# BAB III
# ANALISIS DAN PERANCANGAN SISTEM

## 3.1 Analisis Sistem

Analisis sistem adalah kegiatan untuk melihat sistem yang sudah berjalan, melihat bagian mana yang bagus dan tidak bagus, dan kemudian mendokumentasikan kebutuhan yang akan dipenuhi dalam sistem yang baru. Analisis merupakan tahapan yang paling penting, karena kesalahan dalam tahap ini akan menyebabkan kesalahan di tahap selanjutnya. Sistem yang dibuat merupakan edukasi menggunakan teknologi *Augmented Reality* yaitu media pembelajaran pengenalan anatomi tubuh manusia untuk anak SD khususnya kelas 5. Aplikasi yang dibuat seolah-olah pengguna dapat berinteraksi langsung dengan objek-objek 3D yang dibuat. Aplikasi ini dibuat dengan mengambil latar secara *real time* yang kemudian digabungkan dengan objek 3D melalui kamera.

### 3.1.1 Analisis Masalah

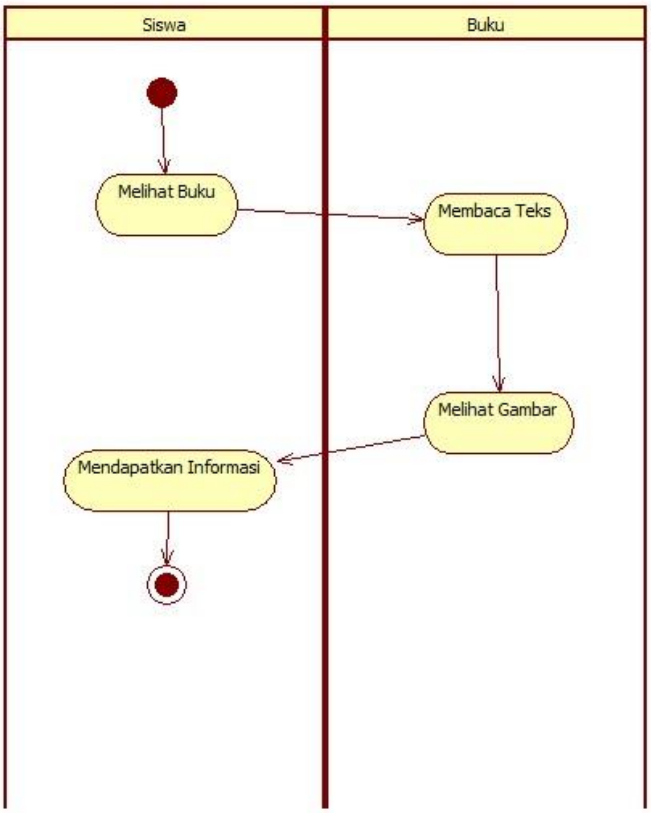
Analisis masalah adalah suatu gambaran masalah yang diangkat dalam penulisan penelitian ini tentang mengimplementasikan teknologi *Augmented Reality.* Analisis masalah yang dimaksud disini adalah pada sistem pembelajaran *Primary Darul Hikam International School* saat ini masih bersifat 2D, *full text*, gambar tidak jelas, dan tidak interaktif. Media tersebut masih belum mampu menjadi media pengenalan anatomi tubuh manusia yang menarik untuk anak kelas 5 SD.

Dengan teknologi yang berkembang saat ini, khususnya teknologi *Augmented Reality* yang dapat menambah nilai dari penyampaian informasi menjadi lebih tinggi dan mendukung animasi 3D serta suara dapat menjadi sarana untuk menutupi kekurangan yang terdapat pada media pembelajaran saat ini. Dengan merujuk dari kekurangan yang ada maka dibangunlah aplikasi *Smart Book* pengenalan anatomi tubuh manusia ini menggunakan teknologi *Augmented Reality. Content* dari *Smart Book* tentang anatomi tubuh manusia ini berdasarkan kurikulum pembelajaran di *Primary Darul Hikam International School.* Dengan

menggunakan teknologi *Augmented Reality* ini dapat memungkinkan pengguna melihat objek anatomi tubuh manusia secara lebih nyata dengan memproyeksikan objek dalam bentuk 3D (tiga dimensi) dan suara.

### 3.1.2 Analisis Sistem yang Berjalan

Analisis sistem yang berjalan merupakan tahapan yang memberi gambaran tentang sistem yang sedang berjalan sekarang. Analisa ini bertujuan untuk memberi gambaran yang lebih detail bagaimana cara kerja sistem yang ada sekarang. Prosedur pada proses media yang sedang berjalan sekarang dapat dilihat pada gambar 3.1.



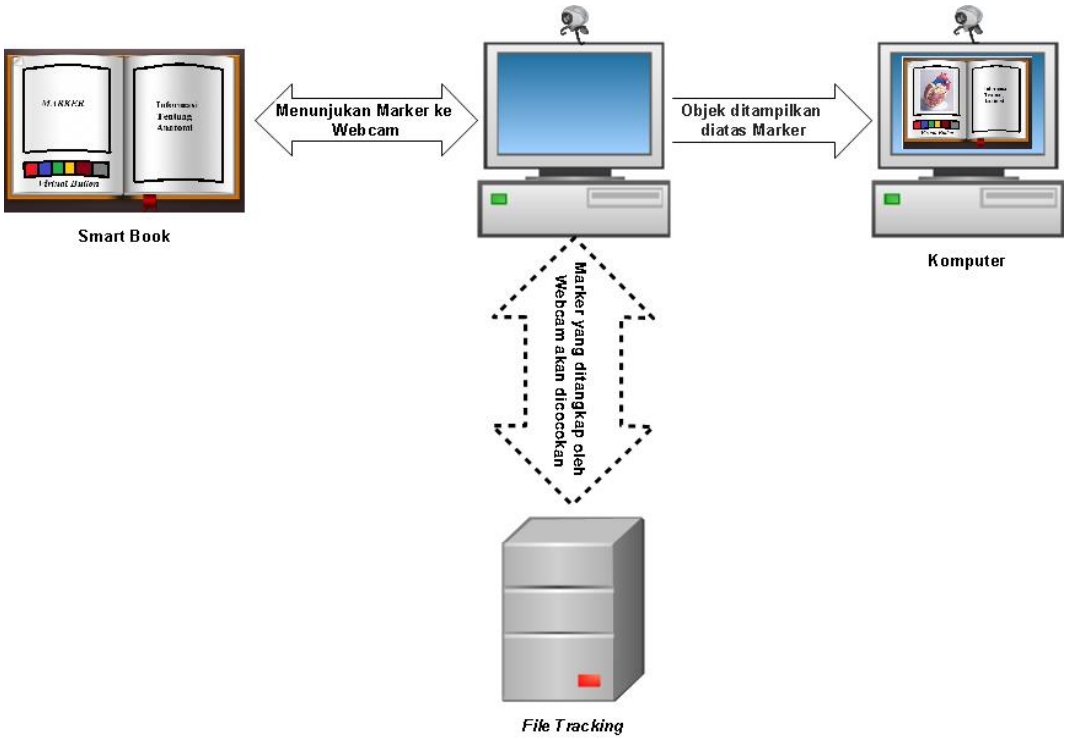

**Gambar 3.1** Alur Sistem yang sedang berjalan

1. Anak-anak adalah objek yang ingin mendapatkan informasi yang ada pada buku.
2. Media buku memberikan informasi dengan menampilkan teks dan gambar dalam bentuk 2D.

Dari gambaran tersebut jelas bahwa media yang didapat untuk mengakses informasi pada buku. Teknologi *Augmented Reality* bisa dijadikan sebagai media alternatif untuk mendapatkan informasi, sehingga konten informasi yang didapat dari buku menjadi lebih menarik dan interaktif.

### 3.1.3 Analisis Arsitektur Sistem

Pada arsitektur aplikasi yang akan dibangun terdiri dari beberapa komponen yaitu ; *user* yang menggunakan aplikasi pengenalan anatomi tubuh manusia, *user* mengarahkan buku yang sudah dilengkapi *marker* (objek AR) ke komputer yang sudah ada *webcam* nya, dari gambar yang di dapat dari *webcam* sistem komputer akan melakukan pelacakan dan pencocokan *marker* guna mengidentifikasi *marker* yang digunakan *user*, kemudian komputer me-*render* objek-objek 3D yang digunakan dalam aplikasi. *User* dapat melihat hasil melalui layar komputer. Gambaran arsitektur sistem dapat dilihat pada gambar 3.2.

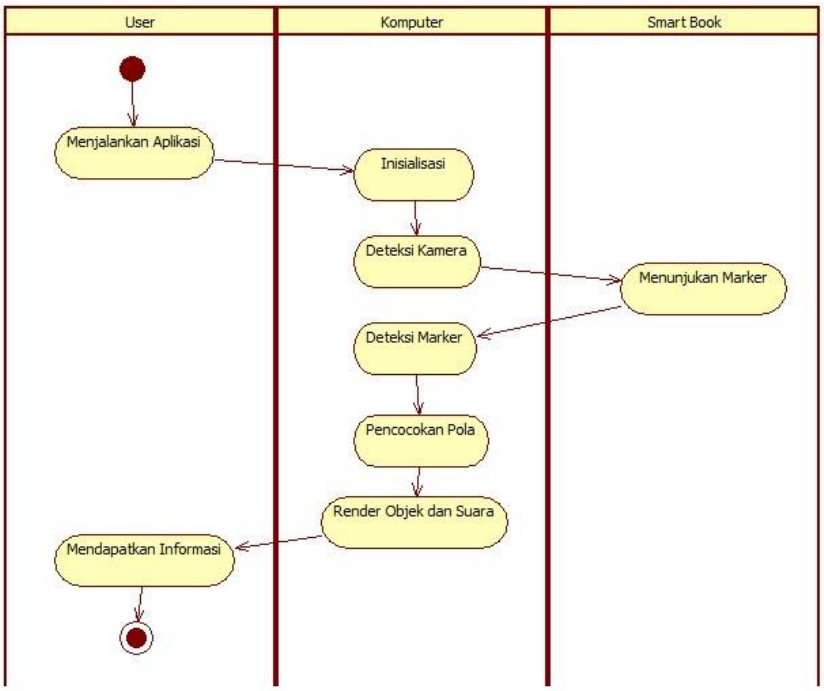



**Gambar 3.2** Arsitektur Sistem *Augmented Reality*

Sistem yang akan dibangun adalah suatu aplikasi berbasis *desktop* dengan menggunakan teknologi *Augmented Reality*. Keunggulan dari teknologi *Augmented Reality* sangat menonjolkan dari segi tampilan yang lebih nyata dan

dijalan kan secara *real-time*. Sehingga dapat diterapkan dalam aplikasi agar dapat menambah nilai dari suatu informasi yang dibutuhkan dan juga dapat menjadi solusi dari permasalahan yang ada.

Proses aplikasi yang akan dibangun dengan menggunakan teknologi *Augmented Reality* dapat dilihat pada gambar 3.3 yang menggambarkan alur sistem yang akan dirancang.

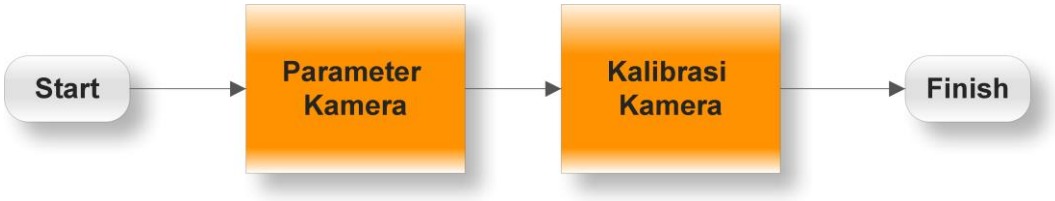



**Gambar 3.3** Alur Sistem

1. *User* adalah pengguna yang akan menggunakan akses pada komputer yang didukung dengan *webcam* dalam mendapatkan informasinya.
2. Pada saat *user* menjalankan aplikasi, aplikasi akan langsung mendeteksi ketersediaan kamera dan secara otomatis untuk mencocokan pola *marker* pada buku serta mensingkronkan *marker* dengan kamera.
3. *Smart Book* merupakan sebuah buku yang didalam nya terdapat *marker* 2D yang memiliki bentuk gambar yang berbeda dan nantinya akan di deteksi oleh kamera *webcam*.
4. Menghasilkan animasi 3D dan suara yang merupakan hasil dari aplikasi yang memiliki teknologi *Augmented Reality*.

### 3.1.4 Analisis Pendeteksian Kamera

Pada tahap ini akan menganalisis bagaimana kamera mendeteksi marker melalui 2 tahapan yaitu parameter kamera dan kalibrasi kamera.




**Gambar 3.4** Proses Pendeteksian Kamera

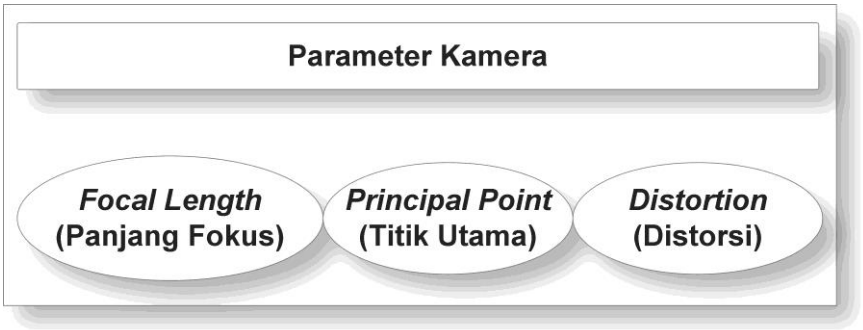
Dalam pendeteksian kamera menggunakan *class Camera* dalam library MetaIO. Didalam *class* tersebut terdapat beberapa *method* yang digunakan untuk mendeteksi kamera yaitu *getCameraList(), vector2di(),* dan *startCamera()*. Definis *class Camera* dapat dilihar pada tabel 3.1.

**Tabel 3.1** Definisi *Class Camera*

| Method | Explanation |
|---|---|
| *getCameraList()* | Mendapatkan semua daftar kamera yang tersedia serta mendefinisikannya |
| *vector2di()* | Mengatur resolusi *output* yang digunakan (default : 640x480) |
| *startCamera()* | Memulai menagkap gambar pada kamera |

#### 3.1.4.1 Parameter Kamera

Pada tahap ini terdiri dari 3 bagian penting yaitu focal length, principal point, dan distorion.

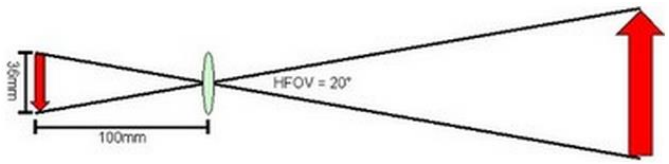



**Gambar 3.5** Proses Parameter Kamera

Setiap kamera didefinisikan oleh seperangkat parameter yang melekat dalam sistem optik. Parameter-parameter ini disebut parameter intrinsik yang meliputi panjang fokus, titik utama dan koefisien distorsi. Dalam konteks *Augmented Reality*, parameter intrinsik penting untuk tujuan deteksi, pelacakan, pengukuran

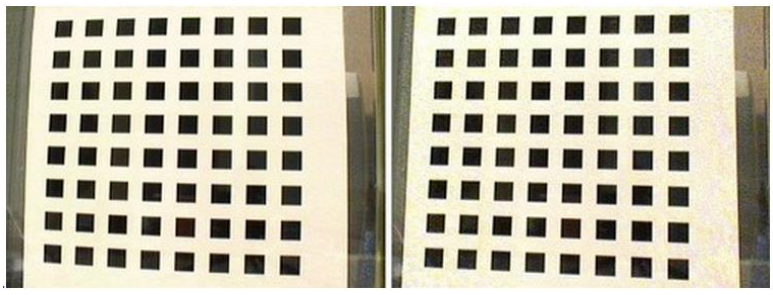
dan lain-lain. Untuk aplikasi sederhana, stabilitas pelacakan tidak menjadi perhatian utama pada nilai-nilai kalibrasi kamera di MetaIO SDK dan dapat diterima. Tapi pada aplikasi industri yang memerlukan pengukuran yang sangat tepat serta pelacakan yang stabil, kalibrasi kamera harus diperhatikan. Oleh karena itu, dianjurkan bahwa kamera disetiap perangkat harus dikalibrasi sebelum menggunakan MetaIO SDK untuk aplikasi pelacakan. Berikut ini analisis tentang parameter intrinsik :

1. Panjang fokus (*Focal Length*) menggambarkan jarak antara pusat optik lensa dan gambar serta menentukan bidang pandang. *TrackingValues* secara langsung tergantung pada itu dan akurasi juga harus cukup. Jika tidak, objek tidak akan bisa di lacak sistem.




