BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Masalah

Permasalahan pada penelitian ini adalah sulitnya mengimplementasikan metode *hand tracking* untuk membangun aplikasi *Augmented Reality* untuk memunculkan objek tiga dimensi berupa desain bangunan arsitektur. Hal ini disebabkan karena sulitnya untuk mendeteksi kontur tangan dan juga mendeteksi jari jari tangan dan estimasi dari sudut tangan, selain itu terdapat juga kesulitan dalam hal menambahkan objek tiga dimensi ke frame karena dalam penambahan objek tiga dimensi tersebut dibutuhkan perhitungan agar objek tiga dimensi tersebut terletak tepat di posisi tangan dengan sudut yang sesuai dengan arah tangan.

3.2 Deskripsi Sistem

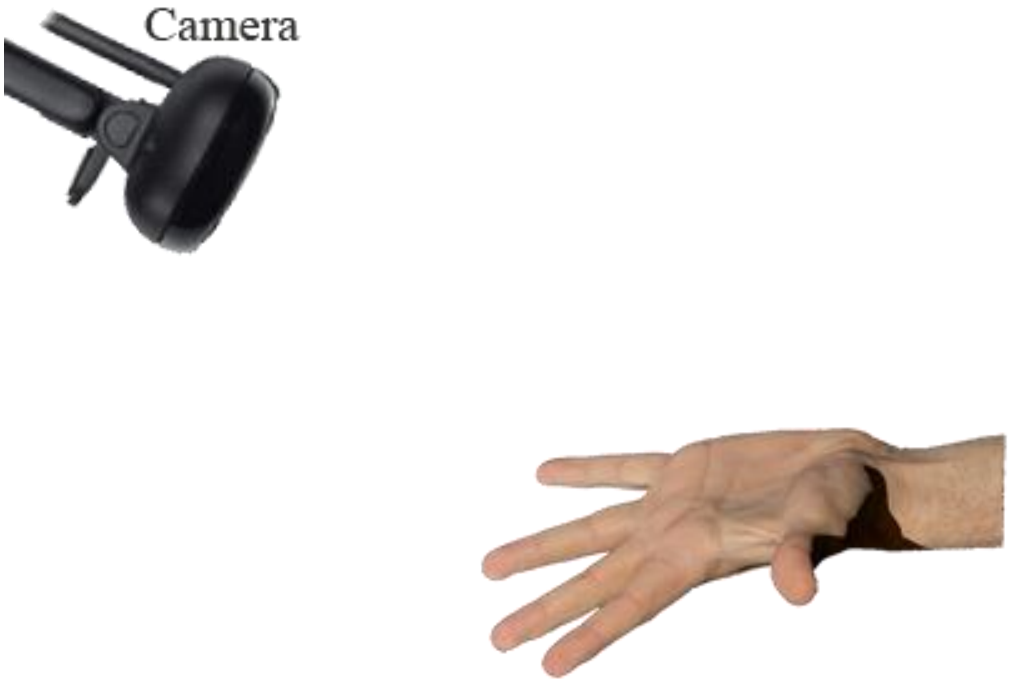
Sistem yang akan dibangun merupakan sistem *augmented reality* dengan mengimplementasikan metode *hand tracking.* Dalam sistem yang dibangun ini terdapat satu *user* yaitu *user* yang bertugas memilih file objek 3 dimensi yang akan dimunculkan pada proses berikutnya, selain itu *user* juga bertugas untuk mengarahkan tangannya untuk di *track* oleh sistem lalu sistem akan memunculkan objek 3 dimensi sesuai dengan lokasi dan rotasi tangan *user.* Alur Sistem yang dibangun dapat dilihat pada gambar 3.1.

1. *User* memilih file input berupa file objek 3 dimensi berformat fbx yang nantinya akan dimunculkan
2. *User* menekan tombol start
3. *User* mengarahkan tangannya ke kamera
4. Sistem mendeteksi keberadaan tangan
5. Sistem mengestimasi lokasi dan rotasi dari objek 3D yang akan dimunculkan
6. Sistem memunculkan objek 3 dimensi yang telah dipilih oleh *user* berdasarkan lokasi dan rotasi tangan yang telah terdeteksi.



Gambar 3.1 Deskripsi Sistem

Sistem akan memunculkan objek 3D berdasarkan lokasi tangan dan juga rotasinya, gambar 3.2 Berikut merupakan contoh posisi tangan terhadap kamera sehingga objek 3D dapat muncul dengan sudut yang baik.



Gambar 3.2 Contoh Posisi Tangan Terhadap Kamera

3.3 Analisis Kebutuhan Data

Sistem yang dibangun adalah aplikasi *Augmented Reality* dengan mengimplementasikan metode *Hand Tracking* untuk memunculkan objek tiga dimensi berupa desain arsitektur bangunan. Sistem ini adalah sistem yang akan memunculkan objek tiga dimensi jika sistem mendeteksi gerakan tangan manusia pada *web-cam*, sehingga sistem ini membutuhkan input berupa citra yang diambil secara *real-time* menggunakan alat input berupa *webcam* dimana citra yang diambil tersebut merupakan kumpulan matriks yang nantinya akan diolah lebih lanjut dalam sistem untuk proses pendeteksian objek tangan manusia untuk memunculkan objek tiga dimensi. Selain input berupa citra yang diambil dari *webcam* secara *real-time* sistem ini juga membutuhkan input dari user yaitu berupa objek tiga dimensi yang akan ditampilkan. Objek tiga dimensi yang dapat dipilih oleh user adalah objek tiga dimensi berformat FBX.

3.4 Analisis Algoritma

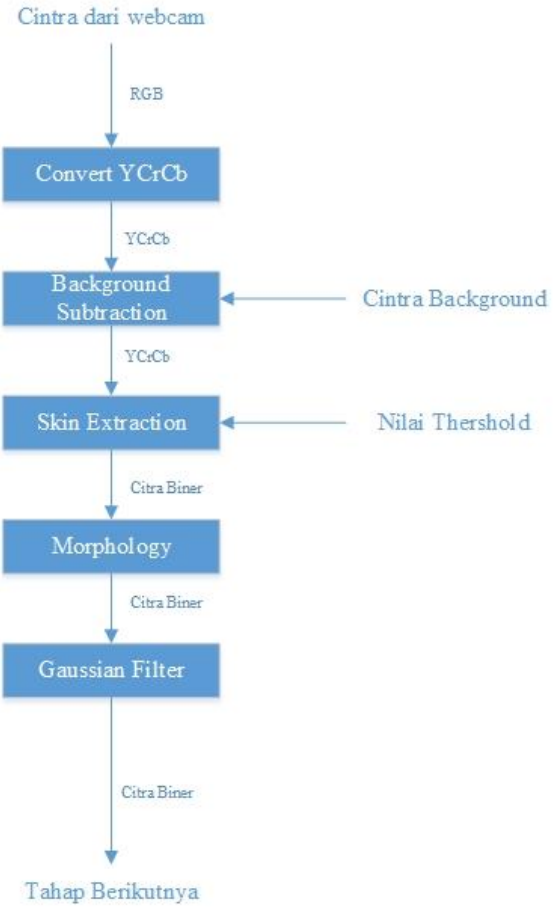
Dalam penelitian ini proses-proses yang dilakukan untuk membangun aplikasi *Augmented Reality ini* dibagi menjadi 3 tahap yang dapat dilihat pada gambar 3.3 berikut :



Gambar 3.3 Proses Algoritma

3.4.1 Tahapan Camera

Tahapan ini adalah tahap dimana proses berhubungan langsung dengan *webcam.* Citra yang diambli dari *webcam* akan diproses sedemikian rupa sehingga menghasilkan citra biner yang *smooth.* Gambar 3.4 berikut menjelaskan proses proses yang terjadi pada tahapan camera.



Gambar 3.4 Proses-Proses Pada Tahapan Camera

3.4.1.1 Konversi YCrCb

Sebelum citra yang ditangkap diproses pada tahap-tahap berikutnya citra perlu dikonversi kebentuk YCrCb karena pada beberapa tahap berikutnya citra yang akan diproses harus berformat YCrCb. Dalam penelitian ini pengkonversian citra dilakukan dengan menggunakan fungsi Convert pada library EmguCv, berikut merupakan fungsi Convert :

```
Image<Ycc,byte> Convert<Ycc,byte>();
```

Fungsi convert akan memberikan nilai balikan berupa matriks citra. Ycc merupakan format tujuan dari konversi citra karena pada penelitian ini konversi dilakukan ke citra YCrCb maka format yang dipilih adalah Ycc sedangkan byte merupakan nilai kedalaman citra dimana nilai byte adalah 0-255.

3.4.1.2 Background Subtraction

Metode background subtractor merupakan metode untuk memisahkan antara foreground dan background sehingga hanya tersisa foregroundnya saja. Langkah ini dimaksudkan agar mempermudah proses pada saat proses skin color extraction apabila terjadi kemiripan antara warna background dan warna kulit. Metode yang digunakan dalam background subtraction ini adalah metode background statis dimana background lingkungan akan diambil terlebih dahulu sebelum sistem dijalankan.

Metode ini berjalan dengan cara membandingkan setiap frame dari inputan webcam dengan background yang telah diambil agar menghasilkan gambar foregroundnya saja. Sistem akan mengambil frame yang sedang berjalan lalu mengubah formatnya menjadi format YCrCb lalu membandingkan perbedaan setiap channelnya dengan Channell yang ada di citra background hasil perbedaan dari setiap channelnya ini lalu akan di filter dengan menggunakan nilai threshold yang berbeda beda jika nilainya berada diantara nilai threshold min dan max maka nilai tersebut akan tetap dan jika tidak maka nilai akan dijadikan nol. Setelah

proses pada setiap channel selesai kemudian channel channel dari hasil tersebut disatukan kembali agar membentuk citra yang baru yaitu citra yang tersisa foregroundnya saja.

