PEMBANGUNAN APLIKASI PENGENALAN OBJEK TERDEKAT UNTUK PENYANDANG TUNANETRA MENGGUNAKAN MLKIT DAN *TEXT TO VOICE* BERBASIS ANDROID

Moch Fachriyan[1], Dian Dharmayanti[2]

[1,2] Teknik Informatika – Universitas Komputer Indonesia
Jl. Dipatiukur 112-114 Bandung
E-mail: mochfachriyan@gmail.com [1], dian.dharmayanti@email.unikom.ac.id[2]

ABSTRAK

Minimnya aplikasi yang dapat mendukung penderita keterbelakangan (cacat), terutama penyandang tunanetra dalam mengenali objek yang ada disekitarnya. Meskipun beberapa objek dapat dirasakan melalui indera peraba, namun ada beberapa objek yang sulit atau bahkan tidak dapat dikenali dengan indra peraba seperti mengenali lembar upah atau fitur gambar timbul. Untuk mengatasi masalah tersebut diperlukan aplikasi yang dapat memungkinkan para penyandang tunanetra dapat mengenali objek yang ada disekitarnya. Melalui *smartphone android* didukung dengan *fitur google assistant* dapat membantu penyandang tunanetra untuk menjalankan aplikasi tanpa navigasi. Selain itu didukung dengan teknologi MLKit. Sehingga penyandang tunanetra dapat mengenali objek dengan adanya bantuan perangkat keras kamera yang terdapat pada *smartphone*. Penyandang tunanetra juga dapat dengan mudah mengetahui deskripsi dari objek yang telah tertangkap kamera tersebut melalui *fitur Google Voice To Text*. Berdasarkan penerapan MLKit dan *Voice To Text* berbasis android tersebut memberikan kemudahan kepada penyandang tunanetra untuk dapat mengenali objek tanpa harus membutuhkan orang lain dalam mengenali objek

Kata kunci: *Firebase MLKit, Pengenalan Objek, Tunanetra, Text To Voice, Kamera, Android.*

1. PENDAHULUAN

1.1 Latar Belakang

Survei Sosial Ekonomi Nasional (SUSENAS) yang dilaksanakan Biro Pusat Statistik (BPS) pada tahun 2012 menyebutkan jumlah penyandang tunanetra di Indonesia sebanyak 1.780.200 orang. Jenis kesulitan tertinggi ditempati oleh kesulitan melihat, yaitu sebesar 3,05% dibandingkan kesulitan lainnya yang hanya berkisar sebesar 1-2% [1]. Penyandang tunanetra masih dipandang sebelah mata dan tidak memiliki kemampuan dalam menjalani hidupnya secara mandiri. Dibutuhkan peran serta lembaga dan partisipasi masyarakat lainnya melalui bimbingan dan pendampingan, salah satunya yakni melalui panti sosial. Panti Sosial Bina Netra Wyata Guna Bandung adalah unit pelaksana teknis di bidang rehabilitasi dan pelayanan sosial di lingkungan Kementerian Sosial. Panti yang sudah berdiri sejak tanggal 6 Agustus 1901 ini menjadi panti tunanetra tertua dan terbesar di Indonesia. BRSPDSN Wyata Guna Bandung terletak di Jalan Pajajaran Nomor 52 Kelurahan Cicendo, Kecamatan Pasir Kaliki, Kota Bandung.

Penggunaan *smartphone* Android tidak dapat dipisahkan dari aktivitas manusia saat ini, termasuk di dalam BRSPDSN Wyata Guna Bandung yang semakin menyesuaikan dengan kecanggihan perkembangan teknologi. Pemanfaatan teknologi berupa sebuah aplikasi dapat membantu penyandang tunanetra dalam mengenali objek secara lebih lebih efektif dan efisien. Penyandang tunanetra adalah kondisi seseorang yang mengalami gangguan atau hambatan dalam penglihatannya. Berdasarkan tingkat gangguannya, penyandang tunanetra dibagi menjadi dua, yaitu buta total (*total blind*) dan yang masih mempunyai sisa penglihatan (*low vision*) [2]. Bagi masyarakat normal, tentu mudah untuk dapat mengenali objek di sekitar tanpa menggunakan alat bantu. Tetapi, penyandang tunanetra membutuhkan indra peraba untuk mengenali objek yang ada di sekitar. Tentunya tidak mudah karena ada beberapa objek yang tidak bisa dikenali dengan indra peraba saja, seperti uang kertas yang masih kesulitan diraba oleh tunanetra. Seringkali penyandang tunanetra tidak bisa mengenal dengan baik lembar upah saat melakukan transaksi jual beli, fitur gambar timbul masih sulit diraba.

Melalui penggunaan *smartphone* android, penyandang tunanetra dapat membuka aplikasi tanpa menggunakan navigasi. Dengan bantuan *google assistant*, penyandang tunanetra dapat membuka aplikasi dengan menyebutkan nama aplikasi tersebut. Selain itu dengan memaanfaatkan perkembangan teknologi *smartphone* untuk membantu pengenalan objek, kemudahan lainnya dapat dibantu dengan menggunakan teknologi MLKit. MLKit merupakan teknologi google yang mempunyai kecerdasan buatan berupa *image labeling*, *face recognition*, dan *barcode scanner*, sehingga dapat memudahkan dalam mengenali objek sekitar dengan menggunakan perangkat keras kamera yang tertanam di *smartphone*. Sedangkan

dalam memberikan detail objek apa saja yang ada di depannya di gunakan Google *Voice To Text* sehingga dapat mendeskripsikan apa yang telah tertangkap oleh kamera.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang sudah dipaparkan diatas maka yang mendasari penyusunan tugas akhir ini adalah kesulitannya penyandang tunanetra dalam mengenali objek.

1.3 Maksud dan Tujuan

Maksud dari penelitian tugas akhir ini adalah untuk membuat aplikasi pengenalan objek untuk penyandang tunanetra dengan mengimplementasikan teknologi MLKit dan *Text To Voice* berbasis android. Sedangkan tujuan dari dibuatnya aplikasi ini yaitu untuk membantu penyandang tunanetra untuk bisa mengenali objek berupa keluaran suara.

