BAB II
LANDASAN TEORI

2.1 Kamus

Kata kamus diserap dari bahasa Arab *qamus* (قاموس), dengan bentuk jamaknya *qawamis*. Kata Arab itu sendiri berasal dari kata Yunani Ωκεανός (*okeanos*) yang berarti 'samudra'[5]. Sejarah kata itu jelas memperlihatkan makna dasar yang terkandung dalam kata kamus, yaitu wadah pengetahuan, khususnya pengetahuan bahasa, yang tidak terhingga dalam dan luasnya. Dewasa ini kamus merupakan khazanah yang memuat perbendaharaan kata suatu bahasa, yang secara ideal tidak terbatas jumlahnya.

Kamus adalah buku acuan yg memuat kata dan ungkapan, biasanya disusun menurut abjad berikut keterangan tentang makna, pemakaian, atau terjemahannya. Kamus disusun sesuai abjad dari A-Z dengan tujuan untuk memudahkan pengguna kamus dalam mencari istilah yang diinginkan. Kamus berfungsi untuk membantu seseorang mengenal perkataan baru. Selain menerangkan maksud kata, kamus juga mempunyai pedoman sebutan, asal usul (*etimologi*) suatu perkataan dan contoh penggunaan bagi suatu perkataan.

Kamus bisa ditulis dalam satu atau lebih. Oleh karena itu, kamus dibagi menjadi beberapa jenis, yaitu :

a. Kamus Ekabahasa.
b. Kamus Dwibahasa.
c. Kamus Aneka Bahasa.

2.2 Aplikasi

Aplikasi berasal dari kata *application* yang artinya penerapan. Aplikasi adalah penggunaan dalam suatu komputer, instruksi (*instruction*) atau pernyataan (*statement*) yang disusun sedemikian rupa sehingga komputer dapat memproses input menjadi output[6]. Aplikasi adalah penerapan dari rancang sistem untuk mengolah data yang menggunakan aturan atau ketentuan bahasa pemrograman tertentu.

Beberapa aplikasi yang digabung bersama menjadi suatu paket kadang disebut sebagai suatu paket atau *suite* aplikasi (*application suite*). Contohnya adalah *Microsoft Office* dan *OpenOffice*, yang menggabungkan suatu aplikasi pengolah kata, lembar kerja, serta beberapa aplikasi lainnya. Aplikasi-aplikasi dalam suatu paket biasanya memiliki antarmuka pengguna yang memiliki kesamaan sehingga memudahkan pengguna untuk mempelajari dan menggunakan tiap aplikasi. Sering kali, mereka memiliki kemampuan untuk saling berinteraksi satu sama lain sehingga menguntungkan pengguna. Contohnya, suatu lembar kerja dapat dibenamkan dalam suatu dokumen pengolah kata walaupun dibuat pada aplikasi lembar kerja yang terpisah.

2.3 Basis Data

Database atau basis data dapat diartikan sebagai kumpulan data *store* yang tersimpan dalam disk magnetik, disk optikal atau media penyimpanan sekunder lainnya, yang terdiri dari satu atau lebih tabel yang terintegrasi satu sama lain, di mana setiap pemakai (*user*) diberi wewenang (otorisasi) untuk dapat mengakses (mengubah, menghapus, menganalisis, menambah, memperbaiki) data dalam tabel-tabel tersebut[7]. Dengan kata lain, basis data merupakan sekumpulan data yang saling terkait (berelasi) antara satu dengan yang lainnya yang menggambarkan suatu organisasi (*enterprise*). Sedangkan sistem basis data adalah suatu sistem yang mengelola data dan dapat menyediakan data tersebut ketika diperlukan.

2.3.1 Komponen Utama Sistem Basis Data

Komponen utama sistem basis data adalah sebagai berikut :

a. Data yang disimpan dalam basis data.
b. Perangkat keras
c. Perangkat lunak
d. DBMS, Report-writer, design arts, dll.
e. Basis data
f. Pengguna.

2.3.2 Bahasa dalam Sistem Basis Data

Secara umum terdapat dua jenis bahasa dalam sistem basis data, yaitu:

a. *Data Definition Language*(DDL)

Perintah-perintah yang digunakan oleh *database administrator* untuk mendefinisikan sekema ke DBMS. Secara detil hal yang perlu dijabarkan pada DBMS adalah:

1) Nama basis data.
2) Nama seluruh berkas pada basis data.
3) Nama rekaman dan medan.
4) Deskripsi berkas, rekaman dan medan.
5) Nama medan kunci.
6) Nama indeks dan medan yang menjadi indeks.
7) Hal lain seperti ukuran basis data.

DDL juga digunakan untuk menciptakan, mengubah, dan menghapus basis data.

b. *Data Manipulation Language*(DML)

DML adalah perintah-perintah yang digunakan untuk mengubah, memanipulasi dan mengambil data pada basis data. Tindakan seperti menghapus, mengubah, dan mengambil data menjadi bagian dari DML. DML dibagi atas 2 jenis, yaitu:

1) Prosedural

   Menuntut pengguna menetukan data apa saja yang diperlukan dan bagaimana cara mendapatkannya.

2) Nonprosedural

Menuntut pengguna menentukan data apa yang diperlukan tetapi tidak perlu menyebutkan cara mendapatkannya.

2.3.3 Operasi Basis Data

Operasi dasar basis data diantaranya adalah pembuatan, penghapusan, pembuatan tabel, pengisian tabel yang dijelaskan sebagai berikut.

a. Pembuatan *database* baru (*create database*) yaitu menciptakan *database* baru.
b. Penghapusan *database* (*drop database*) yaitu penghapusan suatu *database*.
c. Pembuatan file/tabel baru ke dalam suatu databse yaitu penambahan file/tabel ke dalam database.
d. Penambahan/pengisian data baru ke sebuah file/tabel disebuah *database* (*insert*).
e. Perubahan data yang adapada sebuah file/tabel disebuah *database* (*update*).

Penghapusan data yang ada pada sebuah file/tabel disebuah *database* (*delete*).

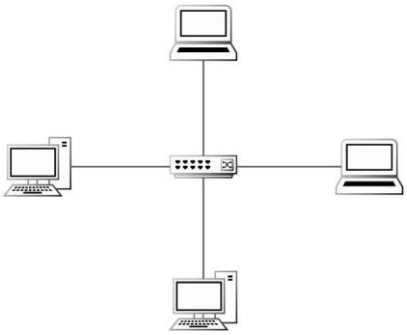
2.4 Jaringan Komputer

Jaringan komputer (*computer network*), adalah satu himpunan interkoneksi sejumlah komputer *autonomous* dalam suatu himpunan interkoneksi sejumlah komputer *autonomous*[8]. Jaringan komputer adalah kumpulan antara teknologi komputer, dan teknologi komunikasi. Gabungan teknlogi ini melahirkan pengolahan data yang dapat didistribusikan, mencakup pemakaian *database*, *software* dan *hardware*. Setiap komputer, printer atau periferal yang terhubung dengan jaringan disebut *node*. Sebuah jaringan komputer dapat memiliki dua, puluhan, ribuan atau bahkan jutaan node.

Jaringan komputer dapat diklasifikasikan menurut skala sebagai berikut :

a. Local Area Network (LAN):

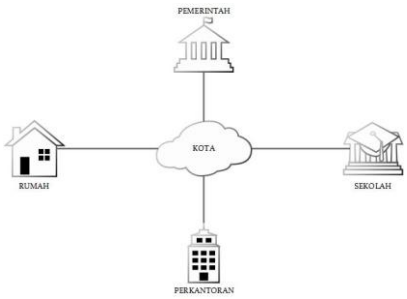
Suatu jaringan komputer yang menghubungkan suatu komputer dengan komputer lain dengan jarak yang terbatas. jaringan ini berada pada satu bangunan atau lokasi yang sama dengan jarak antar komputer yang dihubungkannya bisa mencapai 5 sampai 10 km dengan kecepatan transmisi data yang tinggi (mulai dari 10 Mbps sampai 100Mbps), dan menggunakan peralatan tambahan seperti *repeater*, *hub*, dan sebagainya. Gambar 2.1 menunjukan jaringan komputer dengan bentuk LAN.



Gambar 2.1 Local Area Network

b. Metropolitan Area Network (MAN):

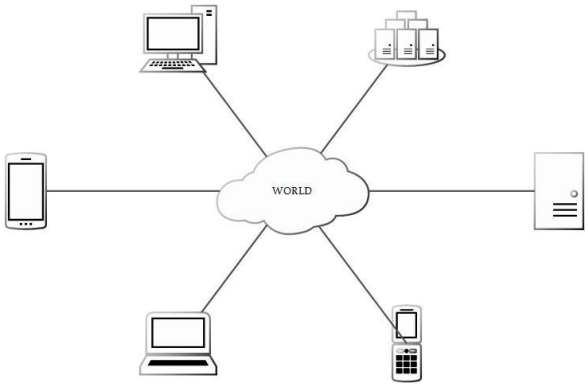
Jaringan ini merupakan gabungan beberapa *LAN* yang terletak pada satu kota (jangkauan 50-75 mil) yang dihubungkan dengan kabel khusus atau melalui saluran telepon, dengan kecepatan transmisi antara 34 sampai 155Mbps, dan menggunakan peralatan seperti *router*, telepon, *ATM switch*, dan antena parabola. Gambar 2.2 menunjukan jaringan komputer dengan bentuk MAN.