Langkah pertama yang dilakukan dalam metode background subtraction ini adalah mencari nilai absolut dari hasil pengurangan antara frame yang sedang berjalan dengan citra background dengan munggunakan fungsi *AbsDiff* pada library EmguCV, berikut merupakan fungsi *AbsDiff* :

```
Image<Ycc,byte> AbsDiff(
     Image<Ycc,byte> img2
     );
```

Fungsi *AbsDiff* akan mengembalikan nilai berupa image pada kasus ini format image adalah YCrCb. Parameter img2 merupakan parameter citra yang menjadi nilai pengurang yang nantinya akan mengurangi nilai dari citra yang dimaksud.

Setelah citra hasil dari pengurangan didapatkan maka citra tersebut menjadi 3 channel yaitu channel Y, Cr dan Cb kemudian setiap channel tersebut di proses dengan menggunakan nilai threshold yang berbeda beda dengan menggunakan fungsi *InRange* pada library EmguCV, berikut merupakan fungsi dari *InRange* :

```
Image<Gray,byte> InRange(
     Gray lower,
     Gray higher
     );
```

Fungsi *Inrange* merupakan fungsi yang akan memeriksa setiap nilai pada matriks citra apabila nilai berada diantara batas bawah dan batas atas maka akan mengembalikan nilai 0 dan jika tidak akan mengembalikan nilai 255. Kembalian dari fungsi *Inrange* merupakan sebuah citra berformat biner sedangkan parameter

lower merupakan nilai batas bawah dan nilai parameter higher merupakan nilai batas atas.

Nilai threshold yang dipakai pada masing masing channel pada penelitian ini adalah sebagai berikut :

Min Y = 34 ; Max Y = 255;

Min Cr = 10 ; Max Cr = 255;

Min Cb = 24 ; Max Cb = 255;

Kemudian setelah ketiga channel tersebut di proses menggunakan nilai threshold setiap channel tersebut kemudan diproses kembali menggunakan fungsi morfologi untuk menghilangkan noise pada hasil filter. Kemudian ketiga channel tersebut disatukan kembali dengan menggunakan operator add untuk dijadikan mask pada citra frame yang sedang berjalan agar dihasilkan citra foreground. Gambar 3.5 berikut merupakan contoh hasil dari proses *background subtraction* gambar sebelah kiri adalah citra sebelum proses *background subtraction* sedangkan gambar sebelah kanan merupakan hasil dari *background subtraction*.



Gambar 3.5 Contoh *Background Subtraction*

3.4.1.3 Skin Color Extraction

Setelah citra foreground dihasilkan maka citra tersebut siap diproses pada tahap selanjutnya yaitu tahap *skin color extraction*. Pada tahap ini citra akan di seleksi berdasarkan warna kulit untuk menghasilkan citra biner. Citra di filter menggunakan fungsi *InRange* pada format YCrCb, berdasarkan [11] nilai threshold yang dipakai adalah sebagai berikut :

Min Y = 0; Max Y = 245;

Min Cr = 141; Max Cr = 185;

Min Cb = 80; Max Cb = 135;

Hasil seleksi dengan fungsi *InRange* tersebut kemudian dioperasikan dengan operasi *morfologi* dan *image smoothing* agar menghasilkan citra biner yang bebas dari noise. Contoh hasil dari operasi *skin color extraction* dapat dilihat pada gambar 3.6.



Gambar 3.6 Hasil Skin Color Extraction

3.4.1.4 Operasi Morfologi dan Image Smoothing

Operasi ini berfungsi untuk menghaluskan citra biner hasil dari operasi seperti pada tahap operasi skin extraction, background extraction, dan lain lain. Terkadang hasil dari threshold pada beberapa operasi sering mengandung noise berupa bintik bintik putih. Untuk menangani hal tersebut citra dari hasil operasi diioperasikan dengan menggunakan operasi morfologi yaitu operasi erosi (Erode) dan dilasi (Dilate) kemudian diikuti dengan operasi *image smoothing* yaitu operasi *Gaussian filter*. Berikut merupakan fungsi dari *Erode*, *Dilate*, dan *Gaussian filter* pada library EmguCV:

```
Image<Gray,byte> Erode(
     int iterations
     );
```

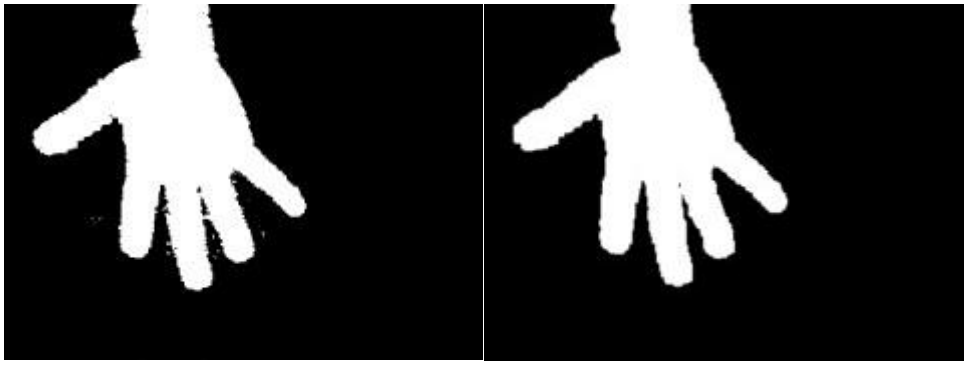
Fungsi Erode akan mengembalikan matriks citra berformat gray. Parameter iteration adalah interasi yang dilakukan dalam melakukan fungsi erode.

```
Image<Gray,byte> Dilate(
     int iterations
     );
```

Fungsi Dilate akan mengembalikan matriks citra berformat gray. Parameter iterations merupakan iterasi yang dilakukan dalam melakukang fungsi dilate.

```
public static void cvSmooth(
     IntPtr src,
     IntPtr dst,
     SMOOTH_TYPE type,
     int param1,
     int param2,
     double param3,
     double param4
     );
```

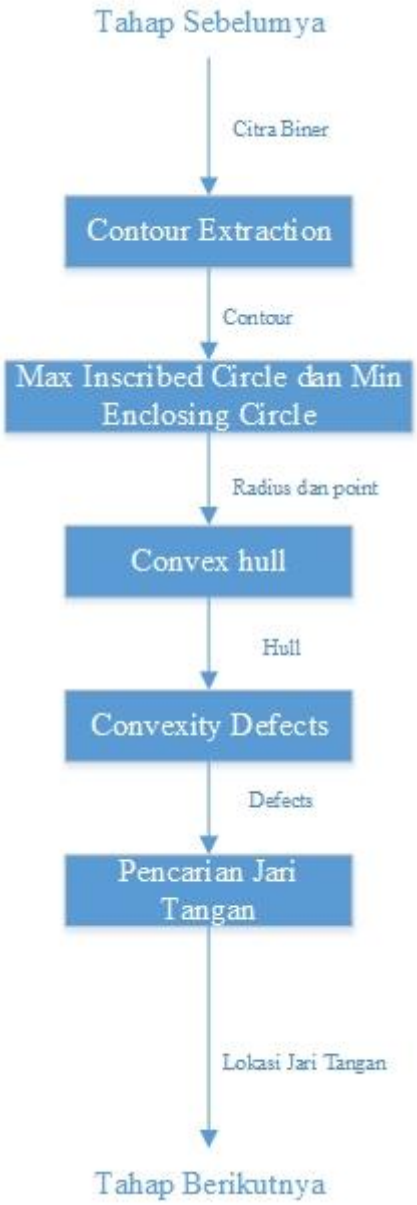
Gambar 3.7 merupakan contoh dari operasi morpologi dan *image smoothing.* Gambar pada bagian kiri adalah citra biner sebelum dilakukan operasi morpologi dan image smoothing dan gambar citra biner kanan merupakan citra yang telah dioperasikan operasi morfologi dan *image smoothing.*



Gambar 3.7 Contoh Hasil Morpologi dan Image Smoothing

3.4.2 Tahap Deteksi

Pada tahapan ini sistem akan mendeteksi letak kontur tangan dari hasil output tahap pertama yang berupa gambar biner. Tahapan ini akan menghasilkan nilai tengah dari telapak tangan, dan letak jari tangan untuk keperluan pada tahap ketiga yaitu tahap augmented. gambar 3.8 berikut menjelaskan proses-proses yang terjadi pada tahapan ini.



Gambar 3.8 Proses-Proses pada tahapan deteksi

3.4.2.1 Contour Extraction dan Polygon Approximation

Output dari tahap sebelumnya yang merupakan sebuah citra biner mengandung gumpalan berwarna putih dengan background berwarna hitam. Gumpalan putih tersebut merupakan sebuah kontur yang harus di ekstrak terlebih dahulu. pada citra hasil output dari tahap sebelumnya kemungkinan masih mengandung noise berupa bintik bintik putih kecil maka untuk memilih kontur yang berupa kontur tangan dipilihlah kontur dengan area terbesar agar kontur

kontur yang tercipta dari noise tidak terpilih. Kemdian kontur yang telah terpilih dioperasikan dengan operasi *Polygon Approximation* agar bentuk kontur lebih mudah untuk dianalisis pada tahap selanjutnya.