**Gambar 3.6** Panjang Fokus

2. Titik utama (*Principal Point*) mendefinisikan pusat proyeksi pada bidang gambar kamera. Pada dasarnya, itu adalah titik dimana sumbu optik mencapai bidang gambar.
3. Parameter distorsi (*Distortion*) menggambarkan distorsi lensa kamera. Dalam gambar kamera terdistorsi, garis lurus didunia nyata tidak muncul lurus pada gambar kamera kecuali dibalik dengan kalibrasi kamera. Seperti pada gambar 3.7, gambar yang sebelah kiri sangat jelas garis distorsi yang bengkok dan gambar sebelah kanan menunjukan seberapa efektif kalibrasi kamera bisa untuk distrorsi reversal.



**Gambar 3.7** Distorsi Kamera

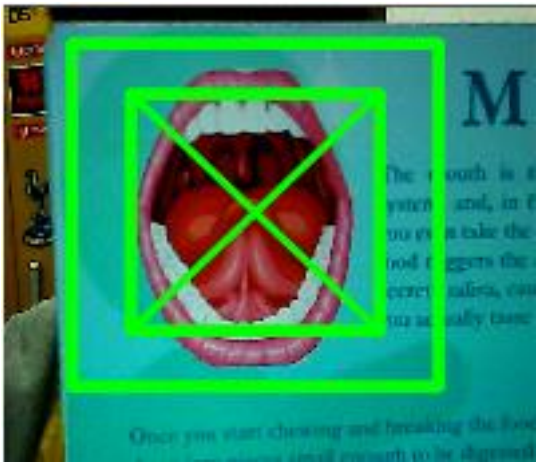
### 3.1.4.2 Kalibrasi Kamera

Pada tahap kalibrasi kamera pertama yang harus dilakukan adalah mengarahkan kamera pada *marker* yang akan dikalibrasi. Pada tahap ini perlahan-lahan arahkan *marker* ke kamera sampai kamera mengenali dan juga melacak *marker* nya. Bila *marker* tersebut diakui akan di tandai dengan kotak target merah seperti pada gambar 3.8.



**Gambar 3.8** Kamera Menandai *Marker* dengan Kotak Target Merah

Ketika kalibrasi selesai kotak target sekitar marker akan beralih warna dari merah ke hijau, lihat gambar 3.9.



**Gambar 3.9** Kalibrasi Selesai

Pada tahap kalibrasi selesai, dalam rangka untuk menguji kesalahan kalibrasi adalah mungkin untuk memindahkan *marker* (digerakan) jauh dari kamera, dan memverifikasi bahwa pelacakan dapat dilakukan dengan sukses.

Kalibrasi kamera yang dihasilkan akan disimpan dalam file XML, dengan *sub tag* yang sesuai parameter kamera, seperti yang ditunjukkan pada gambar 3.10.

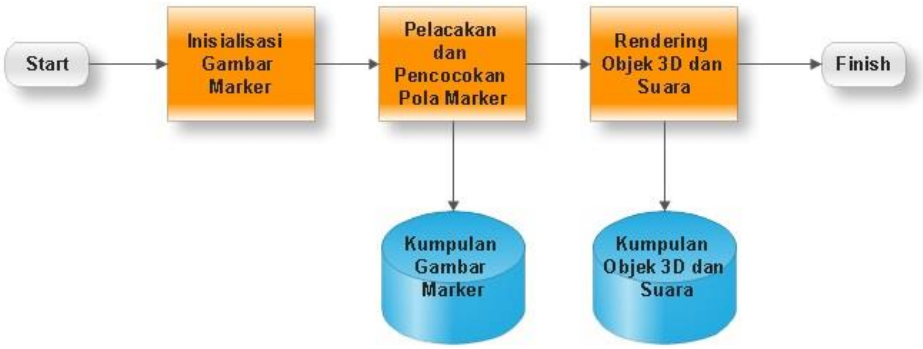
```
<?xml version="1.0"?>
<Camera>
	<Name>Manual Calibration</Name>
	<Info />
	<CalibrationResolution>
		<X>640</X>
		<Y>480</Y>
	</CalibrationResolution>
	<FocalLength>
		<X>577.2347</X>
		<Y>580.0264</Y>
	</FocalLength>
	<PrincipalPoint>
		<X>317.850983</X>
		<Y>240.612549</Y>
	</PrincipalPoint>
	<Distortion>
		<k1>0.0656162351</k1>
		<k2>-0.228799954</k2>
		<p1>0.00270225317</p1>
		<p2>0.0007854185</p2>
	</Distortion>
</Camera>
```

**Gambar 3.10** Proses Kalibrasi Kamera

Berdasarkan gambar 3.10, *CalibrationResolution* merupakan tahap resolusi gambar kamera pada MetaIO dalam piksel yang mana lebar diinisialisasikan dengan X dan panjang diinisialisasikan dengan Y. *FocalLength* merupakan panjang fokus kamera dalam piksel X untuk horizontal dan Y untuk vertical. *PrincipalPoint* adalah titik utama dari kamera dan juga dalam satuan piksel. Serta *Distortion* merupakan koefisiensi distorsi kamera. Metaio SDK menyediakan API untuk mengatur parameter kamera kalibrasi yaitu *setCameraParameters*.

### 3.1.5 Analisis *Tracking Marker*

Analisis *tracking marker* merupakan analisis yang mendiskripsikan bagaimana proses AR pada sistem dapat berjalan, dari tahap inisialisasi gambar, tahap *tracking marker*, sampai tahap me-*render* objek 3D yang telihat pada gambar 3.11.

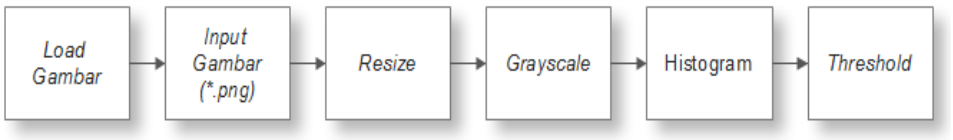



**Gambar 3.11** Alur *Tracking Augmented Reality*

Secara garis besarnya, dalam perancangannya ada tiga bagian utama yaitu :

1. **Inisialisasi Gambar *Marker*.**

   Pada tahap ini ditentukan proses dimana MetaIO SDK bekerja untuk mengambil gambar sebagai *input*-an yang akan diproses. Inisialisasi merupakan tahapan awal yang digunakan sebagai media dalam menangkap gambar. Untuk melakukan penangkapan gambar maka dibutuhkan sebuah *tracking file* dimana dalam penulisan ini file bernama *TrackingAnatomi.xml.*




**Gambar 3.12** Alur Proses Inisialisasi Gambar

Untuk me-*load tracking image* maka dibutuhkan untuk mengaktifkan sensor. Awalnya kamera harus dideteksi dahulu dengan *script* sebagai berikut :

```
bool setCameraParameters(const std::string &calibrationFile)=0
```

Parameter diisi sesuai dengan kebutuhan. Jika sudah memenuhi kualifikasi yang dibutuhkan maka aktifkan sensor untuk *load tracking image*.

```
<SensorCOS>
  <SensorCosID>JANTUNG</SensorCosID>
  <Parameters>
    <ReferenceImage>hearth.png</ReferenceImage>
    <SimilarityThreshold>0.7</SimilarityThreshold>
  </Parameters>
</SensorCOS>

<SensorCOS>
  <SensorCosID>PEREDARAN_DARAH</SensorCosID>
  <Parameters>
    <ReferenceImage>circulatory.png</ReferenceImage>
    <SimilarityThreshold>0.7</SimilarityThreshold>
  </Parameters>
</SensorCOS>
```

**Gambar 3.13** *Load Tracking Image*

Seperti yang terlihat pada gambar 3.13, setiap *ReferenceImage* wajib ditampung dalam sebuah objek *SensorCOS*. Selain *ReferenceImage*, terdapat parameter lain yaitu *SimilarityThreshold* yang merupakan nilai penentu kecocokan antara *ReferenceImage* dengan gambar yang nantinya di-*capture* oleh kamera.

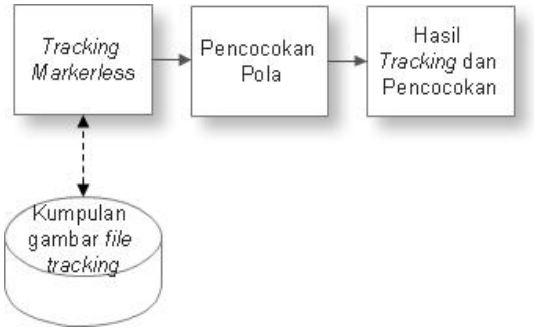
Nilai *SimilarityThreshold* bernilai dari -1 hingga 1 dimana 1 adalah nilai tertinggi yang berarti *ReferenceImage* mutlak sama dengan dengan gambar yang di-*capture*. Sedangkan jika nilai ini diatur sebagai 0.7 dimana ini adalah nilai default yang ditetapkan oleh metaIO, maka nilai ini menunjukkan batas minimum kemiripan. Jika pada suatu kondisi dimana gambar yang di-*capture* oleh kamera memiliki nilai kemiripan dibawah nilai 0.7 maka gambar tersebut dianggap *lost* sehingga sistem akan membaca bahwa proses *tracking* gagal.

*SensorCOS* pada dasarnya adalah sebuah sistem koordinat yang berhubungan dengan gambar template yang harus dilacak. *SensorCosID* merupakan sebuah ID yang dikaitkan dengan COS. *ReferenceImage* sebuah pemanggilan untuk memanggil gambar yang sudah di atur untuk di lacak. *SimilarityThreshold* akan mengesampingkan kesamaan nilai ambang batas standar yang ditentukan untuk sensor.

Setalah melakukan *load* gambar, selanjutnya adalah mempersiapkan gambar yang akan di *imput* kan dengan format gambar yang umum digunakan di MetaIO adalah .png, walaupun bisa juga dengan gambar yang berformat .jpg, .jpeg, .bmp, dll.

Pada tahap selanjutnya yang dilakukan saat menginisialisasi gambar adalah tahap *resize*, *grayscale*, *histogram*, dan *threshold* yang pada MetaIO SDK itu sudah diproses didalamnya serta hanya diperuntukan untuk langsung dipakai saja. Oleh karena itu, MetaIO SDK tersebut tidak bisa di analisis lebih lanjut untuk mengetahui proses didalamnya, dikarenakan keterbatasan yang ada.

**2. Pelacakan dan Pencocokan Pola *Marker*.**




**Gambar 3.14** Alur Proses Pelacakan dan Pencocokan

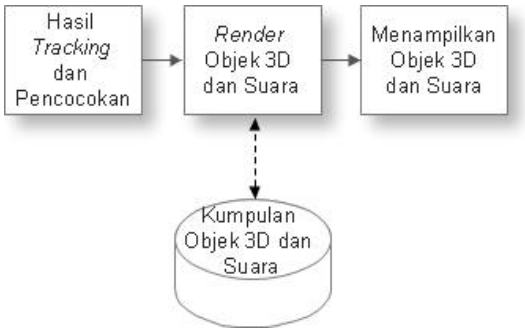
Pelacakan dan pencocokan pola *marker* merupakan proses mendeteksi sebuah *marker* yang kemudian di integrasikan untuk menghasilkan objek 3D. Setelah melakukan konfigurasi pada *tracking file*, selanjutnya memanggil *file* tersebut untuk menyesuaikan dengan hasil gambar yang di*capture* oleh sistem dari gambar cetak.

```
if(!m_pMetaioSDK->setTrackingConfiguration("/Assets/anatomi.xml"))
{
	qCritical("Failed to load tracking configuration");
}
```

Pada saat melakukan *tracking image*, metaIO akan memeriksa file XML yang digunakan sebagai *tracking configuration.* Dalam hal ini, seperti yang tertulis di script tersebut, maka file XML yang digunakan adalah “anatomi.xml” pada folder “Assets”.

Dalam file konfigurasi inilah akan ditentukan beberapa pengaturan untuk objek *image tracker* atau dalam *framework* metaIO disebut sebagai *ReferenceImage*. Untuk menentukan sebuah *ReferenceImage* yang digunakan sebagai *tracker*, maka dibutuhkan sebuah *Sensor* sebagai penampung dari masing-masing *ReferenceImage.*

3. ***Rendering* Objek.**




**Gambar 3.15** Alur Proses Menampilkan Objek

Setelah *tracking* berhasil dilakukan terhadap *marker*, maka sistem akan menempatkan bentuk 3D yang sesuai dengan objek gambar *marker*. Posisi animasi 3D tersebut akan terlihat berada di atas gambar *marker*.

Setelah proses *tracking* berhasil dilakukan maka sistem selanjutnya akan merender objek yang akan dibangun. Pada *framework* metaIO, proses ini akan ditangani oleh class IGeometry dimana setiap objek yang akan di-*render* wajib ditampung sebagai objek IGeometry. Pada smartbook ARnatomi ini, objek akan berupa 3D *Movie Texture* dimana objek 3D yang dirender adalah objek *plane* atau datar yang merupakan objek *default* dari metaIO dan kemudian akan ditambahkan *movie texture* yang menutupi objek 3D *plane* tersebut. Dalam metaIO, proses ini akan dikerjakan oleh method *createGeometryFromMovie*.

```
metaio::IGeometry* geometry = m_pMetaioSDK-> createGeometryFromMovie
("1_AR.3g2",false,true);

geometry->setScale(new Vector3d(3,3,3));
```

Seperti yang tertulis pada *script* tersebut, terdapat 3 parameter untuk masukan yang dibutuhkan oleh *method* yaitu *filepath*, *transparentMovie*, *displayAsBillboard*. *filepath* merupakan alamat dari objek *movie texture* yang akan ditampilkan. Dalam metaIO hanya file dengan ekstensi .3g2 yang didukung oleh *framework* ini dengan spesifikasi sebagai berikut.

1) Kompresi Video : MPEG4 codec inside 3G2 container.
2) Resolusi Video : Disarankan 176x144px @ 20fps (e.g. 288kbps) perangkat yang kuat (Dual Core) dapat menangani resolusi yang lebih tinggi.
3) Kompresi Audio : AAC LC
4) Resolusi Audio : 22050kHz Stereo (e.g. 48kbps)
5) Perbandingan Aspek : Geometri yang disediakan harus memiliki aspek yang sama sebagai pengkodean film.

Parameter yang kedua adalah *transparentMovie*. Nilai ini akan mengatur apakah *movie texture* yang ditampilkan akan menggunakan "alpha-plane" sehingga *movie texture* terlihat "tembus pandang". Parameter terakhir adalah *displayAsBillboard* dimana nilai ini menentukan apakah objek 3D plane yang ditampilkan akan diperlakukan sebagai objek 2D yang selalu menghadap kamera. Dengan memberi nilai true, maka objek 3D yang di-*render* akan selalu menghadap kamera meskipun objek buku tidak tegak lurus terhadap kamera.

Dari uraian tersebut, dapat terlihat bahwa untuk me-*render* sebuah objek 3D, maka akan dibangun sebuah *object IGeometry* yang merupakan bawaan dari *library MetaIO*. *Class* inilah yang digunakan untuk menampung objek 3D dalam format .3g2 yang telah di-*render*. Jika hasil *capture image* dan *tracking image* sesuai maka gambar akan di *load* ke layar pengguna. *Method setScale* yang ditunjukkan sebelumnya adalah pengaturan untuk menentukan besarnya objek 3D dengan 3 parameter untuk sumbu X,Y dan Z.




**Gambar 3.16** Hasil dari *Render* Objek 3D

### 3.1.6 Analisis *Marker (Smart Book)*

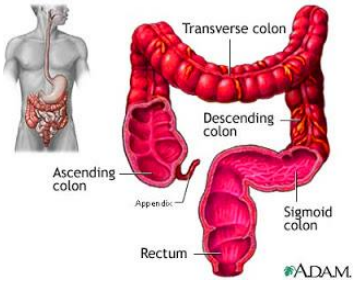
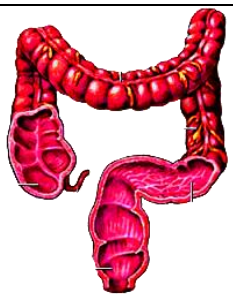
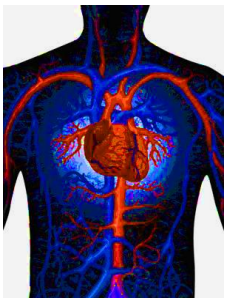
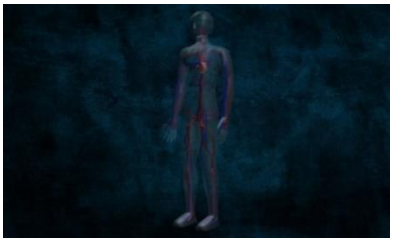
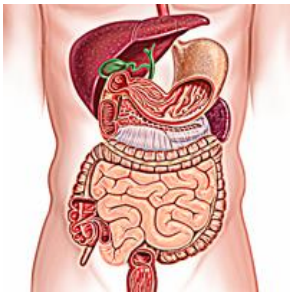
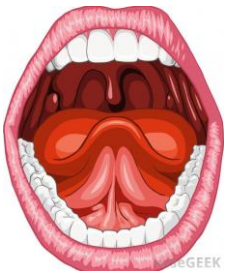
Pada tahapan ini merupakan tahap yang menguraikan tentang *marker* dan tahap ini juga akan menganalisa bentuk pada *marker* yang digunakan serta *marker* apa saja yang digunakan.