Gambar 2.2 Metropolitan Area Network

c. Wide Area Network (WAN):

Jaringan ini merupakan gabungan dari komputer *LAN* atau *MAN* yang ada di seluruh permukaan bumi ini yang dihubungkan dengan saluran telepon, gelombang elektromagnetik, atau satelit. Dengan kecepatan transmisi yang lebih lambat dari 2 jenis jaringan sebelumnya, dan menggunakan peralatan seperti *router, modem, WAN switches*. Gambar 2.3 menunjukan jaringan komputer dengan bentuk WAN.



Gambar 2.3 Wide Area Network

Pada dasarnya setiap jaringan komputer ada yang berfungsi sebagai *client* dan juga *server*. Tetapi ada jaringan yang memiliki komputer yang khusus didedikasikan sebagai *server* sedangkan yang lain sebagai *client*. Ada juga yang tidak memiliki komputer yang khusus berfungsi sebagai *server* saja. Karena itu berdasarkan fungsinya maka ada dua jenis jaringan komputer :

a. *Client-server*

Yaitu jaringan komputer dengan komputer yang didedikasikan khusus sebagai *server*. Sebuah *service* (layanan) bisa diberikan oleh sebuah komputer atau lebih. Contohnya adalah sebuah domain seperti www.unikom.ac.id yang dilayani oleh banyak komputer *webserver*. Atau bisa juga banyak *service*/layanan yang diberikan oleh satu komputer. Contohnya adalah server if.unikom.ac.id yang merupakan satu komputer dengan multi *service* yaitu *mailserver*, *webserver*, file *server*, *databaseserver* dan lainnya.

b. *Peer-to-peer*

Pada jaringan *peer to peer* semua komputer memiliki posisi setara/sejajar, dalam hierarki yang sama. Setiap komputer dapat menjadi klien terhadap komputer peer lainnya, setiap komputer dapat pula berbagi sumber daya dengan komputer yang berada dalam jaringan *peer-to-peer* ini. Sumber daya diletakkan secara desentralisasi pada setiap anggota jaringan, dan tidak memerlukan administrator jaringan.

c. *Hybrid*

Jaringan ini merupakan gabungan dari sifat pada *jaringan peer to peer* dan c*lient-server. Workgroup* yang terdiri dari beberapa komputer yang saling terhubung dapat mengelola sumber daya tanpa membutuhkan otorisasi dari administrator jaringan atau *server*. Pada jenis jaringan ini, terdapat pula sifat dari jaringan *client-server* sedemikian sehingga tingkat keamanan dapat lebih terjaga dan adanya *server* yang mempunyai suatu fungsi layanan tertentu, seperti sebagai *file server, print server, Database server, mail server*, dan lainnya.

Berdasarkan media transmisi datanya, jaringan komputer dapat dibedakan atas beberapa kategori, diantaranya :

a. Jaringan Berkabel (*WiredNetwork*)

Pada jaringan ini, untuk menghubungkan satu komputer dengan komputer lain diperlukan penghubung berupa kabel jaringan. Kabel jaringan berfungsi dalam mengirim informasi dalam bentuk sinyal listrik antar komputer jaringan.

b. Jaringan Nirkabel (*WirelessNetwork*)

Merupakan jaringan dengan medium berupa gelombang elektromagnetik. Pada jaringan ini tidak diperlukan kabel untuk menghubungkan antar komputer karena menggunakan gelombang elektromagnetik yang akan mengirimkan sinyal informasi antar komputer jaringan.

2.5 Internet

Internet (*interconnection-networking*) merupakan jaringan komputer yang saling terhubung secara mendunia yang memungkinkan jutaan jaringan terhubung antara satu dengan yang lainnya sehingga *user* lebih mudah terhubung dengan *user* lainnya[9]. Sedangkan menurut Onno W. Purbo, "Internet dengan berbagai aplikasinya seperti *web*, VoIP, E-Mail pada dasarnya merupakan media uang digunakan untuk mengefisienkan proses komunikasi".

Asal usul internet dari jaringan komputer yang disusun oleh ARPA (*Advance Research Project Agency*) yang bernaung dibawah departemen pertahanan Amerika Serikat atau DoD (*Departement of Defense*) sehingga terbentuk ARPAnet, jaringan komputer bentukan departemen pertahanan Amerika Serikat, tahun 70-an. Pada awalnya, ARPAnet hanya menghubungkan empat buah situs saja, yaitu:

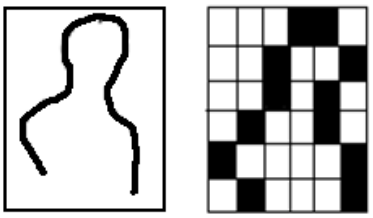
a. SRI (Stanford Research Institute)
b. USCB (University of California at Santa Barbara)
c. UCLA (University of California at Los Angeles)
d. University of Utah at Utah

Jaringan tersebut selanjutnya diperbaharui dan dikembangkan, dan saat ini digunakan sebagai tulang punggung jaringan informasi yang disebut dengan internet. Penelitian di Stanford menghasilkan sebuah protokol yang disebut TCP/IP, inilah yang berkembang terus hingga sekarang dan menjadi protokol standar dalam internet.

2.6 Citra Digital

Citra digital didefinisikan sebagai fungsi f(x,y) dua dimensi, dimana x dan y adalah koordinat spasial dan f(x,y) adalah disebut dengan intensitas atau tingkat keabuan citra pada koordinat x dan y. Jika x, y, dan nilai f terbatas dalam diskrit, maka disebut dengan citra digital. Citra digital dibentuk dari sejumlah elemen terbatas, yang masing-masing elemen tersebut memiliki nilai dan koordinat intensitas warna[10].

Citra digital dapat dibagi menjadi dua jenis. Jenis pertama adalah citra digital yang dibentuk oleh kumpulan *pixel* dalam array dua dimensi. Citra jenis ini disebut citra bitmap (*bitmap image*) atau citra raster (*raster image*). Jenis citra yang kedua adalah citra yang dibentuk oleh fungsi-fungsi geometri dan matematika. Jenis citra ini disebut grafik vektor (*vector graphics*).



Gambar 2.4 Citra Analog dan Citra Digital

2.6.1 Pengolahan Citra Digital

Pengolahan citra merupakan proses pengolahan dan analisis citra yang banyak melibatkan persepsi visual. Proses ini mempunyai ciri data masukan dan informasi keluaran yang berbentuk citra. Istilah pengolahan citra digital secara umum didefinisikan sebagai pemrosesan citra dua dimensi dengan komputer. Citra digital adalah barisan bilangan nyata maupun kompleks yang diwakili oleh bit-bit tertentu. Umumnya citra digital berbentuk persegi panjang atau bujur sangkar dengan lebar dan tinggi tertentu. Ukuran ini biasanya dinyatakan dalam banyaknya titik atau *pixel* sehingga ukuran citra selalu bernilai bulat. Setiap *pixel* memiliki koordinat sesuai posisinya dalam citra. Koordinat ini biasanya dinyatakan dalam bilangan bulat positif, yang dapat dimulai dari 0 atau 1 tergantung pada sistem yang digunakan. Setiap titik juga memiliki nilai berupa angka digital yang merepresentasikan informasi yang diwakili oleh titik tersebut. Beberapa pemrosesan gambar yang bisa dilakukan seperti memperbaiki kualitas gambar, dilihat dari aspek radiometrik (peningkatan kontras, transformasi warna, restorasi citra) dan dari aspek geometrik (rotasi, translasi, skala, transformasi geometrik), melakukan pemilihan ciri citra (*feature images*) yang optimal untuk tujuan analisis, melakukan proses penarikan informasi atau deskripsi obyek atau pengenalan obyek yang terkandung pada citra, melakukan kompresi atau reduksi data untuk tujuan penyimpanan data, transmisi data, dan waktu proses data.

2.6.2 Computer Vision

Terminologi lain yang berkaitan erat dengan pengolahan citra adalah *computer vision* atau *machine vision*. *Computer vision* mencoba meniru cara kerja sistem visual manusia (*human vision*)[11]. *Human vision* sesungguhnya sangat kompleks. Manusia melihat objek dengan indera penglihatan (mata), lalu citra objek diteruskan ke otak untuk diinterpretasi sehingga manusia mengerti objek apa yang tampak dalam pandangan matanya. Hasil interpretasi ini mungkin digunakan untuk pengambilan keputusan. *Computer Vision* merupakan proses otomatis yang mengintegrasikan sejumlah besar proses untuk persepsi visual, seperti akuisisi citra, pengolahan citra, analisis citra, pengenalan (*recognition*) dan membuat keputusan. Proses-proses yang terlibat dalam *computer vision* adalah sebagai berikut:

a. Memperoleh atau mengakuisisi citra digital
b. Melakukan teknik komputasi untuk memproses atau memodifikasi data citra (operasi-operasi pengolahan citra)
c. Menganalisis dan menginterpretasi citra dan menggunakan hasil pemrosesan untuk tujuan tertentu.
d. *Image understanding* yang digunakan untuk mengenali pola dan membuat keputusan.