Untuk mengekstrak contour digunakan fungsi *FindContours* yang ada pada library EmguCV dan untuk operasi *Polygon Approximation* digunakan fungsi *ApproxPoly* yang juga terdapat pada library EmguCV yang akan dijabarkan sebagai berikut :

```
Contour<Point> FindContours(
     CHAIN_APPROX_METHOD method,
     RETR_TYPE type
     MemStorage storage
     );
```

Fungsi *FindContours* akan mengembalikan titik-titik yang membentuk sebuah contour yang direpresentasikan oleh kumpulan point pada list. Parameter method merupakan metode yang dipakai dalam pencarian contour, parameter type merupakan type yang dipakai dalam pencarian contour, sedangkan parameter storage adalah storage atau memori penyimpanan yang akan dipakai dalam pencarian contour.

```
Contour<Point> ApproxPoly(
     double accuracy,
     MemStorage storage
     );
```

Fungsi *ApproxPoly* akan mengembalikan titik-titik dalam contour yang telah dihaluskan agar analisi lebih mudah. Parameter accuracy merupakan nilai akurasi yang akan digunakan dalam proses ini, sedangkan parameter storage merupakan strage atau memori penyimpanan yang akan dipakai dalam pencarian contour.

3.4.2.2 Pencarian *Max Inscribed Circle* dan *Min Enclosing Cicle*

Max inscribed circle merupakan lingkaran terbesar yang dapat dibentuk pada sebuah kontur tampa melewati batas kontur. *Max inscribed circle*

merepresentasikan nilai tengah dari telapak tangan. Nilai ini dapat dicari dengan mencari jarak terdekat dari setiap titik yang ada di dalam kontur dengan sisi-sisi pembatas kontur, nilai terbesar dari jarak tersebut merupakan radius dari *Max inscribed circle* dan titik tersebut merupaka titik tengah dari telapak tangan.

Dalam implementasinya pencarian *Max inscribed circle* dapat memakan waktu yang lama sehingga pencarian harus diperkecil dengan melakukan pembatasan pencarian pada area tengah *bounding rectangle* dari contour, dan pencariannya pun dilakukan setiap N titik. Nilai N yang dipakai pada penelitian ini adalah 4.

Untuk pencarian jarak terdekat titik didalam kontur dengan sisi-sisi kelilingnya digunakan fungsi *cvPointPolygonTest*, berikut merupakan fungsi *cvPointPolygonTest* yang terdapat dalam library EmguCV*:*

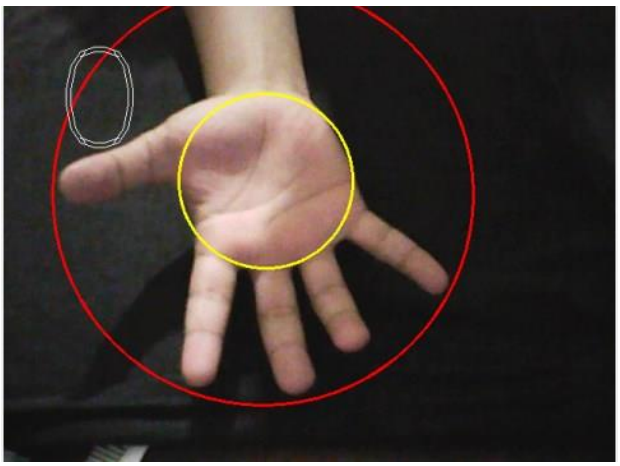
```
double cvPointPolygonTest(
     IntPtr contour,
     PointF pt,
     bool measureDist
     );
```

Fungsi *cvPointPolygonTest* akan mengembalikan nilai jarang terdekat dari suatu titik ke kelilikng sebuah contour yang bertipe data double. Parameter yang dipakan dalam fungsi ini adalah parameter contour yang merupakan parameter yang mewakili contour yang akan diukur jaraknya dengan titik yang dimaksud, parameter Pt adalah point atau ttitik yang akan diukur jaraknya ke keliling contour, sedangkan parameter measureDist merupakan nilai Boolean yang apabila nilainya true fungsi akan mencari nilai terdekat dari titik ke keliling contour.

Min enclosing circle merupakan lingkaran terkecil yang dapat dibentuk dengan mengelilingi kontur. Untuk mencari nilai radius dan titik dari *min enclosing circle* digunakan fungsi *MinEnclosingCircle*, berikut merupakan fungsi *MinEnclosingCircle* :

```
CircleF MinEnclosingCircle(
     PointF[] points
     );
```

Nilai *min enclosing circle* dan *max inscribed cicle* akan dipakai sebagai acuan untuk mencari titik-titik dari jari jari tangan pada analisis berikutnya. Gambar 3.9 berikut menunjukan contoh *Max inscribed circle dan min enclosing circle*, lingkaran merah merupakan *min enclosing circle* sedangkan lingkaran berwarna kuning merupakan *max inscribed circle;*



Gambar 3.9 Contoh *Max inscribed circle* dan *min enclosing circle*

3.4.2.3 Convex hull dan convexity defects extraction

Pada tahapan ini akan dicari *convex hull* dan *convexity defects* dari kontur tangan yang telah ditemukan pada tahap sebelumnya untuk analisis pada tahap berikutnya. *Convex hull* merupakan garis-garis lurus yang mengelilingi kontur sedangkan *convexity defets* merupakan titik-titik yang merepresentasikan sebuah lekukan pada setiap vertex di *convex hull*, *convexity defect* terdiri atas *start point, end point, depth point,* dan *depth length.*

Untuk mencari *convex hull* digunakan fungsi *GetConvexHull* sedangkan untuk mencari *convexity defects* digunakan fungsi *GetConvexityDefacts*, berikut merupakan fungsi dari *GetConvexHull* dan *GetConvexityDefacts* pada library EmguCV:

```
Seq<Point> GetConvexHull(
     ORIENTATION orientation
     );
```

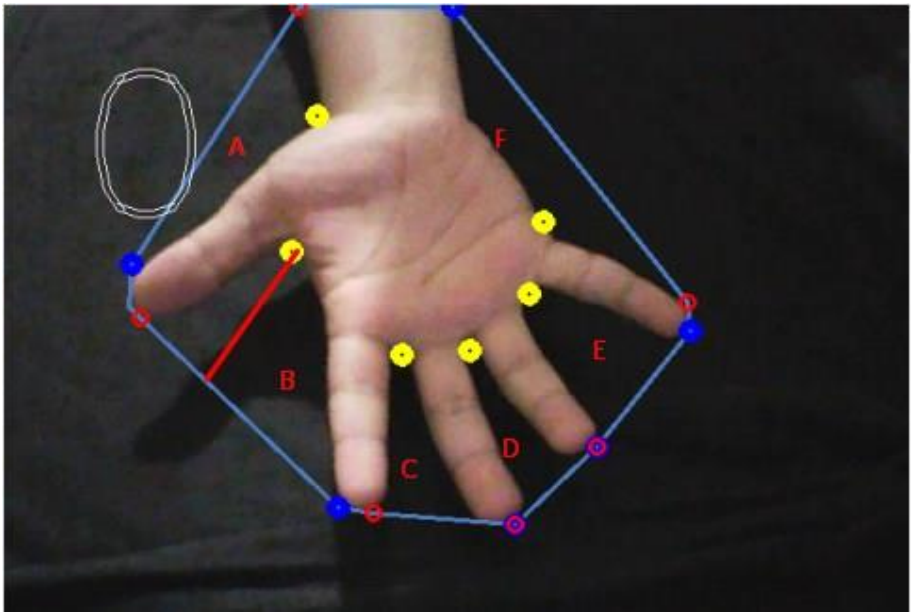
Fungsi GetConvexHull akan mengembalikan seqeuent dari point-point yang mewakili titik-titik yang membentuk hull pada contour. Parameter

orientation merupakan parameter bagaimana convex hull dicari searah jarum jam atau berlawanan arah jarum jam.

```
Seq<MCvConvexityDefect> GetConvexityDefacts(
      MemStorage storage,
      Orientation orientation
      );
```

Fungsi GetConvexityDefacts akan mengembalikan runtunan atau sequen dari defect-defect yang ditemukan setiap defect terdiri dari start point, end point, dan depth point yang merepresentasikan lekukan pada convex hull. Parameter yang dipakai dalam fungsi GetconvexityDefacts adalah parameter storage yaitu tempat penyimpanan atau memori yang dipakai dalam pencarian defect-defect, dan parameter orientation yaitu parameter bagaimana cara mendapatkan defect searah atau berlawanan arah jarum jam.

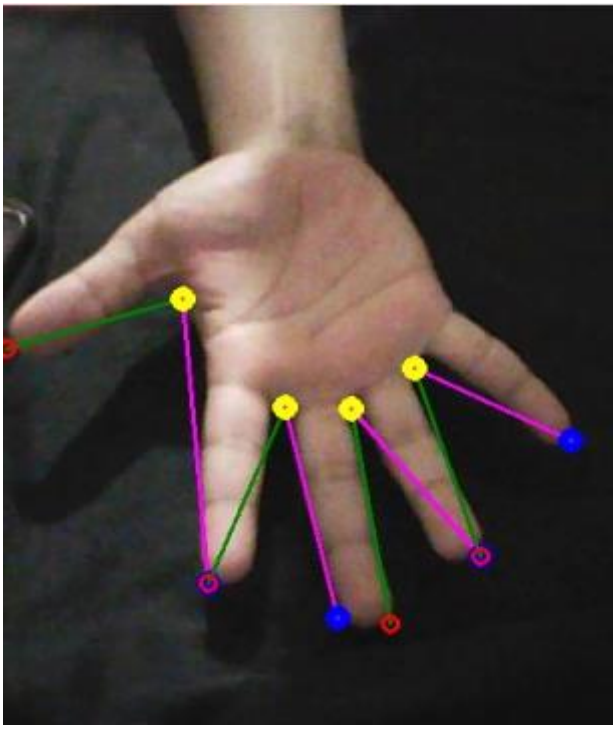
Gambar 3.10 merupakan contoh dari *convex hull dan convexity defects.*Garis berwarna ungu yang mengelilingi tangan merupakan *convex hull* sedangkan area area yang ditandai dengan huruf A,B,C,D,E dan F merupakan area yang dibentuk dari *convexity defects* dengan linkaran merah sebagai start point, lingkaran biru sebagai end point, dan lingkaran kuning sebagai depth point. Sedangkan contoh dari depth length dapat dilihat pada garis merah pada area B.