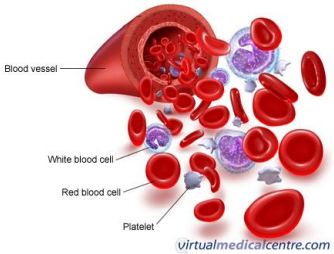
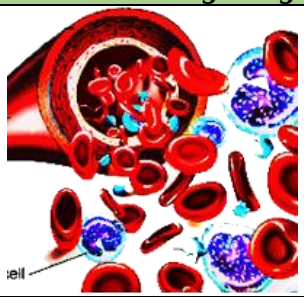
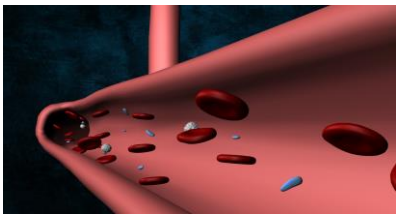
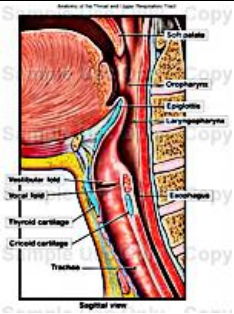
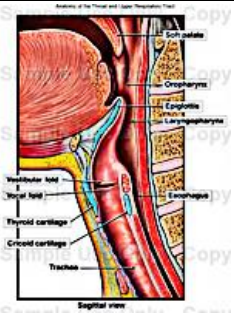
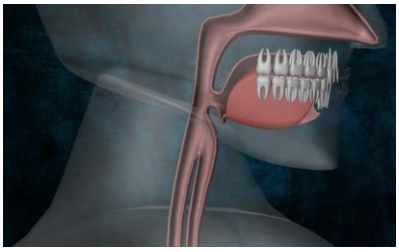
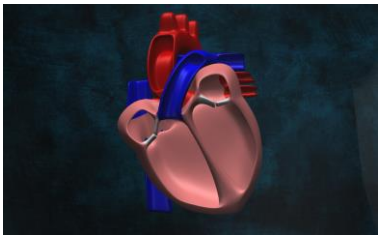
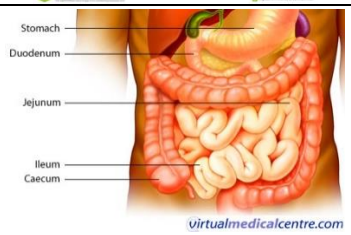
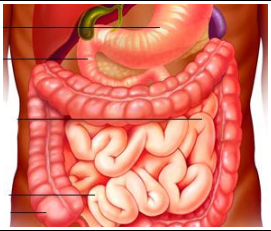
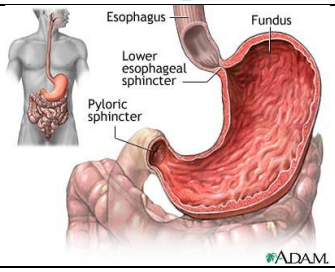
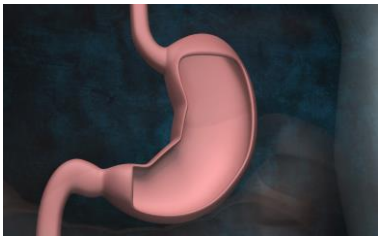
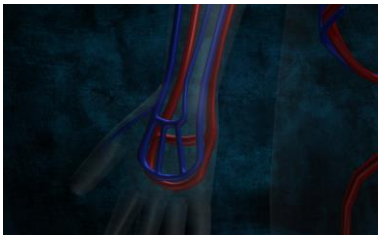
*Markerless* merupakan salah satu teknologi *tracking* yang disediakan oleh MetaIO. *Tracking* jenis ini menggunakan gambar sebagai *marker*-nya. Proses *tracking* jenis ini menggunakan tekstur gambar (dua dimensi) yang disimpan dalam *library* sebagai sumber referensinya, dan membandingkan tekstur yang ditangkap oleh kamera dengan tekstur gambar yang ada di *library marker*-nya.

#### 3.1.6.1 *Marker* yang digunakan

Pada tahap ini menggunakan metode *markerless 2D*, posisi *marker* digantikan dengan pengenalan gambar yang selanjutnya akan di *tracking* dan menjadi *threshold value* dalam menampilkan objek 3D yang sesuai. Dalam penggunaannya dengan menggunakan *Library MetaIO*, maka digunakan *tracking file* yang berbasis *XML* untuk melacak gambar yang di *capture* untuk menyesuaikan objeck 3D yang akan dimunculkan.

**Tabel 3.2** Jenis *Marker*

| ***File Image* Cetak** | ***File Tracking Image*** | **Objek 3D Movie** |
|---|---|---|
|  |  |  |
|   |  |  |
|  |  |  |
|  |  |  |
|  |   |  |

| *File Image* Cetak | *File Tracking Image* | Objek 3D Movie |
|---|---|---|
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Seperti yang terlihat pada tabel 3.2, *tracking image* tidak seluruh gambar yang ditampilkan dalam *Smart Book* namun hanya dengan dimensi 200x200 (*width* 200 px dan *height* 200 px). Hal ini karena sistem dari MetaIO tidak membutuhkan keseluruhan gambar, namun hanya sebagian gambar saja yang memiliki spesifikasi tinggi sehingga memudahkan sistem dalam melakukan pelacakan atau *tracking*.

Untuk mendapatkan hasil yang stabil pada setiap *tracking image*, maka dibutuhkan penanganan khusus agar gambar tersebut dapat di *tracking* dengan sempurna oleh sistem *Augmented Reality* yang dibangun. Merujuk pada *website* resmi *MetaIO*.

### 3.1.7 Analisis Materi

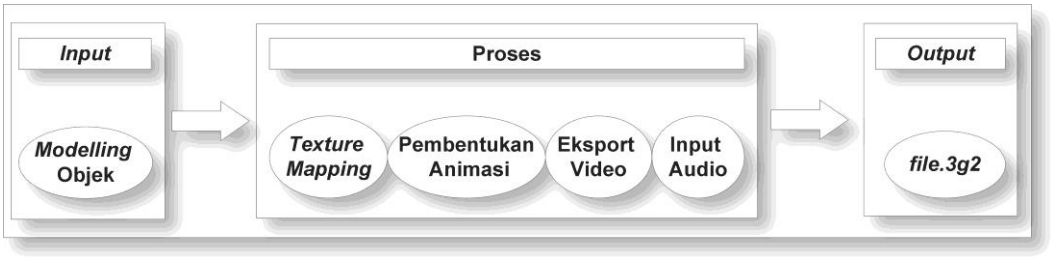
Pada tahap ini akan menguraikan acuan dari materi yang akan diberikan dalam *Smart Book.* Materi pembelajaran di SD tersebut berupa buku yang berjudul "*My Pals Are Here!*". Dimana buku ini tidak mengacu pada kurikulum sistem pendidikan yang ada di Indonesia atau standar materi pembelajaran dari Kementerian Pendidikan Indonesia. Tapi mengacu kepada sistem pembelajaran *International.* Gambar 3.17 dibawah ini menjelaskan materi apa saja yang di ambil dari buku tersebut untuk di masukan ke dalam *Smart Book.*




**Gambar 3.17** Materi yang digunakan *Smart Book*

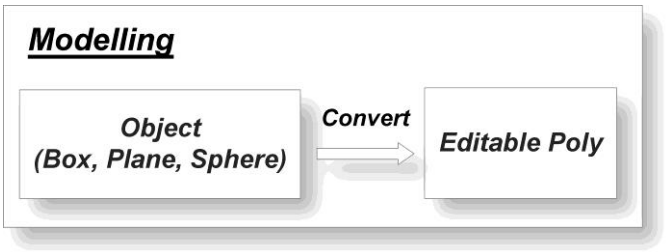
### 3.1.8 Analisis Pembentukan Objek dan Suara

Pada tahap ini akan menguraikan bagaimana cara pembentukan objek, penambahan tekstur, ekspor ke video, penginputan audio hingga konversi ke format yang sesuai pada lib MetaIO yaitu .3G2.




**Gambar 3.18** Proses Pembentukan Objek

### 3.1.8.1 Proses *Modelling* Objek

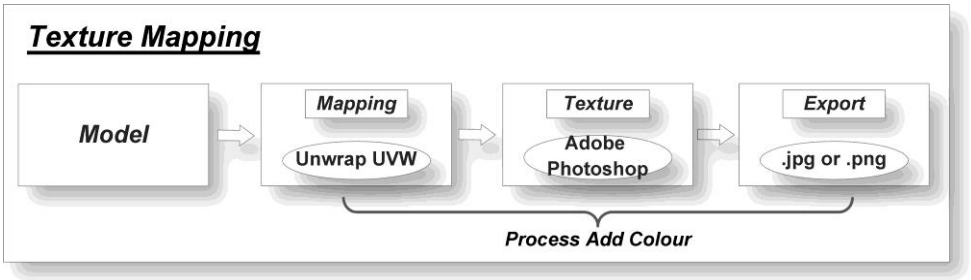



**Gambar 3.19** Proses *Modelling*

Proses *modeling* objek menggunakan software 3D Studio Max 2010. Langkah awal membentuk objek adalah dengan membuat objek bentuk dasar di 3DS max (*box, plane, sphere*) kemudian mengkonversinya menjadi bentuk *Editable Poly*, bentuk ini memungkinkan penambahan segmen, maupun *vertex* pada objek sekaligus memungkinkan pemindahan *vertex* sehingga dapat dibentuk menjadi objek yang diinginkan. *Modelling* dilakukan dengan referensi bentuk objek sebenarnya agar menghasilkan bentuk yang sesuai.

### 3.1.8.2 Proses *Texture Mapping*

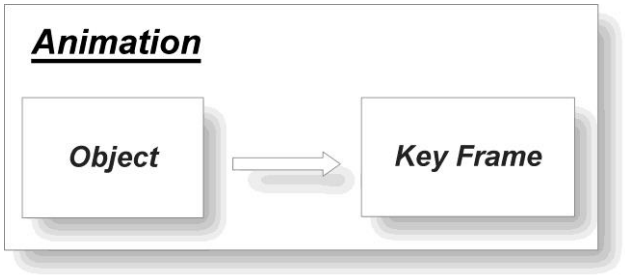
Setelah proses *modelling* selesai, selanjutnya adalah proses *texture mapping* yaitu proses pemasangan warna, gambar, ataupun tekstur pada permukaan objek yang telah jadi. Proses *mapping* dimulai dengan memetakan bentuk objek menjadi bentuk jaring-jaring dengan menggunakan *modifier Unwrap UVW*.




**Gambar 3.20** Proses *Texture Mapping*

Bentuk jaring-jaring yang dihasilkan akan dijadikan sebagai acuan penambahan teksture yang dilakukan di software Adobe Photoshop. Lalu mengatur tekstur pada Adobe Photoshop. Setelah tekstur dari Adobe Photoshop telah terekspor menjadi berkas .jpeg atau .png langkah selanjutnya adalah memasang pada objek 3D di software 3DS max dengan membuka menu material editor dan memasangnya pada slot untuk tiap-tiap objek.

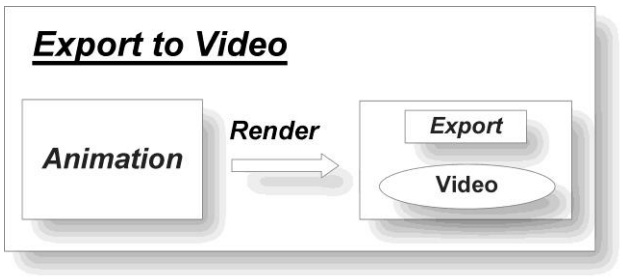
#### 3.1.8.3 Proses *Animation*




**Gambar 3.21** Proses *Animation*

Setelah objek 3D dianggap siap tahap selanjutnya adalah proses animasi. Proses ini pada prinsipnya menetapkan tiap pose kunci tiap gerakan (*keyframe*) sesuai koordinat pada objek secara manual dalam *timeline* yang nantinya software akan mengisi gerakan *inbetween* secara otomatis menyesuaikan dengan perubahan koordinat antar *keyframe*.

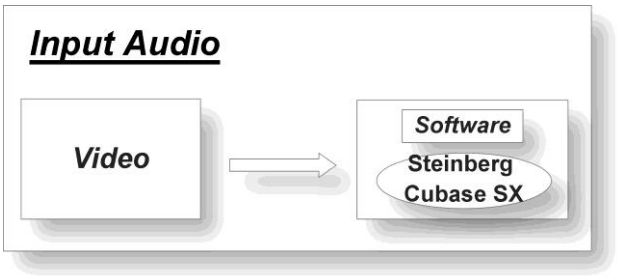
#### 3.1.8.4 Proses *Export to Video*




**Gambar 3.22** Pengaturan *Render*

Setelah animasi selesai tahap selanjutnya adalah me*render* file mentah 3DS max menjadi file video. Setting untuk render video menggunakan resolusi 176x144 pixel sebagai resolusi yang mampu digunakan pada software MetaIO.

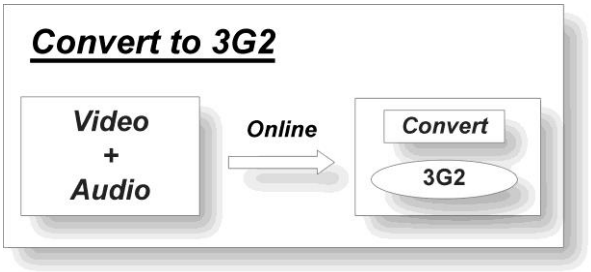
### 3.1.8.5 Proses *Input Audio*




**Gambar 3.23** Proses *Editing Audio*

Video yang dihasilkan kemudian ditambah dengan audio yang berisi narasi sebagai keterangan dari visualisasi dalam video. Proses penambahan audio ini menggunakan software *Steinberg Cubase SX*.

### 3.1.8.6 Proses *Convert to 3G2*




**Gambar 3.24** Proses Konversi 3G2

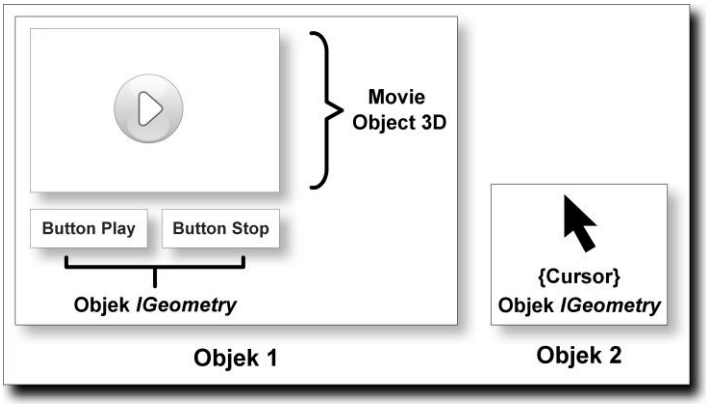
Setelah file video dan audio menjadi satu kemudian dilakukan proses pengkonversian file menjadi file dengan format 3G2 sesuai dengan ketentuan pengaturan yang dapat didukung oleh software MetaIO. Pengkonversian dilakukan menggunakan bantuan konversi file video secara online.

### 3.1.9 Analisis Metode *Occlusion Based*

*Occlusion* adalah hubungan antara suatu benda dengan benda lain jika kita lihat dari suatu sudut pandang. Pada tahap analisis ini akan menjelaskan ada tidaknya *occlusion* dalam menampilkan objek 3D. Secara sederhananya *occlusion* mendefinisikan dimana suatu objek *marker* tidak dapat terdeteksi dikarenakan terhalang oleh objek lain.

