pengirim. *Messages*(*call*) digambarkan sebagai sebuah garis panah putus-putus (Pressman, 2010, 849).

Gambar 2.24 *Messages (return)*

6. *Interaction frames*

   Interaction frames digunakan untuk menggambarkan proses perulangan, seleksi dan struktur kontrol lainnya dalam *sequence diagram*. *Interaction frames* digambarkan sebagai sebuah persegi yang meliputi sebagian isi dari *sequence diagram* (Pressman, 2010:850).



Gambar 2.25 *Interaction frames*

7. *Object destruction*

   Object destruction digunakan untuk menggambarkan suatu objek yang tidak lagi digunakan. *Object destruction* digambarkan sebagai sebuah tanda X di akhir *lifelines* (Pressman, 2010:851).

2.2 Teori Khusus

2.2.1 *Computer Vision*

Computer Vision adalah suatu bidang ilmu yang bertujuan untuk melakukan transformasi data dari suatu gambar atau *video camera* ke dalam suatu keputusan atau representasi yang baru. Semua transformasi yang dilakukan bertujuan untuk mencapai tujuan tertentu. Suatu sistem mesin hanya dapat melihat urutan angka dari suatu gambar. Oleh karena itu, *Computer Vision* bertujuan mengubah sistem mesin dapat mengidentifikasi suatu gambar, seperti wajah dan rupa manusia. *Computer Vision* dapat diterapkan dalam berbagai area seperti robotik, *text recognition*, analisis medis, *game* dengan *Augmented Reality* (*AR*), dan lainnya (Bradski & Kaehler, 2008:2-3).

2.2.2 *Kinect*

Menurut (Catuhe, 2012:17), *Kinect* adalah alat pendeteksi gerakan dan suara untuk digunakan pada *Xbox* 360 dan *Xbox One* yang diproduksi oleh *Microsoft*. Pada tanggal 16 Juni 2011, *Microsoft* merilis *Kinect Software Development Kit* (*Kinect SDK*) yang dapat digunakan oleh para pengembang piranti lunak untuk memgembangkan aplikasi yang menggunakan *Kinect*. Aplikasi yang ingin dikembangkan dapat menggunakan bahasa pemrograman *C++*, *C#*, atau Visual Basic dengan menggunakan bantuan piranti lunak *Microsoft Visual Studio 2010*. *Kinect* memiliki beberapa fitur utama yang meliputi:

1. *Raw sensor stream* yang dapat digunakan untuk mengakses informasi dari *depth sensor*, *color camera sensor*, dan *microphone array*.
2. *Skeletal Tracking* yang dapat digunakan untuk mendeteksi skeleton dari pengguna yang berada dalam jangkauan jarak pandang *Kinect*.
3. *Advanced audio capabilities* yang memiliki beberapa kegunaan utama seperti *effective noise suppression, acoustic echo cancellation, beamforming and source locatization.*
4. Dokumentasi dan contoh *code*

Kinect menggunakan color camera untuk menghasilkan color stream dan depth stream yang dapat digunakan untuk mengukur jarak antara camera terhadap berbagai titik (Catuhe, 2012:12-14)..

Gambar 2.26 *Kinect*

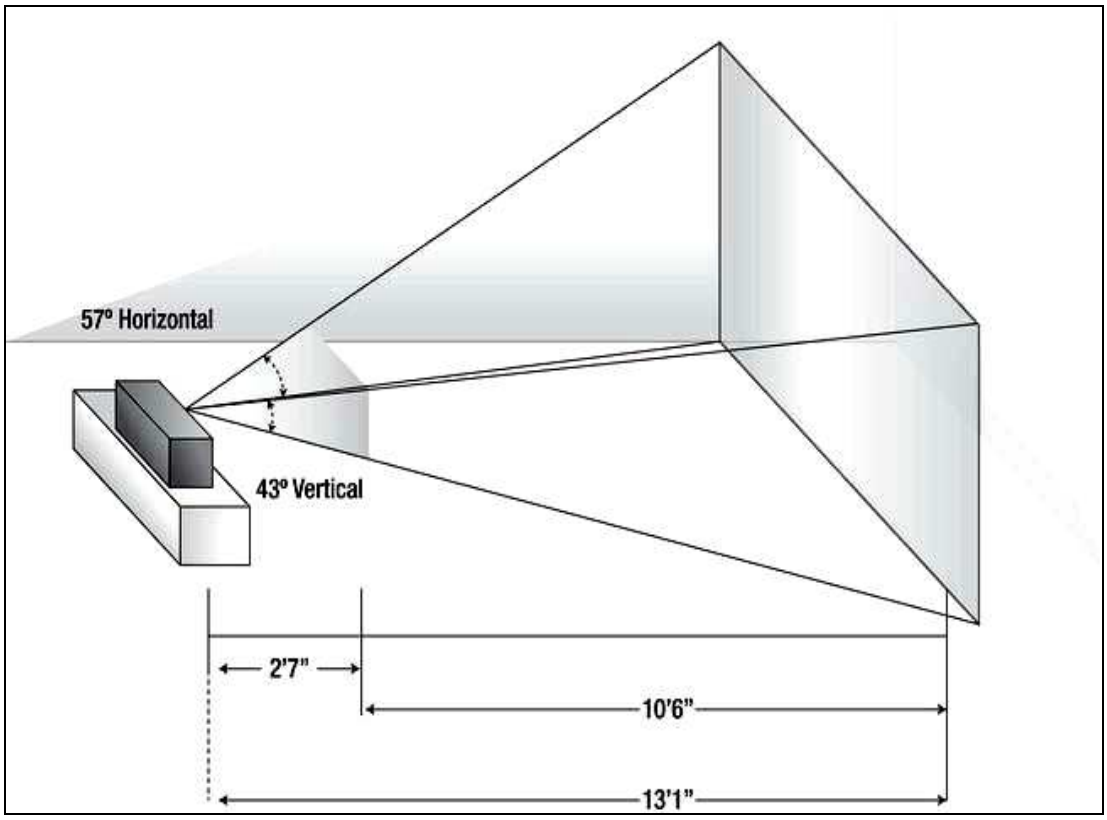
(sumber: https://upload.wikimedia.org/wikipedia/commons/6/67/Xbox-360-Kinect-Standalone.png)

Color camera yang dimiliki oleh *Kinect* dapat menangkap *video* dengan resolusi dan format sebagai berikut:

1. 640 × 480 pada tingkat 30 *frames per second* (*FPS*) dengan format *red*, *green*, and *blue* (*RGB*).
2. 1280 × 960 pada tingkat 12 *frames per second* (*FPS*) dengan format *RGB*.
3. 640 × 480 pada tingkat 15 *frames per second* (*FPS*) dengan format *YUV*.