Gambar 3.10 Contoh *Convex Hull* dan *Convexity defects*

3.4.2.4 Pencarian Lokasi Jari Tangan

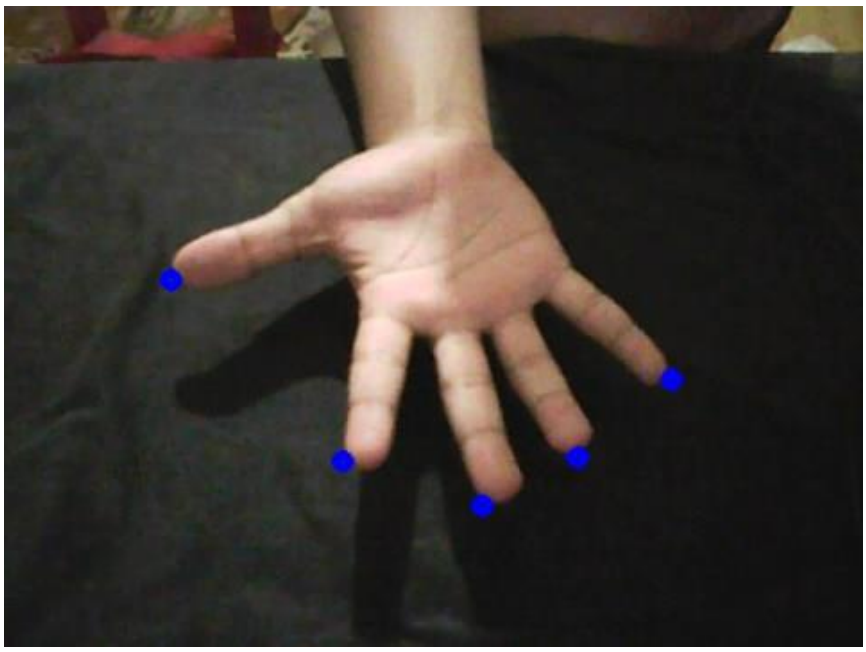
Tahapan ini adalah tahapan dimana posisi jari jari tangan ditentukan dengan menganalisis hasil convexity defect extraxtion dari tahap sebelumnya. Nilai- nilai dari convexity defect yang telah di dapatkan kemudian akan dipilih sesuai dengan karakteristik jari tangan yaitu sudut yang dibentuk antara start point dan end point kurang dari 90 derajat dan juga kedalaman dari defect harus kurang dari radius min enclosing circle dan juga lebih dari radius max inscribed circle. Jika ada titik yang memenuhi syarat tersebut maka untuk nilai start point dan end point pada defect tersebut akan di masukkan kedalam list. Gambar 3.11 merupakan contoh dari hasil pemilihan defect dengan menggunakan syarat diatas.



Gambar 3.11 Hasil Pemilihan Defect

Kemudian dari list yang telah dimasukkan beberapa titik dari hasil pemilihan defect titik-titik tersebut kemudian dikurangi dengan cara menghapus beberapa titik yang saling berdekatan. Hasil dari pengurangan list ini merupakan titik-titik dari jari jari tangan. Gambar 3.12 merupakan contoh dari posisi jari tangan hasil dari pengurangan titik titik yang saling berdekatan.

Hasil pencarian jari-jari tangan yang didapatkan dari proses sebelumnya masih belum optimal maka hasil tersebut perlu dioptimalkan lagi dengan mencari nilai optimal dari *k-curvature* yang sudutnya kurang dari 60 derajat. Proses ini akan memeriksa setiap point pada list jari untuk setiap pointnya akan diperiksa sebanyak n point ke kiri dan n point ke kanan dari kontur yang telah ditemukan untuk setiap point tersebut akan di periksa nilai *k-curvature*nya yaitu nilai sudut yang dibentuk dari titik yang dimaksud dengan titik-titik sejauh k ke kiri dan k ke kanan dari kontur, titik yang memberikan nilai terbesar dan kurang dari 60 derajat merupakan pengganti dari titik jari sebelumnya. Nilai jari yang telah dioptimalkan ini kemudian siap untuk diproses pada tahap selanjutnya yaitu tahap *augmented*.



Gambar 3.12 Hasil Pengurangan Titik

3.4.3 Tahap Augmented

Tahapan ini merupakan tahapan terakhir dalam algoritma dalam tahapan ini aplikasi akan melihat apakah kelima jari tangan berhasil ditemukan atau tidak, apabila jumlah jari tangan yang berhasil dideteksi berjumlah sama dengan lima maka aplikasi akan memunculkan objek 3 dimensi berdasarkan lokasi dan juga rotasi yang dibentuk dari jari tangan. Proses-proses yang ada dalam tahap ini dapat dilihat pada gambar 3.13.



Gambar 3.13 Tahapan Augmented

3.4.3.1 Mengestimasi Rotasi dan Lokasi Tangan

Pada tahapan ini aplikasi akan mengestimasi rotasi dan juga lokasi dari tangan berdasarkan lokasi jari-jari tangan dan telapak tangan dari proses sebelumnya. Pada tahapan ini lokasi jari-jari tangan yang merupakan bidang 2 dimensi akan diestimasi ke bidang 3 dimensi dengan acuan parameter intrinsic dari kamera yang telah didapatkan dari proses kalibrasi kamera. Hasil estimasi bidang 3 dimensi ini akan dibandingkan dengan objek 3 dimensi berupa objek tangan yang telah diinisialisasi untuk menghasilkan vector rotasi dan juga translasi, matriks rotasi ini merupakan matriks yang akan dipakai pada tahap berikutnya yaitu tahap memunculkan objek 3 dimensi. Pada tahap ini untuk mendapatkan matriks rotasi dan translasi digunakan fugnsi *solvePNP* pada *library* EmguCV, berikut merupakan fungsi *solvePNP* :

```
bool SolvePnP(
     IInputArray objectPoints,
```

```
    IInputArray imagePoints,
    IInputArray intrinsicMatrix,
    IInputArray distortionCoeffs,
    IOutputArray rotationVector,
    IOutputArray translationVector,
    bool useExtrinsicGuess = false,
    SolvePnpMethod flags = SolvePnpMethod.Iterative
)
```

Fungsi SolvePnP adalah fungsi untuk mencari vector translasi dan rotasi suatu objek 3 dimensi. Parameter objectPoints merupakan parameter point-point pada bidang 3 dimensi pada kasus ini adalah point-point jari-jari tangan pada bidang 3 dimensi atau model tangan yang telah diinisialisasi sebelumnya. Parameter imagePoints merupakan parameter point-point dari jari tangan dalam bidang 2 dimensi pada kasus ini adalah loksi jari tangan yang didapatkan dari proses sebelumnya. Parameter intrinsicMatrix merupakan matriks intrinsic dari kamera yang didaptkan dari proses kalibrasi kamera. Parameter distortion adalah parameter distorsi kamera yang didapatkan dari proses kalibrasi juga. Parameter rotationVector merupakan parameter keluaran parameter ini akan mengeluarkan nilai dari vector rotasi yang akan digunakan dalam proses memunculkan objek 3 dimensi pada tahap berikutnya. Parameter translationVector merupakan parameter keluaran yang mengularkan nilai dari vector translasi.

3.4.3.2 Memunculkan Objek 3D

Tahapan ini merupakan tahapan terakhir pada tahapan augmented dan juga pada algoritma. Pada tahapan ini objek 3 dimensi yang dimaksud yaitu objek 3 dimensi berupa arsitektur bangunan akan munculkan di atas telapak tangan. Pada tahap ini aplikasi akan memanggil objek 3 dimensi yang telah didefinisikan ke layar dengan lokasi berdasarkan letak dari telapak tangan yang telah dideteksi kemudian objek tersebut dirotasikan berdasarkan nilai vector rotasi yang telah di temukan dalam proses sebelumnya sehingga objek 3 dimensi akan mengikuti

pergerakan tangan dan tetap berada di atas telapak tangan dengan berdasarkan sudut tangan yang dibentuk.

3.5 Analisis Kebutuhan Nonfungsional

Kebutuhan non fungsional yang dibutuhkan untuk membangun sistem ini mencakup dua hal yaitu kebutuhan perangkat keras dan perangkat lunak

3.5.1 Analisis Kebutuhan Perangkat Keras

Adapun kebutuhan perangkat keras yang diperlukan untuk menjalankan sistem ini adalah :

1. Processor minimal Pentium 4 3.0 Ghz
2. Memrory 384 MB atau lebih
3. *Web-cam* minimal 2MP
4. Hardisk minimal 850 MB untuk .NET

3.5.2 Analisis Kebutuhan Perangkat Lunak

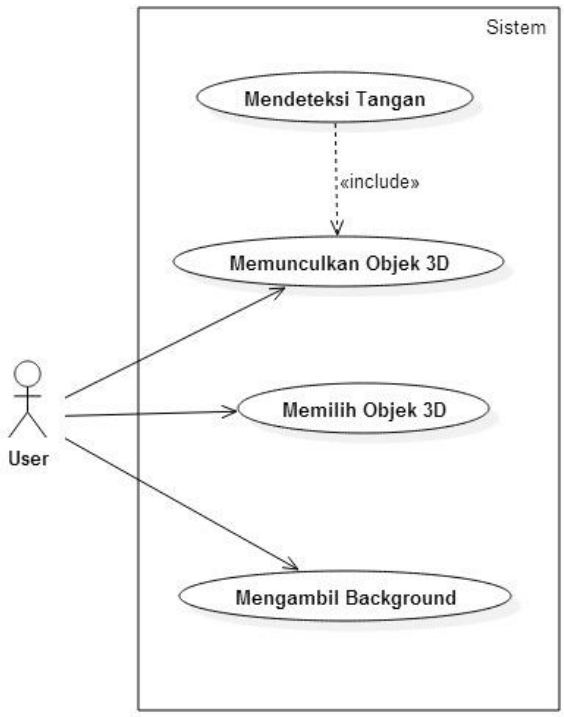
Perangkat lunak yang dibutuhkan untuk menjalankan software *Augmented Reality* ini adalah :

1. Sistem Operasi Windows 7 atau lebih
2. .NET 4 Framework

3.6 Analisis Kebutuhan Fungsional

3.6.1 Diagram Use Case

Diagram Use case merupakan model untuk mendeskripsikan hubungan-hubungan yang terjadi antar aktor dengan aktivitas yang terdapat pada sistem. Pada sistem ini terdapat satu aktor yaitu user. Untuk diagram use case pada sistem ini dapat dilihat pada gambar 3.14.