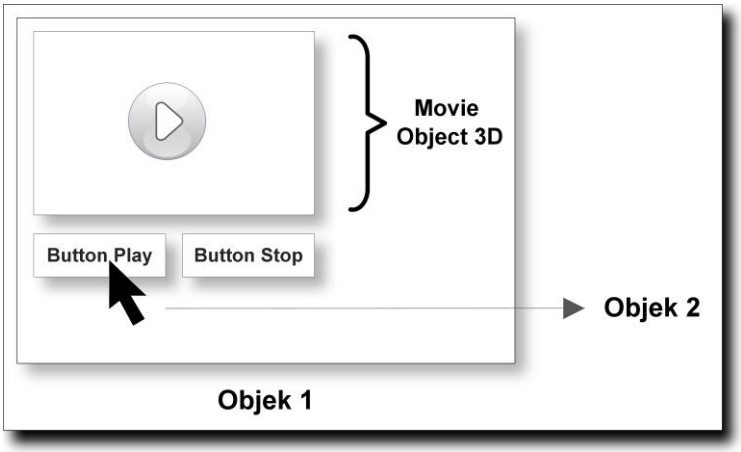
Namun pada MetaIO, *occlusion* ini diterapkan pada persinggungan dua objek, objek digital (*virtual button*) dan objek nyata (*cursor/touch*). Kedua objek tersebut telah ada objek IGeometry sebelumnya. *Object IGeometry* yang merupakan

bawaan dari *library MetaIO* digunakan untuk menampung objek 3D dalam format .3g2 yang akan di-*render*



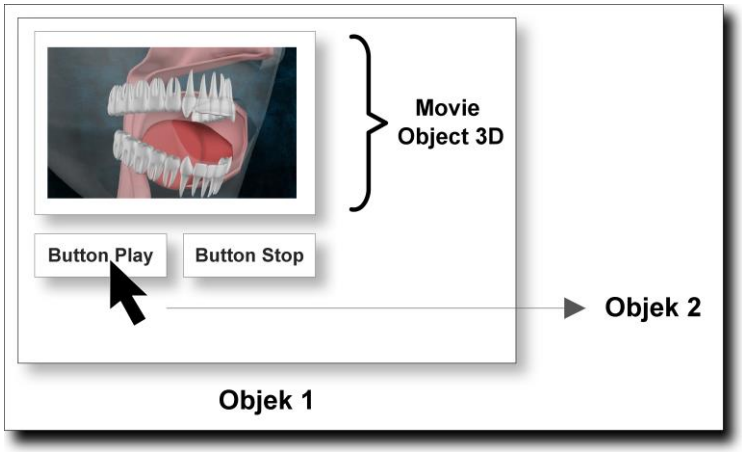

**Gambar 3.25** Kondisi Awal Sebelum Proses *Occlusion* Pada MetaIO

Sebagai contoh, ketika sebuah *image marker* di-*tracking*, maka akan tampil satu objek digital sebagai model 3D dengan objek IGeometry yang pada awalnya telah ada. Lalu objek kedua (*cursor/touch*) juga telah memiliki IGeometry sendiri untuk mengecek apakah objek yang di klik atau di sentuh nya mempunyai IGeometry yang sama. Jika objek IGeometry sama akan terjadi suatu *event*, *event* tersebut akan mengaktifkan atau menjalankan *movie* animasi tersebut.




**Gambar 3.26** Memulai Proses Persinggungan Antar Objek IGeometry

Dalam hal ini, objek *virtual button* yang dibangun akan tertutup oleh objek IGeometry hasil *input* pengguna melaui *cursor mouse*. Pada gambar 3.27 menunjukkan bagaimana cursor mouse telah menutupi objek lain. Kemudian mengaktifkan suatu *event*, *event* tersebut akan memulai menjalankan *movie* animasi.




**Gambar 3.27** Terjadi Proses *Occlusion* dan Mengaktifkan *Event*

Dalam *tool metaIO* nya metode *occlusion based interaction* dapat dilihat pada uraian dibawah ini.

```
1 ---- >        occlusionObject        =        m_pMetaioSDK-
>getGeometryFromScreenCoordinates(x,y,TRUE);

2 ---- > if(occlusionObject==playArfi){
3 ---- >    m_arfi->startMovieTexture(true);
4 ---- > }else if(occlusionObject==stopArfi){
```

Script pada baris ke-1 berguna untuk menentukan / menginisialisasi teksture yang digunakan untuk menyelimuti object *occlusionObject*. Berbeda dengan cara yang digunakan pada button play / stop yang memberikan *texture* dengan *createGeometryFromImage*, *occlusionObject* memberikan tekture dengan method *getGeometryFromScreenCoordinate* yang ditanamkan pada method *CMyWindow::OnLButtonDown*

Hal ini menunjukkan dimana object *occlusionObject* hanya akan dibuat ketika tombol kiri mouse / touchpad di klik. Selanjutnya ketika layar diklik kiri oleh pengguna, maka *coordinate* tersebut akan diisikan pada parameter “x” dan “y” yang dibutuhkan oleh method *getGeometryFromScreenCoordinates*. Jadi intinya, ketika mouse di klik kiri, maka system akan mencatat koordinat x dan y dari lokasi di klik tersebut. Selanjutnya system akan mengecek apakah yang di klik adalah area kosong atau sebuah geometry. Jika sebuah geometry, maka texture geometry tersebut akan digunakan sebagak texture dari object geometry *occlusionObject*.

## 3.2 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non fungsional merupakan analisis yang dibutuhkan untuk menentukan spesifikasi kebutuhan sistem. Spesifikasi ini juga meliputi elemen ataupun komponen apa saja yang dibutuhkan sistem yang akan dibangun sampai sistem tersebut diimplementasikan. Pada analisis kebutuhan non fungsional ini dijelaskan mengenai analisis perangkat keras (*hardware*), perangkat lunak (*software*), dan pengguna (*user*).

### 3.2.1 Analisis Kebutuhan Perangkat Keras

Komputer adalah sebuah perangkat keras yang tidak luput dari perangkat lunak sebagai interaksinya. Perangkat lunak memberikan perintah agar perangkat keras berfungsi dengan baik. Dalam pembangunan aplikasi menggunakan perangkat keras sebagai pendukungnya sebagai berikut :

1) *Processor Intel Core i5* 2.30GHz
2) RAM 6 GB
3) *Harddisk* dengan kapasitas 1TB
4) Sistem Operasi *Windows 8.1*
5) *Motherboard* dengan *chipset* yang kompatibel dengan VGA *card* yang dipakai
6) VGA *card* dengan kemampuan grafis 3D, seperti NVIDIA GeForce dan ATI Radeon
7) *Webcam* 1.3 *Mega Pixel*
8) *Speaker*

Sedangkan spesifikasi minimum perangkat keras yang dibutuhkan oleh pihak pengguna untuk teknologi AR seperti berkut ini :

1) *Processor Dual Core* minimal 1.7GHz (*Quad Core* disarankan)
2) RAM minimal 2 GB
3) *Harddisk* dengan kapasitas minimal 50 GB
4) Sistem Operasi *Windows 7*
5) *Motherboard* dengan *chipset* yang kompatibel dengan VGA *card* yang dipakai

6) VGA *card* dengan kemampuan grafis 3D, seperti NVIDIA GeForce dan ATI Radeon
7) *Webcam* minimal 2 *Mega Pixel*
8) *Speaker*

### 3.2.2 Analisis Kebutuhan Perangkat Lunak

Perangkat lunak digunakan dalam sebuah sistem merupakan kumpulan perintah-perintah yang diberikan kepada perangkat keras agar saling berinteraki untuk melakukan suatu tugas. Perangkat lunak yang dibutuhkan oleh pengembang tersebut adalah sebagai berikut :

1) Sistem Operasi Windows 8.1
2) Visual Studio 2010
3) 3D Studio Max
4) Astah Profesional
5) Adobe Photoshop
6) ConceptDraw Pro
7) MetaIO SDK

Sedangkan perangkat lunak yang dibutuhkan oleh pihak pengguna untuk aplikasi ini adalah sebagai berikut :

1) Sistem Operasi Windows 7
2) MetaIO SDK

### 3.2.3 Analisis Pengguna

Analisis pengguna dimaksudkan untuk mengetahui siapa saja pengguna yang terlibat dengan sistem. Pengguna yaitu orang yang dapat mengerti dan memahami komputer sehingga dapat menggunakan aplikasi yang akan dibangun. Disini sebagai pengguna adalah anak kelas 5 SD yang tentunya akan didampingi oleh guru yang mengajar. Hak akses yang diberikan hanya melihat animasi 3D dan suara yang dihasilkan oleh aplikasi serta dapat berinteraksi dengan objek.

**Tabel 3.3** Karakteristik Pengguna

| User | Hak Akses | Tingkat Pendidikan | Pengalaman |
|---|---|---|---|
| Siswa Kelas 5 SD | Melihat dan mendapatkan Informasi | Kelas 5 SD | Minimal dapat menjalankan komputer. |

## 3.3 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional menggambarkan proses kegiatan yang akan diterapkan dalam sebuah sistem dan menjelaskan kebutuhan yang diperlukan sistem agar sistem dapat berjalan dengan baik serta sesuai dengan kebutuhan. Kebutuhan fungsional juga berhubungan dengan fitur perangkat lunak yang akan dibuat. Analisis akan dimodelkan dengan menggunakan UML (*Unified Modelling Language*). Tahap-tahap pemodelannya antara lain identifikasi *actor, use case diagram,* skenario *use case diagram*, *activity diagram*, *class diagram, sequence diagram*.

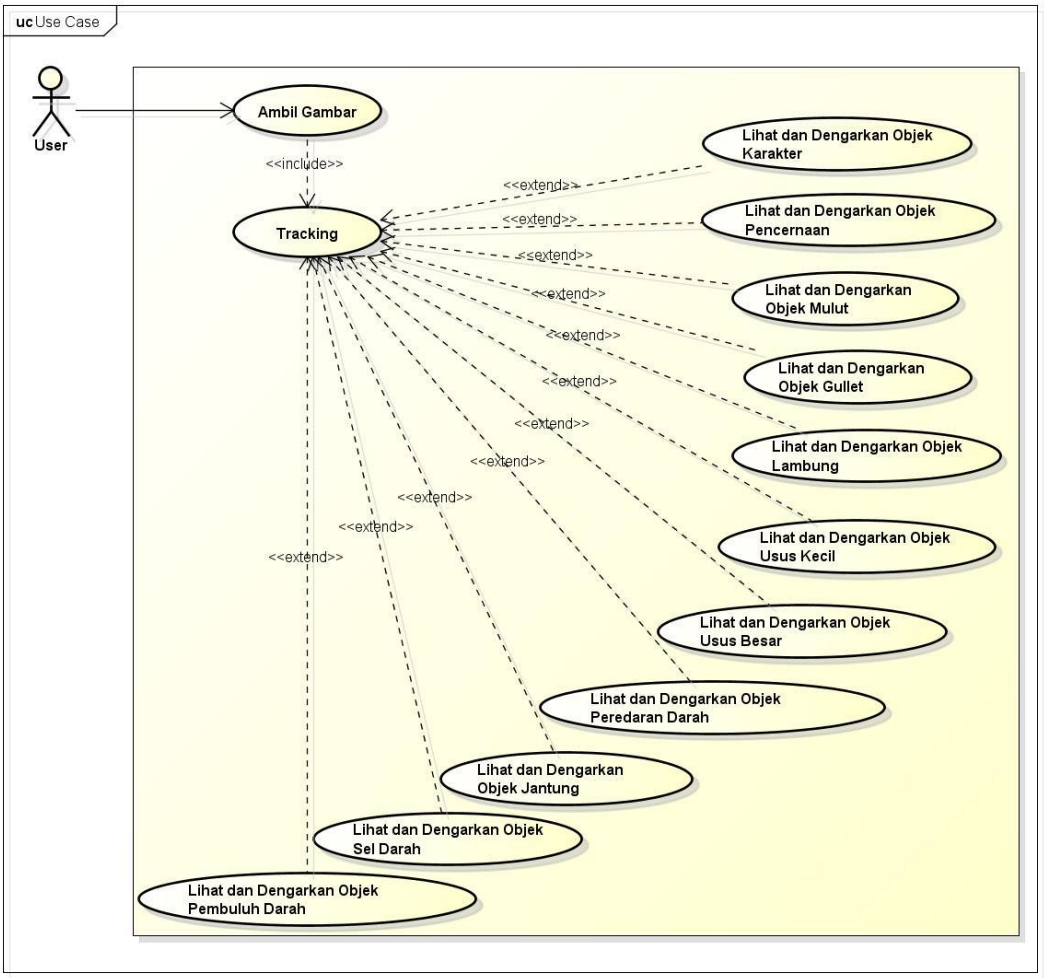
### 3.3.1 *Use Case Diagram*

*Use Case Diagram* merupakan pemodelan untuk kelakuan (*behavior*) sistem yang akan dibuat. *Use Case* mendekripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Pada *use case* hanya ini terdapat 1 aktor yaitu, *user*. Penjelasannya lihat pada tabel 3.4.

**Tabel 3.4** Aktor dalam *Use Case*

| Nama Aktor | Definisi |
|---|---|
| User | Orang yang berinteraksi dengan sistem dan dapat mengakses semua fungsi yang ada pada sistem. |

Dari hasil analisa aplikasi yang ada, maka *use case diagram* untuk aplikasi ini dapat dilihat pada gambar 3.28.




**Gambar 3.28** *Use Case Diagram*

### 3.3.2 Definisi *Use Case Diagram*

Bagian ini menjelaskan definisi dari masing-masing dari use case diagram yang dilakukan oleh pengguna. Mulai dari proses memilih masuk kedalam aplikasi sampai dengan objek ditampilkan.

**Tabel 3.5** Definisi *Use Case Diagram*

| No | Kode *Use Case* | Nama *Use Case* | Keterangan |
|---|---|---|---|
| 1 | UC-AR01 | Ambil Gambar | Fungsional untuk mengambil gambar menggunakan kamera *webcam*. |
| 2 | UC-AR02 | *Tracking* | Fungsional untuk membandingkan gambar tangkap dengan gambar yang ada pada *file tracking*. |

| No | Kode *Use Case* | Nama *Use Case* | Keterangan |
|---|---|---|---|
| 3 | UC-AR03 | Lihat dan Dengarkan Objek Karakter | Fungsional untuk menampilkan informasi objek karakter yang akan menjelaskan semua isi didalam aplikasi. |
| 4 | UC-AR04 | Lihat dan Dengarkan Objek Pencernaan | Fungsional untuk menampilkan informasi tentang objek pencernaan. |
| 5 | UC-AR05 | Lihat dan Dengarkan Objek Mulut | Fungsional untuk menampilkan informasi tentang objek mulut. |
| 6 | UC-AR06 | Lihat dan Dengarkan Objek *Gullet* | Fungsional untuk menampilkan informasi tentang objek *gullet*. |
| 7 | UC-AR07 | Lihat dan Dengarkan Objek Lambung | Fungsional untuk menampilkan informasi tentang objek lambung. |
| 8 | UC-AR08 | Lihat dan Dengarkan Objek Usus Kecil | Fungsional untuk menampilkan informasi tentang objek usus kecil. |
| 9 | UC-AR09 | Lihat dan Dengarkan Objek Usus Besar | Fungsional untuk menampilkan informasi tentang objek usus besar. |
| 10 | UC-AR10 | Lihat dan Dengarkan Objek Peredaran Darah | Fungsional untuk menampilkan informasi tentang objek peredaran darah. |
| 11 | UC-AR11 | Lihat dan Dengarkan Objek Jantung | Fungsional untuk menampilkan informasi tentang objek jantung. |
| 12 | UC-AR12 | Lihat dan Dengarkan Objek Sel Darah | Fungsional untuk menampilkan informasi tentang objek sel darah. |
| 13 | UC-AR13 | Lihat dan Dengarkan Objek Pembuluh Darah | Fungsional untuk menampilkan informasi tentang objek pembuluh darah. |

### 3.3.3 Skenario *Use Case Diagram*

Skenario *use case* mendiskripsikan urutan dari tahapan-tahapan dalam proses bisnis, baik yang dilakukan aktor terhadap sistem maupun yang dilakukan sistem terhadap aktor. Skenario ambil gambar terkait dengan use case ambil gambar dapat dilihat pada tabel 3.6.

**Tabel 3.6** Skenario *Use Case* Ambil Gambar

| **Identifikasi** | |
|---|---|
| **Nama Use Case** | Ambil Gambar |
| **Kode** | UC-AR01 |
| **Aktor** | *User* |
| **Tujuan** | Menjalankan aplikasi dan mendeteksi keberadaan kamera serta menjalankan kamera secara *real-time* . |
| **Kondisi Awal** | Sistem menampilkan gambar yang akan di *tracking*. |
| **Skenario Utama** | |
| **Aksi Aktor** | **Reaksi Sistem** |
| 1. *User* membuka aplikasi *ARnatomi*. | |
| | 2. Mendeteksi keberadaan perangkat kamera. |
| | 3. Menjalankan kamera secara real-time agar bisa melakukan proses *tracking*. |
| **Skenario Gagal** | |
| **Aksi Aktor** | **Reaksi Sistem** |
| | 1. "Layar Kosong". |
| **Kondisi Akhir** | Aplikasi menampilkan objek yang sedang disorot kamera *webcam*. |

Skenario *tracking* terkait dengan *use case tracking* dapat dilihat pada tabel 3.7.