Kinect memiliki batasan-batasan penggunaan tertentu agar penggunaan sensor dan camera yang ada pada *Kinect* dapat bekerja secara optimum. Batasan-batasan tersebut adalah:

1. Sudut pandang horizontal sebesar 57 derajat.
2. Sudut pandang vertikal sebesar 43 derajat.
3. Berada pada suhu ruangan 5^0 sampai dengan 35^0 Celcius.
4. Jarak pengguna yang berada pada 1.2 sampai dengan 4 meter untuk *standard mode* (0.4 sampai dengan 3 meter untuk *near mode*).
5. Jangkauan *depth* yang dihasilkan berada pada jangkauan 400 mm untuk *near mode* sampai 8000 mm untuk *standard mode*.



Gambar 2.27 Jangkauan jarak pandang *Kinect*

(sumber: (Webb & Ashley, 2012:69-75))

Penelitian ini menggunakan *Kinect* karena dapat mendeteksi nilai *depth* dengan cukup akurat dan mendeteksi informasi *joint location* dari *skeleton* pengguna.

Kinect Depth Data

Depth Camera yang dimiliki oleh *Kinect* dapat menghasilkan *depth stream* dengan resolusi sebagai berikut:

1. 640 × 480.
2. 320 × 240.
3. 80 × 60.

Sebagai contoh, resolusi 640 × 480 berarti terdapat 640 × 480 *pixel* pada *depth stream*, yang berarti terdapat 307200 *pixel* pada *depth stream* yang dihasilkan. Setiap pixel pada *depth frame* direpresentasikan dalam bentuk 16 bit data, dengan perincian sebagai berikut:

1. 13 bit untuk jarak sensor terhadap objek (*depth*).
2. 3 bit untuk deteksi keberadaan objek manusia (*player index*).

Gambar 2.28 Struktur *16-bit* pada setiap *depth pixel*

Untuk mendapatkan nilai jarak *depth* berdasarkan 13 bit di atas, dibutuhkan operasi bitwise *shift right* sebanyak 3 bit pada *depth pixel*. Sebagai contoh, misalkan diketahui struktur 16 bit pada depth pixel adalah sebagai berikut:

0011 1001 0110 1100

Maka, hasil operasi *bitwise shift right* sebanyak 3 bit adalah sebagai berikut:

0000 0111 0010 1101

Depth Pixel Bits	0011	1001	0110	1100
Shifted 3 Bits Right	0000	0111	0010	1101

2.2.2.1 *Pixel Filtering*

Depth Camera yang dimiliki oleh *Kinect* dapat menghasilkan *depth stream* dengan resolusi sebagai berikut:

1. 640 × 480.
2. 320 × 240.
3. 80 × 60.

Dari tiga jenis resolusi *depth stream* yang ada, resolusi 640 × 480 merupakan resolusi *depth stream* terbesar yang dihasilkan oleh *Kinect Sensor*. Namun, kualitas *depth stream* yang dihasilkan masih memiliki tingkat *noise* yang cukup besar. Gambar berikut ini merupakan contoh dari depth *stream* yang dihasilkan oleh *Kinect Sensor*:

Gambar 2.29 Contoh *Depth Stream*

(sumber: http://www.codeproject.com/KB/audio-video/317974/noisyintro.JPG)

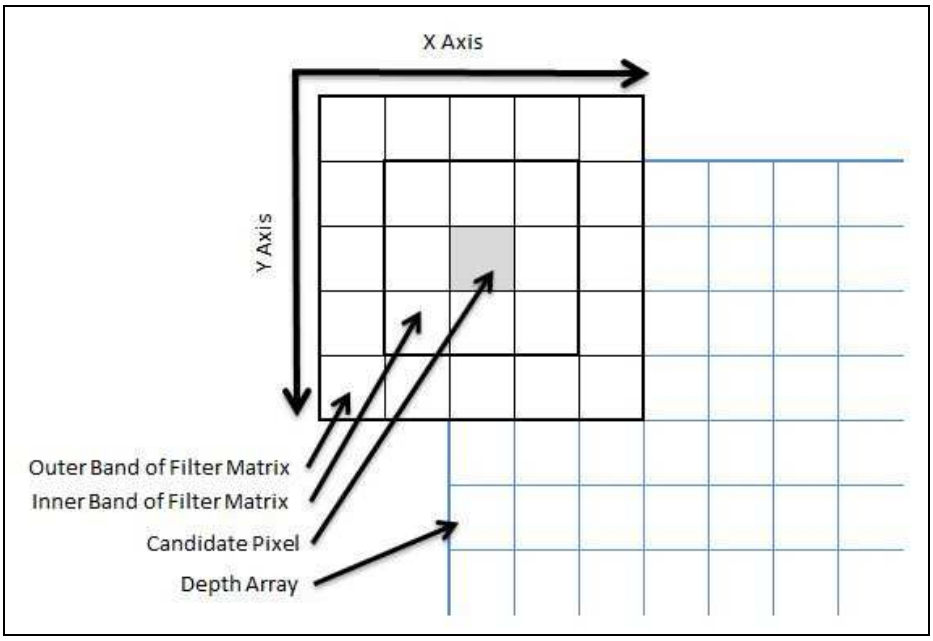
Berdasarkan gambar di atas, objek yang berada lebih dekat ke *sensor Kinect* akan berwarna lebih terang, sedangkan objek yang berada lebih jauh akan berwarna lebih gelap. Apabila objek berada di luar jangkauan jarak pandang sensor *Kinect*, maka objek akan digambarkan berwarna putih. Selain itu, pada gambar di atas setiap objek tergambarkan kurang memiliki bentuk yang teratur. Bentuk yang kurang teratur seperti pada gambar di atas akan mempengaruhi hasil pada proses pengukuran berdasarkan *Kinect Depth Data*.

Menurut (Sanford, 2012), *Pixel Filtering* adalah metode yang digunakan untuk memperhalus *depth data* yang dihasilkan dengan cara mengurangi jumlah *noise* yang ada pada *depth data* yang dihasilkan. Algoritma yang dilakukan dalam metode *Pixel Filtering* adalah sebagai berikut:

1. Hitung jarak dari sensor terhadap objek untuk setiap pixel yang ada pada *depth stream*.
2. Cari pixel yang memiliki jarak 0 milimeter atau berada di luar jangkauan jarak pandang Kinect. *Pixel* ini yang akan disebut sebagai *pixel candidate*.
3. Buat dua buah area yang mengelilingi pixel *candidate* tersebut. Kedua area ini akan disebut sebagai *inner band* dan *outer band*. Fungsi dari kedua area ini adalah untuk mencari pixel-pixel yang memiliki jarak lebih dari 0 milimeter di dalam areanya masing-masing.
4. Untuk area di dalam *inner band* dan *outer band*, hitung jumlah *pixel* yang memiliki jarak lebih dari 0 milimeter.

5. Jika jumlah non-0 pixel pada salah satu dari kedua area telah melebihi batas tertentu, maka nilai depth untuk *pixel* candidate akan digantikan dengan nilai modus dari non-0 *pixel* yang ada pada kedua area.