Gambar 3.14 Diagram Use Case

3.6.2 Definisi Aktor

Aktor adalah pihak mana saja yang mengkases Use Case. Daftar aktor yang terlibat beserta deskripsi dapat dilahat pada tabel 3.1.

Tabel 3.1 Daftar Aktor

No	Nama	Deskripsi Tugas
1.	User	Mengarahkan tangan pada kamera, memilih objek 3D yang akan ditampilkan

3.6.3 Definisi Use Case

Diagram use case merupakan konstruksi untuk mendeskripsikan hubungan-hubungan yang terjadi antar aktor dengan aktivits yang terdapat pada sistem. Tujuan pemodelan use case diantaranya adalah mendefinisikan kebutuhan fungsional dan operasi sistem. Definisi dari setiap use case yang ada pada sistem dapat dilihat pada tabel 3.2.

Tabel 3.2 Deskripsi Use Case

Nomor	Nama Use Case	Deskripsi
UC-01	Memunculkan Objek 3D	Fungsionalitas untuk memunculkan objek 3D pada bidang perspektif
UC-02	Mendeteksi Tangan	Fungsionalitas untuk mendeteksi dan mentracking tangan
UC-03	Mengambil Background	Fungsionalitas untuk mengambil background pada lingkungan sekitar
UC-04	Memilih Objek 3D	Fungsionalitas untuk memilih objek 3D yang akan dimunculkan

3.6.4 Skenario Use Case

Skenario use case menjelaskan urutan langkah-langkah dalam proses yang dilakukan sebuah Use Case. Adapun skenario use case dari sistem yang dibangun adalah sebagai berikut :

3.6.4.1 Skenario Use Case Mendeteksi Tangan

Skenario use case mendeteksi tangan dapat dilihat pada tabel 3.3.

Tabel 3.3 Skenario Use Case Mendeteksi Tangan

Use Case Name	Mendeteksi Tangan	
Goal In Context	Sistem dapat mendeteksi tangan dan mengikuti gerakannya	
Precondition	User berada pada form utama AR	
Description	Fungsionalitas untuk mendeteksi dan mentracking tangan	
Related Use Case	-	
Successful End Condition	Sistem berhasil mendeteksi tangan	
Failed End Condition	-	
Actors	User	
Trigger	*Webcam* mengambil citra dari lingkungan	
Main Flow	**Step**	**Step**
	1.	Sistem mengambil citra dari *webcam*
	2.	Sistem mengubah format citra
	3.	Sistem mengekstrak *foreground*
	4.	Sistem mendeteksi warna kulit
	5.	Sistem mencari kontur tangan
	6.	Sistem mencari letak telapak tangan
	7.	Sistem mencari letak jari-jari tangan
	8.	Sistem berhasil mendeteksi tangan

3.6.4.2 Skenario Use Case Memunculkan Objek 3D

Skenario use case memunculkan objek 3d dapat dilihat pada tabel 3.4.

Tabel 3.4 Skenario Use Case Memunculkan Objek 3D

Use Case Name	Memunculkan Objek 3D
Goal In Context	Sistem memunculkan objek 3d pad bidang perspektif
Precondition	*User* menekan tombol Start
Description	Fungsionalitas untuk memunculkan objek 3D pada bidang perspektif

Related Use Case	Mendeteksi Tangan	
Successful End Condition	Sistem berhasil menampilkan objek 3D pada bidang perspektif	
Failed End Condition	File 3D tidak valid,Tangan tidak terdeteksi	
Actors	-	
Trigger	User menekan tombol start	
Main Flow	**Step**	**Step**
	1.	Sistem meload objek tiga dimensi
	2.	Menampilkan form utama AR
	3.	User mengarahkan tangan ke kamera
	4.	*Include::*Mendeteksi Tangan
	5.	Sistem mengestimasi lokasi objek 3D pada bidang perspektif
	6.	Sistem memunculkan objek 3D
Extensions	**Step**	**Step**
	1.1	Objek 3D tidak valid
	4.1	Tangan tidak terdeteksi

3.6.4.3 Skenario Use Case Mengambil Background

Skenario use case mengambil background dapat dilihat pada tabel 3.5.

Tabel 3.5 Skenario Use Case Mendeteksi Telapak Tangan

Use Case Name	Mengambil background	
Goal In Context	Background berhasil diganti dengan background yang diambil	
Precondition	-	
Description	Fungsionalitas untuk mengambil background pada lingkungan sekitar	
Related Use Case	-	
Successful End Condition	Sistem berhasil mengganti background	
Failed End Condition	-	
Actors	User	
Trigger	User menekan tombol take background	
Main Flow	**Step**	**Step**
	1.	Sistem mengambil *background*
	2.	Sistem menyimpan *background*

3.6.4.4 Skenario Use Case Memilih Objek 3D

Skenario use case mengambil background dapat dilihat pada tabel 3.6.

Tabel 3.6 Skeneraio Use Case Memilih Objek 3D

Use Case Name	Memilih Objek 3D
Goal In Context	Berhasil memilih objek 3D
Precondition	-
Description	Fungsionalitas untuk memilih objek 3D yang akan dimunculkan
Related Use Case	-
Successful End Condition	User berhasil memilih objek 3D
Failed End Condition	-
Actors	User
Trigger	User menekan tombol browse

Main Flow	Step	Step
	1.	Sistem menampilkan form pemilihan file
	2.	User memilih file
	3.	User menekan tombol OK

3.6.5 Activity Diagram

Activity diagram menggambarkan berbagai aliran aktivitas dalam sistem yang dirancang, bagaimana masing-masing fungsionalitas bekerja, dan bagaimana suatu fungsional berakhir. Activity Diagram memodelkan event-event yang terjadi pada use case. Adapun Activity Diagram dari aplikasi yang akan dibangun adalah sebagai berikut :

3.6.5.1 Activity Diagram Memunculkan Objek 3D

Activity Diagram Memunculkan Objek 3D dapat dilihat pada gambar 3.15.

3.6.5.2 Activity Diagram Mendeteksi Tangan

Activity diagram mendeteksi tangan dapat dilihat pada gambar 3.16.

3.6.5.3 Activity Diagram Mengambil Background

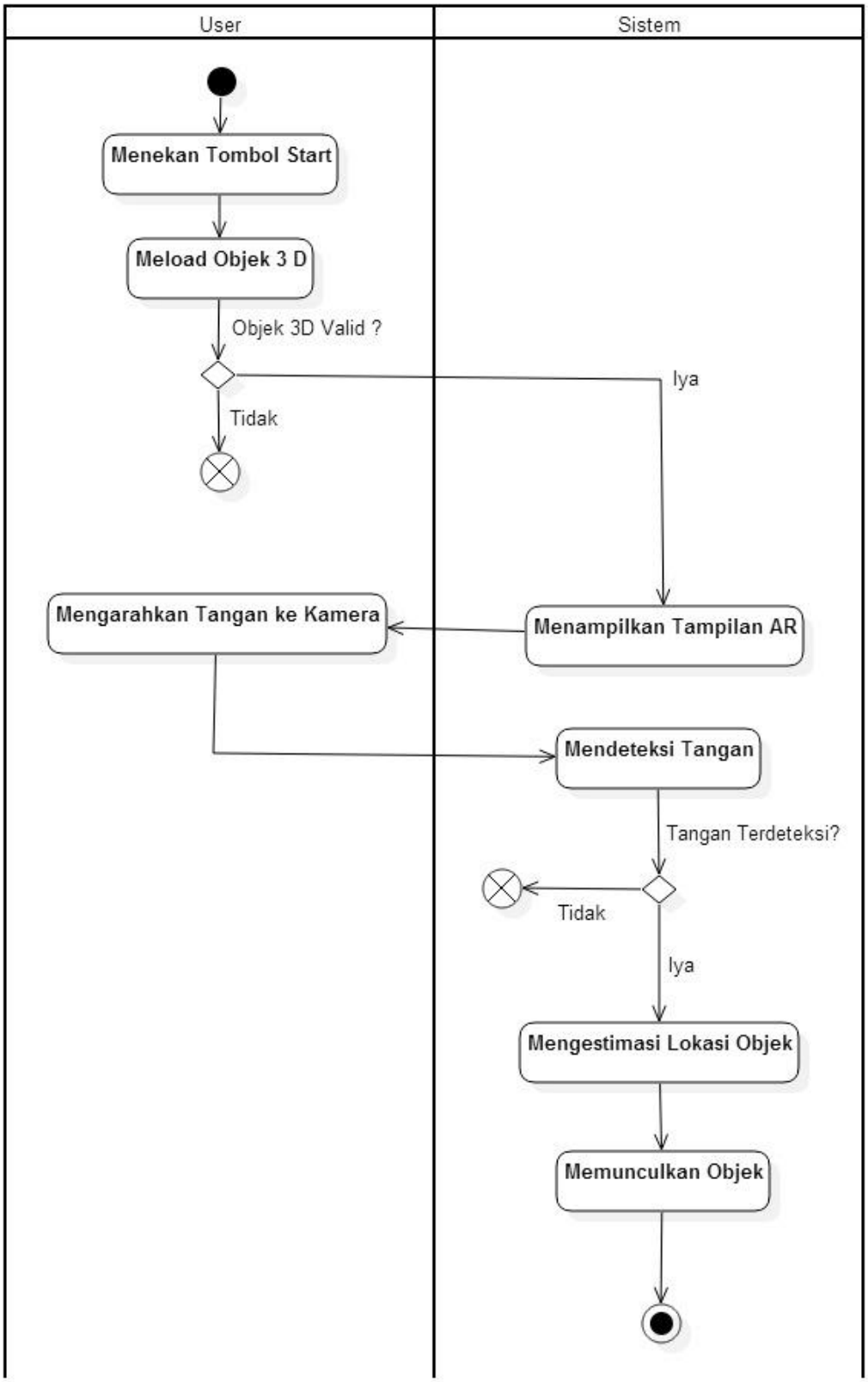
Activity diagram mengambil background dapat dilihat pada gambar 3.17

3.6.5.4 Activity Diagram Memilih Objek 3D

Activity diagram memilih objek 3D dapat dilihat pada gambar 3.18.