Pengolahan citra merupakan tahap *preprocessing* (proses awal) pada *computer vision*, sedangkan pengenalan pola merupakan proses untuk menginterpretasikan citra. Teknik-teknik di dalam pengenalan pola memainkan peranan penting dalam *computer vision* untuk mengenali objek.

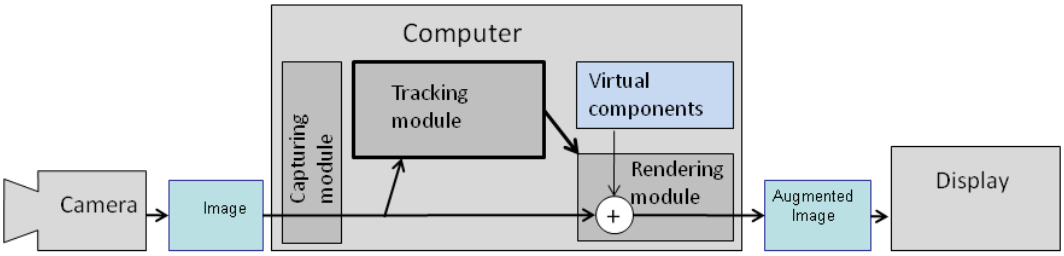
2.7 *Augmented Reality*

Augmented Reality (AR) adalah campuran dari gambar dunia nyata dengan benda-benda buatan atau informasi yang dihasilkan oleh komputer[12]. AR juga didefinisikan sebagai perpanjangan lingkungan pengguna dengan konten buatan. Benda-benda nyata menampilkan informasi yang tidak dapat diterima oleh pengguna dalam inderanya sendiri. Hal ini membuat realitas tertambah sesuai sebagai alat untuk membantu persepsi dan interaksi penggunanya dengan dunia

nyata. Informasi yang ditampilkan oleh benda maya membantu pengguna melaksanakan kegiatan-kegiatan dalam dunia nyata. AR dapat diaplikasikan untuk semua indera, termasuk pendengaran, dan sentuhan. Selain digunakan dalam bidang-bidang seperti kesehatan, militer, industry manufaktur, AR juga dapat digunakan untuk penerjemahan teks dalam berbagai bahasa dengan tambahan OCR yang dimiliki pada AR.

Ada tiga definisi dari AR yang diterima secara umum oleh Ronald Azuma pada tahun 1997, Azuma mendifinisikan bahwa:

a. Menggabungkan lingkungan nyata dan virtual.
b. Berjalan secara interaktif dan real time.
c. Tergolong kedalam lingkungan 3D[13].



Gambar 2.5 Proses *Augmented Reality*

Pada gambar 2.5 menunjukan gambaran sederhana dari proses *augmented reality*. *Capturing Module* menangkap gambar dari kamera. *Tracking Module* menghitung lokasi yang benar dan penempatan untuk lapisan virtual. *Rendering Module* menggabungkan gambar asli dan komponen virtual menggunakan pose yang dihitung dan kemudian membuat gambar virtual pada layar.

Tracking Module adalah inti dari sistem *augmented reality* yang menghitung pose relatif dari kamera secara *realtime*. Maksud dari pose menunjukan 6 posisi Degrees of Freedom (DOF) yaitu lokasi 3D dan penempatan 3D dari objek. *Tracking Module* memungkinkan sistem untuk menambahkan komponen virtual sebagai bagian dari adegan nyata. Perbedaan mendasar dibandingkan dengan alat pengolahan gambar lainnya adalah bahwa dalam *augmented reality* benda-benda virtual dipindahkan dan diputar dalam koordinat 3D bukannya koordinat gambar 2D.

2.7.1 Arsitektur

Pada tahun 1994, Milgram dan Kishino merumuskan kerangka kemungkinan penggabungan dan peleburan dunia nyata dan dunia maya ke dalam sebuah ***v****irtuality continuum.* Gambar 2.6 menunjukan,*augmented reality* lebih dekat ke sisi kiri yang menjelaskan bahwa lingkungan bersifat nyata dan benda bersifat maya. Sebaliknya *augmented virtuality* lebih dekat ke sisi kanan dalam kerangka tersebut, yang menjelaskan bahwa lingkungan bersifat maya dan benda bersifat nyata. Sehingga jika terjadi penggabungan antara a*ugmented reality* dengan *virtual reality* akan tercipta *mixed reality* [14].



Gambar 2.6 *Mixed Reality*

2.7.2 Metode Pengenalan Target

Augmented reality memiliki 2 metode pengenalan target. Metode pengenalan target yang pertama adalah *marker based tracking* dan yang kedua adalah *markerless.*

2.7.2.1 *Marker Based Tracking*

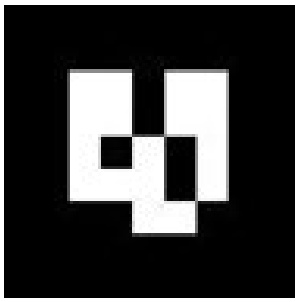
Augmented reality menyajikan informasi dalam konteks dunia nyata yang benar. Dalam melakukan hal ini sistem perlu tahu di mana pengguna berada dan apa yang pengguna cari. Biasanya, pengguna mengeksplorasi lingkungan melalui tampilan pada layar yang diperoleh dari kamera bersama dengan informasi berbentuk *augmented*. Dengan demikian dalam prakteknya, sistem perlu menentukan lokasi dan penempatan kamera. Dengan kalibrasi kamera, sistem ini kemudian mampu membuat benda-benda virtual di tempat yang benar. Maksud dari Tracking berarti menghitung pose relatif (lokasi dan penempatan) dari kamera secara real time, ini adalah salah satu komponen dasar dari *augmented reality*.

Augmented reality pada *marker based tracking* merupakan metode yang dikembangkan sejak tahun 80an. *Marker* merupakan ilustrasi dari hitam dan putih persegi dengan batas hitam tebal dan latar belakang putih. Komputer akan mengenali posisi dan orientasi *marker* dan menciptakan dunia *virtual* 3D yaitu titik (0,0,0) dan 3 sumbu yaitu X,Y,dan Z [15].



Gambar 2.7 Model virtual dari bangunan

Gambar 2.7 adalah hasil augmented dari AR ScaleModel yaitu model virtual dari bangunan yang ditempatkan diatas denah dalam skala dan pose yang benar menggunakan marker. Gambar 2.8 adalah contoh marker yang digunakan.



Gambar 2.8 Contoh marker yang digunakan

Dalam *visual tracking*, sistem menyimpulkan pose kamera berdasarkan pengamatan dari apa yang dilihatnya. Dalam lingkungan yang tidak diketahui, dibutuhkan beberapa waktu untuk mengumpulkan cukup data untuk dapat menyimpulkan pose dan kemudian menghitung estimasi pose dari waktu ke waktu. Sebagai lingkungan tidak diketahui ke sistem, sistem akan memilih orientasi sumbu koordinat secara acak. Selain itu, tidak mungkin untuk menyimpulkan skala yang benar semata-mata berdasarkan pengamatan visual. Salah satu solusi untuk mengatasi masalah-masalah ini adalah menambahkan

penanda yang mudah terdeteksi dalam lingkungan dan untuk mendeteksinya digunakan teknik *computer vision*. Marker adalah penanda dimana sistem dapat mendeteksi dari gambar yang didapat menggunakan pengolahan citra, pengenalan pola dan teknik *computer vision* . Setelah terdeteksi, kemudian mendefinisikan baik skala yang benar dan pose kamera. Pendekatan ini disebut pelacakan berbasis marker, dan secara luas digunakan dalam AR.