Gambar berikut ini merupakan penjelasan area *inner band* dan *outer band* dari algoritma *Pixel Filtering*:



Gambar 2.30 *Inner Band* dan *Outer Band* dari *Candidate Pixel*

(sumber: http://www.codeproject.com/KB/audio-video/317974/filter.JPG)

2.2.2.2 *Weighted Moving Average*

Menurut (Sanford, 2012), *Weighted Moving Average* adalah metode yang digunakan untuk mengurangi *flicker* pada *depth data* yang disebabkan oleh adanya noise pada *depth data*. Tahapan algoritma yang digunakan dalam metode *Weighted Moving Average* adalah sebagai berikut:

1. Inisialisasi *queue* yang akan digunakan untuk menyimpan beberapa *frame* terakhir dari *depth stream*.
2. Ambil *frame* terakhir dari *depth stream* dan simpan ke dalam *queue*.
3. Beri nilai beban dengan ketentuan nilai terbesar untuk *frame* terbaru dan nilai terkecil untuk *frame* yang tersimpan paling awal.
4. Jumlahkan nilai dari setiap pixel untuk koordinat x dan y tertentu dari setiap *frame* dengan formula sebagai berikut:

$$sum(x,y) = \sum_{i=1}^{5} f_i(x,y) * w_i$$

Di mana:

- $sum(x, y)$ merupakan nilai penjumlahan untuk setiap *pixel* pada 5 *frame* terakhir untuk koordinat (x, y)
- $f_i(x, y)$ merupakan nilai *pixel* untuk koordinat (x, y) pada 5 *frame* terakhir
- w_i merupakan bobot yang diberikan untuk setiap *frame* dan nilainya bervariasi antara 1-5

5. Hitung nilai rata-rata untuk setiap nilai $sum(x, y)$ dan simpan sebagai nilai baru bagi *pixel* pada koordinat x dan y pada *depth data*. Formula yang digunakan untuk menghitung nilai rata-rata adalah sebagai berikut:

$$avg(x,y) = \frac{sum(x,y)}{5}$$

Di mana:

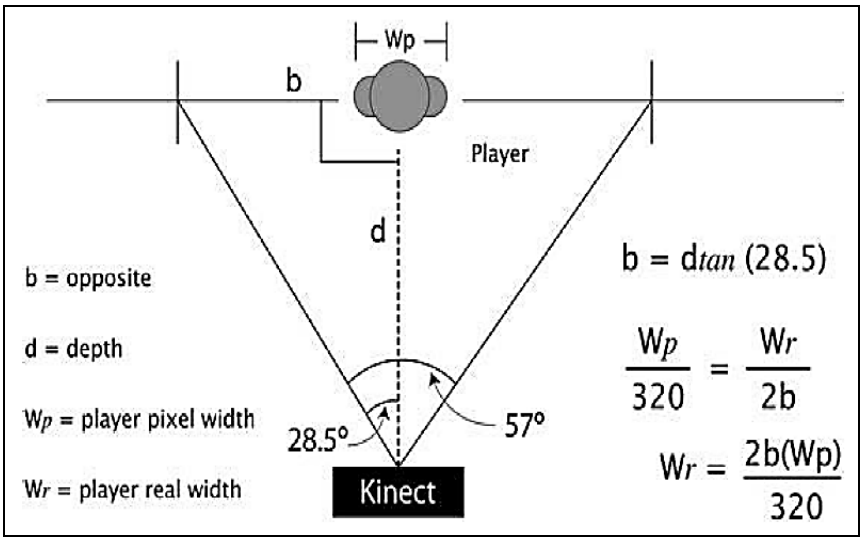
- $avg(x,y)$ merupakan nilai rata-rata untuk setiap *pixel* pada 5 *frame* terakhir untuk koordinat (x, y)
- $sum(x, y)$ merupakan nilai penjumlahan untuk setiap *pixel* pada 5 *frame* terakhir untuk koordinat (x, y)

2.2.2.3 *Kinect Depth Data Measurement*

Menurut Webb dan Ashley (2012:69-75), posisi koordinat X dan Y dari masing-masing *depth pixel* yang dihasilkan tidak berhubungan dengan pengukuran lebar dan tinggi dari sebuah objek yang sebenarnya. Namun, pengukuran lebar dan tinggi dari sebuah objek berdasarkan *depth pixel* yang dihasilkan sangat memungkinkan untuk dilakukan. Faktor-faktor yang dapat digunakan untuk melakukan pengukuran berdasarkan *depth pixel* adalah:

1. Jangkauan pandangan kamera Kinect sebesar 57^0 horizontal dan 43^0 vertikal.
2. Jarak *depth value*.
3. Perhitungan trigonometri

Gambar berikut akan memberikan ilustrasi pengukuran berdasarkan *depth pixel* dengan menggunakan faktor-faktor di atas:



Gambar 2.31 Ilustrasi pengukuran lebar dan tinggi menggunakan *depth pixel* (sumber: (Webb & Ashley, 2012:69-75))

Dimana:

- d adalah jarak *depth value* setiap *pixel*.
- Wp adalah lebar objek yang diukur dalam satuan *pixel*.
- Wr adalah lebar objek yang sesungguhnya dalam satuan meter atau centimeter.
- W adalah lebar dari resolusi *depth stream* yang dihasilkan oleh *Kinect sensor* (biasanya bernilai 640 atau 320).
- H adalah tinggi dari resolusi *depth stream* yang dihasilkan oleh *Kinect sensor* (biasanya bernilai 480 atau 240).

Sebagai contoh, diketahui jarak *depth value* (d) berdasarkan *sensor Kinect* adalah 1500 mm, lebar objek dalam *pixel* (Wp) adalah 90 *pixel*, lebar resolusi *depth stream* (W) adalah 640 *pixel*. Maka lebar objek sesungguhnya (Wr) berdasarkan pengukuran didapat berdasarkan perhitungan berikut:

$$b = d \times \tan(28.5) = 1500\ mm \times 0.543 = 814.434$$

$$Wr = \frac{2b \times Wp}{640} = \frac{2 \times 814.434 \times 90}{640} = 229.059\ mm$$

Perhitungan tinggi objek berdasarkan *depth pixel* menggunakan standar yang sama, namun parameter yang digunakan adalah tinggi objek dalam satuan pixel (Hp), tinggi resolusi *depth stream* (H) yang dihasilkan oleh *Kinect sensor*, dan besar sudut pandang dari *Kinect sensor* sebesar 43^0 vertikal.

