BAB 2
LANDASAN TEORI

2.1 Hand Gesture Recognition

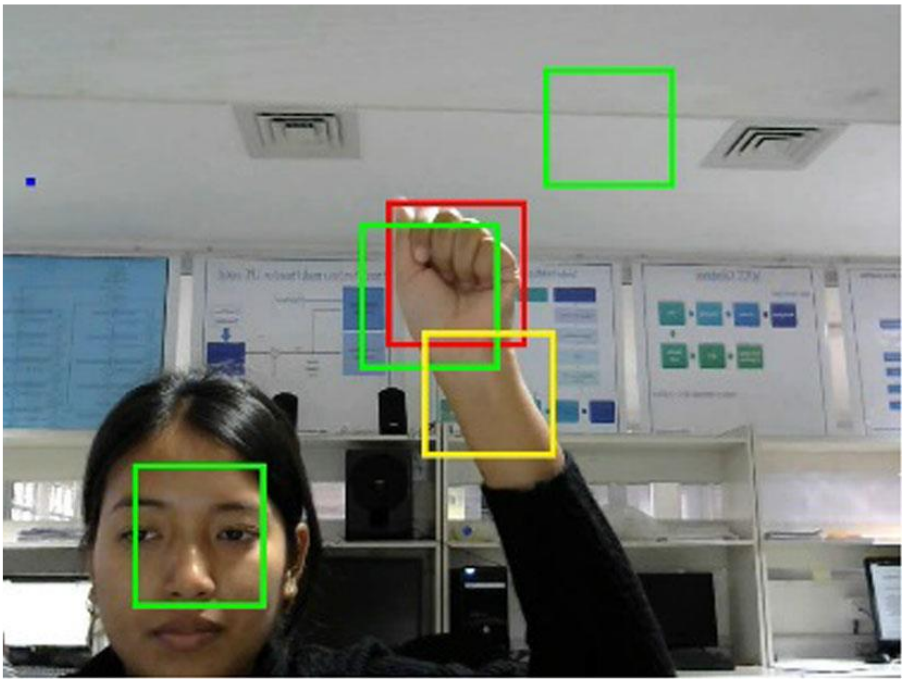
Hand gesture recognition, merupakan teknologi yang mampu membaca gerak tangan kemudian dirubah menjadi teks dan atau suara. *Gesture recognition* merupakan topik dalam *computer science* dan *language technology* yang bertujuan agar komputer bisa memahami gerakan manusia yang umumnya berasal dari tangan atau wajah [8].

2.1.1 Teknologi Pengenalan Gestur Tangan

Teknologi yang bisa mengenali gesture diantaranya: *vision based* (berbasis visi), *glove based* (menggunakan sarung tangan), dan *color markers* (menggunakan penanda warna).

1) *Vision Based*

Metode menggunakan *vision based* (berbasis visi) membutuhkan kamera untuk mengambil gambar yang akan diproses. *Vision based* termasuk metode yang sederhana namun tantangnnya cukup banyak, seperti faktor pencahayaan, warna objek yang dideteksi sama dengan warna di sekitar, dan tantangan dari background gambar yang akan dideteksi. Selain itu, dibutuhkan kecepatan, ketepatan serta efisiensi sistem pendeteksian. Contoh hasil pendeteksian menggunakan *vision based* dapat diamati pada gambar 2.1 Contoh Hasil Pendeteksian Dengan *Vision Based*.

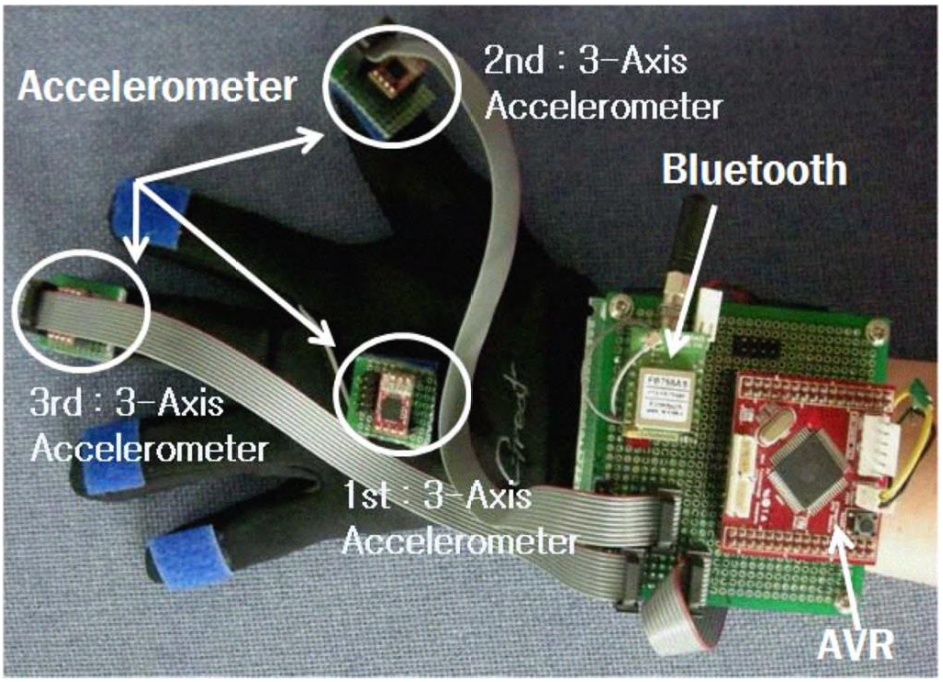


Gambar 2.1 Contoh Hasil Pendeteksian Dengan *Vision-based*

Sumber: Singha, J., Roy, A., & Laskar, R. H. (2018). [35]

2) *Glove Based*

Metode menggunakan *glove based* (sarung tangan) membutuhkan sensor untuk menangkap koordinat tangan dan perpindahannya. Keuntungan menggunakan metode ini dapat mempermudah penerimaan informasi seperti koordinat telapak tangan, jari, dan orientasi. Kekurangan metode ini memerlukan koneksi langsung antara pengguna dan sistem, serta biaya pembuatan alatnya cukup mahal. Contoh teknologi yang menggunakan *glove based* dapat diamati pada Gambar 2.2 Contoh *Glove Based.*