**Tabel 3.7** Skenario *Use Case Tracking*

| **Identifikasi** | |
|---|---|
| **Nama Use Case** | *Tracking* |
| **Kode** | UC-AR02 |
| **Aktor** | *User* |
| **Tujuan** | Mencocokan *marker* dengan data yang ada pada *file tracking*. |
| **Kondisi Awal** | Sistem menampilkan gambar marker pada aplikasi di layar komputer. |
| **Skenario Utama** | |
| **Aksi Aktor** | **Reaksi Sistem** |
| 1. *User* menunjukan *marker* (*Smart Book*) pada kamera. | |
| | 2. Melakukan pencocokan pola *marker* dengan *file tracking*. |
| **Skenario Gagal** | |
| **Aksi Aktor** | **Reaksi Sistem** |
| | 1. "Layar Kosong". |

| | |
|---|---|
| **Kondisi Akhir** | Sistem menampilkan gambar yang akan di *tracking*. |

Skenario objek karakter terkait dengan *use case* lihat dan dengarkan objek karakter dapat dilihat pada tabel 3.8.

**Tabel 3.8** Skenario *Use Case* Lihat dan Dengarkan Objek Karakter

| **Identifikasi** | |
|---|---|
| **Nama Use Case** | Lihat dan Dengarkan Objek Karakter |
| **Kode** | UC-AR03 |
| **Aktor** | *User* |
| **Tujuan** | Menampilkan objek dan menjelaskan aplikasi serta materi. |
| **Skenario Utama** | |
| **Aksi Aktor** | **Reaksi Sistem** |
| 1. Menjalankan aplikasi dan menyorotkan kamera *webcam* kearah *marker* (*Smart Book*). | |
| | 2. Melakukan pencocokan pola *marker* dengan *file tracking*.<br>3. Pemilihan objek 3D yang akan di *render*. |
| **Kondisi Akhir** | Aplikasi menampilkan objek karakter dan suara tersebut secara *real time*. |
| **Skenario Gagal** | |
| **Aksi Aktor** | **Reaksi Sistem** |
| | 1. "Layar Kosong". |
| **Kondisi Akhir** | Aplikasi tetap menampilkan gambar yang sedang di *tracking*. |

Skenario objek pencernaan terkait dengan *use case* lihat dan dengarkan objek pencernaan dapat dilihat pada tabel 3.9.

**Tabel 3.9** Skenario *Use Case* Lihat dan Dengarkan Objek Pencernaan

| **Identifikasi** | |
|---|---|
| **Nama Use Case** | Tampilkan Objek Pencernaan |
| **Kode** | UC-AR04 |
| **Aktor** | *User* |
| **Tujuan** | Menampilkan objek dan mendengarkan penjelasan materi. |
| **Skenario Utama** | |
| **Aksi Aktor** | **Reaksi Sistem** |
| 1. Menjalankan aplikasi dan menyorotkan kamera *webcam* kearah *marker* (*Smart Book*). | |
| | 2. Melakukan pencocokan pola *marker* dengan *file tracking*. |

| | |
|---|---|
| | 3. Pemilihan objek 3D yang akan di *render*. |
| **Kondisi Akhir** | Aplikasi menampilkan objek mulut secara *real time*. |
| **Skenario Gagal** | |
| **Aksi Aktor** | **Reaksi Sistem** |
| | 1. "Layar Kosong". |
| **Kondisi Akhir** | Aplikasi tetap menampilkan gambar yang sedang di *tracking*. |

Skenario objek mulut terkait dengan *use case* lihat dan dengarkan objek mulut dapat dilihat pada tabel 3.10.

**Tabel 3.10** Skenario *Use Case* Lihat dan Dengarkan Objek Mulut

| **Identifikasi** | |
|---|---|
| **Nama Use Case** | Lihat dan Dengarkan Objek Mulut |
| **Kode** | UC-AR05 |
| **Aktor** | *User* |
| **Tujuan** | Menampilkan objek dan mendengarkan penjelasan materi. |
| **Skenario Utama** | |
| **Aksi Aktor** | **Reaksi Sistem** |
| 1. Menjalankan aplikasi dan menyorotkan kamera *webcam* kearah *marker* (*Smart Book*). | |
| | 2. Melakukan pencocokan pola *marker* dengan *file tracking*.<br>3. Pemilihan objek 3D yang akan di *render*. |
| **Kondisi Akhir** | Aplikasi menampilkan objek mulut secara *real time*. |
| **Skenario Gagal** | |
| **Aksi Aktor** | **Reaksi Sistem** |
| | 1. "Layar Kosong". |
| **Kondisi Akhir** | Aplikasi tetap menampilkan gambar yang sedang di *tracking*. |

Skenario objek *gullet* terkait dengan *use case* lihat dan dengarkan objek *gullet* dapat dilihat pada tabel 3.11.

**Tabel 3.11** Skenario *Use Case* Lihat dan Dengarkan Objek *Gullet*

| **Identifikasi** | |
|---|---|
| **Nama Use Case** | Lihat dan Dengarkan Objek *Gullet* |
| **Kode** | UC-AR06 |
| **Aktor** | *User* |
| **Tujuan** | Menampilkan objek dan mendengarkan penjelasan materi. |
| **Skenario Utama** | |

| **Aksi Aktor** | **Reaksi Sistem** |
|---|---|
| 1. Menjalankan aplikasi dan menyorotkan kamera *webcam* kearah *marker* (*Smart Book*). | |
| | 2. Melakukan pencocokan pola *marker* dengan *file tracking*.<br>3. Pemilihan objek 3D yang akan di *render*. |
| **Kondisi Akhir** | Aplikasi menampilkan objek *gullet* secara *real time*. |
| **Skenario Gagal** | |
| **Aksi Aktor** | **Reaksi Sistem** |
| | 1. "Layar Kosong". |
| **Kondisi Akhir** | Aplikasi tetap menampilkan gambar yang sedang di *tracking*. |

Skenario objek lambung terkait dengan *use case* lihat dan dengarkan objek lambung dapat dilihat pada tabel 3.12.

**Tabel 3.12** Skenario *Use Case* Lihat dan Dengarkan Objek Lambung

| **Identifikasi** | |
|---|---|
| **Nama Use Case** | Lihat dan Dengarkan Objek Lambung |
| **Kode** | UC-AR07 |
| **Aktor** | *User* |
| **Tujuan** | Menampilkan objek dan mendengarkan penjelasan materi. |
| **Skenario Utama** | |
| **Aksi Aktor** | **Reaksi Sistem** |
| 1. Menjalankan aplikasi dan menyorotkan kamera *webcam* kearah *marker* (*Smart Book*). | |
| | 2. Melakukan pencocokan pola *marker* dengan *file tracking*.<br>3. Pemilihan objek 3D yang akan di *render*. |
| **Kondisi Akhir** | Aplikasi menampilkan objek lambung secara *real time*. |
| **Skenario Gagal** | |
| **Aksi Aktor** | **Reaksi Sistem** |
| | 1. "Layar Kosong". |
| **Kondisi Akhir** | Aplikasi tetap menampilkan gambar yang sedang di *tracking*. |

Skenario objek usus kecil terkait dengan *use case* lihat dan dengarkan objek usus kecil dapat dilihat pada tabel 3.13.

**Tabel 3.13** Skenario *Use Case* Lihat dan Dengarkan Objek Usus Kecil

| **Identifikasi** | |
|---|---|
| **Nama Use Case** | Lihat dan Dengarkan Objek Usus Kecil |
| **Kode** | UC-AR08 |
| **Aktor** | *User* |
| **Tujuan** | Menampilkan objek dan mendengarkan penjelasan materi. |
| **Skenario Utama** | |
| **Aksi Aktor** | **Reaksi Sistem** |
| 1. Menjalankan aplikasi dan menyorotkan kamera *webcam* kearah *marker* (*Smart Book*). | |
| | 2. Melakukan pencocokan pola *marker* dengan *file tracking*.<br>3. Pemilihan objek 3D yang akan di *render*. |
| **Kondisi Akhir** | Aplikasi menampilkan objek usus kecil secara *real time*. |
| **Skenario Gagal** | |
| **Aksi Aktor** | **Reaksi Sistem** |
| | 1. "Layar Kosong". |
| **Kondisi Akhir** | Aplikasi tetap menampilkan gambar yang sedang di *tracking*. |

Skenario objek usus besar terkait dengan *use case* lihat dan dengarkan objek usus besar dapat dilihat pada tabel 3.14.

**Tabel 3.14** Skenario *Use Case* Lihat dan Dengarkan Objek Usus Besar

| **Identifikasi** | |
|---|---|
| **Nama Use Case** | Lihat dan Dengarkan Objek Usus Besar |
| **Kode** | UC-AR09 |
| **Aktor** | *User* |
| **Tujuan** | Menampilkan objek dan mendengarkan penjelasan materi. |
| **Skenario Utama** | |
| **Aksi Aktor** | **Reaksi Sistem** |
| 1. Menjalankan aplikasi dan menyorotkan kamera *webcam* kearah *marker* (*Smart Book*). | |
| | 2. Melakukan pencocokan pola *marker* dengan *file tracking*.<br>3. Pemilihan objek 3D yang akan di *render*. |
| **Kondisi Akhir** | Aplikasi menampilkan objek usus besar secara *real time*. |
| **Skenario Gagal** | |
| **Aksi Aktor** | **Reaksi Sistem** |

| | |
|---|---|
| | 1. "Layar Kosong". |
| **Kondisi Akhir** | Aplikasi tetap menampilkan gambar yang sedang di *tracking*. |

Skenario objek peredaran darah terkait dengan *use case* lihat dan dengarkan objek peredaran darah dapat dilihat pada tabel 3.15.

**Tabel 3.15** Skenario *Use Case* Lihat dan Dengarkan Objek Peredaran Darah

| **Identifikasi** | |
|---|---|
| **Nama Use Case** | Lihat dan Dengarkan Objek Peredaran Darah |
| **Kode** | UC-AR10 |
| **Aktor** | *User* |
| **Tujuan** | Menampilkan objek dan mendengarkan penjelasan materi. |
| **Skenario Utama** | |
| **Aksi Aktor** | **Reaksi Sistem** |
| 1. Menjalankan aplikasi dan menyorotkan kamera *webcam* kearah *marker* (*Smart Book*). | |
| | 2. Melakukan pencocokan pola *marker* dengan *file tracking*.<br>3. Pemilihan objek 3D yang akan di *render*. |
| **Kondisi Akhir** | Aplikasi menampilkan objek peredaran darah secara *real time*. |
| **Skenario Gagal** | |
| **Aksi Aktor** | **Reaksi Sistem** |
| | 1. "Layar Kosong". |
| **Kondisi Akhir** | Aplikasi tetap menampilkan gambar yang sedang di *tracking*. |

Skenario objek jantung terkait dengan *use case* lihat dan dengarkan objek jantung dapat dilihat pada tabel 3.16.

**Tabel 3.16** Skenario *Use Case* Lihat dan Dengarkan Objek Jantung

| **Identifikasi** | |
|---|---|
| **Nama Use Case** | Lihat dan Dengarkan Objek Jantung |
| **Kode** | UC-AR11 |
| **Aktor** | *User* |
| **Tujuan** | Menampilkan objek dan mendengarkan penjelasan materi. |
| **Skenario Utama** | |
| **Aksi Aktor** | **Reaksi Sistem** |
| 1. Menjalankan aplikasi dan menyorotkan kamera *webcam* kearah *marker* (*Smart Book*). | |

| | |
|---|---|
| | 2. Melakukan pencocokan pola *marker* dengan *file tracking*.<br>3. Pemilihan objek 3D yang akan di *render*. |
| **Kondisi Akhir** | Aplikasi menampilkan objek jantung secara *real time*. |
| **Skenario Gagal** | |
| **Aksi Aktor** | **Reaksi Sistem** |
| | 1. "Layar Kosong". |
| **Kondisi Akhir** | Aplikasi tetap menampilkan gambar yang sedang di *tracking*. |

Skenario objek sel darah terkait dengan *use case* lihat dan dengarkan objek sel darah dapat dilihat pada tabel 3.17.

**Tabel 3.17** Skenario *Use Case* Lihat dan Dengarkan Objek Sel Darah

| **Identifikasi** | |
|---|---|
| **Nama Use Case** | Lihat dan Dengarkan Objek Sel Darah |
| **Kode** | UC-AR12 |
| **Aktor** | *User* |
| **Tujuan** | Menampilkan objek dan mendengarkan penjelasan materi. |
| **Skenario Utama** | |
| **Aksi Aktor** | **Reaksi Sistem** |
| 1. Menjalankan aplikasi dan menyorotkan kamera *webcam* kearah *marker* (*Smart Book*). | |
| | 2. Melakukan pencocokan pola *marker* dengan *file tracking*.<br>3. Pemilihan objek 3D yang akan di *render*. |
| **Kondisi Akhir** | Aplikasi menampilkan objek sel darah secara *real time*. |
| **Skenario Gagal** | |
| **Aksi Aktor** | **Reaksi Sistem** |
| | 1. "Layar Kosong". |
| **Kondisi Akhir** | Aplikasi tetap menampilkan gambar yang sedang di *tracking*. |

Skenario objek pembuluh darah terkait dengan *use case* lihat dan dengarkan objek pembuluh darah dapat dilihat pada tabel 3.18.

**Tabel 3.18** Skenario *Use Case* Lihat dan Dengarkan Objek Pembuluh Darah

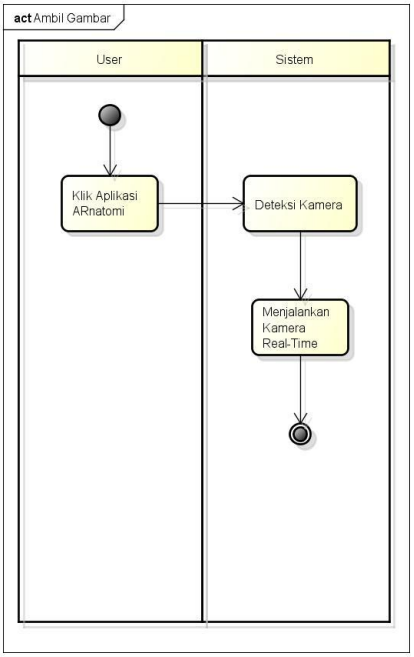
| **Identifikasi** | |
|---|---|
| **Nama Use Case** | Lihat dan Dengarkan Objek Pembuluh Darah |
| **Kode** | UC-AR13 |
| **Aktor** | *User* |

| | |
|---|---|
| **Tujuan** | Menampilkan objek dan mendengarkan penjelasan materi. |
| **Skenario Utama** | |
| **Aksi Aktor** | **Reaksi Sistem** |
| 1. Menjalankan aplikasi dan menyorotkan kamera *webcam* kearah *marker* (*Smart Book*). | |
| | 2. Melakukan pencocokan pola *marker* dengan *file tracking*.<br>3. Pemilihan objek 3D yang akan di *render*. |
| **Kondisi Akhir** | Aplikasi menampilkan objek pembuluh darah secara *real time*. |
| **Skenario Gagal** | |
| **Aksi Aktor** | **Reaksi Sistem** |
| | 1. "Layar Kosong". |
| **Kondisi Akhir** | Aplikasi tetap menampilkan gambar yang sedang di *tracking*. |

### 3.3.4 *Activity Diagram*

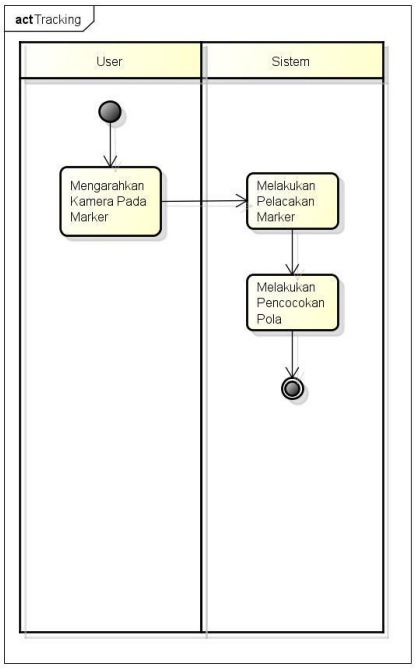
*Activity Diagram* atau diagram aktifitas menggambarkan *workflow* (aliran kerja) aktor pada saat kamera *webcam* menyorot *marker* agar dapat melakukan *tracking* yang akan dicocokan polanya.