2.7.2.2 *Markerless*

Berbeda dengan *Marker Based Tracking*, metode *Markerless Augmented Reality* tidak lagi memerlukan penggunaan sebuah *marker* untuk menampilkan objek-objek digital. Metode ini memiliki beberapa teknik-teknik khusus yaitu *Face Tracking, 3D Object Tracking, Motion Tracking, GPS Based Tracking,* dan *Text Recognition* .

a. *Face Tracking*

Ciri pada wajah setiap manusia berbeda-beda, namun pasti setiap manusia memilki mata, hidung dan mulut. Penentuan titik koordinat bentuk pola wajah menggunakan algoritma Viola-Jones. Dengan algoritma Viola-Jones, teknik ini dapat mengenali pola pada mata, hidung dan mulut dan mengabaikan objek sekitarnya sehingga objek digital dapat diimpelementasikan pada wajah melalui sebuah kamera.

b. *3D Object Tracking*

Berbeda dengan *Face Tracking* yang hanya mengenali pola wajah secara umum, teknik *3D object Tracking* dapat mengenali semua bentuk benda yang ada disekitarnya seperti televisi, handphone, mobil, rumah dan lain-lain. *3D object Tracking* menggunakan hubungan geometris sebagai pendeteksi bentuk objek 3D. Hubungan geometris digunakan karena pada *3D object Tracking* hanya menggunakan fungsi kamera untuk mendeteksi benda-benda nyata yang ada di sekitar sebagai sebuah *marker*. Ada 2 Metode pengembangan *3D object Tracking* yang dipakai yaitu *bottom-up approach*(BUA) dan *top-down approac*h(TDA). 2 metode ini cocok untuk dipakai dalam penentuan titik koordinat bentuk objek 3D berbasis kamera.

c. *Motion Tracking*

Metode *marker*less dengan teknik ini yaitu dengan menangkap gerakan yang dilakukan pengguna. *Motion tracking* lebih dekat ke dunia *virtual reality*, karena di sini hasil yang tampak hampir keseluruhan dunia virtual. Dalam penerapannya, *motion tracking* memerlukan sebuah alat pendeteksi gerakan dari tubuh penggunanya. Salah satu alat tersebut adalah *Head Mounted Display*(HMDs), alat ini berfungsi sebagai indra penglihatan dunia *virtual*. Metode pengembangan yang dipakai dalam *motion tracking* yaitu metode TDA. Metode ini dibutuhkan karena *inertial orientation sensors* yang menjadi solusi dalam menangkap sebuah gerakan

d. *GPS Based Tracking*

Pengembangan teknik ini lebih diarahkan pada *smartphone,* karena teknologi GPS dan kompas yang tertanam pada *smartphone* tersebut. Dengan memanfaatkan fitur GPS yang berfungsi sebagai penentu lokasi pengguna pada saat itu berada sehingga lokasi terdekat yang ingin dituju dapat dilihat melalui implementasi augmented reality.

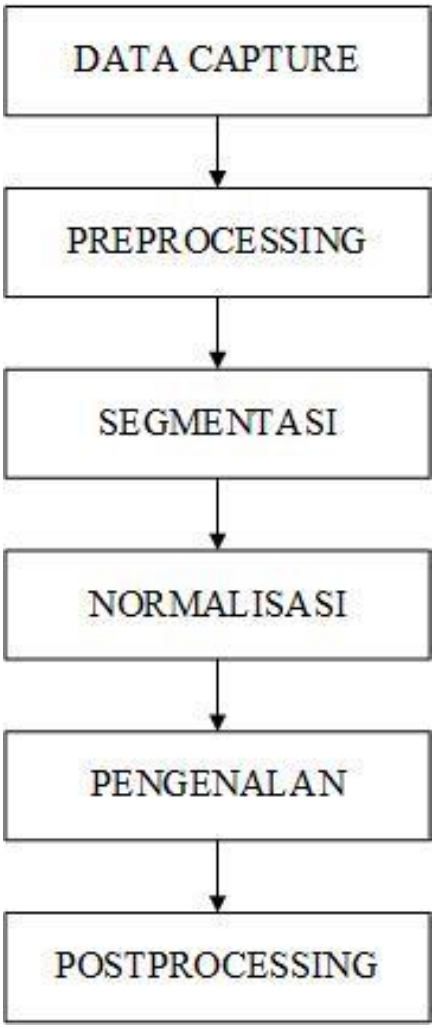
Teknik GPS *based tracking* sebenarnya membutuhkan peran kompas dan akselerometer sebagai pengatur ukuran layar secara *horizontal* dan *vertical* agar marker lokasi dapat dilihat ketika kamera *handset* berada posisi yang sesuai dengan lokasi tersebut. Namun ketika *handset* tidak berada dalam sudut pandang lokasi tersebut maka *marker* tersebut tidak akan tampak. Kebanyakan teknik GPS *based tracking* sudah memiliki *engine* pembantu(AR *browser*) yang telah dikembangkan oleh beberapa perusahaan sehingga mempermudah untuk mengembangkan teknik ini sesuai dengan keinginan.

e. *Text Recognition*

Teknik pendeteksian teks pada dasarnya menggunakan teknologi OCR (*Optical Character Recognition*). OCR yaitu sebuah sistem komputer yang dapat membaca huruf, baik yang berasal dari sebuah pencetak (printer atau mesin ketik) maupun yang berasal dari tulisan tangan. Penggunaan OCR pada *augmented reality* telah dikembangkan sebagai media translator pada *smartphone* untuk kemudahan proses terjemahan secara *realtime*.

2.8 *Optical Character Recognition*

Optical character recognition (OCR) adalah sebuah sistem komputer yang dapat membaca huruf, baik yang berasal dari sebuah pencetak (printer atau mesin ketik) maupun yang berasal dari tulisan tangan. OCR adalah aplikasi yang menerjemahkan gambar karakter (*image character*) menjadi bentuk teks dengan cara menyesuaikan pola karakter per baris dengan pola yang telah tersimpan dalam *database* aplikasi. Hasil dari proses OCR adalah berupa teks sesuai dengan gambar *output scanner* dimana tingkat keakuratan penerjemahan karakter tergantung dari tingkat kejelasan gambar dan metode yang digunakan[16]. Gambar 2.9 menunjukan urutan kerja pada OCR.



Gambar 2.9 Diagram kerja OCR

a. Data Capture

Data capture merupakan proses konversi suatu dokumen (hardcopy) menjadi suatu file gambar digital.

b. Preprocessing

Preprocessing merupakan suatu proses untuk menghilangkan bagian-bagian yang tidak diperlukan pada gambar input untuk proses selanjutnya.

c. Segmentasi

Segmentasi adalah proses memisahkan area pengamatan (region) pada tiap karakter yang akan dideteksi.

d. Normalisasi

Normalisasi adalah proses merubah dimensi region tiap karakter. Dalam OCR algoritma yang digunakan pada proses ini adalah algoritma scaling.

e. Pengenalan

Pengenalan merupakan proses untuk mengenali karakter yang diamati dengan cara membandingkan ciri-ciri karakter yang diperoleh dengan ciri-ciri karakter yang ada pada database.

f. Postprocessing

Pada umumnya proses yang dilakukan pada tahap ini adalah proses koreksi ejaan sesuai dengan bahasa yang digunakan.

2.8.1 *Preprocessing*

Untuk mengambil informasi dari *pixel* yang ada pada gambar, setiap komponen pada setiap sistem pengenalan karakter dirancang untuk mengurangi jumlah data. Langkah pertama adalah mengubah intensitas *pixel* pada gambar agar mudah digunakan pada proses selanjutnya. Tahapan yang dilakukan dalam *image processing* adalah *grayscale* dan binerisasi.

2.8.1.1 *Grayscale*

Citra digital *grayscale* atau *greyscale* adalah suatu citra dimana nilai dari setiap *pixel* merupakan *sample* tunggal. Citra yang ditampilkan dari citra jenis ini terdiri atas warna abu-abu, bervariasi pada warna hitam pada bagian yang intensitas terlemah dan warna putih pada intensitas terkuat. Citra *grayscale*

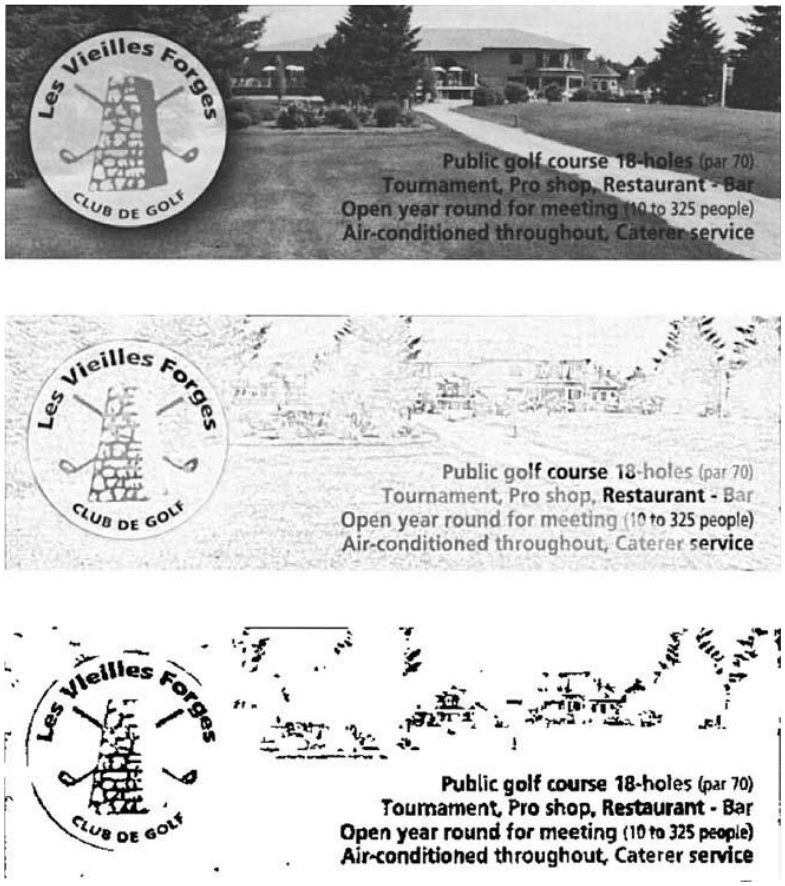
berbeda dengan citra ”hitam-putih”, dimana pada konteks komputer, citra hitam putih hanya terdiri atas dua warna saja yaitu ”hitam” dan ”putih” saja. Citra *grayscale* disimpan dalam format delapan bit untuk setiap sample *pixel*, yang memungkinkan sebanyak 256 intensitas. Nilai intensitas paling rendah merepresentasikan warna hitam dan nilai intensitas paling tinggi merepresentasikan warna putih.