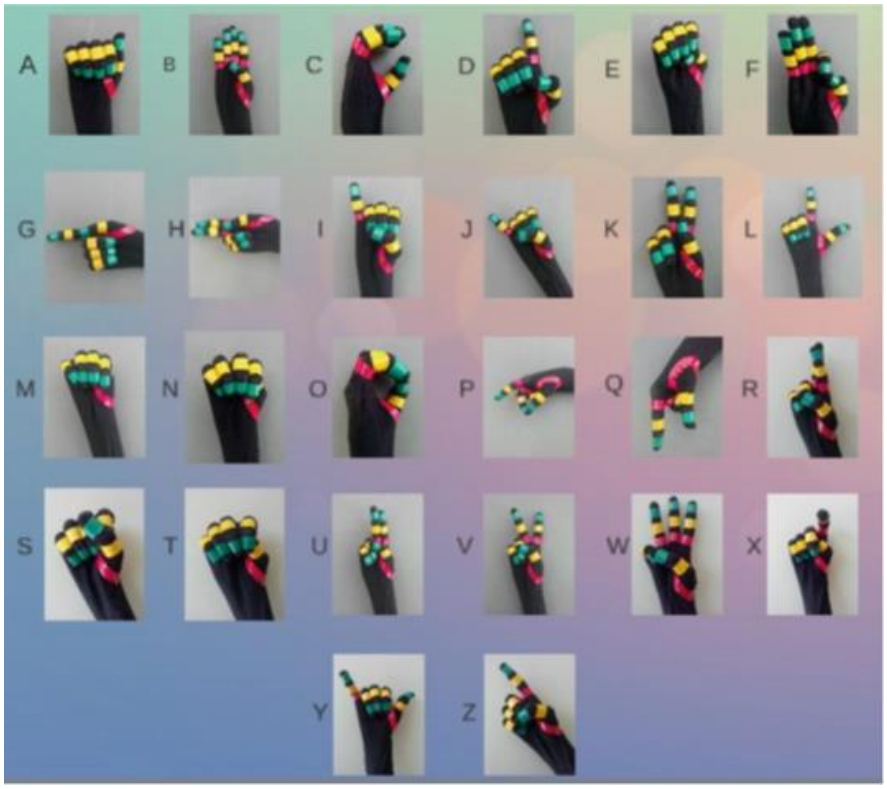


Gambar 2.2 Contoh *Glove Based*

Sumber : Kim, J. H., Thang, N. D., & Kim, T. S. (2009, July). [12]

3) *Color Marker Based*

Metode menggunakan *color marker* (penanda warna) membutuhkan sarung tangan yang diberikan beberapa warna untuk mengarahkan proses tracking tangan beserta lokasi telapak dan jari tangan. Metode ini juga menyediakan kemampuan untuk membentuk tangan dengan mengekstraksi ciri ciri geometris. Contoh teknologi yang menggunakan penanda warna dapat diamati pada Gambar 2.3 *American Sign Language* Menggunakan *Color Markers.*



Gambar 2.3 *American Sign Language* Menggunakan *Color Markers*

Sumber: Zaman, M., Rahman, S., Rafique, T., Ali, F., & Akram, M. U. (2016, November). [34]

2.1.2 Lingkungan Pengaplikasian Hand Gesture

Kemampuan komputer atau mesin dalam memahami gerakan tangan akan membuka banyak potensi dalam pengembangan aplikasi. Pengembangan aplikasi ini nantinya akan banyak membantu manusia dalam kehidupan sehari hari. Diantara pengaplikasiannya adalah:

1. Pengenalan bahasa isyarat. Media komunikasi antara tunarungu dengan sekitarnya yang mencakup beberapa kategori, yaitu *fingerspelling*, *isolation word*, *lexicon of words*, dan *continous sign*.
2. Robotika. Pergerakan lengan, badan, kaki dan bagian robotik lainnya yang dapat digerakkan dengan simulasi tindakan manusia.
3. *Game virtual reality*. Interaksi yang nyata antara pengguna game dengan lingkungan *virtual 3D* untuk mensimulasikan pergerakan pengguna

didalam game.

4. Interaksi pengguna (manusia) dengan komputer (HCI). HCI Contohnya seperti kontrol gerakan militer, bidang medis, manupulasi grafik, alat desain, anotasi atau pengeditan dokumen.

2.2 Bahasa Isyarat

Bahasa isyarat dalam KBBI merupakan bahasa yang tidak disampaikan dalam bentuk tulisan atau perkataan yang mengeluarkan suara. Bahasa isyarat memakai bahasa yang diciptakan oleh Teman Tuli berupa gerakan tangan dan terkadang dikombinasikan dengan bahasa non-verbal lainnya [20].

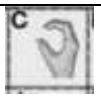
Terdapat dua bahasa isyarat di Indonesia, SIBI dan BISINDO. Sistem isyarat bahasa indonesia SIBI digunakan di sekolah-sekolah sebagai bahasa resmi. Sedangkan Bahasa Isyarat Indonesia atau BISINDO banyak digunakan sebagai media komunikasi antara Teman Tuli di kehidupan sehari-hari [9].

2.2.1 SIBI

SIBI atau sistem isyarat bahasa isyarat Indonesia merupakan bahasa yang pemerintah sahkan sebagai bahasa isyarat resmi di sekolah. Bahasa isyarat SIBI diciptakan pertama kali oleh Alm. Anton Widyatmoko. Pada tahun 2001 pemerintah menerbitkan kamus SIBI untuk disebarluaskan ke SLB di seluruh Indonesia.

Walaupun telah disahkan oleh pemerintah, nyatanya penggunaan SIBI di kehidupan sehari-hari Teman Tuli sangat jarang terjadi. Alasannya, karena SIBI menerjemahkan bahasa lisan Indonesia kedalam bahasa isyarat beserta imbuhannya, seperti ber-, me-, -nya dan lain sebagainya. Hal tersebut membuat SIBI menjadi tidak praktis dan tidak wajar bagi Teman Tuli [9]. Contoh isyarat abjad menggunakan SIBI dapat diamati pada tabel 2.1 Contoh Isyarat Abjad SIBI.