Gambaran *activity diagram* terkait *use case* ambil gambar dapat dilihat pada gambar 3.29.



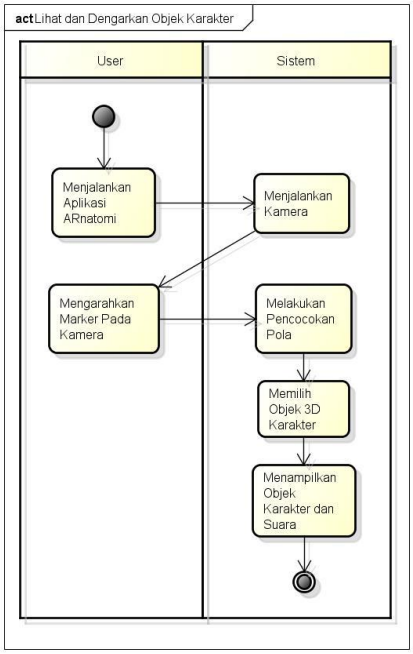

**Gambar 3.29** *Activity Diagram* Ambil Gambar

Gambaran *activity diagram* terkait *use case tracking* dapat dilihat pada gambar 3.30.



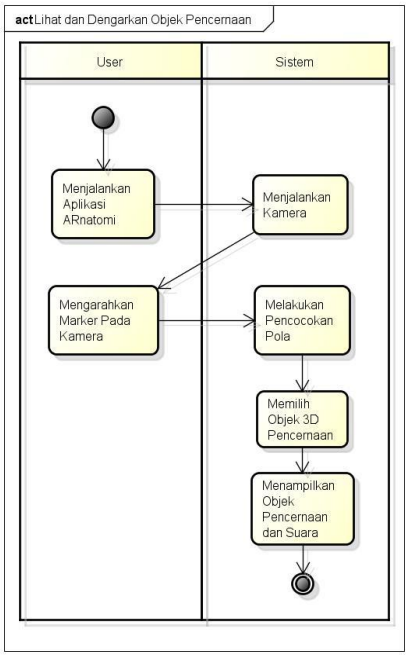

**Gambar 3.30** *Activity Diagram Tracking*

Gambaran *activity diagram* terkait *use case* lihat dan dengarkan objek karakter dapat dilihat pada gambar 3.31.



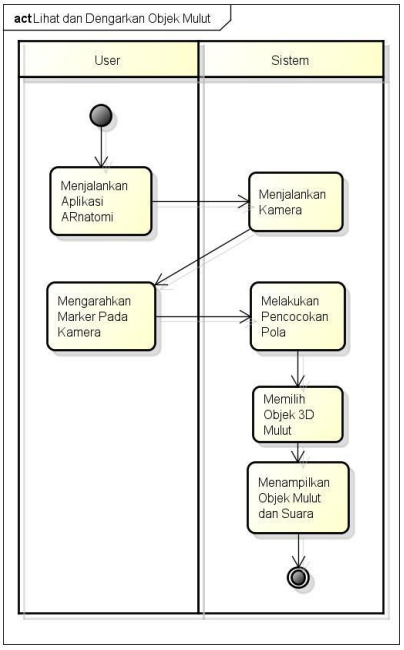

**Gambar 3.31** *Activity Diagram* Lihat dan Dengarkan Objek Karakter

Gambaran *activity diagram* terkait *use case* lihat dan dengarkan objek pencernaan dapat dilihat pada gambar 3.32.



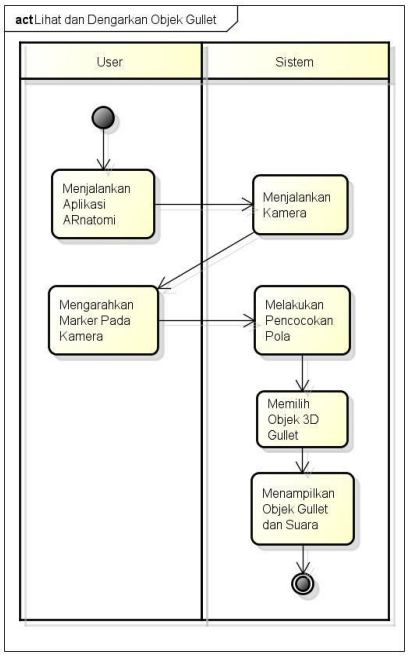

**Gambar 3.32** *Activity Diagram* Lihat dan Dengarkan Objek Pencernaan

Gambaran *activity diagram* terkait *use case* lihat dan dengarkan objek mulut dapat dilihat pada gambar 3.33.



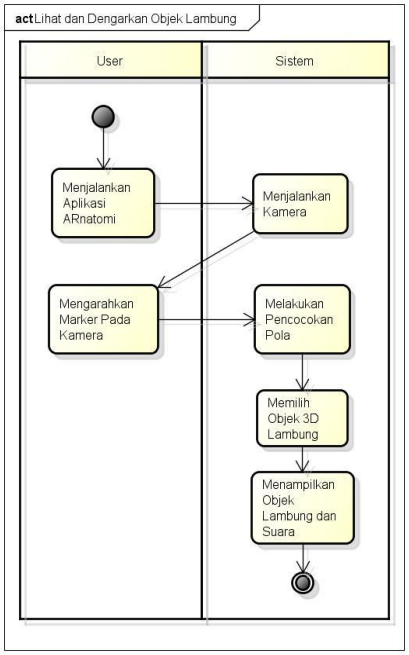

**Gambar 3.33** *Activity Diagram* Lihat dan Dengarkan Objek Mulut

Gambaran *activity diagram* terkait *use case* lihat dan dengarkan objek *gullet* dapat dilihat pada gambar 3.34.



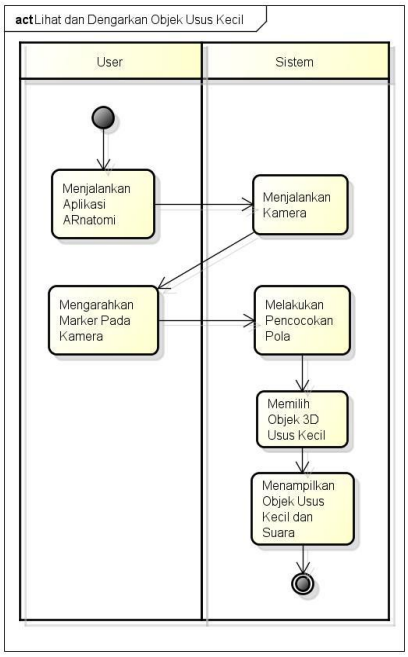

**Gambar 3.34** *Activity Diagram* Lihat dan Dengarkan Objek *Gullet*

Gambaran *activity diagram* terkait *use case* lihat dan dengarkan objek lambung dapat dilihat pada gambar 3.35.



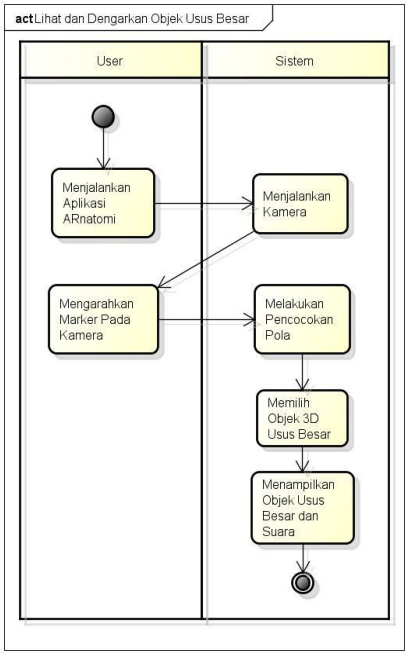

**Gambar 3.35** *Activity Diagram* Lihat dan Dengarkan Objek Lambung

Gambaran *activity diagram* terkait *use case* lihat dan dengarkan objek usus kecil dapat dilihat pada gambar 3.36.



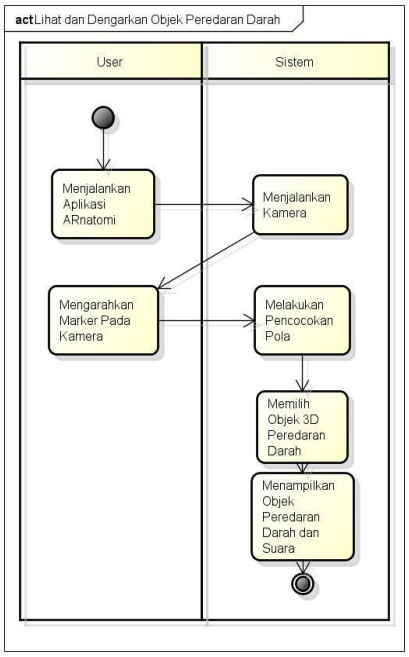

**Gambar 3.36** *Activity Diagram* Lihat dan Dengarkan Objek Usus Kecil

Gambaran *activity diagram* terkait *use case* lihat dan dengarkan objek usus besar dapat dilihat pada gambar 3.37.



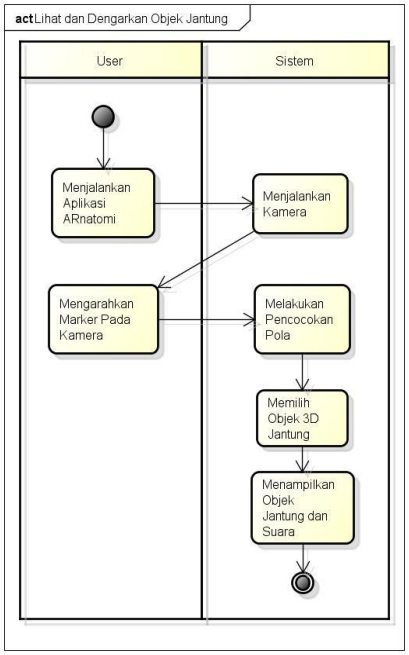

**Gambar 3.37** *Activity Diagram* Lihat dan Dengarkan Objek Usus Besar

Gambaran *activity diagram* terkait *use case* lihat dan dengarkan objek peredaran darah dapat dilihat pada gambar 3.38.



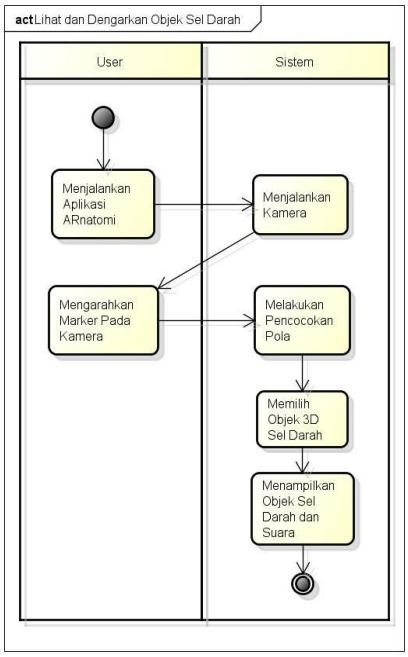

**Gambar 3.38** *Activity Diagram* Lihat dan Dengarkan Objek Peredaran Darah

Gambaran *activity diagram* terkait *use case* lihat dan dengarkan objek jantung dapat dilihat pada gambar 3.39.



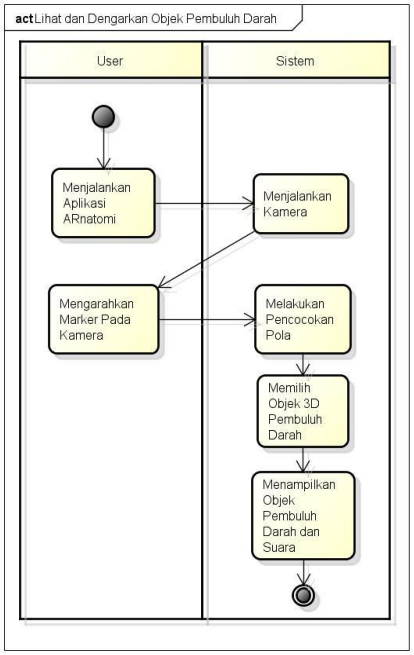

**Gambar 3.39** *Activity Diagram* Lihat dan Dengarkan Objek Jantung

Gambaran *activity diagram* terkait *use case* lihat dan dengarkan objek sel darah dapat dilihat pada gambar 3.40.




**Gambar 3.40** *Activity Diagram* Lihat dan Dengarkan Objek Sel Darah

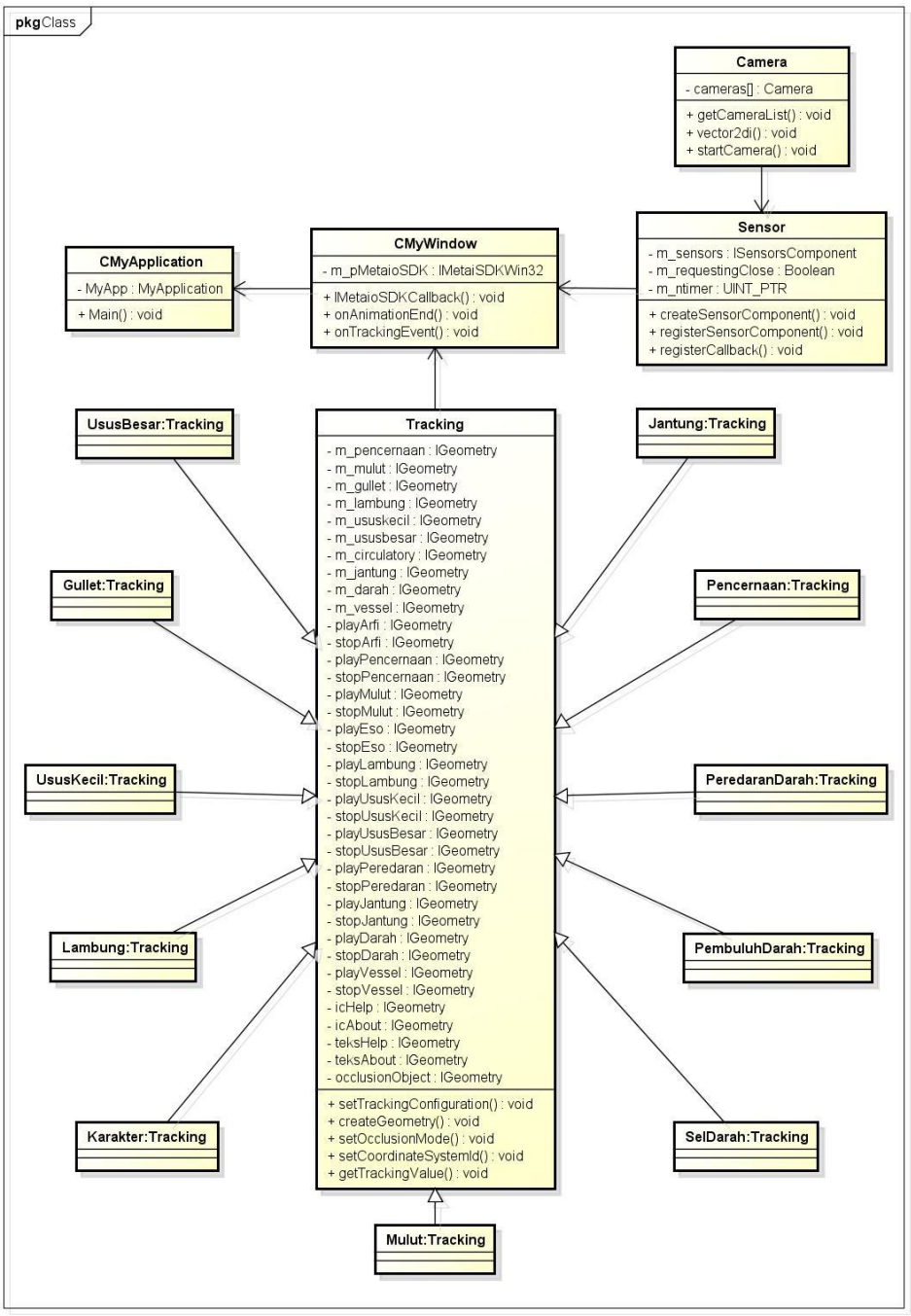
Gambaran *activity diagram* terkait *use case* lihat dan dengarkan objek pembuluh darah dapat dilihat pada gambar 3.41.




**Gambar 3.41** *Activity Diagram* Lihat dan Dengarkan Objek Pembuluh Darah

### 3.3.5 *Class Diagram*

*Class Diagram* merupakan gambaran struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Berikut adalah *class diagram* dari sistem aplikasi *ARnatomi*.