1.4 Batasan Masalah

Dalam penelitian ini dibuat beberapa batasan masalah agar pembahasan lebih terfokus sesuai dengan tujuan yang akan dicapai. Adapun batasan masalahnya sebagai berikut:

1. Aplikasi bersifat publik.
2. Memanfaatkan teknologi MLKit sebagai *image labeling*.
3. Aplikasi ini hanya bisa mengenali objek seperti barang rumah, barang elektronik.
4. Deteksi Objek hanya menggunakan MLKit.

1.5 Metodologi Penelitian

Metodologi penelitian merupakan suatu proses yang digunakan untuk memecahkan suatu masalah yang logis, dimana memerlukan data-data untuk mendukung terlaksananya suatu penelitian. Metodologi penelitian yang digunakan adalah metode analisis deskriptif. Metode analisis deskriptif merupakan metode yang menggambarkan fakta-fakta dan informasi dalam situasi atau kejadian sekarang secara sistematis, faktual dan akurat. Metode penelitian ini memiliki dua tahapan, yaitu tahap pengumpulan data dan tahap pembangunan perangkat lunak.

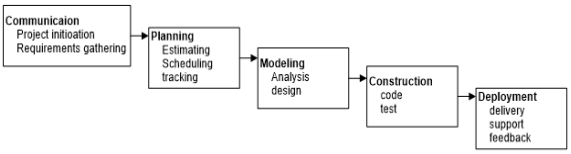
1.5.1 Metode Pengumpulan Data

Metode yang dilakukan dalam rangka mengumpulkan data penelitian ini adalah sebagai berikut:

1. Studi Literatur
   Pengumpulan data dengan cara mengumpulkan literatur, *paper*, dan jurnal yang ada kaitannya dengan judul penelitian.
2. Observasi
   Teknik pengumpulan data dengan mengadakan penelitian dan peninjauan langsung terhadap yang diambil.
3. Wawancara
   Pengumpulan data yang dilakukan dengan cara bertanya/mewawancarai kepada narasumber yang berkaitan dengan penelitian.

1.5.2 Metode Pembangunan Perangkat Lunak

Metode yang digunakan dalam pembuatan perangkat lunak menggunakan paradigma secara *waterfall* [3]. Menurut Pressman model *waterfall* adalah model klasik yang bersifat sistematis, berurutan dalam membangun sebuah *software*. Berikut ini gambaran dari *waterfall* yang meliputi beberapa proses, yaitu:



Gambar 1.1. Tahapan Pembangunan Perangkat Lunak.

1. Komunikasi
   Komunikasi merupakan tahap analisis kebutuhan perangkat lunak dan pengumpulan data. Kebutuhan perangkat lunak diidentifikasi dengan cara melakukan observasi di tempat penelitian, kemudian dengan cara menyebar lembar pertanyaan kuisioner yang berkaitan dengan judul pemelitian, dan mengumpulkan data-data yang dibutuhkan.
2. Perencanaan
   Perencanaan merupakan tahap penyusunan rencana-rencanan yang akan dilakukan selama pembangunan perangkat lunak meliputi pembuatan jadwal dari setiap tahap yang dilakukan dan mempersiapkan semua kebutuhan.
3. Pemodelan
   Selama tahap ini, dilakukan implementasi dari kebutuhan pembuatan aplikasi dalam bentuk presentasi antarmuka serta arsitektur aplikasi sebagai serangkaian perancangan aplikasi untuk *front-end* dalam memberikan informasi yang informatif kepada pengguna aplikasi ini dari hasil pengolahan sistem *back-end*.
4. Pembangunan
   Pada tahap ini, perancangan aplikasi diimplementasikan dalam bentuk kode atau serangkaian unit program. Pengimplementasian pada tahap ini menggunakan android studio.
5. Penyebaran
   Tahapan ini bisa dikatakan final dalam pembuatan sebuah *software* atau sistem. Setelah melakukan analisis, desain dan pengkodean maka sistem yang sudah jadi akan digunakan oleh *user*. Kemudian pada tahap ini didapat hasil dan juga umpan balik dari penggunaan aplikasi yang telah dirancang.

2. LANDASAN TEORI

Landasan teori merupakan penjelasan dari berbagai konsep dan teori-teori yang berkaitan dengan pembangungan aplikasi pengenalan objek.

2.1 Definisi Tunanetra

Tunanetra merupakan suatu kondisi hilangnya daya pengelihatan untuk dapat berfungsi sebagaimana mestinya sehingga individu yang mengalami ketunanetraan harus menggunakan indera pendengaran, perabaan dan penciuman dalam menempuh pendidikannya [4]. Pendapat tersebut menerangkan bahwa tunanetra merupakan suatu kondisi yang dialami oleh seseorang, yaitu tidak berfungsinya indera pengelihatan sebagaimana mestinya. Oleh karena itu, seseorang yang tunanetra harus menggunakan indra-indra selain pengelihatan dalam menempuh pendidikan.

Anak tunanetra merupakan seorang anak yang membutuhkan alat bantu, metode atau teknik-teknik tertentu dalam kegiatan pembelajarannya sehingga anak tersebut dapat belajar tanpa pengelihatan atau dengan pengelihatan fungsionalnya. Pendapat tersebut menjelaskan bahwa dari sudut pandang pendidikan, anak tunanetra membutuhkan alat bantu, metode, dan teknik-teknik tertentu dalam kegiatan pembelajaran. Hal tersebut penting agar anak tunanetra bisa tetap mengikuti pembelajaran walaupun tanpa pengelihatan atau dengan pengelihatan fungsional [5].

2.1.1 Penyebab Tunanetra

Secara garis besar, berdasarkan ketajaman pengelihatan yang masih tersisa, tunanetra diklasifikasikan menjadi dua golongan besar, yaitu kurang lihat (low vision) dan buta total atau totally blind. Seseorang anak dikatakan kurang lihat (low vision) bila anak tersebut masih mampu menerima rangsang cahaya dari luar, tetapi ketajaman indera pengelihatannya lebih dari 6/21, atau bila anak tersebut hanya mampu membaca headline yang ada di koran. Berdasarkan pendapat tersebut, seorang anak dikatakan low vision bila anak tersebut hanya mampu membaca headline yang ada pada koran. Bila diukur ketajaman pengelihatannya, anakk low vision ketajaman pengelihatannya lebih dari 6/21, artinya anak tersebut hanya dapat melihat/membaca dengan jelas objek yang berjarak 6 meter, padahal objek tersebut dabat dilihat dengan jelas oleh orang yang memiliki pengelihatan normal dari jarak 21 meter [6].