2.2.2.4 Pengukuran Langsung Bagian Tubuh

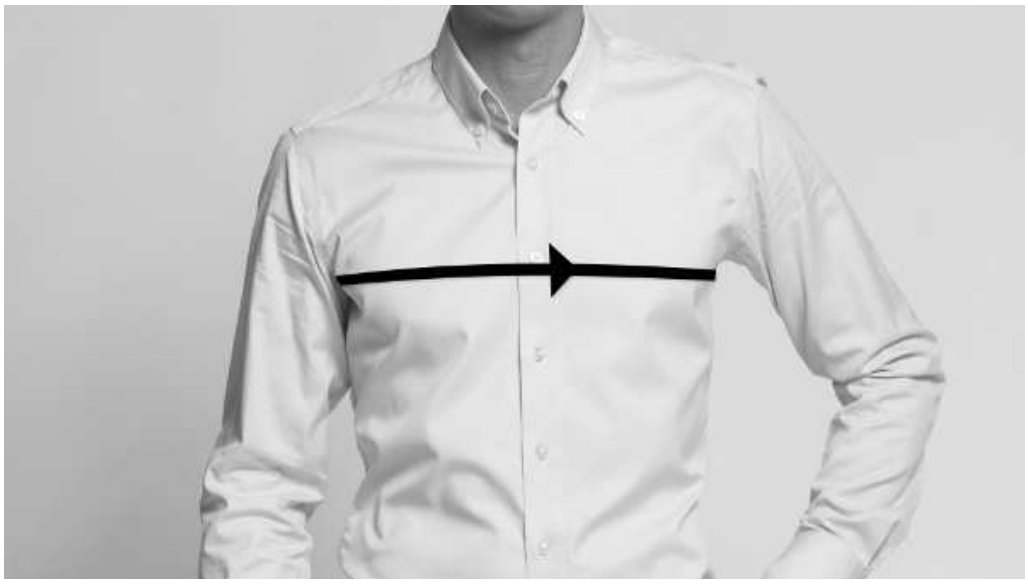
Pengukuran langsung pada bagian tubuh dilakukan dengan alat ukur panjang. Pengukuran dilakukan pada bagian tubuh lebar bahu (*shoulder width*), lingkar dada (*chest width*), lingkar pinggul (*hip width*), dan panjang tubuh bagian atas.

Pengukuran pada lebar bahu (*shoulder width*) dilakukan dengan mengukur panjang antara 2 titik perpanjangan ketiak pada bahu secara vertikal. Gambar berikut ini menunjukkan pengukuran pada lebar bahu (*shoulder width*).

Gambar 2.32 Pengukuran pada Lebar Bahu (*shoulder width*)

(sumber: https://www.tailorstore.com/mens-body-measurements.pdf)

Pengukuran pada lingkar dada (*chest width*) dilakukan dengan melingkarkan alat ukur panjang pada bagian dada di titik-titik terluar pada dada. Gambar berikut ini menunjukkan pengukuran pada lingkar dada.



Gambar 2.33 Pengukuran pada Lingkar Dada (*chest width*)

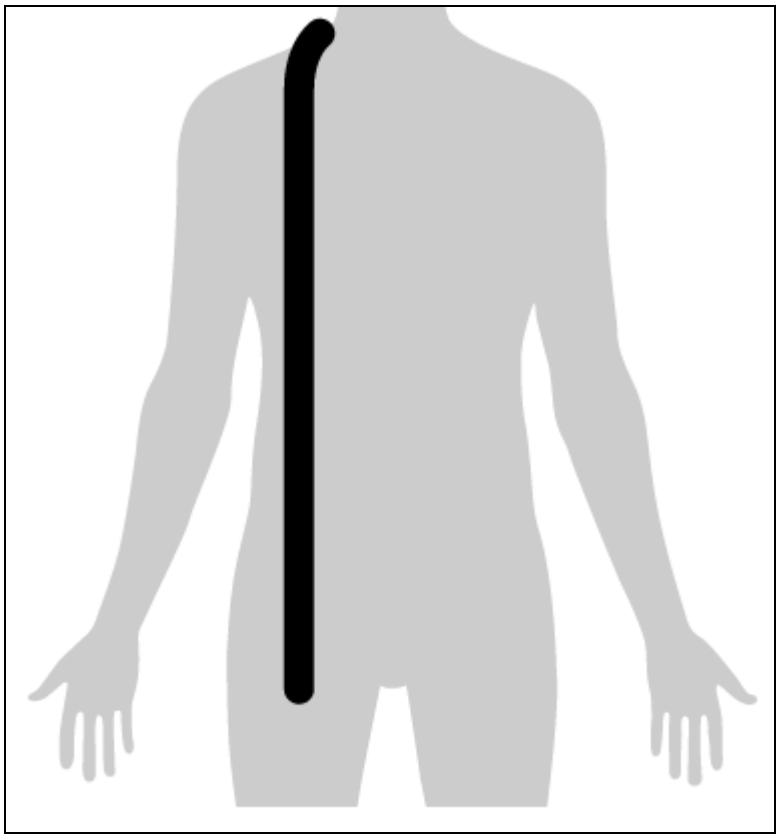
(sumber: https://www.tailorstore.com/mens-body-measurements.pdf)

Pengukuran pada lingkar pinggul dilakukan dengan melingkarkan alat ukur panjang pada bagian pinggul di titik-titik terluar pada pinggul. Gambar berikut ini menunjukkan pengukuran pada lingkar pinggul.

Gambar 2.34 Pengukuran pada Lingkar Pinggul (*hip width*)

(sumber: https://www.tailorstore.com/mens-body-measurements.pdf)

Pengukuran pada panjang tinggi bagian atas tubuh pengguna dilakukan dengan mengukur panjang antara bahu dan pinggul. Gambar berikut ini menunjukkan pengukuran pada tinggi tubuh bagian atas (*upper body height*).