3.6.6 Class Diagram

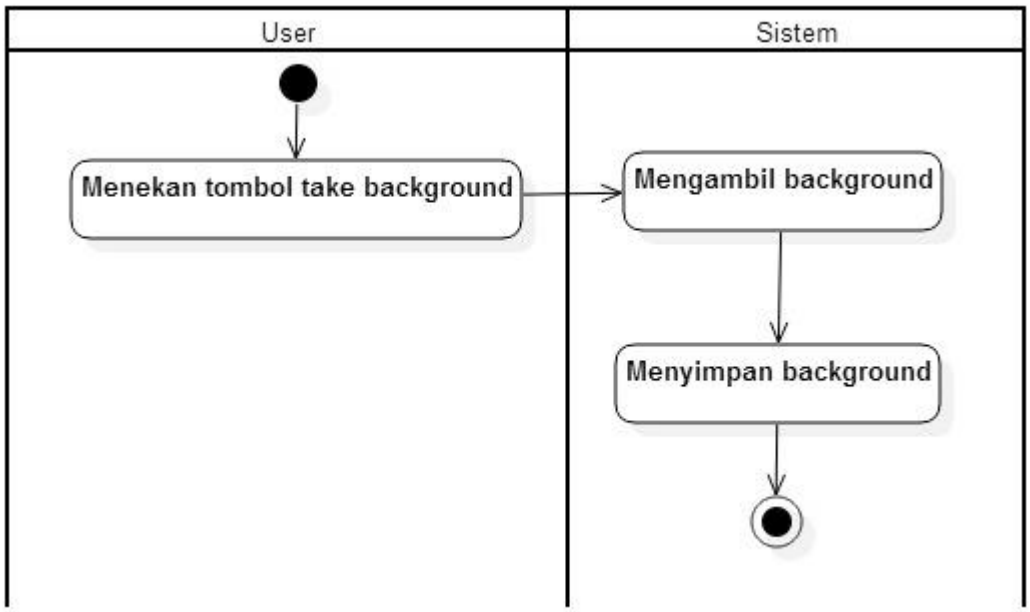
Class Diagram menggambarkan struktur dan hubungan antra objek-objek yang ada pada sistem. Struktur itu meliputi atribut-atribut dan metode-metode yang ada pada masing-masing kelas. Untuk class diagram pada penelitian ini dapat dilihat pada gambar 3.19.



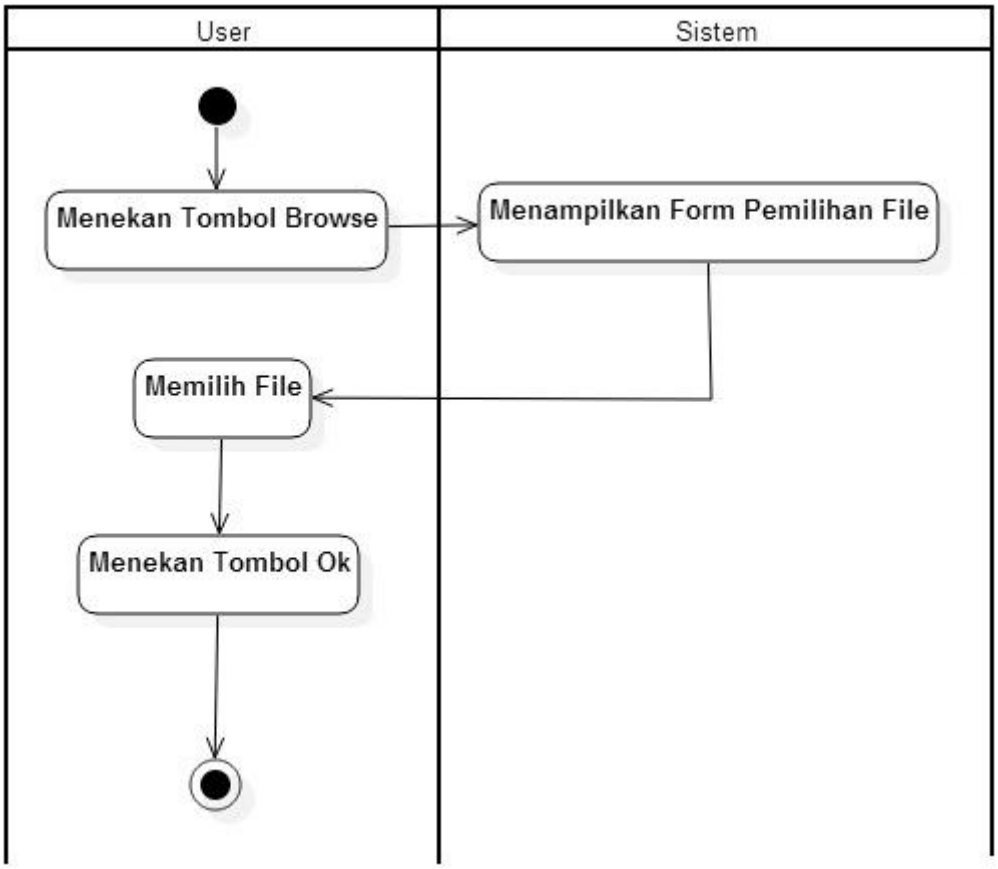
Gambar 3.15 Activity Diagram Memunculkan Objek 3D



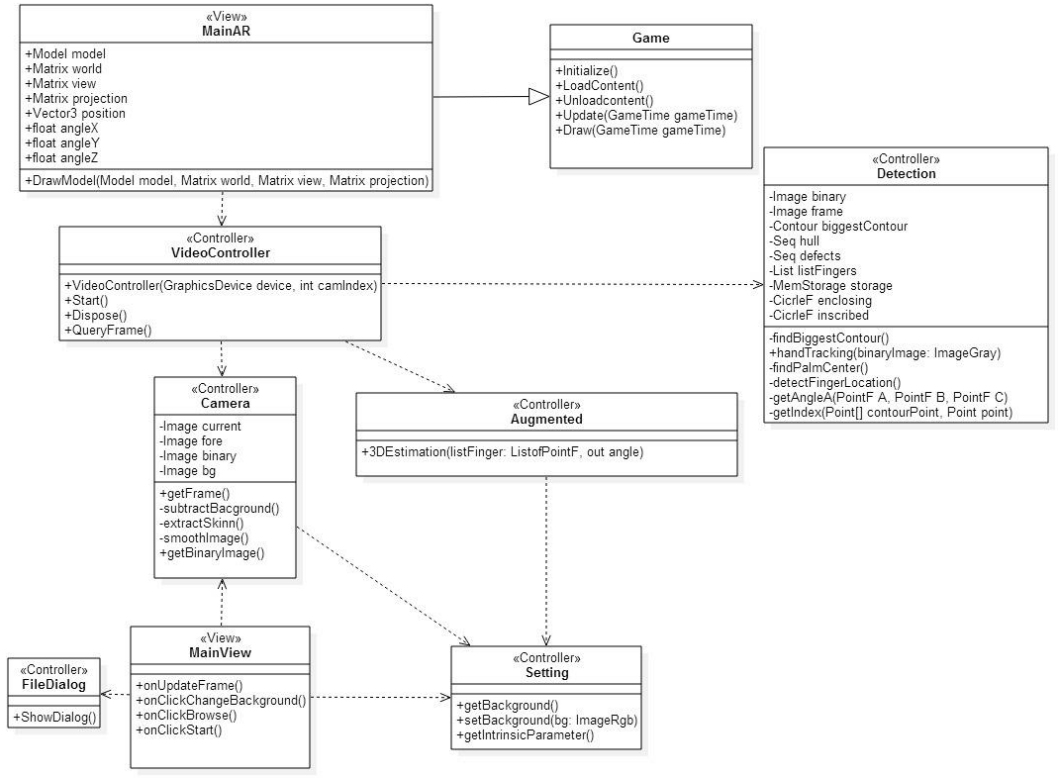
Gambar 3.16 Activity Diagram Mendeteksi Tangan



Gambar 3.17 Activity Diagaram Mengambil Background



Gambar 3.18 Activity Diagram Memilih Objek 3D



Gambar 3.19 Class Diagram

3.6.7 Sequence Diagram

Sequence diagram menggambarkan interaksi antara masing masing objek pada setiap use case dalam urutan waktu. Interaksi berupa pengiriman serangkaian data antar objek-objek yang saling berinteraksi. Adapun sequence diagram dari palikasi yang dibangun adalah sebagai berikut.

3.6.7.1 Sequence Diagram Memunculkan Objek 3D

Sequence diagram memunculkan objek 3D dapat dilhat pada gambar 3.20.