Pendapat lain mengemukakan bahwa seorang anak termasuk ke dalam kategori kurang lihat (low vision) bila anak tersebut masih memiliki pengelihatan yang buruk walaupun telah dikoreksi, tetapi fungsi pengelihatannya masih dapat ditingkatkan melalui penggunaan alat-alat bantu optik dan modifikasi lingkungan [5]. Pendapat tersebut menjelaskan bahwa walaupun telah dikoreksi, pengelihatan anak low vision masih buruk. Namun, fungsi pengelihatannya masih dapat ditambah atau ditingkatkan dengan bantuan alat-alat optik dan modifikasi lingkungan.

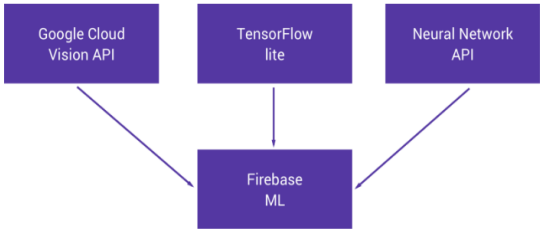
Seorang anak tunanetra yang termasuk ke dalam kategori kurang lihat (low vision) membutuhkan beberapa adaptasi, seperti kaca pembesar atau buku yang dicetak besar. Dari pendapat tersebut, anak low vision memerlukan modifikasi dalam kegiatan pembelajaran, seperti penggunaan kaca pembesar untuk membaca dan buku-buku yang tulisannya sudah diperbesar [2].

Berdasarkan pendapat-pendapat tersebut, dapat ditegaskan bahwa bila seorang anak masih bisa membaca headline atau setelah diukur ketajaman pengelihatannya lebih dari 6/21 (hanya mampu melihat suatu objek dalam jarak 6 meter, sedangkan orang normal dapat melihatnya dalam jarak 21 meter) dan masih dapat dioptimalkan dengan penggunaan alat-alat bantu optik, maka anak tersebut termasuk ke dalam anak tunanetra kategori kurang lihat (*low vision*). Anak tunanetra kategori kurang lihat (*low vision*) memerlukan adaptasi di dalam pembelajarannya, seperti membutuhkan kaca pembesar untuk membaca, memerlukan buku dengan tulisan yang sudah diperbesar, dan modifikasi lingkungan. Hal ini tentu saja berbeda dengan kebutuhan dari anak tunanetra kategori buta total (*totally blind*).

2.2 *MLKit For Firebase*

Di Google I/O tahun ini kita melihat perkenalan dari *Firebase* MLKit, yaitu sebuah bagian dari rangkaian *Firebase* yang hadir memberikan aplikasi berupa kemampuan untuk mendukung fitur kecerdasan buatan dengan lebih mudah. SDK saat ini hadir dengan koleksi kemampuan yang telah ditentukan sebelumnya dan biasanya diperlukan dalam sebuah aplikasi. Kamu dapat menerapkannya di aplikasimu, terlepas dari apakah kamu terbiasa dengan pembelajaran secara mesin atau tidak [7].

Sekarang, apa yang ditawarkan *Firebase* MLKit kepada kita semua sudah memungkinkan untuk penerapan diri kita menggunakan berbagai teknologi pembelajaran secara mesin.



Sumber gambar : https://joebirch.co/2018/05/22/exploring-firebase-mlkit-on-android-introducing-mlkit-part-one/

Gambar 2.1 Diagram MLKit

Meskipun kita dapat menerapkan berbagai hal tanpa *Firebase* MLKit, ada beberapa alasan kenapa kita mungkin tidak dapat melakukannya, yakni kemungkinan karena:

1. Kurangnya pengetahuan tentang pembelajaran secara mesin dapat menahan kita untuk tidak dapat menerapkan beragam fitur lain, mungkin kita merasa kewalahan atau tidak punya waktu untuk dapat meningkatkan kemampuan di dalam bidang ini.
2. Menemukan model pembelajaran mesin yang super akurat dan terlatih dengan baik, pada waktu yang sama sulit memilih mana yang akan digunakan dan kemudian dioptimalkan untuk *platform* kamu.
3. Hosting model MLKit kamu untuk akses penyimpanan awan yang juga bisa menjadi sesuatu sulit bagi penerapan MLKit kamu. Mengemas dalam aplikasi kamu terkadang bisa jadi pendekatan lebih mudah, tetapi hal itu sendiri memiliki beberapa kelemahan.

Dengan mengingat poin berikut, mungkin sulit untuk mengetahui darimana harus memulai. Hal ini adalah salah satu tujuan utama *Firebase* MLKit untuk membuat pembelajaran mesin ke aplikasi Android dan *iOS*, secara lebih mudah diakses oleh pengembang dan tersedia di lebih banyak aplikasi [6]. Saat ini, MLKit menawarkan kemampuan untuk:

1. Mengenali *text*,
2. Mengenali *landmark*,
3. Pengenalan wajah
4. Pemindaian kode batang
5. Pemberian label pada gambar

2.2.1 *Image Labeling*

Pelabelan Gambar (*Image Labeling*) adalah proses mengenali perbedaan wujud dari sebuah gambar. Kita dapat mengenali beragam wujud seperti hewan, tumbuhan, makanan, aktivitas, warna benda, karakter fiksi, minuman, dan lain-lain. Dengan teknologi MLKit's *Image Labeling* API kita dapat mengenali kurang lebih 400 wujud [8].

	On-device	Cloud
Pricing	Free	Free for first 1000 uses of this feature per month: see Pricing
Label coverage	400+ labels that cover the most commonly-found concepts in photos. See below.	10,000+ labels in many categories. See below. Also, try the Cloud Vision API demo to see what labels can be found for an image you provide.
Knowledge Graph entity ID support	✓	✓

Sumber gambar : https://firebase.google.com /docs/ml-kit/label-images

Gambar 2.2 Tabel API *Image Labeling*.