Tabel 2.1Contoh Isyarat Abjad SIBI

NO	GERAKAN	GAMBAR
1.	A	
2.	B	
3.	C	
4.	D	
5.	E	
6.	F	
7.	G	
8.	H	
9.	I	
10.	J	
11.	K	
12.	L	
13.	M	
14.	N	
15.	O	

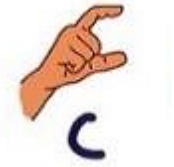
NO	GERAKAN	GAMBAR
16.	P	
17.	Q	
18.	R	
19.	S	
20.	T	
21.	U	
22.	V	
23.	W	
24.	X	
25.	Y	
26.	Z	

Sumber: Angkoso, C. V., Fuad, M., & Hadiwineka, D. R. (2016). [32]

2.2.2 BISINDO

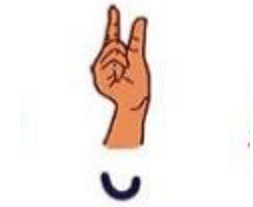
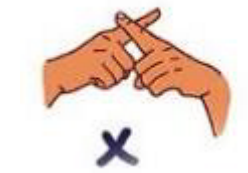
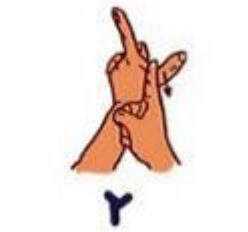
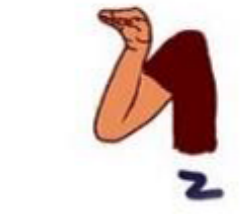
BISINDO merupakan bahasa ibu yang banyak digunakan oleh Teman Tuli dalam percakapan sehari-hari. Selain bahasanya yang lebih sederhana, BISINDO lebih menekankan kepada pesan yang ingin disampaikan, seperti mengatakan “aku sedih” diikuti raut muka yang murung, atau “aku mendapat nilai bagus” diikuti raut muka riang [9]. Contoh isyarat abjad menggunakan BISINDO dapat diamati pada tabel 2.2 Contoh Isyarat Abjad BISINDO.

Tabel 2.2 Contoh Isyarat Abjad BISINDO

NO	GERAKAN	GAMBAR
1.	A	
2.	B	
3.	C	
4.	D	
5.	E	
6.	F	
7.	G	

NO	GERAKAN	GAMBAR
8.	H	
9.	I	
10.	J	
11.	K	
12.	L	
13.	M	
14.	N	

NO	GERAKAN	GAMBAR
15.	O	
16.	P	
17.	Q	
18.	R	
19.	S	
20.	T	

NO	GERAKAN	GAMBAR
21.	U	
22.	V	
23.	W	
24.	X	
25.	Y	
26.	Z	

Sumber: Fajri, B. R., Samala, A. D., & Ranuharja, F. (2020). [33]

2.3 JAVA

Java merupakan *programming language* yang bisa berjalan pada banyak *computer* bahkan *handphone*. James Gosling pertama kali membuat Java ketika masih bergabung dengan Sun Microsystem. Java mempunya gaya bahasa yang mirip seperti C dan C++ karena mengadopsinya, tapi menggunakan sintak model objek yang lebih simple.

Aplikasi yang dibuat menggunakan Java dikompilasi menjadi p-code (*bytecode*) dan hanya akan berjalan pada *virtual machine* yang disebut JVM (*Java Virtual Machine*). Karena bisa berjalan pada sistem operasi yang berbeda, Java mempunyai slogan "*write once, run anywhere*" atau "tulis sekali, jalankan dimanapun". Ini berarti program Java dapat berjalan pada sistem operasi manapun selama sistem operasi tersebut mempunya *Java Virtual Machine*.

Model yang diterapkan dalam Java adalah *three-layer security* (pengamanan tiga lapis) untuk menjaga sistem dari kode yang tidak diinginkan. Pertama, *bytecode verifier* membaca *bytecode* serta menjaminkan *bytecode* telah sesuai dengan peraturan dasar bahasa Java. Kemudian yang kedua, *class loader* mengatasi muatan kelas Java kepada *runtime interpreter*. Lalu yang terakhir manajer keamanan mengatasi keamanan tingkat aplikasi melalui pengendalian program apa yang memilik hak untuk melakukan akses sumber daya, seperti pada port jaringan, sistem file, sistem window dan proses eksternal.

Multithreading ialah salah satu fitur dalam Java. *Thread* berfungsi agar komputer bisa menjalankan satu tugas lebih dalam waktu yang bersamaan. Terdapat kelas yang berfungsi untuk membuat *multithreaded* dalam Java, yang menjadikan program memiliki kemungkinan menyelesaikan sejumlah pekerjaan dengan konkuren.

Java melaksanakan *garbage collection* untuk menghapus sendiri objek yang tak dipakai lagi. Adanya fasilitas tersebut meminimalisir beban program dalam mengelola memori, serta mengeliminasi atau mengurangi sumber kesalahan paling fatal dalam pengalokasian secara dinamis.

Mekanisme *exception handling* yang dimiliki oleh Java terbilang sangat

ampuh. Mekanisme ini memberikan fasilitas dalam memecah kode normal dengan bagian yang mengatasi kesalahan, agar aplikasi menjadi lebih tangguh terhadap kemungkinan kesalahan yang akan muncul. Di saat terdapat kesalahan fatal, maka Java akan membentuk *exception*, kemudian *exception* bisa dikelola dan ditangkap program tanpa adanya risiko yang menimbulkan penurunan sistem.

Metode *native* yang merupakan fungsi dalam bahasa lain seperti C++/C mendapatkan dukungan dari program Java. *Native method* membuat programmer dapat menuliskan fungsi yang bisa dijalankan jauh lebih efektf dari pada fungsi ekivalen pada Java.