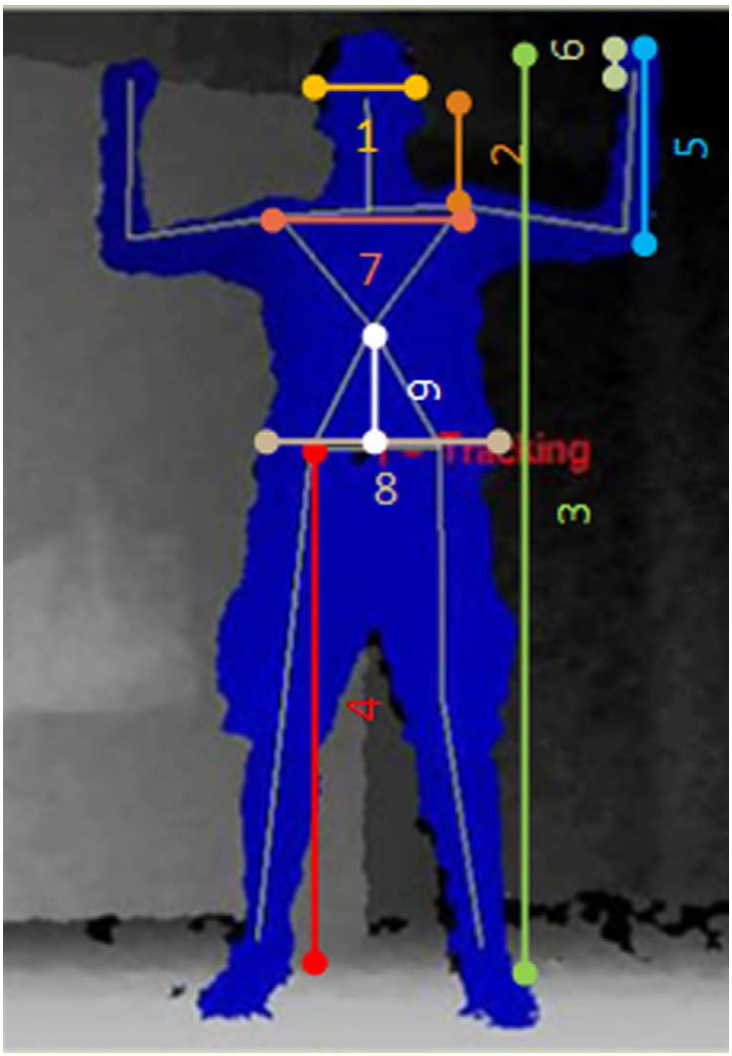


Gambar 2.35 Gambar Pengukuran Tubuh Bagian Atas (*Upper Body Height*)

(sumber: https://www.tailorstore.com/mens-body-measurements.pdf)

2.2.2.5 *Pengukuran Tidak Langsung*

Pengukuran tidak langsung dilakukan dengan menggunakan *Kinect* sebagai alat bantu. Pengukuran dilakukan untuk panjang lebar bahu (*shoulder width*), lebar dada (*chest width*), lebar pinggul (*hip width*), dan panjang tubuh bagian atas (*upper body height*). Pengukuran pada keempat bagian tubuh tersebut dihitung berdasarkan panduan pada gambar berikut ini.



Gambar 2.36 Panduan Pengukuran Panjang Bagian Tubuh

(sumber: (Gültepe & Güdükbay, 2014))

Pengukuran pada lebar bahu (*shoulder width*) dilakukan berdasarkan pada poin nomor 7 pada gambar 2.43. Lebar bahu didapat berdasarkan titik *skeleton ShoulderCenter* pada tubuh yang dideteksi oleh *Kinect*. Berdasarkan titik skeleton *ShoulderCenter*, ditarik garis horizontal pada tubuh pengguna dan akan dihitung jumlah pixel sepanjang garis horizontal tersebut berdasarkan metode *Kinect Depth Data Measurement.*

Pengukuran pada lebar pinggul (*hip width*) dilakukan berdasarkan pada poin nomor 8 pada gambar 2.43. Lebar pinggul didapat berdasarkan titik *skeleton HipCenter* pada tubuh yang dideteksi oleh *Kinect*. Berdasarkan titik skeleton *HipCenter*, ditarik garis horizontal pada tubuh pengguna dan akan dihitung jumlah

pixel sepanjang garis horizontal tersebut berdasarkan metode *Kinect Depth Data Measurement.*

Pengukuran pada lebar dada (*chest width*) dilakukan berdasarkan titik *skeleton ShoulderLeft* dan *ShoulderRight* pada tubuh yang dideteksi oleh *Kinect*. Nilai panjang lebar dada (*chest width*) dihitung berdasarkan jarak antara kedua titik tersebut dengan menggunakan formula *Euclidean Distance*.

2.2.3 *Microsoft XNA*

Microsoft XNA adalah kumpulan *tools* dengan *runtime environment* yang terorganisir yang dikembangkan oleh *Microsoft* dengan tujuan untuk pengembangan *video game* dan manajemen *video game*. XNA dibangung berdasarkan *.NET Framework* yang dirancang untuk berjalan pada *Windows NT, Windows Phone, Xbox 360, Windows XP*, *Windows Vista*, dan *Windows 7*. Aplikasi yang menggunakan *XNA Framework* secara teknis dapat ditulis dengan bahasa pemrograman yang mendukung *.NET Framework* seperti *C#.NET* dan *VB.NET*. Keuntungan penggunaan *XNA Framework* dalam pembuatan aplikasi adalah dukungan dalam menampilkan model 3D (Jaegers, 2012:1-2).

2.2.4 *KinectXNAGui Framework*

KinectXNAGui Framework dibuat dengan tujuan untuk mempermudah proses pembuatan tampilan aplikasi yang menggunakan *XNA Framework* sebagai dasar dari aplikasi. Kelemahan dari penggunaan *XNA Framework* adalah tidak adanya dukungan untuk membuat tampilan layar yang mudah dan cepat, sehingga tampilan layar perlu dibangun sejak awal pembuatan aplikasi. *KinectXNAGui Framework* mengintegrasikan dasar-dasar tampilan layar yang menggunakan *XNA Framework* dan interaksi pengguna dengan *Kinect*.

2.2.5 *Fuzzy Logic*

Fuzzy Logic merupakan salah satu bentuk dari *multi-value logic* yang menentukan *reasoning* berdasarkan perkiraan, dibandingkan dengan *reasoning* yang tetap dan pasti. Apabila dibandingkan dengan logika biner, variabel pada *fuzzy* logic dapat memiliki nilai kebenaran yang berada pada rentang $[0-1]$ (Ross, 2010:132). Proporsi dari nilai kebenaran ($T(P)$) dapat dinyatakan sebagai berikut:

$$T(P) = \mu_A(x), \qquad 0 \leq \mu_A \leq 1$$

Pendekatan *fuzzy* memiliki kelebihan pada hasil yang terkait dengan pengenalan pola dan pengambilan keputusan dalam kondisi yang tidak pasti. Berikut ini adalah berbagai alasan mengenai penerapan *Fuzzy Logic* dalam kehidupan:

1. Konsep *Fuzzy Logic* mudah dimengerti.
2. *Fuzzy Logic* sangat fleksibel.
3. *Fuzzy Logic* memiliki toleransi terhadap data yang tidak tentu.
4. *Fuzzy Logic* mampu memodelkan berbagai fungsi nonlinear.
5. *Fuzzy Logic* dapat membangun pengalaman para pakar dalam aplikasi tanpa harus melalui proses pelatihan.

2.2.5.1 Himpunan *Fuzzy* (*Fuzzy* Set)

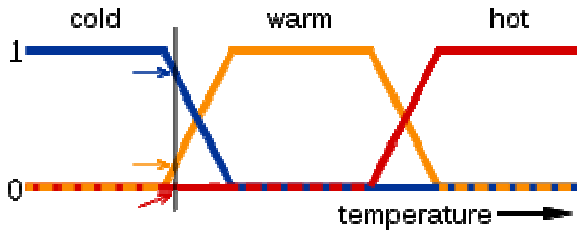
Menurut (Ross, 2010:34), himpunan *fuzzy* (*fuzzy set*) adalah himpunan yang mengandung berbagai variasi derajat keanggotaan. Sebuah himpunan *Fuzzy* (*A*) dapat dinyatakan sebagai berikut:

$$A = \left\{ \frac{\mu_A(x_1)}{x_1} + \frac{\mu_A(x_2)}{x_2} + \dots \right\} = \{ \sum_i^n \frac{\mu_A(x_i)}{x_i} \}$$

Himpunan *Fuzzy* memiliki 2 atribut, yaitu:

1. Linguistik, yaitu penamaan suatu grup yang mewakili suatu keadaan atau kondisi tertentu dengan menggunakan bahasa alami, seperti: panas, dingin, dan hangat.
2. Numeris, yaitu suatu nilai (angka) yang menunjukkan ukuran dari suatu variabel seperti: 10, 20, dan 30.