MLKit's *Image Labeling* menyediakan API yang berbasis perangkat dan penyimpanan. Kita dapat memilih salah satu untuk digunakan tergantung pada sebuah kasus. API yang berbasis perangkat cepat dan tidak membutuhkan koneksi internet. Sebaliknya, yang berbasis penyimpanan lebih akurat tetapi membutuhkan koneksi internet. Kita dapat melihat level akurasi dari API yang berbasis peragkat dan penyimpanan pada gambar di bawah ini:

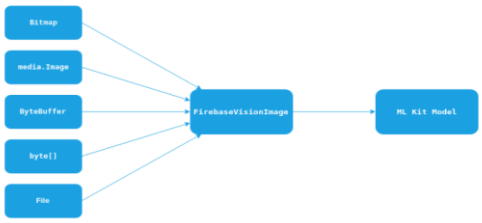
Sumber gambar : *Google I/O 2018.*

Gambar 2.3 Gambar Akurasi API.

Yang berbasis perangkat: infrastruktur neon di langit yang menyenangkan orang yang berbasis awan atau penyimpanan: kincir angin di taman hiburan rekreasi malam luar ruangan.

Berikut langkah langkah penerapan atau pembuatan kode pada *Image Labeling*

1. Langkah pertama: Tambahkan *Firebase* pada aplikasi.
2. Langkah kedua: Membutuhkan MLKit Dependensi.
   Selanjutnya spesfikasikan model ML (kondisional), Untuk API berbasis perangkat kamu dapat memilih konfigurasi di aplikasi secara download otomatis model ML setelah *install* dari play store. Jika tidak, model akan *download* pada saat pertama kali kamu aplikasi terpasang di perangkat. Untuk memasang fitur ini, kamu membutuhkan spesifikasi model di aplikasi.
3. Langkah ketiga: Ambil Gambar
   MLKit menyediakan langkah mudah untuk label gambar dari beragam tipe gambar seperti Bitmap, media. *Image*, *ByteBuffer*, *byte*, atau *file* dari perangkat. Kamu hanya perlu membuat sebuah objek *Firebase Vision Image* dari tipe gambar yang telah disebutkan dan meneruskannya ke model.



Sumber gambar : https://developers.google.com/ android/reference/com/google/firebase/ml/vision/co mmon/FirebaseVisionImage

Gambar 2.4 Diagram Image Labeling

Di contoh ini menggunakan tipe gambar Bitmap untuk membuat objek *FirebaseVision Image.*

4. Langkah keempat: Atur model
   Sekarang waktunya untuk mempersiapkan gambar kita. Pelabelan gambar.
   a. Model berbasis perangkat:
      Pelabelan gambar berbasis perangkat mengembalikan tidak lebih dari 10 label. Kamu dapat mengubah konfigurasi dengan melewati objek dari *Firebase Vision Label Detector Options* untuk model pelabelan gambar.
   b. Model berbasis penyimpanan Cloud
      Pelabelan gambar berbasis penyimpanan Cloud menggunakan versi STabel dari model dan mengembalikan tidak lebih dari 10 label. Kamu dapat mengubah konfigurasi dengan melewati objek dari *Firebase Vision Label Detector Options* untuk model pelabelan gambar.
5. Langkah Kelima:
   Akhirnya, kita dapat melewati gambar untuk model pelabelan gambar.
6. Langkah keenam: Ekstrak Informasi
   Jika pelabelan gambar berhasil, pendengar akan berhasil menerima daftar dari objek *Firebase Vision Label*. Representasi objek berwujud dari label dan mengandung semua informasi yang berhubungan dengan label itu. Kamu dapat menggali semua informasi seperti ini [8].

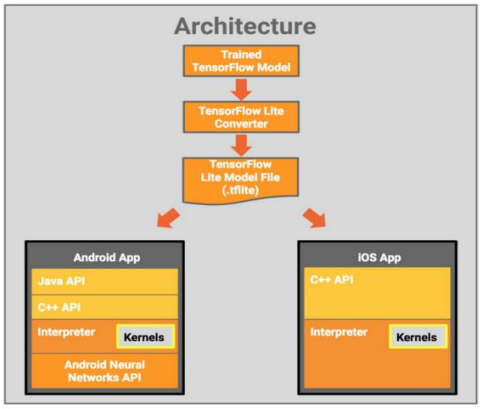
2.3 *TensorFlow Lite*

TensorFlow Lite adalah solusi *TensorFlow* untuk *mobile phone* dan perangkat tertanam. *TensorFlow* akan menjalankan model mesin pencari di perangkat mobile dengan penglihatan rendah. Jadi kamu dapat mengambil keuntungan dari klasifikasi yang ada. Pemulihan atau apapun yang mungkin diinginkan tanpa harus berkunjung terus menerus ke server.

TensorFlow didukung pada Android dan *IOS* tipe C++ API yang memiliki *Java Wrapper* untuk pengembang android. Selain itu, pada perangkat android yang mendukung, penerjemah juga dapat menggunakan Android *Neural Networks* API untuk akselerasi perangkat keras. Jika perangkat android tidak mendukung, maka pengaturan *default* akan mengarah ke CPU untuk dieksekusi. Dalam artikel ini, akan lebih fokus pada bagaimana kamu menggunakannya di aplikasi android [9].

TensorFlow Lite terdiri dari *software* untuk menjalankan model yang sudah ada sebelumnya, dan seperangkat alat yang dapat digunakan untuk menyiapkan model untuk digunakan pada perangkat seluler tertanam.

Hal ini belum dirancang untuk model pelatihan, sebagai gantinya kamu dapat melatih model pada mesin yang bertenaga lebih tinggi, kemudian mengkonversi model tersebut ke format. *TensorFlow Lite,* yang mana model tersebut digunakan ke penerjemah seluler.



Sumber gambar : https://www.tensorflow.org/lite/guide

Gambar 2.5. *Arsitekture Tesnsorflow Lite.*

TensorFlow Lite saat ini digunakan dalam peninjauan pengembangan, sehingga mungkin tidak mendukung beroperasi di semua model *TensorFlow*. Meskipun demikian, *TensorFlow* dapat berjalan dengan model klasifikasi gambar yang umum termasuk *inception* dan *mobilenets.* Dalam artikel ini, kamu akan melihat bagaimana penggunaan model *mobilenet* di Android. Aplikasi ini akan melihat umpan kamera dan menggunakan *mobilenet* untuk mengklasifikasikan gambar dominan ke dalam gambar utama [9].