Citra *grayscale* digunakan untuk menyederhanakan model citra. Citra berwarna terdiri dari tiga *layer* matrik yaitu R- *layer*, G-*layer* dan B-*layer*. Sehingga untuk melakukan proses-proses selanjutnya tetap diperhatikan tiga *layer* tersebut. Bila setiap proses perhitungan dilakukan menggunakan tiga *layer*, berarti dilakukan tiga perhitungan yang sama. Konsep tersebut diubah dengan mengubah tiga *layer* menjadi satu *layer* matrik *grayscale* dan hasilnya adalah citra *grayscale*. Dalam citra ini tidak ada lagi warna, yang ada adalah derajat keabuan. Untuk mengubah citra berwarna yang mempunyai nilai matrik masingmasing r, g dan b menjadi citra *grayscale* dengan nilai s, maka konversi dapat dilakukan dengan mengambil rata-rata dari nilai r, g dan b sehingga dapat dituliskan pada persamaan 2.1 :

$$s = \frac{r + g + b}{3} \tag{2.1}$$

2.8.1.2 Binerisasi

Binerisasi adalah proses mengubah citra berderajat keabuan menjadi citra biner atau hitam putih sehingga dapat diketahui daerah mana yang termasuk objek dan *background* dari citra secara jelas. Input untuk proses *thresholding* adalah *grayscale image* dan outputnya adalah *binary image.* Gambar 2.10 adalah contoh citra *grayscale* yang dirubah menjadi citra biner.



Gambar 2.10 Perubahan *grayscale image* menjadi *binary image*

Thresholding digunakan untuk memisahkan bagian citra yang dibutuhkan dan menghilangkan bagian citra yang tidak dibutuhkan [17]. Dalam pelaksanaannya *thresholding* membutuhkan suatu nilai yang digunakan sebagai nilai pembatas antara intensitas objek-objek utama dengan latar belakang, dan nilai tersebut dinamakan *threshold*. *Thresholding* digunakan untuk mempartisi citra dengan mengatur nilai intensitas semua *pixel* yang lebih besar dari nilai *threshold T* sebagai latar depan dan yang lebih kecil dari nilai *threshold T* sebagai latar belakang atau sebaliknya.

2.8.2 Segmentasi

Segmentasi merupakan suatu proses untuk mendapatkan area atau obyek yang diinginkan pada suatu citra dengan memisahkan area atau objek dari latar belakangnya. Cara yang digunakan untuk memisahkan objek yang akan dikenali adalah dengan memeriksa intensitas pixel di sekitar objek.

2.8.3 *Connected Component Analysis*

Proses *connected component analysis* memeriksa pixel-pixel di sekitar objek. Citra di-*scan* secara mendatar dari kiri ke kanan, dimulai dari atas citra hingga ke bawah citra untuk mencari pixel yang memiliki nilai yang sama dengan pixel di sekitarnya. Pada proses ini apabila ditemukan pixel hitam, akan dilakukan pengecekan nilai pixel hitam di sekitarnya. Jika semua pixel yang berada di sekitar pixel hitam tersebut memiliki nilai yang sama, maka pixel tersebut merupakan kesatuan dari suatu objek.

Analisis pixel-pixel tetangga bisa dilakukan dengan empat tetangga atau delapan tetangga. Analisis menggunakan teknik empat tetangga dilakukan dengan memeriksa pixel sebelah atas, bawah, kiri, dan kanan pixel. Dengan teknik tersebut, pixel tetangga dengan intensitas yang sama namun berada pada posisis diagonal akan dianggap objek lain. Sedangkan analisis dengan teknik delapan tetangga, pixel tetangga dengan intensitas yang sama namun berada pada posisi diagonal, akan dianggap objek yang sama.

2.8.4 Normalisasi

Normalisasi karakter merupakan proses operasi paling penting untuk pengenalan karakter. Citra dipetakan pada *pixel* dengan ukuran tertentu sehingga memberikan representasi dimensi yang tetap. Tujuan dari normalisasi citra adalah mengurangi resolusi citra yang berguna saat proses pengenalan citra dan juga meningkatkan akurasi pengenalan. Proses yang digunakan pada tahap normalisasi ini adalah proses penskalaan citra.

Scaling atau penskalaan pada citra disebut juga *image zooming*, yaitu proses untuk mengubah ukuran citra asli (*zoom in* / memperbesar ukuran citra asli atau *zoom out* / memperkecil ukuran citra asli). Proses perubahan ukuran resolusi citra dibutuhkan untuk menyesuaiakan resolusi citra masukan dengan resolusi citra *template*.

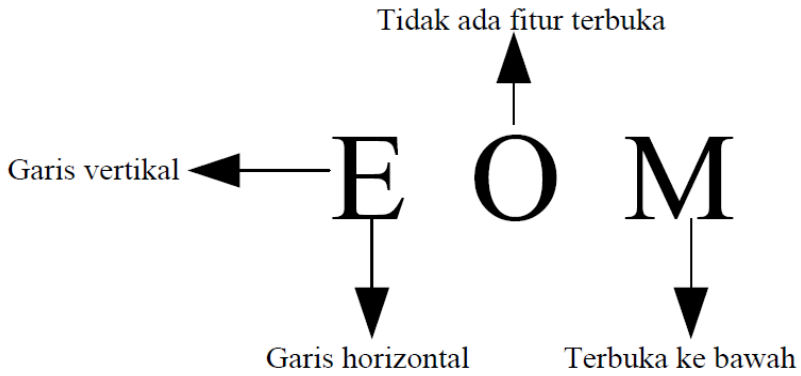
2.8.5 Pengenalan Pola

Pengenalan pola dapat dikatakan sebagai kemampuan mengenali objek berdasarkan ciri-ciri dan pengetahuan yang pernah diamatinya dari objek-objek tersebut. Tujuan dari pengenalan pola adalah mengklasifikasi dan mendeskripsikan pola atau objek kompleks melalui pengetahuan sifat-sifat atau ciri-ciri objek tersebut.

Ada tiga pendekatan dalam pengenalan pola yaitu secara sintaks, statistik, dan semantik. Pengenalan pola secara sintaks dilakukan berdasarkan ciri-ciri objek. Pengenalan pola secara statistik dilakukan berdasarkan komputasi matematis. Pendekatan dengan semantik berarti pola dikenali dalam tataran yang lebih abstrak.

2.8.5.1 *Feature Extraction*

Feature extraction merupakan salah satu cara untuk mengenali suatu objek dengan melihat ciri-ciri khusus yang dimiliki objek tersebut. Tujuan dari *feature extraction* adalah melakukan perhitungan dan perbandingan yang bisa digunakan untuk mengklasifikasikan ciri-ciri yang dimiliki oleh suatu citra.

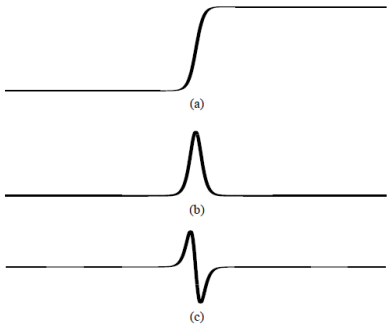


Gambar 2.11 Ilustrasi *Feature Extraction*

Gambar 2.11 merupakan ilustrasi citra karakter dengan ciri-cirinya. Ciri-ciri dari masing-masing citra *template* akan di simpan. Citra masukan yang akan dibandingkan akan dianalisis berdasarkan ciri-ciri citra. Ciri-ciri yang dimilik citra masukan akan diklasifikasikan terhadap ciri-ciri citra *template*.

a. Edge Detection

Edge atau ujung tepi adalah *visual feature* yang paling penting yang digunakan dalam proses deteksi teks. *Edge* atau ujung tepi adalah diskontinuitas tajam dalam profil level keabuan. *Edge detection* merupakan proses yang sangat kompleks, karena adanya *noise* dan ambiguitas dari "ketajaman"[20]. Dalam citra 2D, *edge* ditentukan oleh besar dan arahnya. *Edge detection* sering dilakukan oleh turunan spasial dan tresholding.