2.4 Android

Android merupakan perangkat mobile yang dikembangkan oleh Android Inc yang berisi sistem infromasi berbasis Linux. Google secara resmi membeli Android pada tahun 2015, dan menjadi pengembang sepenuhnya terhadap Android [25]. Android secara konsisten memberikan pembaruan terhadap perangkatnya hampir setiap tahun. Versi android dapat diamati pada Tabel 2.3 Versi Android

Tabel 2.3 Versi Android

NO	Versi	Tanggal Rilis
1.	Apple Pie / Alpha (1.0)	September 2008
2.	Banana Bread (1.1)	Febrari 2009
3.	Cupcake (1.5)	April 2009
4.	Donut (1.6)	September 2009
5.	Eclair (2.0 – 2.1)	Desember 2009
6.	Froyo : Frozen Yoghurt (2.2)	Mei 2010
7.	Gingerbread (2.3)	Desember 2010
8.	Honeycomb (3.0 – 3.2)	Februari 2011
9.	Ice Cream Sandwich (4.0)	Oktober 2011
10.	Jelly Bean (4.1 – 4.3)	Juli 2012
11.	Kitkat (4.4)	Oktober 2013
12.	Lolipop (5.0)	Juni 2014
13.	Marshmallow (6.0)	Agustus 2015

NO	Versi	Tanggal Rilis
14.	Nougat (7.0)	Agustus 2016
15.	Oreo (8.0)	Agustus 2017
16.	Pie (9.0)	Maret 2018

Semua aplikasi Android mempunyai level dan tingkat yang serupa. Andorid tidaklah membeda-bedakan aplikasi pihak ketiga dan aplikasi inti. API yang disediakan Android memberikan hak akses pada *hardware*, data ponsel bahkan kedalam sistem. User bisa meng*uninstall* aplikasi inti kemudian mengubahnya dengan aplikasi yang bukan bawaan Android. Sementara Android SDK (*Software Development Kit*) memberikan fasilitas API dan tools yang dibutuhkan dalam melakukan pengembangan aplikasi dalam platform Andorid.

A. Kelebihan Android

Android ialah satu dari sejumlah sistem operasi yang paling banyak digunakan dalam *smartphone* dikarenakan kelebihannya, seperti:

1. *Open source*. Sebagai turunan dari Linux, maka Android secara otomatis berbasis *open source*. Hal tersebut menjadikan banyak pengembang membuat aplikasi dalam *operating system* Android.
2. Android terus mengalami perkembangan sering berjalannya waktu, hal tersebut ditunjukkan melalui adanya pengembangan versi Android dalam perangkat *smartphone* yang saat ini merembet juga kepada tablet komputer.
3. Mudahnya akses terhadap pasar aplikasi. Dikarenakan Andorid mempunyai *playstore* dalam penyebaran aplikasinya, membuat pengguna dapat dengan mudah mencari aplikasi yang dibutuhkan, bahkan banyak aplikasi yang dapat diakses secara gratis.
4. Adanya dukungan dari Google seperti mempunyai akses mudah kepada Google. Google memberikan kebebasan dalam mengakses layanannya seperti gmail, gmaps, gdocs, dan sebagainya.
5. Memungkinkan untuk melakukan custom ROM, yang memberikan kemudahan dalam memodivikasi *interface* tanpa khawatir dengan

keamanan perangkatnya.

B. Kelemahan Android

Adanya sejumlah kelebihan dan keuntungan dari *operating system* Android tidaklah menutup kemungkinan adanya kekurang pada sistem ini, berikut sejumlah kekurangan dalam Android:

1. Perangkat Andorid lebih boros dalam penggunaan baterai, dikarenakan mempunyai banyak proses system yang *run on background* yang membuat baterai cepat habis.
2. Minimnya pengawasan atau kontrol terhadap *playstore* yang membuat aplikasi Android bisa diupload dengan mudah. Dengan mudahnya penguploadan aplikasi ini, membuka celah adanya aplikasi *malware* penyusup yang menyerupai aplikasi normal.

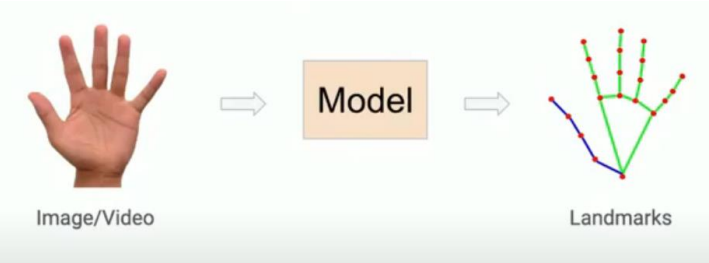
2.5 Mediapipe

Mediapipe merupakan sebuah *framework* untuk membangun *pipeline* dan akan menyimpulkan data yang masuk secara sembarangan. *Mediapipe* dirancang bagi mereka yang ingin mengimplementasikan kecerdasan buatan kedalam aplikasi yang akan dibangun. *Mediapipe* juga memungkinkan pembangunan aplikasi *cross-platform* yang bisa berjalan di berbagai perangkat keras yang berbeda [20].

Mediapipe merupakan *framework* yang ditulis menggunakan C++ dan bisa digunakan untuk mengembangkan aplikasi lintas platform, seperti windows, linux, macintos, android, bahkan website. Dengan menggunakan *multithreading* dan *GPU acceleration*, membuat *mediapipe* sangat cepat dalam memproses data yang masuk [23]. *Mediapipe* mempunyai *calculator* yang akan berjalan ketika program mulai dan akan berhenti ketika program selesai. *Calculator* ini sangat penting karena *mediapipe* menggunakan konsep *multithreading* [24].

Google menggunakan *mediapipe* sejak tahun 2012 untuk produk internalnya dan menjadikannya *open source* pada juni 2019 di CVPR [37]. Google menggunakan *mediapipe* pada youtube untuk menentukan beberapa karakteristik dalam video, misalnya dalam penentuan *thumbnail* video, identifikasi copyright video, object video yang tidak diizinkan (porno, kekerasan), dan juga iklan.