2.4 *API Google Cloud Vision*

API Google Cloud Vision memungkinkan pengembang memahami konten gambar dengan mengenkapsulasi model pembelajaran mesin yang canggih dengan API *REST* yang mudah digunakan. Dengan cepat mengklasifikasikan gambar menjadi ribuan kategori (misalnya, "perahu layar", "singa", "Menara Eiffel"), mendeteksi benda dan wajah individual dalam gambar, dan menemukan dan membaca kata-kata tercetak yang terdapat di dalam gambar. Anda dapat membangun metadata pada katalog gambar Anda, konten ofensif yang moderat, atau mengaktifkan skenario pemasaran baru melalui analisis sentimen gambar. Analisis gambar yang diunggah dalam permintaan atau integrasikan dengan penyimpanan gambar Anda di *Google Cloud Storage*.

2.5 *Text To Voice*

Text-to-Speech adalah suatu proses di mana teks diubah menjadi audio digital dan kemudian "berbicara." Kebanyakan mesin *Text-to-Speech* dapat dikategorikan menurut metode yang mereka gunakan untuk menterjemahkan fonem ke dalam suara yang dapat didengar.

Secara fungsional, *Text to Speech* atau TTS melakukan proses sebaliknya dari sistem Pengenal Ucapan. Namun demikian pendekatan

implementasinya sama sekali berbeda. Artinya, komponen-komponen pembentuk kedua sistem tersebut sama sekali berbeda [10].

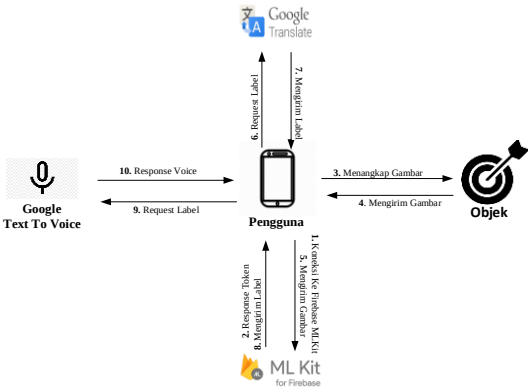
2.6 Analisis Maslah

Analisis masalah bertujuan untuk mengidentifikasi masalah-masalah yang akan dipecahkan. Analisis masalah juga merupakan langkah pertama dalam tahap analisis sistem. Masalah akan dapat teridentifikasi bisa dari suatu pertanyaan. Masalah ini yang menyebabkan sasaran dari sistem tidak tercapai. Maka dari itu, langkah pertama yang harus dilakukan pada tahap analisis masalah adalah mengidentifikasi terlebih dahulu masalah-masalah yang terjadi berikut merupakan analisis masalah:

1. Penyandang tunanetra masih banyak yang salah dalam mengenali objek.
2. Penyandang tunanetra masih membutuhkan orang lain dalam mengenali objek.

2.6.1 Analisis Sistem Yang Dibangun

Arsitektur sistem yang diajukan pada penelitian ini memiliki desain arsitektur sebagai berikut:



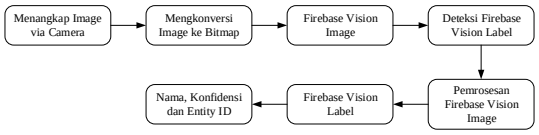
Gambar 2.6. Arsitekture Sistem.

Keterangan dari gambar desain arsitektur diatas adalah sebagai berikut:

1. Aplikasi akan melakukan koneksi ke *Firebase* MLKit API.
2. *Firebase* MLKit API akan memeberikan respon jika token sesuai maka akan memberikan respon http ok jika tidak sesuai akan mengirimkan respon *failed.*
3. Sistem mengaktifkan kamera.
4. Kamera akan mengembalikan informasi berupa resolusi warna dan format dengan camera API 2
5. Dari kamera API 2 selanjutnya sistem akan mengirim setiap *frame* yang di tangkap sebagai media inputan ke via MLKit *Firebase.*
6. Sistem mengirim label yang sudah diproses MLKit *Firebase* ke *Google Translate* untuk diterjemahkan kedalam bahasa Indonesia.
7. *Google Translate* mengirim label yang sudah di terjemahkan.
8. MLKit *Firebase* akan membaca meta data dan mengekstrak kedalam bentuk JSON
9. Selanjutnya JSON tersebut akan dikirim ke *Google Text to Voice* sebagai media input.
10. *Google Text to Voice* akan mengembalikan JSON tersebut ke dalam bentuk suara.

2.6.1.1 Analisis MLKit Firebase

Firebase MLKit adalah layanan API *machine learning* yang memungkinkan menerapkan *mechine learning* dengan menggunakan berbagai model yang sangat akurat diaplikasi Android.



Gambar 2.7. Cara Kerja MLKit Dengan *Firebase Vision Image*.

Dari gambar 2.7 dapat dilihat dalam proses pengenalan objek MLKit menggunakan *firebase vision image* dengan kamera sebagai input yang selanjut nya akan di *convert* ke dalam bitmap dan melakukan *upload* ke *firebase vision image* untuk untuk lebih jelasnya sebagai berikut:

1. Menangkap *Image* Via Kamera
   Pada tahap ini media kamera menjadi sebuah media input dengan membaca frame image yang ditangkap dengan menggunakan *android.hardawe.camera2.API* android *camera* API ini hanya *support JPEG / YUV_420_888 format.*
2. Mengkonversi Media *Input* ke Bitmap
   Mengkonversi menggunkan kamera menjadi media input setiap objek akan di tangkap dengan menggunakan *surfaceview, surfacetexture Surface (SurfaceTexture), MediaCodec, MediaRecorder, Allocation, and ImageReader, Bitmap.createBitmap()*
3. *Firebase Vision Image*
   Dari gambar bitmap yang dikonversi selanjutnya adalah mengenali object dengan menggunakan *Cloud Vision Detector* yang merupakan salah satu layanan API Google dikarenakan menggunakan MLKit fitur ini telah tersedia dalam satu SDK.
4. *Firebase Vision Label*
   Firebase Vision Label merupakan proses mendeteksi dan mencari label dari gambar yang telah diproses oleh *vision image* untuk integrasi nya cukup menggunakan:

```
FirebaseVisionLabelDetector
   labelDetector =
   FirebaseVision.getInstance().getV
   isionLabelDetector();
FirebaseVisionImage image =
FirebaseVisionImage.fromBitmap(bitma
p);
```