3.6.7.2 Sequence Diagram Mendeteksi Tangan

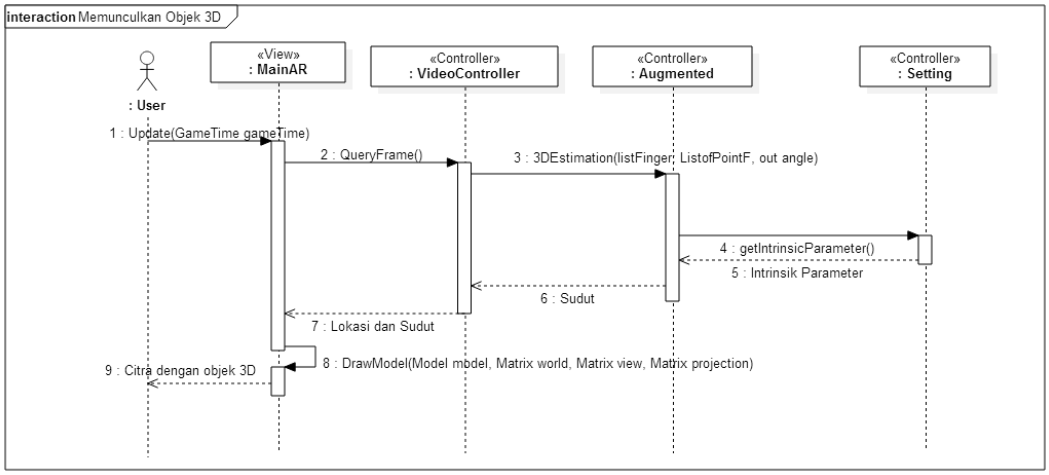
Sequence diagram mendeteksi tangan dapat dilihat pada gambar 3.21.

3.6.7.3 Sequence Diagram Mengambil Background

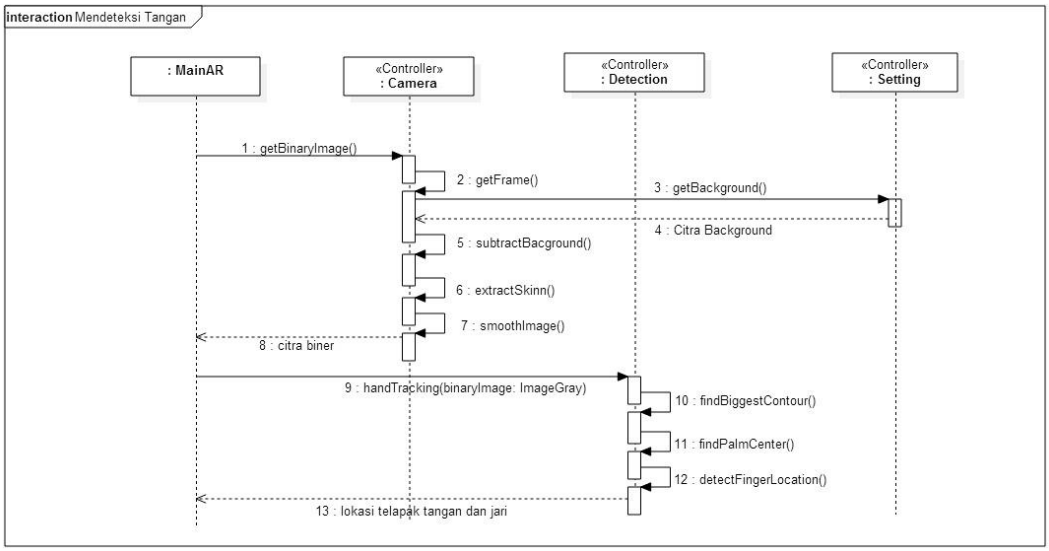
Sequence diagram mengambil background dapat dilihat pada gambar 3.22.

3.6.7.4 Sequence Diagram Memilih Objek 3D

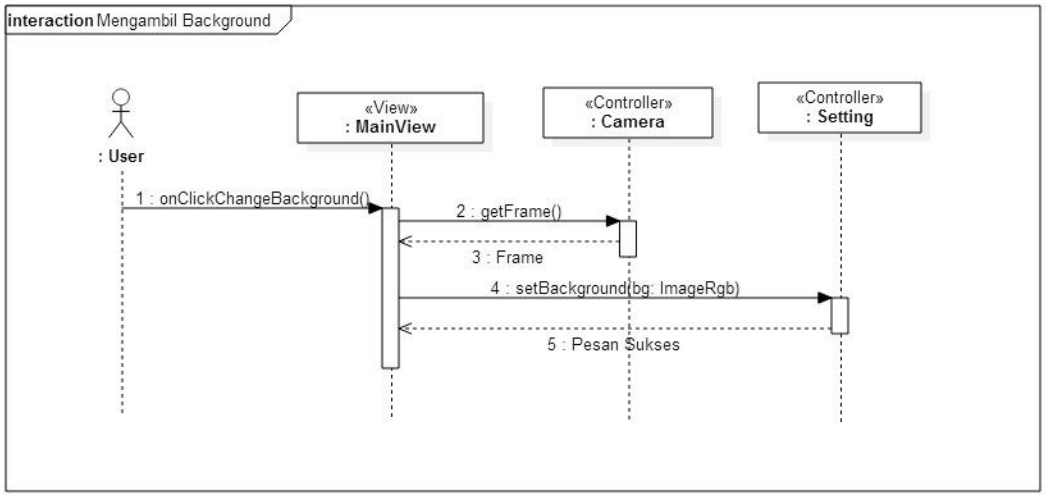
Sequence diagaram memilih objek 3D dapat dilihat pada gambar 3.23.



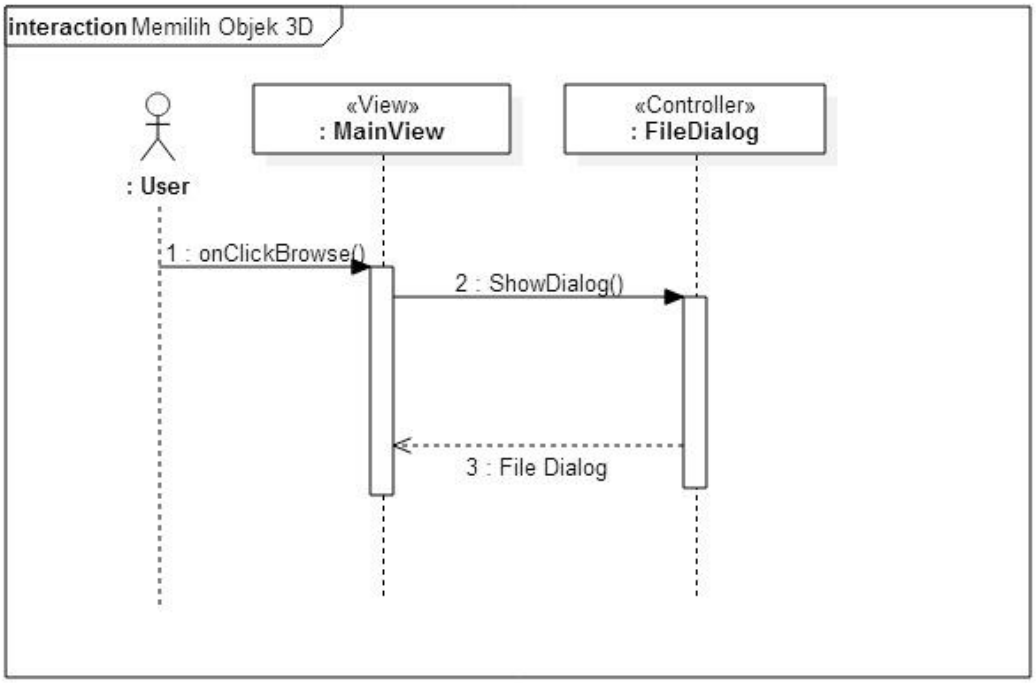
Gambar 3.20 Sequence Diagram Memunculkan Objek 3D



Gambar 3.21 Sequence Diagram Mendeteksi Tangan



Gambar 3.22 Sequence Diagram Mengambil Background



Gambar 3.23 Sequence Diagram Memilih Objek 3D

3.7 Perancangan Sistem

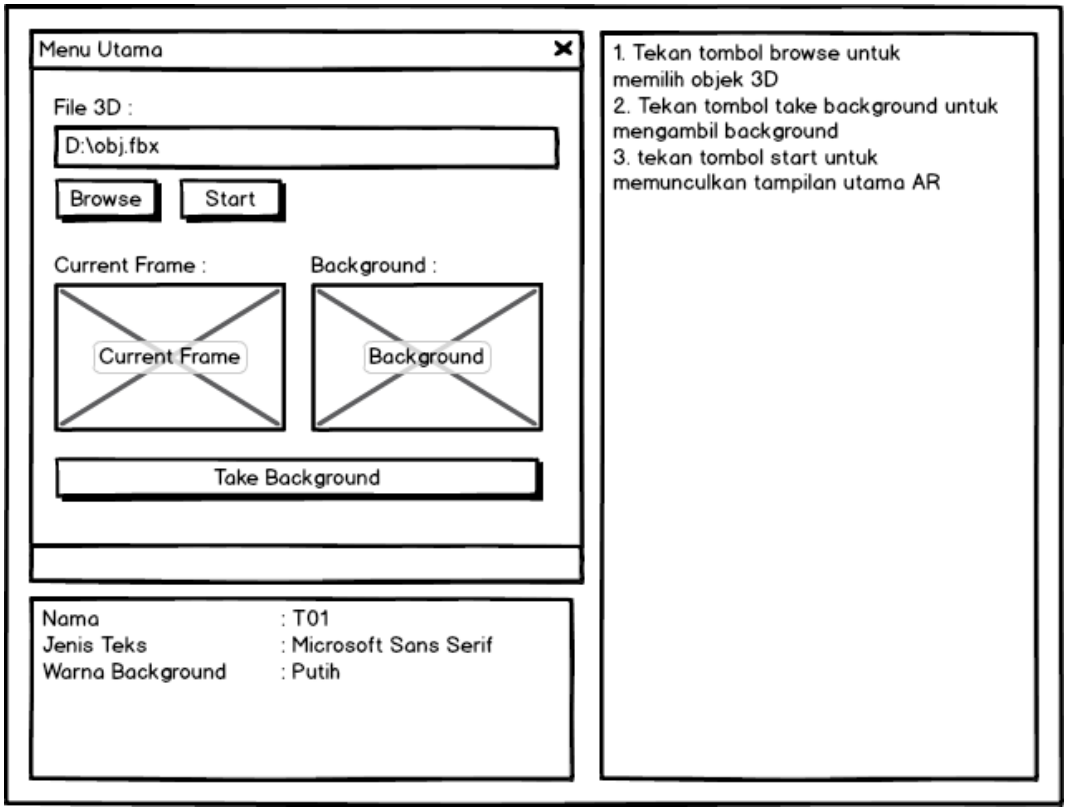
Perancangan sistem adalah penggambaran, perencanaan, dan pembuatan sketsa beberapa element yang terpisah ke dalam satu kesatuan yang utuh dan memiliki fungsi. Perancangan sistem pada aplikasi *augmented reality* ini meliputi perancangan antarmuka dan jaringan semantik.