Berikut ini adalah contoh interpretasi dari himpunan *Fuzzy* dengan variabel berupa temperatur:



Gambar 2.37 Himpunan *Fuzzy* dengan variabel temperatur

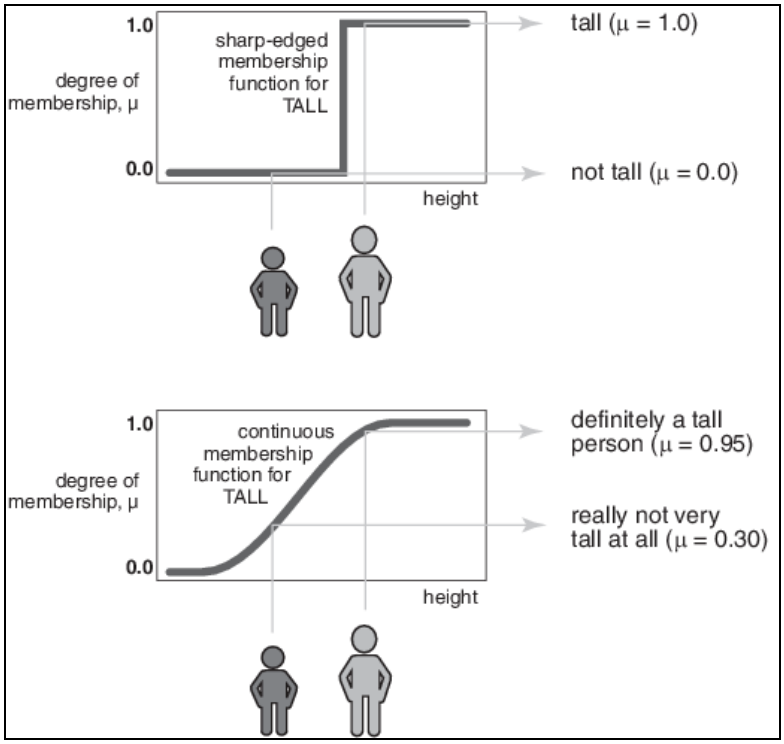
(sumber: http://m.eet.com/media/1158143/fuzzy.jpg)

2.2.5.2 Fungsi Keanggotaan (*Membership Function*)

Fungsi keanggotaan (*membership function*) adalah kurva yang mendefinisikan bagaimana setiap titik pada ruang input dipetakan ke sebuah nilai

keanggotaan (derajat keanggotaan) yang bernilai 0 sampai 1. Sebuah fungsi keanggotaan dapat terdiri dari bagian, yaitu:

1. *Core* adalah daerah dari fungsi keanggotaan dimana elemen daerah tersebut (x)memiliki nilai derajat kenggotaan penuh ($\mu_{A(x)} = 1$).
2. *Support* adalah daerah dari fungsi keanggotaan dimana elemen daerah tersebut (x) memiliki nilai derajat kenggotaan yang lebih besar dari 0 ($\mu_{A(x)} > 0$).
3. *Boundaries* adalah daerah dari fungsi keanggotaan dimana elemen daerah tersebut (x) memiliki nilai derajat kenggotaan yang lebih besar dari 0 tetapi tidak penuh ($0 < \mu_{A(x)} < 1$).



Gambar 2.38 Fungsi Keanggotaan

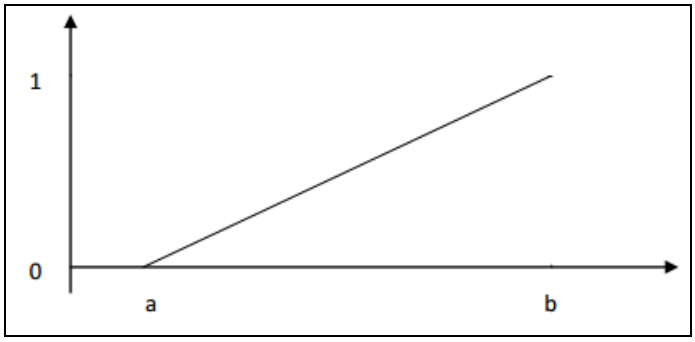
(sumber: http://www.mathworks.com/help/releases/R2015a/fuzzy/fuzzy_tall.png)

Suatu himpunan *Fuzzy* disebut normal jika memiliki minimal 1 elemen yang memiliki nilai derajat kenggotaan penuh. Berikut ini adalah berbagai fungsi sederhana yang dapat digunakan sebagai pendekatan untuk mendapatkan derajat keanggotaan:

1. Representasi Linear

Representasi linear memetakan derajat keanggotaannya sebagai suatu garis lurus sederhana (naik atau turun). Representasi ini digunakan untuk suatu masalah yang belum tentu atau jelas. Fungsi keanggotaan dari representasi linear dinyatakan sebagai berikut:

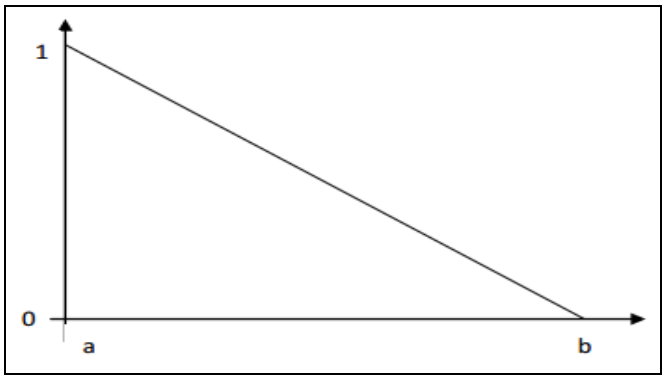
1. Representasi Linear Naik, seperti pada gambar berikut:



Gambar 2.39 Representasi *Linear* Naik

Fungsi keanggotaan: $\mu_{(x)} = \begin{cases} 0; & x \leq a \\ \frac{x-a}{b-a}; & a < x < b \\ 1; & x \geq b \end{cases}$