Gambar 2.12 (a) Sebuah Edge, (b) *First Order Derivative,* (c) *Second Order Derivative*

Potensi *edge* atau ujung tepi yang terdeteksi dengan menggunakan *Second Order Derivative* adalah *zero-crossing* dimana turunan nilai-nilai akan mengubah tanda-tanda seperti yang pada gambar 2.12 (c). *Second Order Derivative* lebih *noise* daripada *First Order Derivative.* Namun, salah satu keuntungan menggunakan *zero-crossing* dalam *edge detection* yaitu lebih mudah melacak *zero-crossing* untuk mendapatkan kontur terdekat daripada melacak titik gradien maksimum dengan threshold. Untuk mencapai edge detection yang stabil diantara noise, Marr dan Hildreth menyarankan memperhalus citra dengan smoother gaussian sebelum menerapkan operator laplacian. Operator yang dihasilkan disebut sebagai Laplacian of Gaussian (LOG), yang juga dapat diimplemantasikan oleh Difference of Gaussian (DOG)

$$\nabla^2 I = \left(\frac{\partial^2 I}{\partial^2 x}, \frac{\partial^2 I}{\partial x \partial y}, \frac{\partial^2 I}{\partial^2 y} \right) \tag{2.2}$$

2.9 Aplikasi Pendukung

Aplikasi pendukung merupakan aplikasi yang mendukung sistem dalam proses pembangunan maupun dalam silkus hidup sistem tersebut. Aplikasi pendukung ini sangat membantu terhadap fungsional dari sistem.

2.9.1 Android

Android adalah sebuah sistem operasi berbasis *Linux kernel* dan dirancang untuk perangkat *mobile* dengan layar sentuh seperti *smartphone* dan komputer tablet. Awal mulanya dikembangkan oleh Android, Inc., yang didukung oleh Google secara finansial dan kemudian dibeli pada tahun 2005. Android ini diresmikan pada tahun 2007 seiring dengan berdirinya *Open Handset Alliance* yang merupakan konsorsium dari *hardware*, *software* dan perusahaan telekomunikasi yang ditujukan untuk memanjukan standar terbuka untuk perangkat *mobile*.

2.9.1.1 *Version*

Version history dari sistem operasi *mobile* Android dimulai dari Android *beta* yang dirilis pada Nopember 2007. Android 1.0 yang merupakan versi komersil pertama, dirilis pada September 2008[18]. Android berada di bawah pengembangan Google dan OHA (*Open Handset Alliance*) dan memiliki banyak perbaharuan pada sistem operasinya sejak pertama kali dirilis. Tabel 2.1 merupakan sejarah versidari Android berdasarkan API levelnya.

Tabel 2.1 Versi Android Berdasarkan API Level

API Level	Versi	Code-Name	Tanggal Rilis
API Level 1	1.0		28 September 2008
API Level 2	1.1		9 Februari 2009
API Level 3	1.5	Cupcake	30 April 2009
API Level 4	1.6	Donut	15 September 2009
API Level 5	2.0	Éclair	26 Oktober 2009
API Level 6	2.0.1		3 Desember 2009
API Level 7	2.1		12 Januari 2010
API Level 8	2.2	Froyo	20 May 2010
	2.2.1		18 Januari 2011
	2.2.2		22 Januari 2011
	2.2.3		21 November 2011
API Level 9	2.3	Gingerbread	6 Desember 2010
	2.3.1		Desember 2010
	2.3.2		Januari 2011
API Level 10	2.3.3		9 Februari 2011
	2.3.4		28 April 2011
	2.3.5		25 Juli 2011
	2.3.6		2September 2011
	2.3.7		21 September 2011
API Level 11	3.0	Honeycomb	22 Februari 2011
API Level 12	3.1		10 Mei 2011
API Level 13	3.2		15 Juli 2011
	3.2.1		20 September 2011
	3.2.2		30 Agustus 2011
	3.2.3		
	3.2.4		Desember 2011
	3.2.5		Januari 2012
	3.2.6		Februari 2012
API Level 14	4.0	Ice Cream Sandwich	19 Oktober 2011
	4.0.1		21 Oktober 2011
	4.0.2		28 November 2011
API Level 15	4.0.3		16 Desember 2011
	4.0.4		29 Maret 2012
API Level 16	4.1	Jelly Bean	9 Juli 2012
	4.1.1		23 July 2012
	4.1.2		9 Oktober 2012
API Level 17	4.2		13 November 2012
	4.2.1		27 November 2012
	4.2.2		11 February 2013
API Level 18	4.3		24 Juli 2013
	4.3.1		3 Oktober 2013
API Level 19	4.4	KitKat	31 Oktober 2013
	4.4.1		5 Desember 2013
	4.4.2		9 Desember 2013

2.9.1.2 Fitur

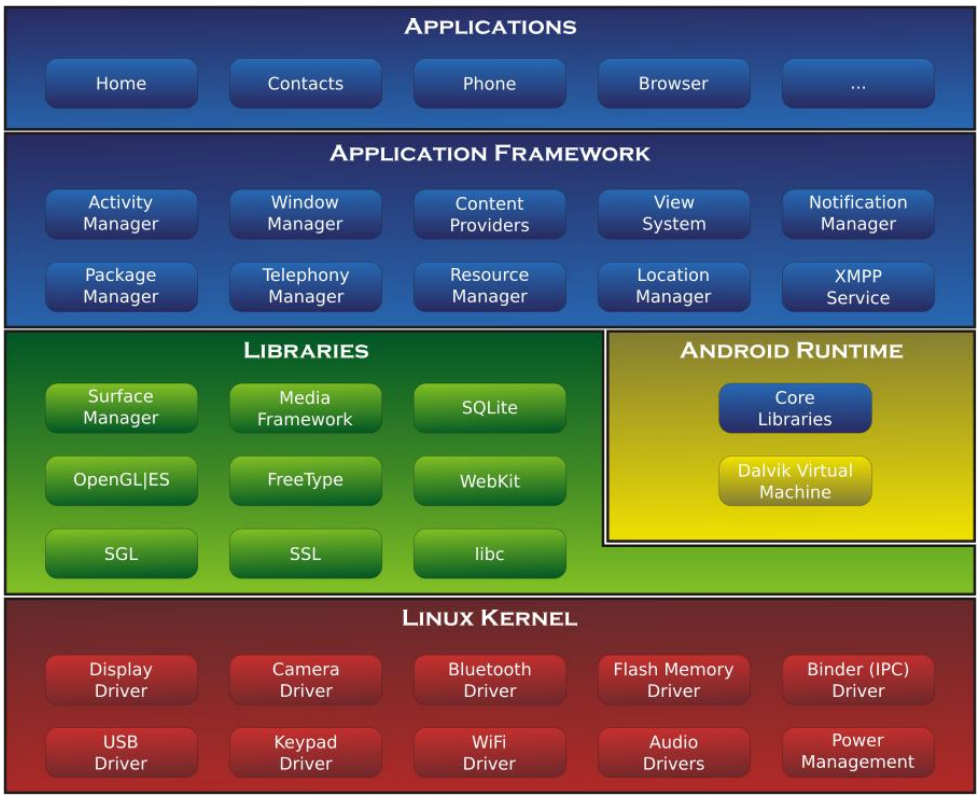
Android memiliki banyak sekali fitur. Fitur-fitur pada Android dapat diklasifikasikan pada tabel 2.2.

Tabel 2.2 Klasifikasi Fitur Android

General	*Messaging*
	Web Browser
	Voice Based Features
	Multi-touch
	Multitasking
	Screen Capture
	Voice Calling
	Multi Language Support
	Accessbility
Connectivity	*Connectivity*
	Bluetooth
	Tethering
Media	*Streaming Media Support*
	Media Support
	External Storage
Hardware Support	
Fitur lainnya	*Java Support*
	Handset Layout
	Storage

2.9.1.3 Arsitektur

Google sebagai pencipta Android yang kemudian diasuh oleh *Open Handset Alliance* mengibaratkan Android sebagai sebuah tumpukan software. Setiap lapisan dari tumpukan ini menghimpun beberapa program yang mendukung fungsi-fungsi spesifik dari sistem operasi. Tumpukan paling bawah adalah kernel. Google menggunakan kernel Linux versi 2.6 untuk membangun Android, yang mencakup memory management, security setting, power management, dan beberapa driver hardware. Bertempat di level yang sama dengan library adalah lapisan *runtime* yang mencakup serangkaian inti library Java. Dengannya, para programmer dapat mengembangkan aplikasi untuk Android menggunakan bahasa pemrograman Java. Lapisan selanjutnya adalah application framework, yang mencakup program untuk mengatur fungsi- fungsi dasar smartphone. Gambar 2.13 menunjukan arsitektur yang dimiliki android.