**Gambar 3.42** *Class Diagram* ARnatomi

Penjelasan dari masing-masing *class diagram* dapat dilihat pada tabel 3.19.

**Tabel 3.19** Deskripsi *Class Diagram ARnatomi*

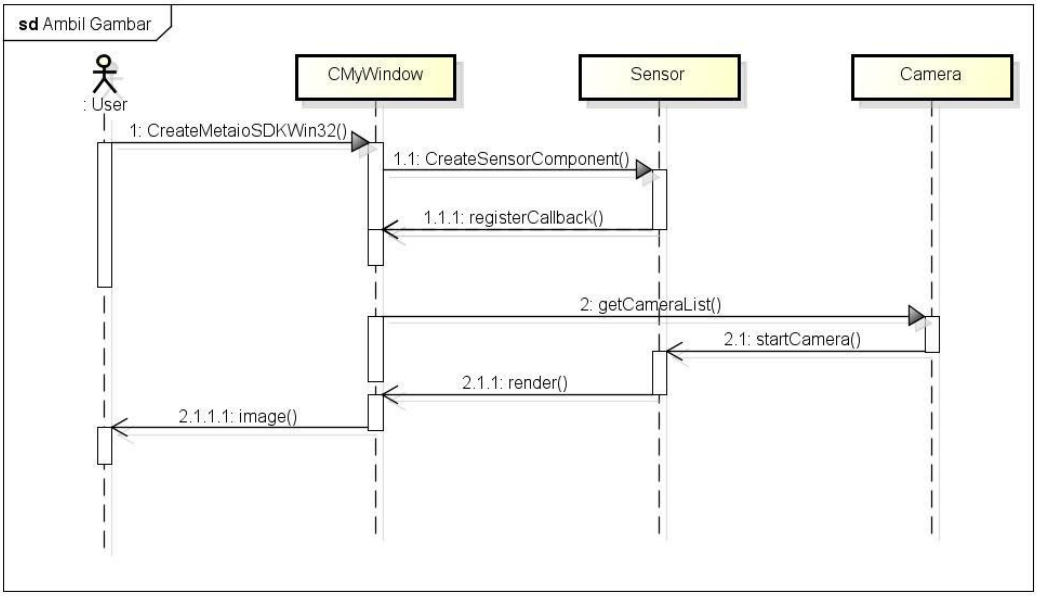
| **User Application** | | |
|---|---|---|
| **Class** | **Jenis Class** | **Deskripsi** |
| CMyApplication | *Control* | *Class* yang mengatur apa yang akan ditampilkan pada layar *desktop user*. |
| Sensor | *Control* | *Class* yang mengatur fungsi *hardware* yang melekat pada perangkat yang digunakan. |
| Camera | *Control* | *Class* yang mengatur fungsi kamera yang terdapat pada komputer. |
| CMyWindow | *Boundary* | *Class* yang menampilkan hasil *tracking* dan *movie* yang dilengkapi animasi dan suara dari proses sistem ke *interface* / layar pengguna. |
| Tracking | *Entity* | *Class* yang bertanggung jawab dalam pengenalan *image tracking* dan *load IGeometry*. |
| Karakter:Tracking | *Entity* | *Class* yang berisikan *method* dari *behavior* karakter seperti pengaturan skala objek karakter, rotasi maupun translasi karakter. |
| Pencernaan:Tracking | *Entity* | *Class* yang menampung *render* objek 3D beserta tekstur *movie* dari Pencernaan. |
| Mulut:Tracking | *Entity* | *Class* yang menampung *render* objek 3D beserta tekstur *movie* dari Mulut. |
| Gullet:Tracking | *Entity* | *Class* yang menampung *render* objek 3D beserta tekstur *movie* dari Gullet. |
| Lambung:Tracking | *Entity* | *Class* yang menampung *render* objek 3D beserta tekstur *movie* dari Lambung. |
| UsusKecil:Tracking | *Entity* | *Class* yang menampung *render* objek 3D beserta tekstur *movie* dari Usus Kecil. |
| UsusBesar:Tracking | *Entity* | *Class* yang menampung *render* objek 3D beserta tekstur *movie* dari Usus Besar. |
| PeredaranDarah:Tracking | *Entity* | *Class* yang menampung *render* objek 3D beserta tekstur *movie* dari Peredaran Darah. |

| Class | Jenis Class | Deskripsi |
|---|---|---|
| Jantung:Tracking | *Entity* | *Class* yang menampung *render* objek 3D beserta tekstur *movie* dari Jantung. |
| SelDarah:Tracking | *Entity* | *Class* yang menampung *render* objek 3D beserta tekstur *movie* dari Sel Darah. |
| PembuluhDarah:Tracking | *Entity* | *Class* yang menampung *render* objek 3D beserta tekstur *movie* dari Pembuluh Darah. |

### 3.3.6 *Sequence Diagram*

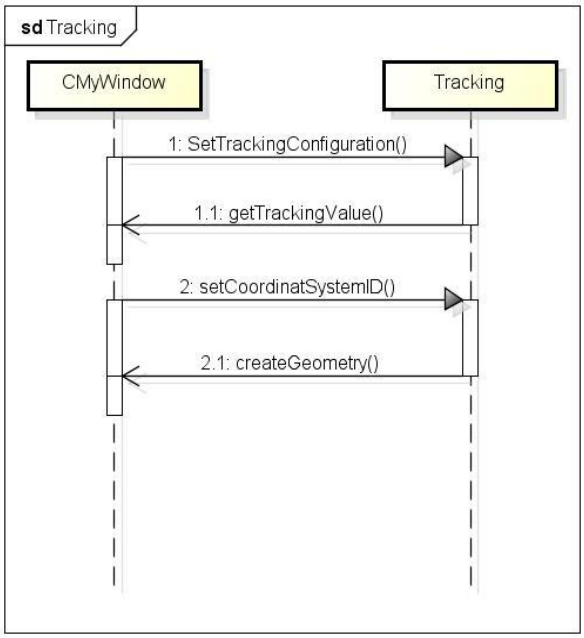
*Sequence Diagram* menggambarkan kelakuan objek pada *use case* dengan mendiskripsikan waktu hidup objek dan *message* yang dikirmkan dan diterima antar objek.

Gambaran yang ada pada *sequence diagram* ambil gambar dapat dilihat pada gambar 3.43.



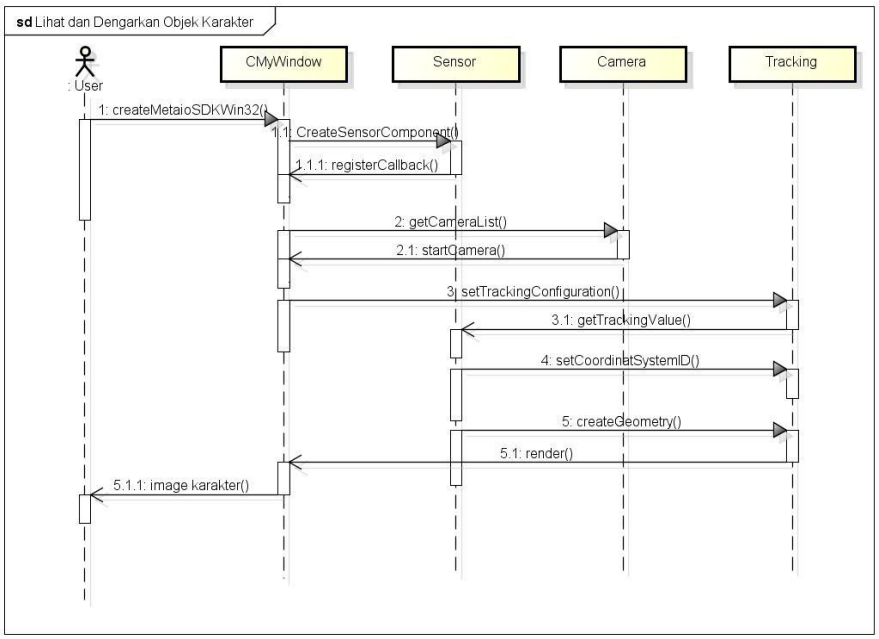

**Gambar 3.43** *Sequence Diagram* Ambil Gambar

Gambaran yang ada pada *sequence diagram tracking* dapat dilihat pada gambar 3.44.



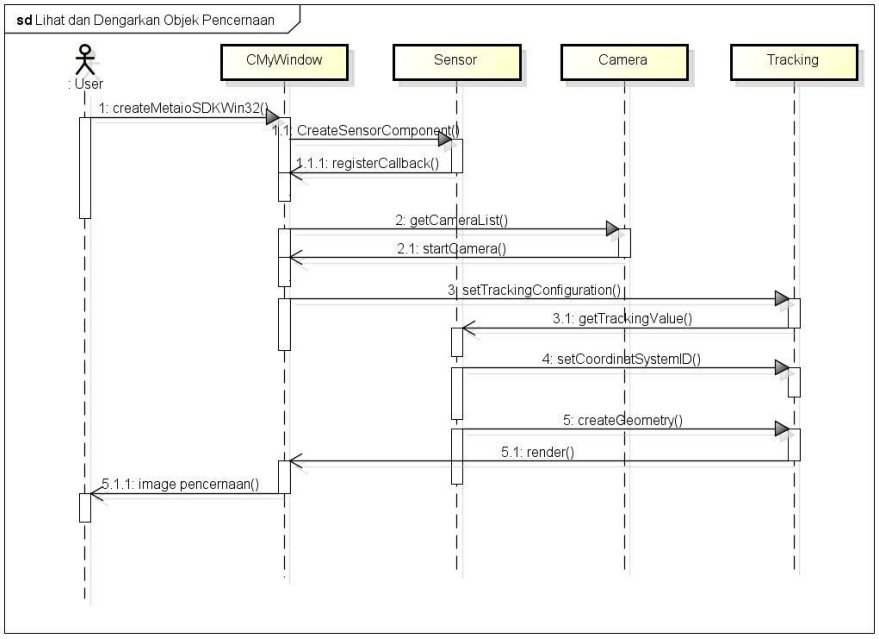

**Gambar 3.44** *Sequence Diagram Tracking*

Gambaran yang ada pada *sequence diagram* karakter dapat dilihat pada gambar 3.45.



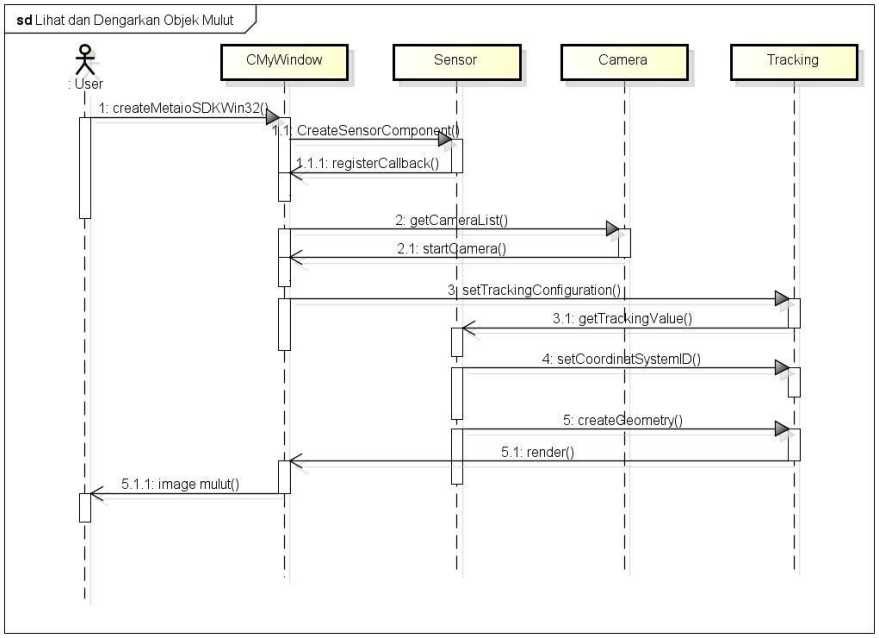

**Gambar 3.45** *Sequence Diagram* Karakter

Gambaran yang ada pada *sequence diagram* pencernaan dapat dilihat pada gambar 3.46.



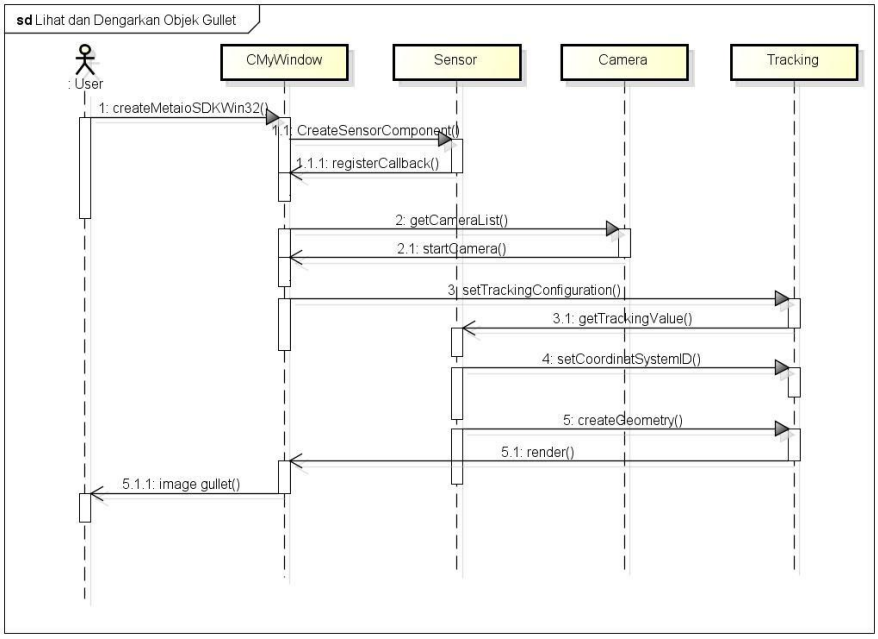

**Gambar 3.46** *Sequence Diagram* Pencernaan

Gambaran yang ada pada *sequence diagram* mulut dapat dilihat pada gambar 3.47.



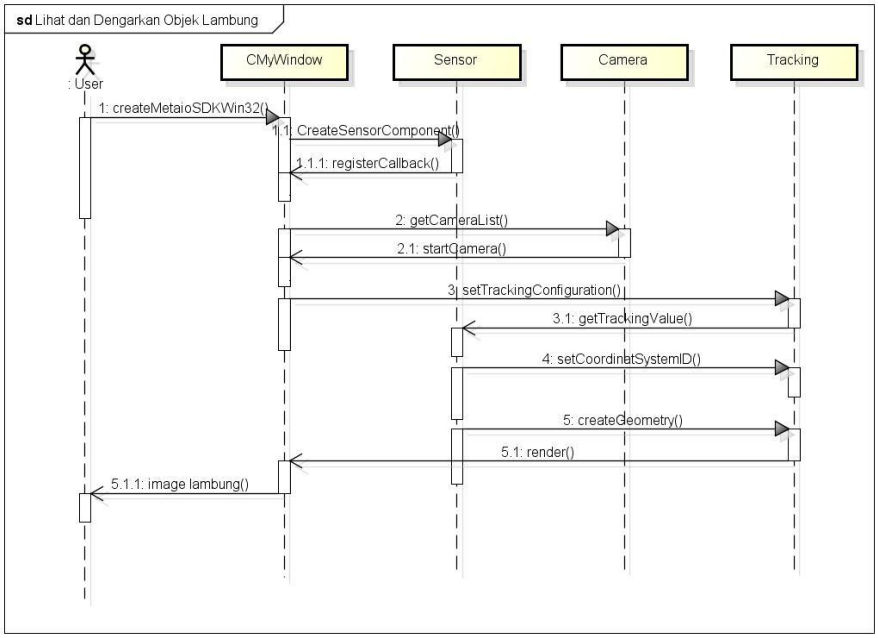

**Gambar 3.47** *Sequence Diagram* Mulut

Gambaran yang ada pada *sequence diagram gullet* dapat dilihat pada gambar 3.48.