5. Pemrosesan *Vision Image*
   Pada tahap ini setelah melewati proses pendeteksian objek dan label akan di

konversi kedalam sebuah meta data dalam sebuah *array*.

```
fromByteBuffer (ByteBuffer
byteBuffer,
FirebaseVisionImageMetadata
metadata)
Creates a FirebaseVisionImage from a
ByteBuffer.
```

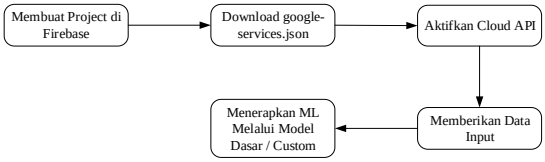
6. *Firebase Vision Label*
   Setelah setiap gambar dengan objek yang dikenali maka selanjutnya adalah pemberian label terhadap setiap objek yang di kenali dengan kembalian *array list*.
7. Pemberian *Confidance* dan *Entity id*
   Setelah pemberian *vision label* tahap selanjutnya adalah dengan mengembalikan *confidance* dari setiap objek yg di kenali dan *entity id* nya dengan menggunakan fungsi *graph* dari google dengan bentuk json.

```
GenericUrl url = new
GenericUrl("https://kgsearch.googlea
pis.com/v1/entities:search");
```

Dapat disimpulkan dalam mengenali objek media kamera dijadikan sebagai media input dalam pengolahan citra setelah itu gambar akan di konversi ke dalam format bitmap dan melakukan pengolahan citra.

2.6.1.1.1 Integrasi MLKit *Firebase*

Terdapat beberapa tahapan dalam mengintegrasikan MLKit di karenakan MLKit merupakan sebuah teknologi *machine learning* dan memudahkan *depelover* dalam mengintegrasikannya. berikut tahapan dalam mengintegrasikan MLKit:

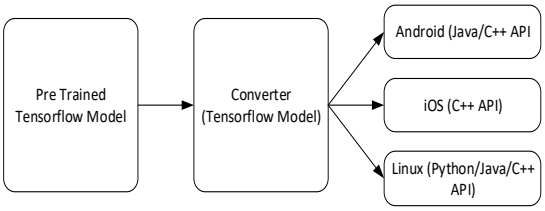


Gambar 2.8 Integrasi MLKit.

Dengan mengintegrasikan MLKit *Firebase* ke dalan *cloud vision* API ini akan mempermudahkan pembuatan aplikasi yang sudah didukung AI yang dapat melakukan tugas tugas kompleks terkait dengan mendeteksi objek disekitar. Akan tetapi hanya bisa mendeteksi objek *standart google vision* yang sudah ada di dalam penyimpanan *cloud vision* API. Jika akan mendeteksi objek yang baru, harus membawa *custom* model yang kita bawa dan nanti nya akan diproses melalui *Tensorflow Lite*.

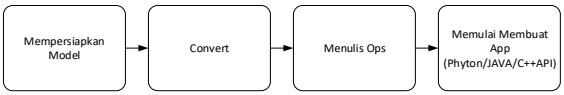
2.6.2 Analisis *Tensorflow Lite*

TensorFlow Lite adalah seperangkat alat untuk membantu pengembang menjalankan model *TensorFlow* pada perangkat seluler, tertanam, dan *IoT*. Inferensi pembelajaran mesin pada perangkat dengan latensi rendah dan ukuran biner kecil.



Gambar 2.9 Platform Yang Didukung.

TensorFlow Lite mendukung *platform* Android / *iOS* serta *platform* Linux (misalnya Raspberry Pi). Pada perangkat seperti Raspberry Pi, API Python membantu. Platform *TensorFlow Lite* juga mendukung model Core ML serta platform *iOS*. Untuk penggunaan *Tensorflow Lite* dapat menggunakan model yang sudah ada, adapun tahapan untuk mulai membangun:



Gambar 2.10 Tahapan *Tensorflow Lite*

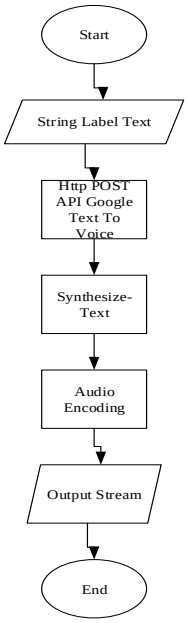
Menggunakan *Tensorflow Lite* secara *real time* terdiri dari empat langkah :

1. Menggunakan model yang sudah ada atau menyiapkan model sendiri dan melatih model tersebut.
2. Setelah model sudah siap dan harus dikonversi ke dalam *TensorFlow Lite* format menggunakan konverter.
3. Kemudian, menulis ops di atasnya untuk segala jenis optimasi.
4. Lalu dapat memulai membuat proyek.

Mengkonversi model ML menjadi model *TensorFlow Lite* dapat dilakukan hanya dalam satu baris kode dengan memanggil metode konversi. Berikut adalah perintah Python sederhana yang mengubah model yang ada menjadi format *TensorFlow Lite*. (tflite).