Gambar 2.13 Arsitektur Android

a. Linux Kernel

Android dibangun di atas kernel Linux 2.6 hingga 3.4 Namun secara keseluruhan android bukanlah linux, karena dalam android tidak terdapat paket standar yang dimiliki oleh linux lainnya. Linux merupakan sistem operasi terbuka yang handal dalam manajemen memori dan proses. Oleh karenanya pada android hanya terdapat beberapa servis yang diperlukan seperti keamanan, manajemen memori, manajemen proses, jaringan dan driver. Kernel linux menyediakan driver layar, kamera, *keypad*, Wi-Fi, *Flash Memory*, audio, dan IPC (*Interprocess Communication*) untuk mengatur aplikasi dan lubang keamanan. Kernel juga berperan sebagai lapisan abstract antara *hardware* dengan tumpukan aplikasi yang berjalan diatasnya.

b. *Libraries*

Android menggunakan beberapa *library* yang terdapat pada C/C++ dengan standar Berkeley Software Distribution (BSD) hanya setengah dari yang aslinya untuk tertanam pada kernel Linux. Beberapa *library* diantaranya:

1) *System C Library* merupakan implementasi dari standar sistem *library*C (libc) untuk perangkat berbasis Linux.
2) Media *Libraries* untuk memutar dan merekam berbagai macam format audio dan video.
3) *Surface Manager* untuk mengatur hak akses layer dari berbagai aplikasi.
4) *Graphic Library* termasuk didalamnya SGL dan OpenGL, untuk tampilan 2D dan 3D.
5) FreeType untuk rendering font bitmap dan vector.
6) SQLite untuk mengatur relasi *database* yang digunakan pada aplikasi.
7) SSl dan WebKit untuk *browser* dan keamanan internet.

c. Android *Runtime*

Pada android tertanam paket *library* inti yang menyediakan sebagian besar fungsi android. Inilah yang membedakan Android dibandingkan dengan sistem operasi lain yang juga mengimplementasikan Linux. Android *Runtime* merupakan mesin virtual yang membuat aplikasi android menjadi lebih tangguh dengan paket *library* yang telah ada.

d. *ApplicationFramework*

Framework aplikasi menyediakan kelas-kelas yang dapat digunakan untuk mengembangkan aplikasi android. Selain itu, juga menyediakan abstraksi generik untuk mengakses perangkat, serta mengatur tampilan *user interface* dan sumber daya aplikasi.

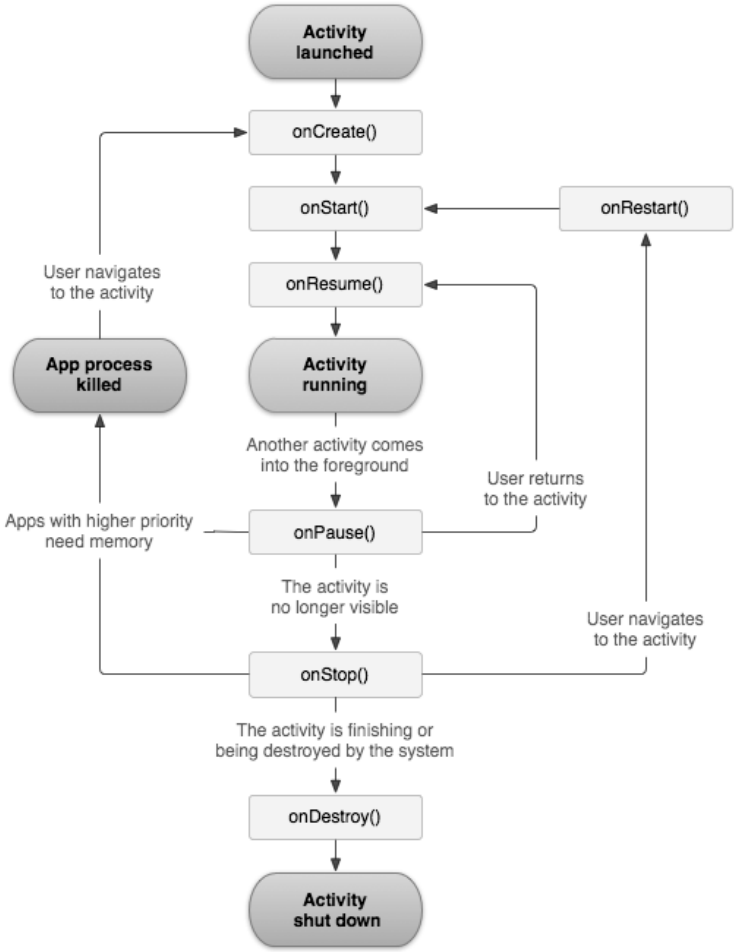
Bagian terpenting dalam *framework* aplikasi android adalah sebagai berikut :

1) *Activity Manager*
2) *Content Providers*
3) *Resuource Manager*
4) *Location Manager*
5) *Notification Manager*

e. *Application Layer*

Puncak dari diagram arsitektur android adalah lapisan aplikasi dan *widget*. Lapisan aplikasi merupakan lapisan yang paling tampak pada pengguna ketika menjalankan program. Pengguna hanya akan melihat program ketika digunakan tanpa mengetahui proses yang terjadi dibalik lapisan aplikasi. Lapisan ini berjalan dalam Android *runtime* dengan menggunakan kelas dan service yang tersedia pada *framework* aplikasi.

Lapisan aplikasi android sangat berbeda dibandingkan dengan sistem operasi lainnya. Pada android semua aplikasi, baik aplikasi inti (*native*) maupun aplikasi pihak ketiga berjalan diatas lapisan aplikasi dengan menggunakan pustaka API (*Application Programming Interface*) yang sama. Android mempunyai siklus hidup aplikasi seperti pada Gambar 2.14.



Gambar 2.14 Silkus Hidup Aplikasi Android

2.9.1.4 Android SDK

Android SDK (Android *Software Development Kit*) merupakan perangkat lunak Android untuk proses pengembangan dimana proses aplikasi baru diciptakan untuk sistem operasi Android. Aplikasi tersebut biasanya dikembangkan dalam bahasa pemrograman Java dengan menggunakan Software Development Kit (SDK) Android, tetapi *tools* lainnya juga tersedia. Pada Juli 2013, lebih dari satu juta aplikasi telah dikembangkan untuk Android, dengan lebih 25 juta unduhan. Sebuah riset menunjukkan bahwa lebih 67% pengembang aplikasi seluler menggunakan platform Android[19]. Pada kuartal 2 tahun 2012, sekitar 105 juta unit telepon cerdas Android telah dipasarkan, dengan total pangsa pasar 68% secara keseluruhan.

Saat ini Android SDK termasuk seperangkat alat pengembangan yang terdiri dari debugger, *library*, sebuah handset emulator berdasarkan QEMU, dokumentasi, kode contoh, dan tutorial. Platform pengembangan saat ini, dapat berjalan pada modern Linux *distribution,* Microsoft Windows XP dan lebih baru serta Mac OS X 10.5.8 keatas.

IDE yang resmi didukung oleh Android adalah Eclipse menggunakan ADT (Android *Development Tools*) *Plugin* meskipun IntelliJ DEA mendukung penuh pengembangan Android dan NetBeans IDE pun mendukung pengembangan Android melalui sebuah *plugin*. Selain itu, pengembangdapat menggunakaneditor teks apapununtuk mengeditfileJava dan XML, kemudianmenggunakan *command line tools*(diperlukanJava *Development* KitdanApacheAnt) untuk menciptakan, membangun dan melakukan debug terhadap aplikasi Android serta pengendalian perangkat Android yang terpasang (misalnya, memicu reboot, dan menginstal paket perangkat lunak jarak jauh).

2.9.2 SQLite

Android memiliki fasilitas untuk membuat *database* yang dikenal dengan SQLite. SQLite adalah salah satu *software* yang bersifat *embedded* yang sangat populer. SQLite memiliki kelebihan yaitu menggunakan kombinasi SQL *interface* dan penggunaan *memory* yang sangat sedikit dengan kecepatan yang sangat cepat. SQLite di android termasuk dalam Android *runtime*, sehingga setiap versi android dapat membuat *database* dengan SQLite.

Untuk membuat dan membuka *database* pada android yang paling baik adalah menggunakan *libraries* Importandroid.database.sqlite.SQLiteOpenHelper yang menyediakan tiga metode yaitu:

a. *Constructor*, menyediakan representasi versi dari *database* dan skema *database* yang digunakan
b. *onCreate(),* menyediakan SQLiteDatabase *object* yang digunakan dalam definisi tabel dan inisialisasi data
c. *onUpgrade(),* menyediakan fasilitas konversi *database* dari *database* versi lama ke *database* versi baru atau sebaliknya.

Adapun fitur-fitur yang ada pada SQLite adalah :

a. *Serverless*, SQLite tidak memerlukan proses pada *server* atau sistem untuk menjalankannya, melainkan hanya sebuah *file* yang diakses oleh *library* SQLite.
b. *Cross Platform*, semua instan *database* berada dalam sebuah *file* yang *cross-platform*, tidak memerlukan administrasi.
c. *Zero Configuration*, Tidak ada *server* berarti tidak perlu setup, membuat sebuah *database* instan adalah semudah anda membuat *file* biasa.
d. *Small Runtime Footprint*, untuk membangun *database* SQLite hanya membutuhkan kurang dari satu *megabyte library* (kode program) dan hanya membutuhkan beberapa *megabyte memory*, bahkan dengan beberapa *adjustment* baik ukuran *library* maupun memory dapat diperkecil.
e. *Self-Contained*, sebuah *library* mengandung keseluruhan dari sistem *database*, yang langsung terintegrasi pada sebuah aplikasi program.
f. *Ransactional*, SQLite transaction memperbolehkan aksi penyimpanan melalui beberapa proses *thread*.
g. *Full Featured*, SQLite mensupport hampir sebagian besar standar SQL92 (SQL2).
h. *Highly Reliable*, Tim pengembang SQLite mengembangkan melalui kode program yang sangat serius serta telah melewati proses *testing*.