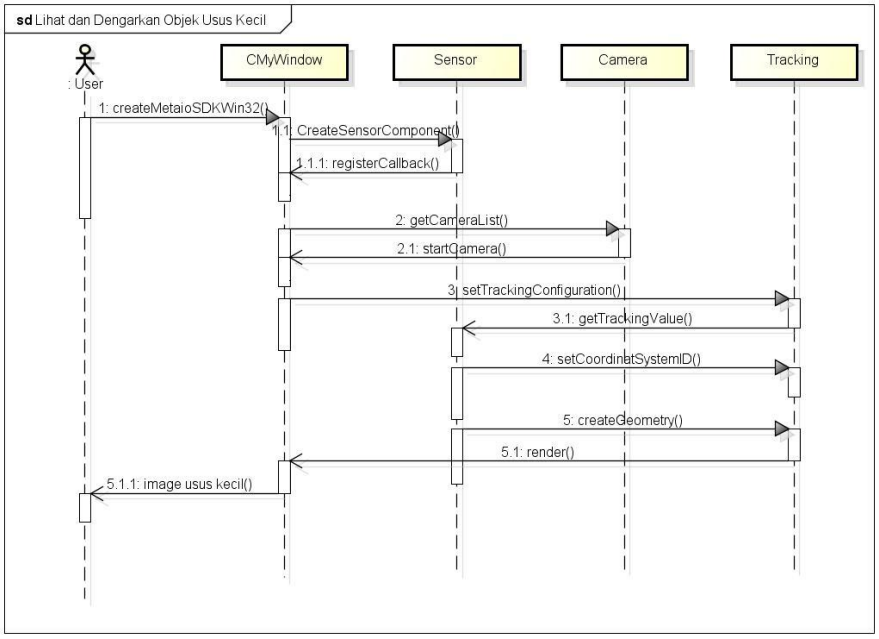

**Gambar 3.48** *Sequence Diagram Gullet*

Gambaran yang ada pada *sequence diagram* lambung dapat dilihat pada gambar 3.49.



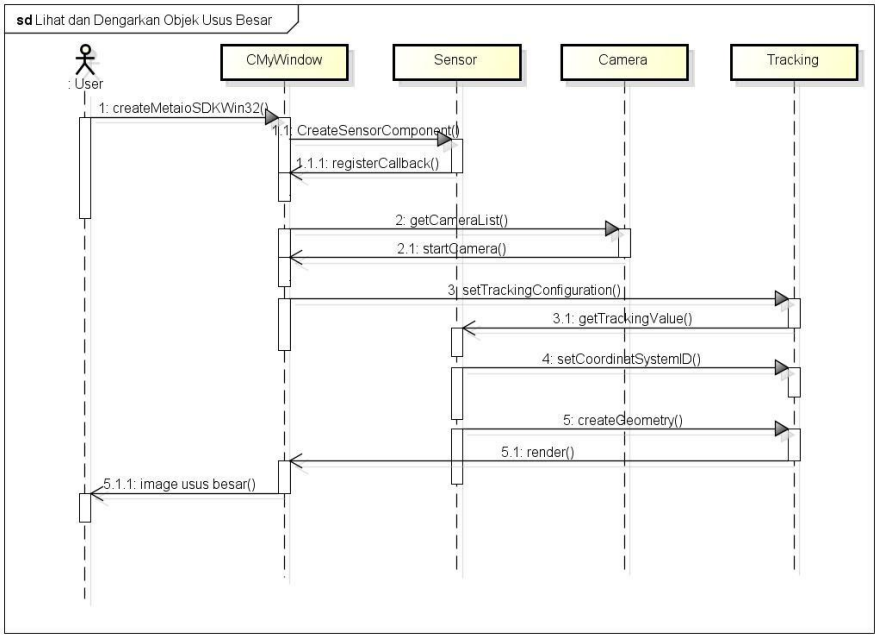

**Gambar 3.49** *Sequence Diagram* Lambung

Gambaran yang ada pada *sequence diagram* usus kecil dapat dilihat pada gambar 3.50.



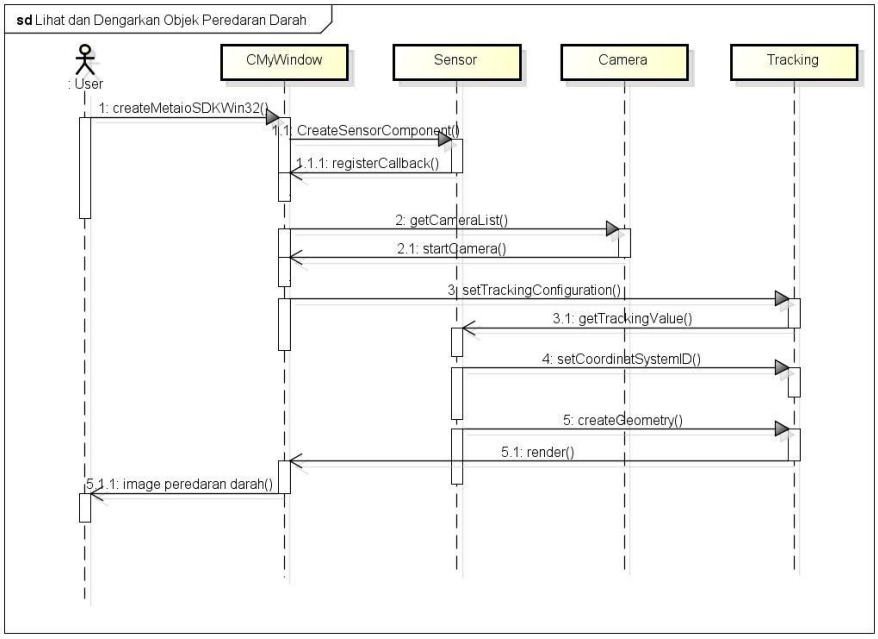

**Gambar 3.50** *Sequence Diagram* Usus Kecil

Gambaran yang ada pada *sequence diagram* usus besar dapat dilihat pada gambar 3.51.



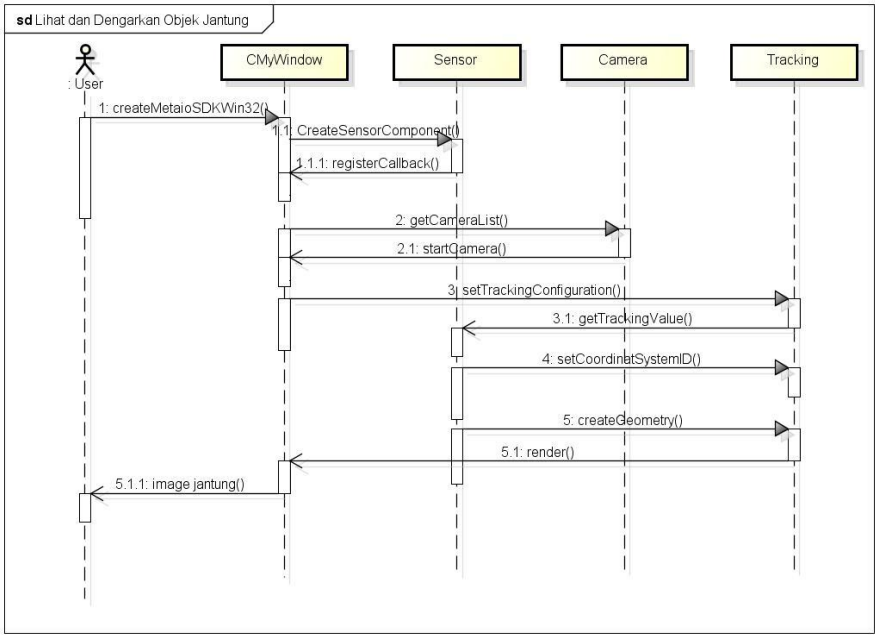

**Gambar 3.51** *Sequence Diagram* Usus Besar

Gambaran yang ada pada *sequence diagram* peredaran darah dapat dilihat pada gambar 3.52.



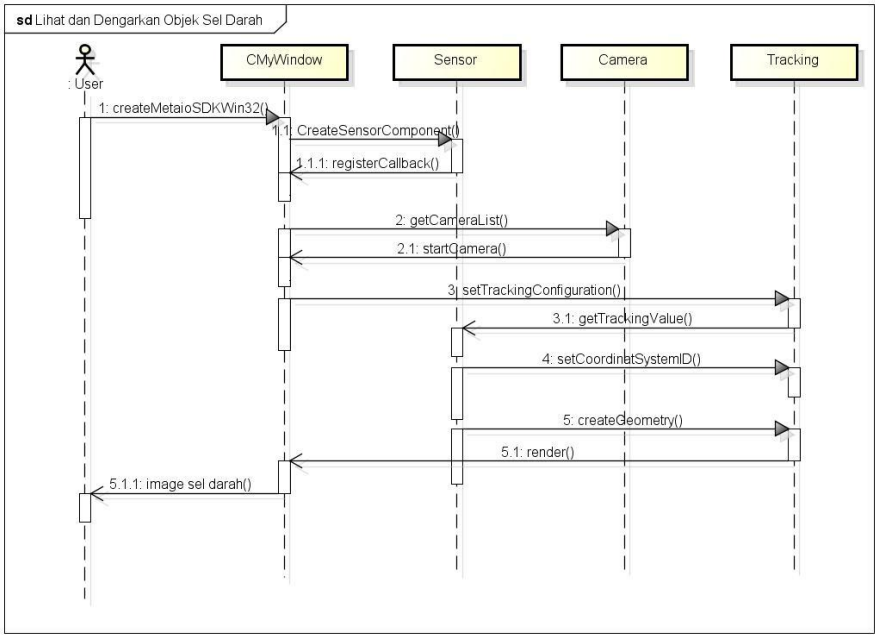

**Gambar 3.52** *Sequence Diagram* Peredaran Darah

Gambaran yang ada pada *sequence diagram* jantung dapat dilihat pada gambar 3.53.



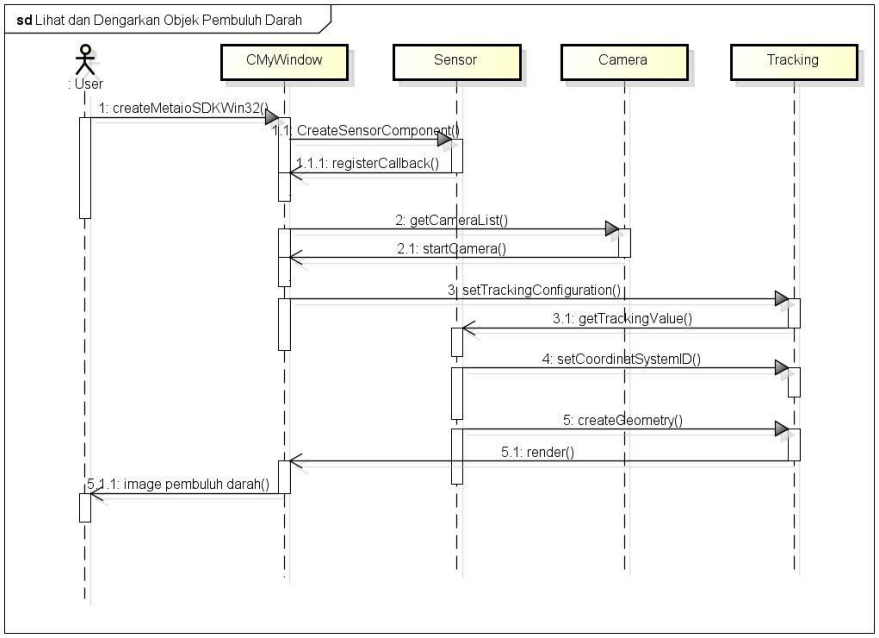

**Gambar 3.53** *Sequence Diagram* Jantung

Gambaran yang ada pada *sequence diagram* sel darah dapat dilihat pada gambar 3.54.




**Gambar 3.54** *Sequence Diagram* Sel Darah

Gambaran yang ada pada *sequence diagram* pembuluh darah dapat dilihat pada gambar 3.55.




**Gambar 3.55** *Sequence Diagram* Pembuluh Darah

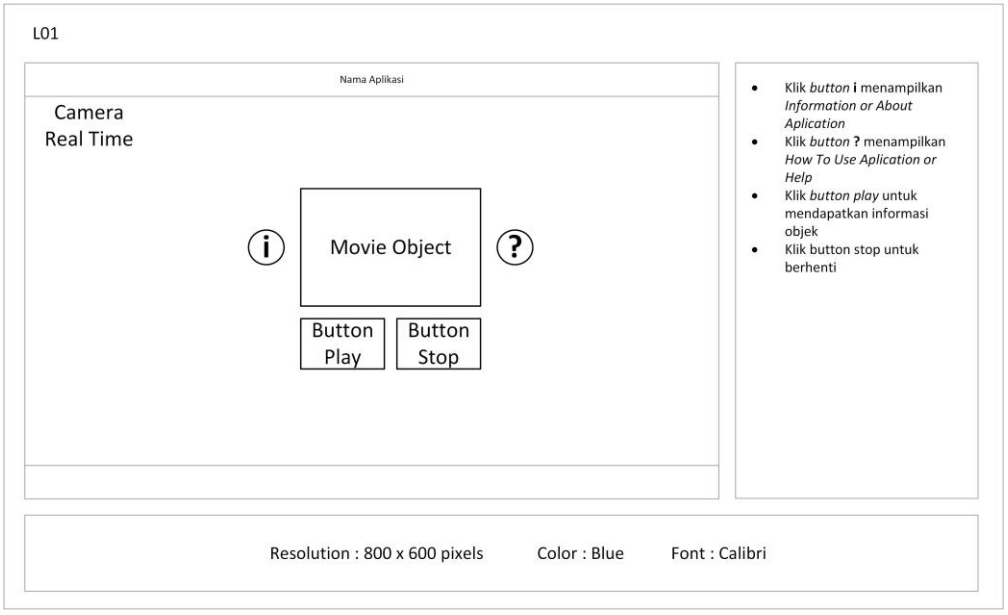
## 3.4 Perancangan Sistem

Perancangan akan dimulai setelah tahap analisis terhadap sistem selesai dilakukan. Perancangan dapat diartikan sebagai penggambaran, perencanaan, dan pembuatan sketsa dari beberapa elemen yang terpisah kedalam satu kesatuan yang utuh dan berfungsi.

### 3.4.1 Perancangan Antarmuka

Perancangan antarmuka bertujuan untuk mencari bentuk yang optimal dari aplikasi yang akan dibangun dengan pertimbangan faktor-faktor permasalahan dan kebutuhan yang ada pada sistem seperti yang telah ditetapkan pada tahap analisis.

Adapun perancangan antarmuka aplikasi *home* yang akan dibangun dapat dilihat dari gambar 3.56.

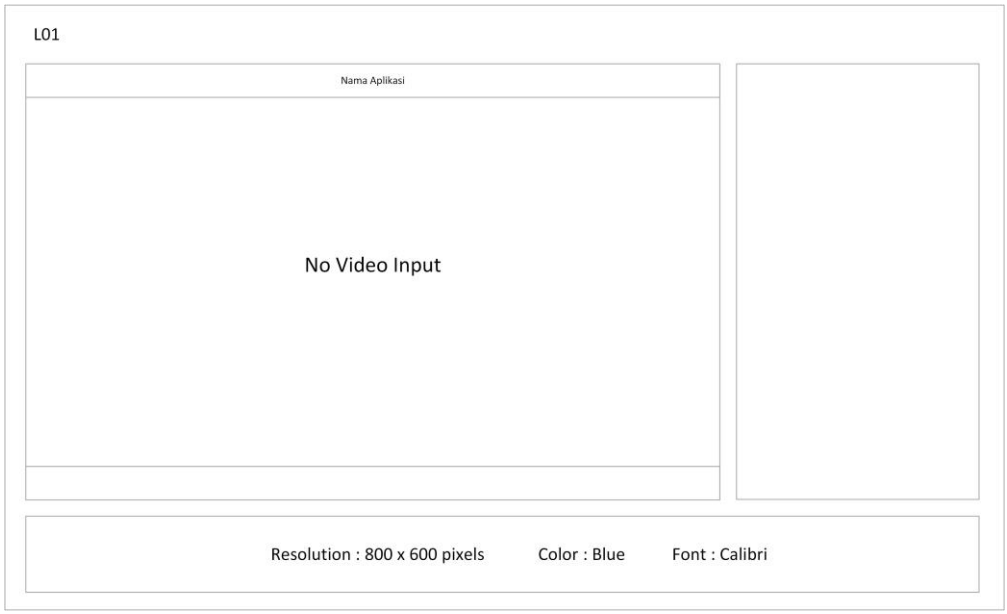



**Gambar 3.56** Perancangan Antarmuka *Home*

Menu *home* merupakan inti dari aplikasi *ARnatomi*. Menu home untuk menyorot gambar *marker* secara *real time.* Serta dapat menampilkan informasi objek 3D dan juga suara. Didalam *tracking* menu *home*, sudah di sisipkan *button about aplication, button help, button play* dan *button stop*.

### 3.4.2 Perancangan Pesan

Pada tahap ini merupakan perancangan pesan kesalahan yang terdapat pada aplikasi *ARnatomi*. Tampilannya dapat dilihat pada gambar 3.57.




**Gambar 3.57** Perancangan Pesan Kesalahan

Pesan tersebut diartikan bahwa kamera *webcam* yang ada pada komputer tidak terdeteksi oleh aplikasi atau kamera tidak mendukung terhadap aplikasi.