Google mengumpulkan semua dataset tangan, kemudian melabeli dan mengidentifikasi setiap landmarknya secara manual. Landmark dalam mediapipe dapat diamati pada gambar 2.4 *Landmark.*



Gambar 2.4 Landmark

Sumber: https://youtu.be/qXs0QZ6VWS8

2.5.1 Keuntungan Mediapipe

Mediapipe memungkinkan untuk melakukan penerjemahan menggunakan *machine learning* secara *realtime (ML solution for live and streaming media)*. Keuntungan yang akan didapat bagi pengguna yang menggunakan *mediapipe* diantaranya [36]:

1. *End-to-end acceleration*: Mampu memproses dan menyimpulkan data yang masuk bahkan pada perangkat keras yang umum digunakan.
2. *Build once, deploy anywhere*: Satu solusi bisa berjalan di iOS, Android, web, desktop/cloud, dan IoT.
3. *Ready to use solution*: Solusi *framework machine learning* yang canggih dan siap digunakan.
4. *Free and open source*: *Framework* dan *solution code* mediapipe berada dibawah lisensi Apache 2.0, dan sepenuhnya bebas untuk dikembangkan dan disesuaikan oleh pengembang.

2.5.2 Solusi Machine Learning Dengan Mediapipe

Selain *hand gesture*, *mediapipe* juga menyediakan solusi *machine learning* yang siap digunakan, dan hampir seluruhnya bisa dipakai untuk membuat program Android, diantaranya [36] :

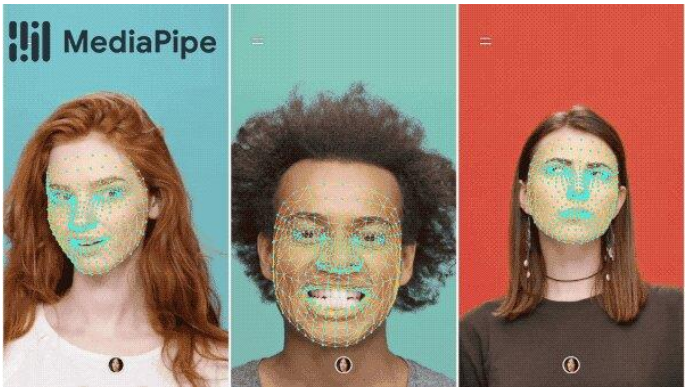
1. *Face Detection* : merupakan solusi ML yang disediakan oleh *mediapie*

untuk mendeteksi wajah dengan 6 *landmark* (tanda) dan memungkinkan *multi-face*. Contoh face detection dapat diamati pada gambar 2.5 Contoh *Face Detection Mediapipe*.

Gambar 2.5 Contoh Face Detection *Mediapipe*

Sumber: https://google.github.io/mediapipe/

2. *Face Mesh Detection* : merupakan solusi ML yang disediakan oleh *mediapie* untuk mendeteksi geometri wajah secara *real time* dan menghasilkan 468 *landmark* (tanda) wajah secara 3D. Contoh *face mesh detection* dapat diamati pada gambar 2.6 Contoh *Face Mesh Mediapipe*.



Gambar 2.6 Contoh *Face Mesh Mediapipe*

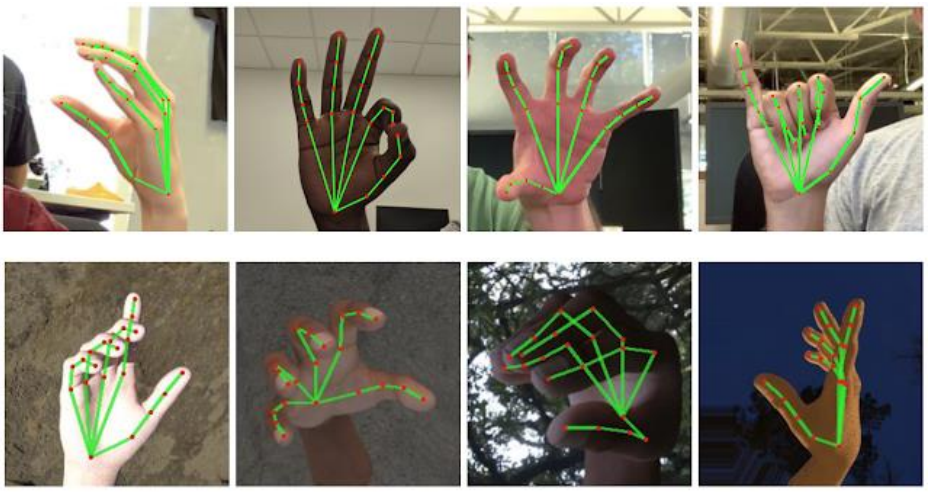
Sumber: https://google.github.io/mediapipe/

3. *Iris* : merupakan solusi ML yang disediakan oleh *mediapipe* untuk melacak *landmark* yang meliputi iris, pupil, dan kontur mata. Dengan error yang kurang dari 10% menjadikan *mediapipe* sangat cocok untuk digunakan. Contoh *iris* dapat diamati pada gambar 2.7 Contoh *Iris Mediapipe*.

Gambar 2.7 Contoh *Iris Mediapipe*

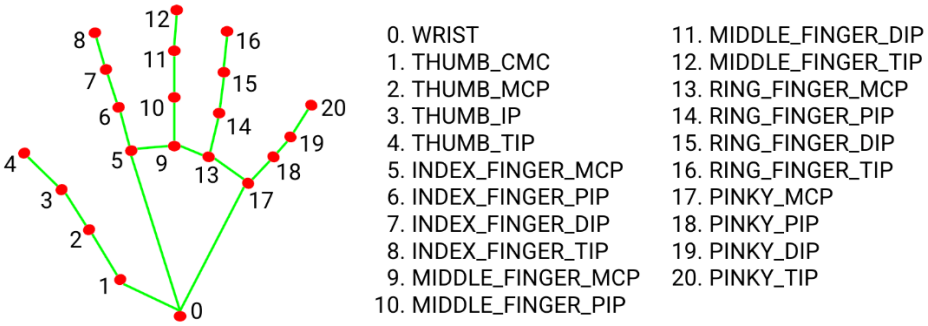
Sumber: https://google.github.io/mediapipe/

4. *Hands* : merupakan solusi ML yang disediakan oleh *mediapipe* untuk melacak tangan dan membentuk 21 3D *landmark* tangan. Proses pendeteksian melibatkan dua buah model, yaitu *palm detection* untuk mendeteksi telapak tangan dan *hand landmark detection* untuk menandai tangan sebanyak 21 titik. Solusi in memungkinkan pendeteksian dua tangan, bahkan Google berencana mengembangkan menjadi *multiple hands*. Contoh *hand landmark* dapat diamati pada gambar 2.8 Contoh *Hand Landmark Detection*. Untuk keterangan tiap *landmark* dapat diamati pada gambar 2.9 *Hand Landmark Mediapipe*.