2.6.3 *Text To Voice API*

Penggunaan *Google Text To Voice* pada aplikasi ini untuk mengkonversikan hasil label yang didapat dari hasil deteksi suatu objek yang sudah diproses oleh MLKit dan output yang dihasilkan berupa keluaran suara (ucapan). Adapun proses yang akan dilakukan seperti gambar berikut:



Gambar 2.11 Alur *Text To Voice.*

Keterangan dari gambar alur *Text to voice* diatas sebagai berikut:

1. Dari hasil deteksi object label, object tersebut digunakan sebagai input text yang selanjut nya akan diproses oleh API Google *Text To Voice*.
2. Label yang dihasilkan akan di *convert* menjadi *string*.

```
String=label_text.xml
```

3. Selanjutnya text tersebut akan di kirim menggunakan metode *Post ke Google Text to Voice.*

```
POST
https://texttospeech.googleapis.com/v1/
text:synthesize
```

4. Google *Text voice* akan melakukan proses Synthesize-text , *Syntesys text* merupakan transformasi dari teks ke arah suara (*speech*) dengan respon body Json seperti berikut:

```
{
  "input": {
    object(SynthesisInput)
},
```

5. Setelah proses Synthesize tahap selanjutnya adalah melakukan *encoding audio* , adalah proses dimana menggenerate *file* mp3 berdasarkan profil yang di gunakan untuk *sample* nya seperti di bawah ini:

```
AudioConfig        audioConfig        =
              AudioConfig.newBuilder()

.setAudioEncoding(AudioEncoding.MP3)
//              MP3              audio.

.addEffectsProfileId(effectsProfile)
//            audio            profile
            .build();
```

6. Setelah berhasil di *encoding* selanjutnya adalah menyiapkan *output stream* ,*output stream* adalah proses dimana melakukan konversi mp3 ke dalam dari *ByteArray*.

```
// Write the response to the output
file.
      try (OutputStream out = new
FileOutputStream("output.mp3"))        {

out.write(audioContents.toByteArray());
      System.out.println("Audio content
written to file \"output.mp3\"");
```

2.6.4 Analisis Kebutuhan Perangkat Lunak

Analisis kebutuhan perangkat lunak memuat kebutuhan perangkat lunak minimum yang harus dipenuhi oleh pengguna. Berikut ini adalah kebutuhan perangkat lunak yang dibutuhkan:

Table 2.1 Analisis Kebutuhan Perangkat Lunak

No	Kebutuhan Perangkat Lunak
1	Versi 4.0 (Ice Cream Sandwich)
2	Kamera minimal 8MP
3	Display 854 x 480 pixel
4	Memory 2 GB

2.6.5 Analisis Kebutuhan Perangkat Keras

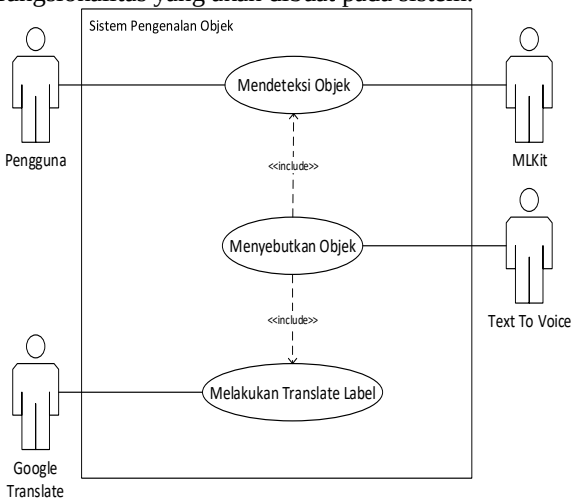
Sistem membutuhkan perangkat keras smartphone pengguna dengan syarat minimum sebagai berikut:

Table 2.2 Analisis Kebutuhan Perangkat Keras

No	Kebutuhan Perangkat Lunak
1	Prosesor QuadCore, 1.2GHz
2	RAM minimum 2 GB

2.6.6 Use Case Diagram

Use Case Diagram merupakan salah satu model diagram UML (*Unified Modeling Language*) yang berfungsi untuk menggambarkan kesepakatan fungsional yang diharapkan dari sebuah sistem. Berikut ini adalah use case diagram dari kesepakatan fungsionalitas yang akan dibuat pada sistem.

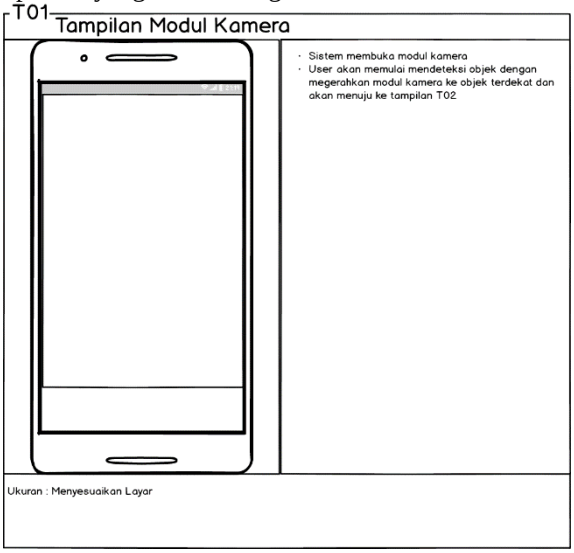


Gambar 2.12. *Use Case Diagram.*

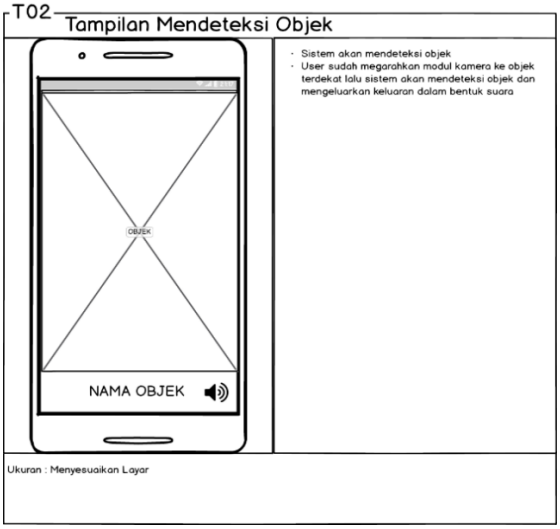
2.6.7 Perancangan Antar Muka

Perancangan antarmuka dilakukan untuk merancang tampilan aplikasi sebelum dibangun, dirancang agar dapat menggambarkan aplikasi yang

nantinya akan dibangun. Berikut ini gambar perancangan antarmuka yang menggambarkan aplikasi yang akan dibangun.



Gambar 2.13. Perancangan Antarmuka Kamera.



Gambar 2.14. Perancangan Antarmuka Mendeteksi Objek.

2.6.8 Hasil Implemantesi Aplikasi

Berikut ini adalah implementasi dari perancangan yang telah dibuat sebelumnya**:**

Gambar 2.15 Hasil Implemntas Antarmuka.