2. Representasi Linear Turun, seperti pada gambar berikut:

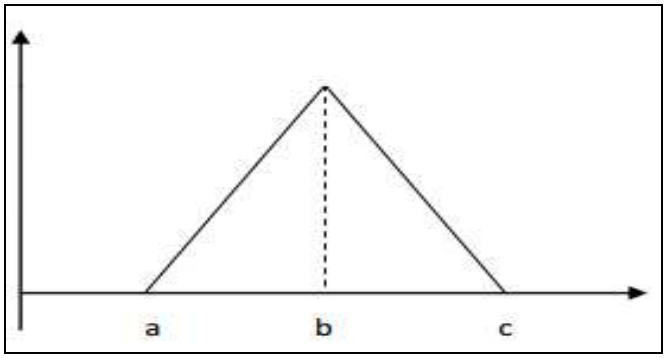


Gambar 2.40 Representasi *Linear* Turun

Fungsi keanggotaan: $\mu_{(x)} = \begin{cases} 1; & x \leq a \\ \frac{b-x}{b-a}; & a < x < b \\ 0; & x \geq b \end{cases}$

2. Representasi kurva segitiga

Representasi linear memetakan derajat keanggotaannya sebagai suatu segitiga, seperti pada gambar berikut:

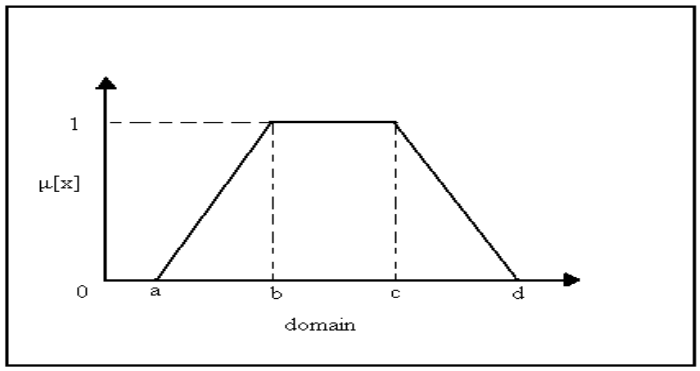


Gambar 2.41 Representasi Kurva Segitiga

Fungsi keanggotaan: $\mu_{(x)} = \begin{cases} 0; x \leq a \vee x \geq c \\ \frac{x-a}{b-a}; a \leq x \leq b \\ \frac{c-x}{c-b}; b \leq x \leq c \end{cases}$

3. Representasi Kurva Trapesium

   Representasi linear memetakan derajat keanggotaannya sebagai suatu trapesium (beberapa titik memiliki nilai derajat keanggotaan penuh), seperti pada gambar berikut:

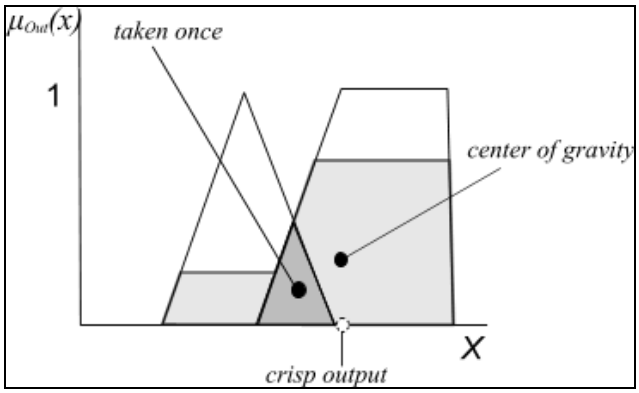


Gambar 2.42 Representasi Kurva Trapesium

Fungsi keanggotaan: $\mu_{(x)} = \begin{cases} 0; x \leq a \vee x \geq d \\ \frac{x-a}{b-a}; a \leq x \leq b \\ 1; b \leq x \leq c \\ \frac{d-x}{d-c}; c \leq x \leq d \end{cases}$

2.2.5.3 *Defuzzification*

Defuzzification adalah proses konversi nilai dari nilai yang bersifat *fuzzy* ke nilai tertentu berdasarkan himpunan *fuzzy* serta komposisi *fuzzy rule*.



Gambar 2.43 Gambar *defuzzification* (*CoG*)

Output fuzzy diperoleh melalui eksekusi dari beberapa fungsi keanggotaan *fuzzy*. Terdapat beberapa metode yang dapat digunakan pada proses *defuzzification*, diantaranya sebagai berikut:

1. *CoG (Center of Gravity)*

   Metode ini mengambil nilai tengah dari seluruh fungsi keanggotaan keluaran *fuzzy* yang ada. Nilai tengah didapat dari nilai titik berat dari suatu bidang. Formula untuk menghitung nilai *output* CoG adalah sebagai berikut:

$$CoG(z) = \frac{\sum \mu(z) \times z}{\sum \mu(z)}$$

   Di mana:

   - *z* adalah titik berat dari masing-masing simetris fungsi keanggotaan
   - $\mu(z)$ adalah nilai pada kurva pada titik z
   - $CoG(z)$ adalah nilai output *fuzzy* berdasarkan *Center of Gravity*

2. *Max Membership Principle*

   Metode ini mengambil nilai fungsi keanggotaan terbesar dari keluaran *fuzzy* yang ada untuk dijadikan sebagai nilai *output*.

2.2.6 Uji *Modified Thompson Tau*

Menurut (John M.Cimbala, 2011), uji statistik *modified Thompson tau* adalah uji statistik yang digunakan untuk menentukan apakah membuang atau menyimpan sebuah data yang disuspect sebagai pencilan (*outlier*) dalam sebuah sampel yang terdiri dari variabel tunggal. Pencilan (*outlier*) didefinisikan sebagai data yang secara statistik tidak konsisten dengan keseluruhan data yang ada. Prosedur yang digunakan dalam uji statistik *Modified Thompson tau* adalah:

1. Hitung nilai rata-rata $(\bar{x})$ dan standar deviasi (S) dari sampel (n)
2. Untuk setiap nilai dari data, hitung $\delta_i = |x_i - \bar{x}|, i = 1,2,3,..,n$

3. Hitung $\tau = \frac{t_{\frac{\alpha}{2}}(n-1)}{\sqrt{n}\sqrt{n-2+t_{\frac{\alpha}{2}}^2}}$

   Dimana:

   - n = Banyaknya nilai dari data.
   - $t_{\frac{\alpha}{2}}$ = Nilai kritis yang ditetapkan.