Gambar 2.8 Contoh *Hand Landmark Detection*

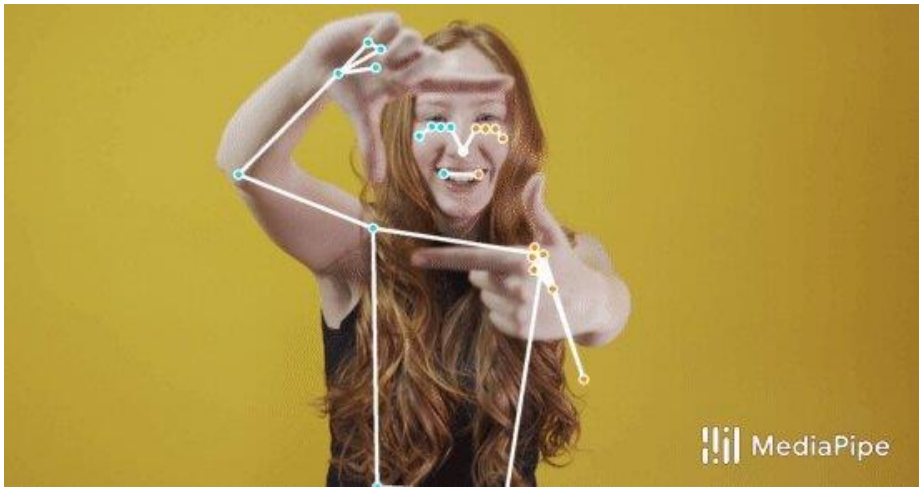
Sumber: https://google.github.io/mediapipe/



Gambar 2.9 *Hand Landmark Mediapipe*

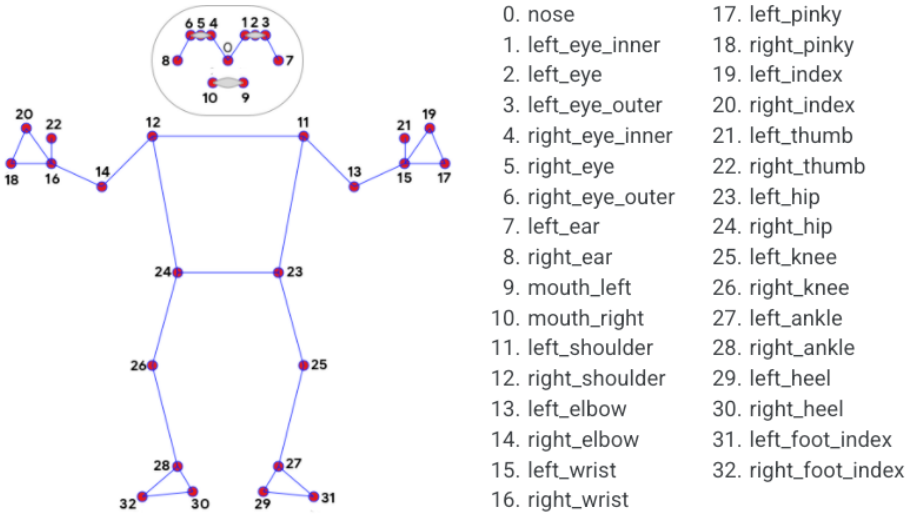
Sumber: https://google.github.io/mediapipe/

5. *Pose* : merupakan solusi ML yang disediakan oleh *mediapipe* untuk melacak pose tubuh dan membentuk 33 *landmark* 2D untuk *full body*, dan 25 *landmark* untuk badan bagian atas. Contoh *Pose* dapat diamati pada gambar 2.10 Contoh *Pose Mediapipe*. Untuk keterangan tiap landmark dapat diamati pada gambar 2.11 *Pose Landmark Mediapipe*.



Gambar 2.10 Contoh *Pose Mediapipe*

Sumber: https://google.github.io/mediapipe/



Gambar 2.11 *Pose Landmark Mediapipe*

Sumber: https://google.github.io/mediapipe/

6. Solusi ML yang lain, seperti *hair segmentation, object detection, box tracking, instant motion tracking, objectron,* dan *knift*.

2.5.3 Teknik Optimasi Mediapipe

Mediapipe menggunakan beberapa teknik agar program yang menggunakan *frameworknya* cepat dan optimal, diantaranya [37]:

1. Memanfaatkan GPU untuk mempercepat proses komputasi.

2. Untuk setiap frame dilakukan *paralel computing*.
3. Membatasi jumlah frame gambar yang masuk, agar tidak terjadi antrian gambar dan data input yang berlebihan, sehingga membebani perangkat dan penggunan memori.
4. Menggunakan hasil dari satu model untuk banyak hal. Misalkan dalam pendeteksian tangan (*hand tracking*) ada dua buah model, yaitu: *palm_detection* dan *hand_landmark_detection*. Untuk frame awal, dijalankan model *palm_detection* kemudian diikuti model *hand_landmark_detection*. Untuk mendeteksi telapak tangan pada frame selanjutnya, cukup menggunakan *landmark* pada frame sebelumnya. Artinya, *palm_detection* akan dijalankan jika prediksi dari *landmark* berada pada *low confidence* (keyakinan yang kurang).

2.6 API (Application Programming Interface)

API atau *Application Programming Interface* ialah sistem yang mengintegrasikan informasi yang berbeda dari aplikasi secara bersamaan dan berjalan di balik layar [25]. API merupakan antarmuka virtual diantara dua fungsi perangkat lunak yang bekerja sama satu sama lain, misalnya saja pada *spreadsheet* dan *word processor*. API menjelaskan bagaimana cara seorang programmer menggunakan sebuah fitur dari komputer.