2.6.9 Skenario Pengujian

Pengujian dilakukan dengan mencoba kemungkinan-kemungkinan yang terjadi dan pengujian dilakukan berulang-ulang, apabila dalam pengujian ditemukan kesalahan maka dilakukan penelusuran atau perbaikan untuk memperbaiki kesalahan yang ada. Jika sudah selesai melakukan perbaikan, maka dilakukan secara terus menerus agar diperoleh hasil yang terbaik. Rencana pengujian alpha yang akan dilakukan pada perangkat lunak ini dapat dilihat sebagai berikut:

Table 2.3 Tabel Perencanaan Pengujian Alpha

Kelas Uji	Poin Pengujian	Jenis Pengujian
Mendeteksi Objek	Mendeteksi objek zdan menampilkan hasil deteksi dalam bentuk label	Blackbox
Melakukan Translate Label	Menampilkan hasil label yang sudah ditranslate ke Bahasa Indonesia	Blackbox
Menyebutkan Objek	Menyebutkan hasil deteksi	Blackbox

Berdasarkan hasil pengujian alpha yang telah dilakukan pada Aplikasi Pengenalan Maket ini maka dapat disimpulkan bahwa sistem sudah berjalan sesuai dengan kebutuhan sistem yang telah dirancang

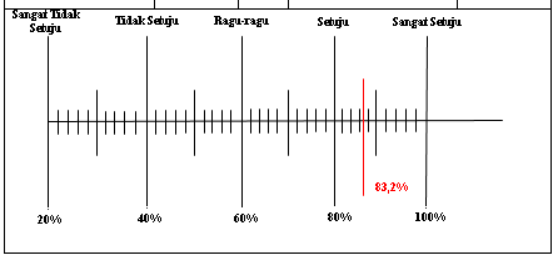
2.6.10 Skenario Pengujian Beta

Pengujian beta merupakan pengujian yang dilakukan secara objektif yang dilakukan secara langsung oleh pengguna yang nantinya akan menggunakan aplikasi pengenalan objek. Hal ini dilakukan untuk dapat mengetahui sejauh mana aplikasi yang dibangun dapat membantu memudahkan dan dapat menyelesaikan masalah yang sudah dijelaskan pada rumusan masalah.

Table 4 Pertanyaan Kuisioner Pengujian.

No	Pertanyaan
1	Apakah anda setuju aplikasi ini dapat membantu dalam mengenali objek ?
2	Apakah anda setuju dengan adanya aplikasi ini dapat digunakan untuk mengenali objek tanpa memerlukan bantuan orang lain ?
3	Apakah anda setuju aplikasi ini mudah dioperasikan ?
4	Apakah anda setuju suara yang dihasilkan dari aplikasi sudah cukup jelas ?
5	Apakah anda setuju aplikasi ini dapat mengenali berbagai macam objek ?

Kategori Jawaban	Skor	Frekuensi Jawaban	Total Skor	Nilai Presentase (%)	Keputusan
Sangat Setuju	5	17	85	(312/75*5))*100 = 83,2 %	Sangat Setuju
Setuju	4	53	212		
Ragu-ragu	3	5	15		
Tidak Setuju	2	0			
Sangat Tidak Setuju	1	0			
Jumlah		75	312		



Gambar 2.16 Kesimpulan Hasil Kuisoner.

3. PENUTUP

Penutup merupakan penjelasan mengenai kesimpulan yang berisi hasil yang diperoleh setelah dilakukannya tahap analisis dan perancangan terhadap pembangunan aplikasi pengenalan objek Serta terdapat beberapa saran agar penelitian selanjutnya lebih baik lagi.

Adapun kesimpulan dan saran yang didapatkan dari hasil penelitian ini yaitu:

3.1 Kesimpulan

Berdasarkan hasil pengujian pada bab sebelumnya, maka diperoleh kesimpulan yaitu Aplikasi yang di bangun memberikan kemudahan kepada penyandang tunanetra untuk dapat mengenali objek tanpa harus membutuhkan orang lain dalam mengenali objek.

3.2 Saran

Aplikasi Pengenalan objek merupakan aplikasi yang dapat memudahkan penyandang tunanetra untuk melakukan pengenalan objek. Oleh karena itu di berikan saran yang dapat digunakan sebagai pengembangan aplikasi ini selanjutnya adalah aplikasi ini harus bisa lebih detail dalam mengenali objek dan juga respon deteksinya harus lebih cepat lagi agar memudahkan penyandang tunanetra dalam mengenali objek.

DAFTAR PUSTAKA

[1] J. J. Tula, Pelayanan Penyandang Disabilitas Dalam Menggunakan Berbagai Sarana Aksebilitas, 2015.

[2] D. P. Hallahan, J. M. Kauffman and P. C. Pullen, EXCEPTIONAL LEARNERS an Introduction to Special Education, 2009.

[3] R. S. Pressman, Rekayasa Perangkat Lunak : Pendekatan Praktisi, Yogyakarta: Andi Offset, 2013.

[4] T. Suharmini, "Psikologi Anak Berkebutuhan Khusus," p. 31, 2009.

[5] A. Widjaya, Seluk - Beluk Tunanetra & Strategi Pembelajarannya, Jogjakarta, 2013.

[6] T. Somantri, Psikologi Anak Luar Biasa, Bandung: PT. Refika Aditama, 2012.

[7] J. Birch, "Exploring Firebase MLKit on Android: Introducing MLKit," 22 May 2018. [Online]. Available: https://joebirch.co/2018/05/22/exploring-firebase-mlkit-on-android-introducing-mlkit-part-one/. [Accessed 05 Mei 2019].

[8] H. Dhawan, "Firebase ML Kit 101 : Image Labeling," 29 Oktober 2018. [Online]. Available: https://firebase.google.com/docs/ml-kit/label-images. [Accessed 24c Mei 2019].

[9] L. Moroney, "Using TensorFlow Lite on Android," Tensorflow Lite, 31 Maret 2018. [Online]. Available: https://www.tensorflow.org/lite/guide. [Accessed 24 Mei 2019].

[10] A. Mariam, "Pembangunan Perangkat Lunak Text To Speech," 2006.

[11] I. A. F Musyafi, "Membangun Aplikasi Chating Dengan Penerjemah Otomatis Berbasis Mobile," *KOMPUTA (Komputer dan Informatika*, 2015.