   Nilai τ juga dapat diperoleh dari tabel *Modified Thompson* τ berikut:

Tabel 2.3 Tabel *Modified Thompson* τ

Values of the Modified Thompson τ					
n	τ	n	τ	n	τ
3	1.1511	21	1.8891	40	1.9240
4	1.4250	22	1.8926	42	1.9257
5	1.5712	23	1.8957	44	1.9273
6	1.6563	24	1.8985	46	1.9288
7	1.7110	25	1.9011	48	1.9301
8	1.7491	26	1.9035	50	1.9314
9	1.7770	27	1.9057	55	1.9340
10	1.7984	28	1.9078	60	1.9362
11	1.8153	29	1.9096	65	1.9381
12	1.8290	30	1.9114	70	1.9397
13	1.8403	31	1.9130	80	1.9423
14	1.8498	32	1.9146	90	1.9443
15	1.8579	33	1.9160	100	1.9459
16	1.8649	34	1.9174	200	1.9530
17	1.8710	35	1.9186	500	1.9572
18	1.8764	36	1.9198	1000	1.9586
19	1.8811	37	1.9209	5000	1.9597
20	1.8853	38	1.9220	($\rightarrow \infty$)	1.9600

4. Suatu nilai dari data disebut pencilan jika $\delta_i > \tau S$, dan sebaliknya.

2.2.7 Analisis Korelasi

Menurut (Walpole, Myers, Ye, 2007:432), analisis korelasi digunakan untuk mengukur kekuatan dari hubungan antara dua buah variabel yang ditandai dengan koefisien korelasi di mana masing-masing variabel diasumsikan sebagai variabel independen. Koefisien korelasi dapat dinyatakan sebagai berikut:

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 \sum_{i=1}^{n}(y_i - \overline{y})^2}}$$

Di mana:

- r adalah koefisien korelasi
- $\overline{x}$ adalah rata-rata variabel X
- $\overline{y}$ adalah rata-rata variabel Y

2.2.8 Klasifikasi Ukuran Pakaian

Penentuan klasifikasi ukuran pakaian dilakukan berdasarkan tabel klasifikasi ukuran pakaian dari suatu produsen pakaian internasional. Tabel berikut ini menunjukkan klasifikasi ukuran pakaian.

	XS	S		M		L		XL		XXL	
EUR SIZE	42	44	46	48	50	52	54	56	58	60	62
US, UK SIZE TOP	32R	34R	36R	38R	40R	42R	44R	46R	48R	50R	52R
US, UK SIZE BOTTOM	28R	30R	32R	33R	34R	36R	38R	40R	42R	44R	46R
CHEST (CM)	84	88	92	96	100	104	108	112	116	120	124
WAIST (CM)	72	76	80	84	88	92	96	100	104	108	112
SEAT (CM)	88	92	96	100	104	108	112	116	120	124	128
INSIDE LEG (CM)	79	80	81	82	83	84	85	86	87	88	89

Gambar 2.44 Tabel Klasifikasi Ukuran Pakaian

(sumber: http://www.hm.com/id/sizeguide/sizeguide_men)

2.3 Penelitian Lainnya

U.Gültepe dan U. Güdükbay (2014) dengan *paper* berjudul "*Real-time Virtual Fitting with Body Measurement and Motion Smoothing"*, mengajukan sebuah metode baru yang menggunakan *depth sensor* agar proses pengepasan (*fitting*) pakaian menjadi lebih realistik. Untuk mengatasi keakuratan *depth data*, maka pengguna akan diukur dulu menggunakan *Kinect*. Selain itu juga digunakan informasi gerakan (*motion*) nya agar pengepasan dapat lebih akurat. Hasil penelitian didapat metode yang cukup efisien tetapi dengan peningkatan kinerja pengepasan yang semakin realistik. Kekurangan dari penelitian ini adalah penggunaan model *virtual 3D avatar* sebagai pengganti citra pengguna.

S. Giovanni et al (2012) dengan *paper* yang berjudul "*Virtual Try-on Using Kinect and HD camera*", merancang aplikasi *VFR* menggunakan bantuan *Kinect* untuk menerima *input* berupa *depth map image* dan kamera *High Definition* (*HD*)

sebagai bantuan untuk menerima *input* berupa rekaman video pengguna. Dalam penelitian ini, peneliti juga membandingkan penggunaan *openNI* dan *Kinect* for *Windows Software Development Kits (SDK)* dalam hal *depth sensing*. Kekurangan dari penelitian ini adalah pengepasan pakaian yang kurang akurat ketika digunakan oleh orang yang memiliki lebar tubuh cukup besar. Selain itu, penggunaan kamera *HD* untuk menerima *input* berupa rekaman *video* pengguna. Penggunaan kamera *HD* menyebabkan total biaya yang dibutuhkan untuk membuat sistem secara keseluruhan menjadi lebih mahal karena harga kamera *HD* yang relatif mahal.

Mao, ye et al (2014) dengan *paper* berjudul "*Real-Time Human Pose and Shape Estimation for Virtual Try-On Ssing a Single Commodity Depth Camera*", mengajukan metode baru untuk mengestimasi pose dan bentuk tubuh dari pengguna secara real-time dengan *template-based approach.* Pendekatan ini secara umum melakukan deteksi pose dan proses kalkulasi bentuk tubuh secara bersamaan. Hasil dari proses ini akan digunakan untuk proses *cloth-simulation* di mana hasilnya akan digabungkan dengan citra gambar pengguna untuk ditampilkan di layar. Kekurangan: algoritma yang digunakan dalam proses *cloth-simulation* menggunakan sumber daya komputasi yang sangat besar.

Khoshelham, Kourosh, dan Sander Oude Elberink (2012) dengan *paper* berjudul "*Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications*" melakukan analisis terhadap akurasi dan resolusi dari *Kinect depth data*. Analisis *error* dilakukan berdasarkan model matematika dari pengukuran tingkat perbedaan *depth data.* Kesimpulan yang didapat dari penelitian ini adalah tingkat *error* dari pengukuran *depth data* meningkat secara kuadratik seiring dengan jarak objek yang semakin menjauh dari sensor, resolusi nilai *depth data* juga menurun secara kuadratik seiring dengan jarak obyek yang semakin menjauh dari sensor, kalibrasi kamera *IR* dan *RGB* sangat diperlukan untuk mengeliminasi distorsi dan ketidaksesuaian antara *colour* dan *depth data*.

Khoshelham (2011) dengan *paper* berjudul "*Accuracy Analysis of Kinect Depth Data*" melakukan analisis terhadap akurasi dari *Kinect Depth Data*. Kesimpulan yang didapat dari penelitian ini adalah *sensor Kinect* yang sudah dikalibrasi dengan baik tidak mengandung *error* sistematik yang besar bila dibandingkan dengan *laser scanning data*, tingkat *error* dari *depth measurement* meningkat secara kuadrat dengan peningkatan jarak dari sensor dan mencapai 4cm pada jarak maksimum, kerapatan titik-titik juga menurun seiring dengan peningkatan

jarak terhadap *sensor*. Faktor yang mempengaruhi adalah *depth resolution* yang sangat rendah pada jarak yang jauh (7cm pada jarak maksimum 5m).

Isikdogan, F. dan Kara, G (2012) dengan *paper* berjudul "A Real Time Virtual Dressing Room Application using Kinect" membuat aplikasi *Virtual Dressing Room* dengan menggunakan *Kinect*. Model pakaian dibuat dalam bentuk dua dimensi dan diletakkan pada citra gambar pengguna berdasarkan *joint location* pada tubuh pengguna.