Awal mulanya teknologi API berkembang melalui pembentukan suatu subrutin sederhana yang memberikan kemampuan *system modifiability* dan *interoperability* dalam mendorong proses tukar data diantara sejumlah aplikasi. Subrutin pada awalnya hanya dapat menghitung proses matematis yang simpel, sampai akhirnya tercipta API perhitungan yang hampir selalu tersedia dalam semua bahasa pemrograman. Melalui suatu subrutin yang sederhana itu, munculah gagasan bagaimana API mesti dikembangkan dan sesuai dengan perkembangan paradigm pemrograman berorientasi objek. Gagasan tersebut dapat menyebabkan sejumlah subrutin dengan jenis yang sama bisa terkumpul menjadi suatu kelas yang membungkus subrutin-subrutin tersebut [27].

2.6.1 Tensorflow

Tensorflow merupakan sebuah framework *machine learning* yang ringan, cepat dan lintas platform untuk keperluan aplikasi seluler dan IoT. Tensoflow pertama kali didevelop oleh Google pada tahun 2015 dan bersifat *open source*. Tensorflow mendukung berbagai platform seperti Android, iOS, Linux *embedded*, dan *microcontroller*. Tensorflow mendukung *multiple language* diantaranya Java, Swift, Objective-C, C++ dan Python [26].

Tensorflow membuat model dengan menggunakan data graphs. Data graphs ini terdiri dari beberapa node yang merepresentasikan sebuah operasi matematik, dan memiliki koneksi antara node yang disebut dengan tensor. Model yang dihasilkan oleh tensorflow dapat diexport dengan format *Flatbuffer* (.tflite) dan dijalankan di *mobile device* dengan menggunakan Tensorflow Lite. Tensorflow Lite yang dikenalkan pada tahun 2017 oleh Google ini memiliki performa yang lebih baik dan *binary size* yang lebih kecil karena kernel yang lebih di-*optimize*, *pre-fused activations*, serta jumlah dependencies yang lebih kecil [30].

Tensorflow banyak digunakan di berbagai bidang kecerdasan buatan seperti:

1. Teks, meliputi *text classification* dan *text prediction*.
2. Speech, meliputi *speech recognition, speech to text* dan *text to speech*.
3. Image, meliputi *object location, object detection, ocr, facial modelling, gesture recognition, segmentation, compression* dan *clustering*.
4. Audio, meliputi *audio translation* dan *voice synthesis.*
5. Content, meliputi *video generation, audio generation* dan *text generation*.

2.6.2 Speech to Text

Suara merupakan sebuah metode paling dasar dalam komunikasi yang umum dan effisien untuk menyampaikan ide antar manusia. Saat ini teknologi pengenalan suara (*speech technology*) sudah banyak tersedia untuk membantu pekerjaan manusia. Teknologi ini memungkinkan komputer atau mesin dapat mengerti suara yang diucapkan oleh manusia, kemudian merespon untuk memberikan layanan yang bermanfaat bagi manusia. Penggunaan suara lebih diutamakan karena berkomunikasi dengan komputer akan terasa lebih cepat dibandingkan dengan

menggunakan keyboard.

Agar komputer dapat mengenali dan mengerti suara yang diucapkan manusia, maka dibutuhkan sebuah teknologi yang bernama *speech-to-text*. Teknologi ini memungkinkan komputer mengenali suara dan menterjemahkannya menjadi teks untuk kemudian diproses sesuai dengan kebutuhan penggunanya. Sistem pengenalan suara merupakan proses mengubah sinyal akustik yang diterima oleh komputer menjadi sekumpulan kata.

Agar komputer mengerti suara yang diucapkan manusia, diperlukan metode yang sesuai. Metode yang bisa digunakan untuk merubah suara menjadi teks adalah *hidden markop model* dan *artificial neural network* [39].

1. *Hidden Markop Model*

   HMM merupakan model statistik yang digunakan dalam teknologi *speech recognition* karena sinyal ucapan dapat dilihat sebagai sinyal stasioner sedikit demi sedikit, atau *short-time stationary signal*. Beberapa parameter dalam pengenalan ini diantaranya:

   a. *Recognition Accuracy*

      Penerjemahan merupakan proses membandingkan the *unknown test pattern* (pola yang tidak diketahui) dengan semua pola suara, kemudian menghitung kesamaan antara pola yang dites dan pola yang dijadikan acuan dalam pendeteksian. Ini merupakan faktor paling penting dalam sistem pengenalan apapun, dan idealnya harus sampai 100%.

   b. *Recognition Speed*

      Sistem yang dibangun haruslah cepat dalam mengenali setiap ucapan yang masuk. Semakin lama sistem memproses data yang masuk maka semakin tidak effisien sistem yang dibangun.

      Sinyal yang masuk melalui beberapa tahap, yaitu:

      1) *Pre-processing*: Sinyal suara yang masuk akan dirubah menjadi *speech frame* dan akan diberikan *unique sample*, dan dihilangkan noisenya.
      2) *HMM Training*: Tahap ini melibatkan pembuatan pola yang

mewakili sebuah fitur dari sebuah class menggunakan satu atau lebih pola test yang sesuai dengan suara ucapan dari kelas yang sama.

3) *HMM Recognition*: Merupakan proses membandingkan *the unknown test pattern* dengan setiap pola referensi dan menghitung kesamaannya.

2. *Artificial Neural Network*

ANN telah didefinisikan dengan berbagai arti oleh banyak peneliti. Tetapi pada dasarnya, semua definisi tersebut mengarah pada satu kesepakatan bahwa ANN terbuat dari beberapa unit pemrosesan yang disebut neuron. Unit pemrosesan ini dilatih menggunakan kumpulan data input-output yang disajikan ke jaringan. Setelah proses pelatihan, jaringan menghasilkan hasil yang sesuai ketika diuji dengan kumpulan data yang serupa, yang berarti mengenali pola yang dikenalkan sebelumnya.

2.7 UML

Unified Modelling Language (UML) ialah sebuah bahasa pemodelan visual yang dipergunakan dalam memvisualisasikan, menspesifikasikan, mendokumentasikan, dan membangun artefak sistem perangkat lunak. UML dipergunakan dalam mendesain, memahami, mengkonfigurasi, mengeksplorasi, mengontrol, dan memelihara informasi tentang sistem.