3.7.1 Perancangan Antarmuka

Perancangan antarmuka merupakan suatu media komnukasi yang digunakan ketika *user* akan berinteraksi dengan sistem. Perancangan antarmuka pada aplikasi *augmented reality* ini adalah sebagai berikut :

3.7.1.1 Perancangan Antarmuka Tampilan Menu Utama

Perancangan antarmuka tampilan menu utama dapat dilihat pada Gambar 3.24.



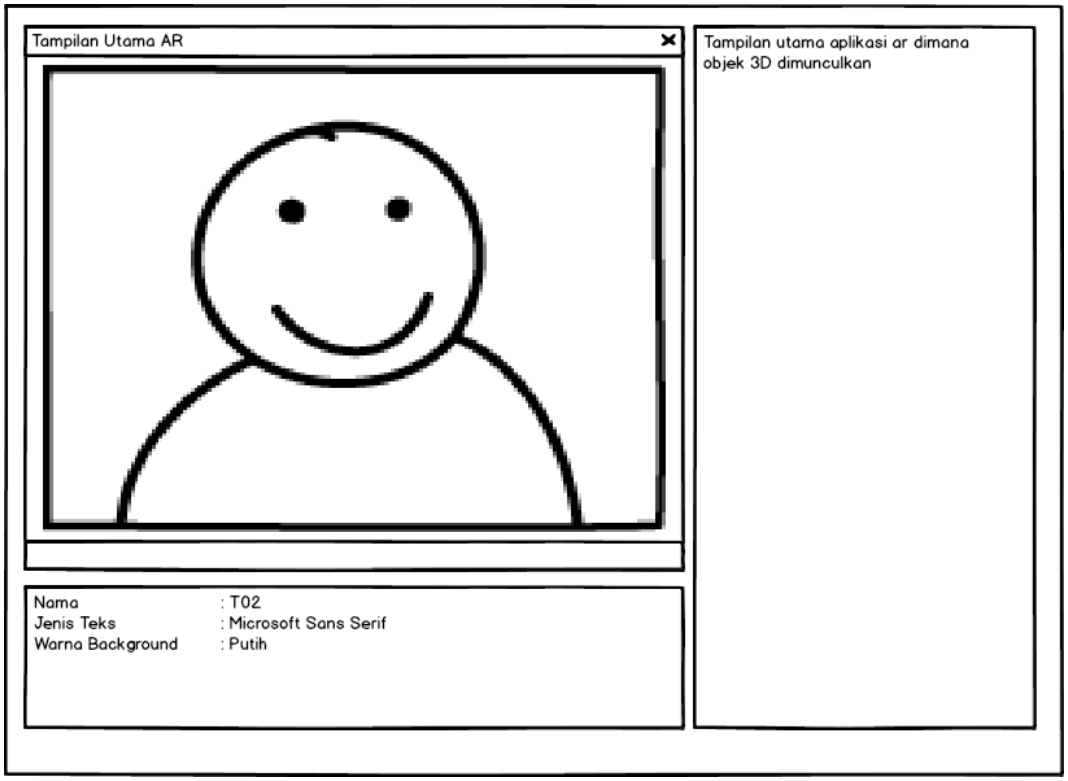
Gambar 3.24 Perancangan Antarmuka Tampilan Menu Utama

3.7.1.2 Perancangan Antarmuka Tampilan Utama AR

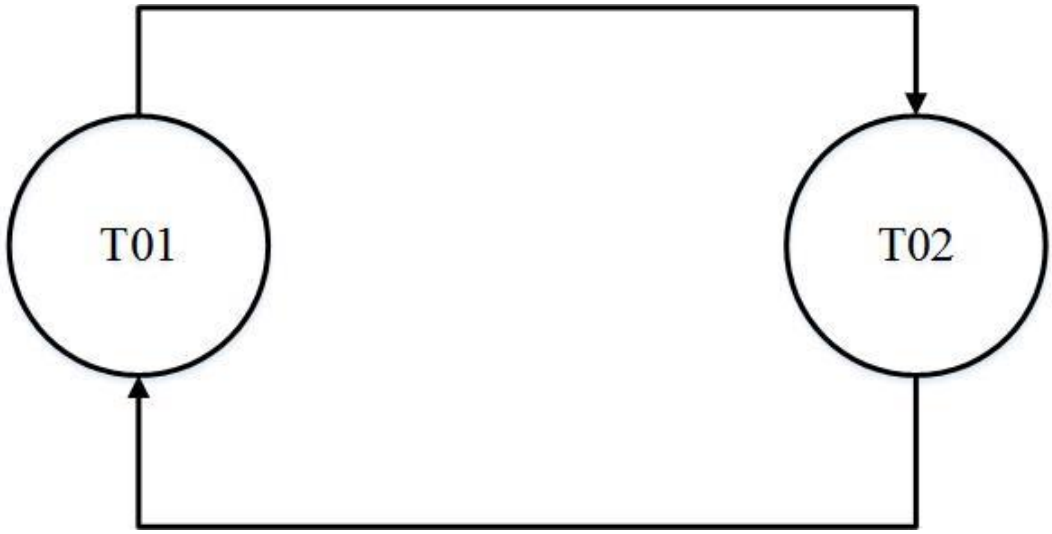
Perancangan antarmuka tampilan utama AR dapat dilihat pada gambar 3.25.

3.7.2 Jaringan Semantik

Jaringan semantik aplikasi *augmented reality* ini dapat dilihat pada gambar 3.26.



Gambar 3.25 Perancangan Antarmuka Tampilan Utama AR



Gambar 3.26 Jaringan Semantik

3.7.3 Perancangan Method

Perancangan komponen method merupakan perancangan yang dibuat setelah perancangan jaringan semantik dan antarmuka. Perancangan ini berfungsi

untuk mendeskripsikan method-method yang berada di dalam aplikasi yang dibangun. Adapun Method yang terdapat dalam aplikasi yang akan dibangun adalah sebagai berikut :

3.7.3.1 Perancangan Method *Hand Tracking*

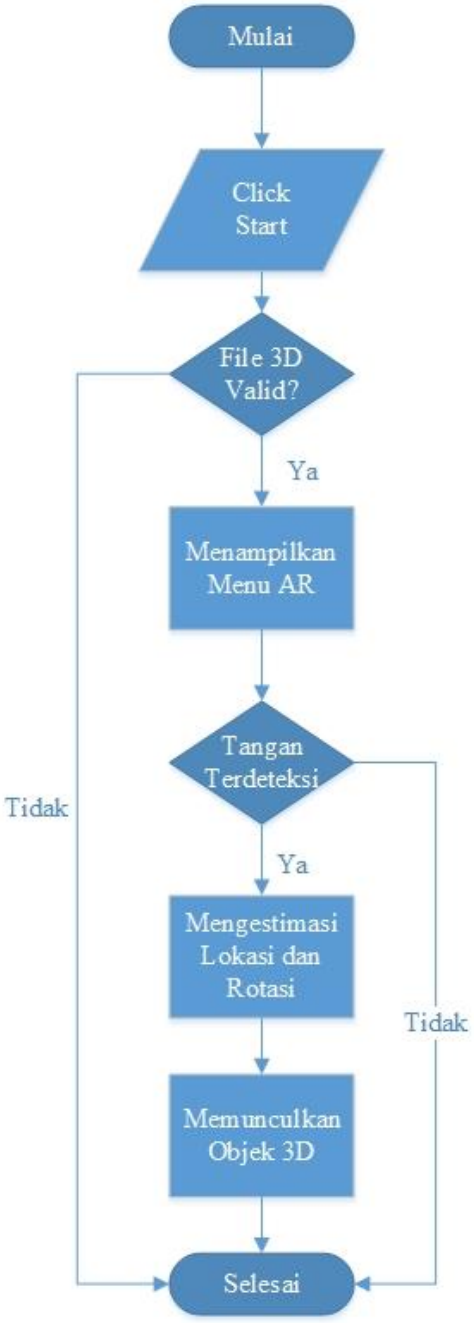
Perancangan Method *hand tracking* dapat dilihat pada gambar 3.27.

3.7.3.2 Perancangan Method Memunculkan Objek 3D

Perancangan Method Memunculkan Objek 3D dapat dilihat pada gambar 3.28.



Gambar 3.27 Method *Hand Tracking*



Gambar 3.28 Method Memunculkan Objek 3D