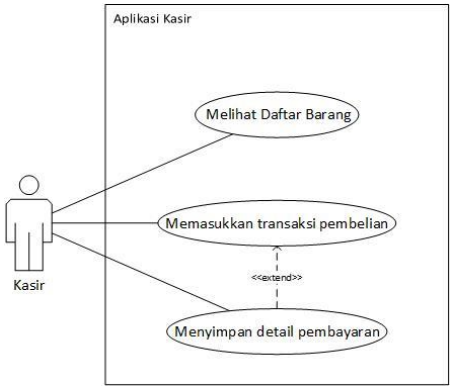
UML bukanlah bahasa pemrograman, melainkan alat yang bisa memberikan generator kode dari UML kepada beragam bahasa pemrograman dan membentuk model rekayasa balik atas program yang tersedia. UML merupakam bahasa pemodelan diskrit, yang dimaksudkan agar menjadi bahasa pemodelan umum bagi sistem diskrit seperti yang tercipta melalui *firmware,* perangkat lunak, ataupun logika digital.

2.7.1 Use Case Diagram

Use case diagram merupakan suatu tahapan proses analisis dalam memperoleh keperluan sistem serta dalam mengetahui aktivitas yang seharusnya dilakukan. *Use case diagram* memetakan interaksi antara aktor dan *use case*.

Deskripsi dari diagram *use case* ini lebih ditekankan dalam system melalui perspektif pemakainya serta ditekankan, juga pada interaksi yang berlangsung antara pemakai dengan sistemnya. *Use case* sangat membantu pengembang sistem untuk lebih jauh mendefinisikan ruang lingkup sistem serta batasan-batasannya.

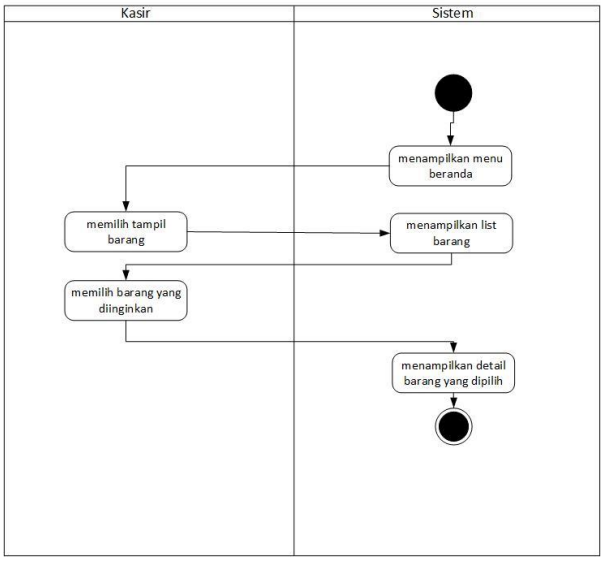
Aktor pada dasarnya ialah seluruh hal yang terdapat pada luar sistem atau perangkat lunak yang sedang dikembangkan. Semua interaksi yang berlangsung diantara sistem dan aktor dimodelkan sebagai *use case*. Contoh *use case* bisa diamati dalam gambar 2.12 Contoh *Use Case*



Gambar 2.12 Contoh Use Case

2.7.2 Activity Diagram

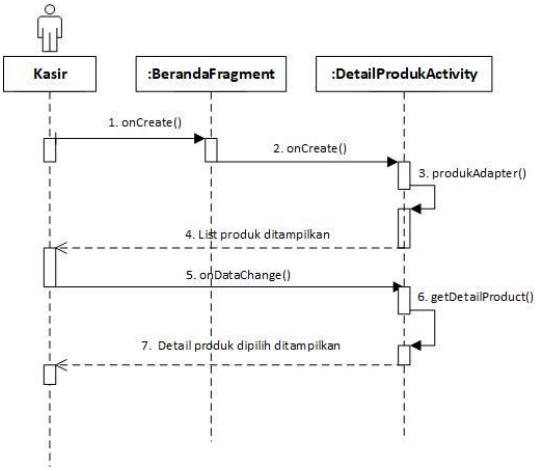
Activity diagram dipergunakan dalam membantu memahami alur kerja dari sebuah proses bisnis beserta urutannya. Mirip seperti *flowchart*, *activity diagram* dapat memodelkan urutan atas suatu aktivitas kepada aktivitas lain. *Activity diagram* fokus terhadap sejumlah aktivitas yang berlangsung yang berhubungan pada sebuah proses tunggal. Diagram tersebut juga menggambarkan bagaimana setiap aktivitas memiliki ketergantungan satu dengan lainnya. *Activity diagram* ini bisa diamati dalam gambar 2.13 Contoh *Activity Diagram*.



Gambar 2.13 Contoh Activity Diagram

2.7.3 Sequence Diagram

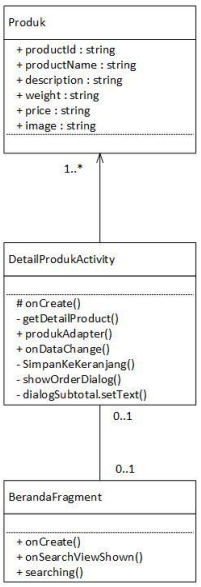
Sequence diagram dipergunakan dalam menunjukan aliran fungsionalitas pada *use case*. Diagram ini adalah bagian dari diagram interaction yang menerangkan bagaimana sebuah operasi dijalankan, pesan (*message*) apa yang dikirimkan serta kapan dilaksanakannya. Diagram ini diatur oleh waktu. Semua objek yang berhubungan dengan proses berlangsungnya operasi diuruti dari kiri kearah kanan. Contoh *sequence diagram* dapat diamati pada gambar 2.14 Contoh *Sequence Diagram.*



Gambar 2.14 Contoh Sequence Diagram

2.7.4 Class Diagram

Class diagram memberi perspektif luas atas sebuah *system* dengan menggambarkan setiap kelas dan hubungannya. Diagram ini sifatnya statis; menunjukkan hubungan apa yang dialami tidaklah apa yang terjadi apabila mereka berhubungan. *Class* menunjukkan property/atribut. Diagram ini menunjukkan deskripsi dan struktur *class*, *package*, ataupun obyek serta hubungan satu dengan lainnya seperti asosiasi, pewarisan, dan lainnya. Contoh dari *class diagram* bisa diamati dalam gambar 2.15 Contoh *Class Diagram*.



Gambar 2.15 Contoh Class Diagram