Eclipse tidak mendukung pembuatan *database* SQlite di dalam fitur-fiturnya. Untuk itu diperlukan *software* pembantu yang dapat membuat sebuah *database* kosong yang nantinya *database* tersebut dapat digunakan oleh Android untuk dimodifikasi atau diakses oleh suatu program Android tertentu.

2.9.3 Eclipse

Eclipse adalah sebuah IDE (*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua platform (*platform-independent*). Berikut ini adalah sifat dari Eclipse:

a. Multi-platform
b. Mulit-language
c. Multi-role

Eclipse pada saat ini merupakan salah satu IDE favorit dikarenakan gratis dan *open source*, yang berarti setiap orang boleh melihat kode pemrograman perangkat lunak ini. Selain itu, kelebihan dari Eclipse yang membuatnya populer adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan komponen yang dinamakan *plug-in*.

2.9.3.1 Arsitektur

Sejak versi 3.0, Eclipse pada dasarnya merupakan sebuah *kernel*, yang mengangkat *plug-in*. Apa yang dapat digunakan di dalam Eclipse sebenarnya adalah fungsi dari *plug-in* yang sudah diinstal. Ini merupakan basis dari Eclipse yang dinamakan *Rich Client Platform* (RCP). Berikut ini adalah komponen yang membentuk RCP:

a. *Core platform*
b. OSGi
c. SWT (*Standard Widget Toolkit*)
d. JFace
e. *Eclipse Workbench*

Secara standar Eclipse selalu dilengkapi dengan JDT (*Java Development Tools*), *plug-in* yang membuat Eclipse kompatibel untuk mengembangkan program Java, dan PDE (*Plug-in Development Environment*) untuk mengembangkan *plug-in* baru[20]. Eclipse beserta *plug-in*-nya diimplementasikan dalam bahasa pemrograman Java.

Konsep Eclipse adalah IDE yang terbuka (*open*), mudah diperluas (*extensible*) untuk apa saja, dan tidak untuk sesuatu yang spesifik. Jadi, Eclipse tidak saja untuk mengembangkan program Java, akan tetapi dapat digunakan untuk berbagai macam keperluan, cukup dengan menginstal *plug-in* yang dibutuhkan. Apabila ingin mengembangkan program C/C++ terdapat *plug-in* CDT (*C/C++ Development Tools*). Selain itu, pengembangan secara visual bukan hal yang tidak mungkin oleh 60

Eclipse, *plug-in* UML2 tersedia untuk membuat diagram UML. Dengan menggunakan PDE setiap orang bisa membuat *plug-in* sesuai dengan keinginannya.

2.9.4 Vuforia SDK *(Software Development Kit)*

Vuforia adalah *Augmented Reality* SDK (*Software Development Kit*) untuk pembangunan AR pada perangkat *mobile* yang disediakan oleh Qualcomm[21]. Target pada vuforia merupakan objek pada dunia nyata yang dapat dideteksi oleh kamera, untuk menampilkan objek virtual. Vuforia menggunakan teknologi *computer vision* untuk mengenali dan membuat pelacakan objek yang ditangkap oleh kamera video secara *realtime*.

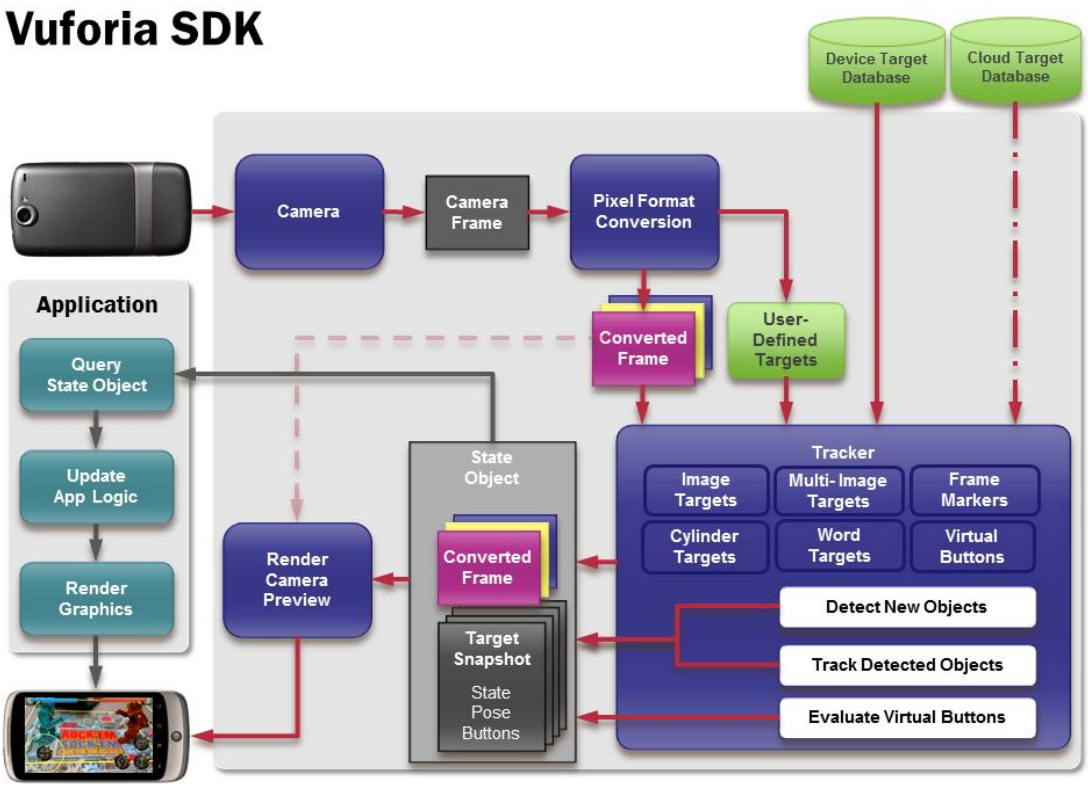
Vuforia SDK mendukung berbagai jenis target baik 2D dan 3D termasuk *multi-target configurations, markerless image targets* dan *frame markers*. Vuforia juga memiliki fitur tambahan lain seperti *localized occlusion detections* menggunakn *virtual buttons, image targets* secara *realtime* dan kemampuan untuk mengkonfigurasi ulang dan menciptakan *target sets*. Vuforia SDK mendukung Android OS minimal versi 2.3, dan OpenGL ES minimal versi 2.0. *Environment* pengembangan yang disaran kan adalah Microsoft Windows 7 32/64-bit atau Windows XP. Komponen untuk membangun *actual code* (JDK, Eclipse+ADT dan gcc) tersedia di beberapa platform.Vuforia SDK juga mendukung pengembangan untuk iOS sehingga aplikasi yang dikembangkan kompatibel dengan berbagai perangkat mobile iPhone, tetapi untuk sementara tidak dapat mendukung platform tersebut dengan dokumen rinci. Gambar 2.15 adalah *environment* pengembangan da *platform* pengembangan dari Vuforia.

Development Environment	Development Platform			
	Native SDK		Unity Extension	
	Android	iOS	Android	iOS
Windows	Yes	--	Yes, multi-platform deployment	
MacOS	Yes	Yes	Yes, multi-platform deployment	
Linux	Yes	--	--	--

Gambar 2.15 *Environment* Pengembangan dan *Platform* Pengembangan

2.9.4.1 Arsitektur

Vuforia SDK berbasis *Augmented Reality* menggunakan layar perangkat mobile sebagai "*magic lens*" sebuah kaca untuk melihat ke dunia *Augmented Reality* dimana dunia nyata dan dunia virtual muncul secara berdampingan. Komponen-komponen yang dimiliki vuforia sebagai pendukung dalam membuat *Augmented reality* dapat dilihat pada gambar 2.16.



Gambar 2.16 Arsitektur Vuforia SDK

Vuforia SDK memerlukan beberapa komponen penting agar dapat bekerja dengan baik. Komponen - komponen tersebut antara lain :

a. Kamera

Komponen kamera dibutuhkan untuk memastikan bahwa setiap *frame preview* ditangkap dan diteruskan secara efisien ke *tracker*. Pengembang hanya perlu menginisialisasi kamera kapan untuk memulai dan menghentikan pengambilan. Frame kamera secara otomatis diberikan dengan format gambar dan ukuran yang tergantung pada perangkat yang digunakan.

b. *Image Converter*

Pixel Format Converter mengkonversi dari format kamera (misalnya YUV12) ke dalam format yang sesuai untuk melakukan render pada OpenGL ES (misalnya RGB565) dan untuk pelacakan (misalnya, *luminance*) secara internal. Konversi ini juga meliputi *downsampling* untuk memiliki gambar kamera dalam resolusi berbeda yang tersedia di *stack* bingkai yang dikonversi.

c. *Tracker*

Komponen *tracker* berisi algoritma *computer vision* yang dapat mendeteksi dan melacak benda-benda dunia nyata yang ada pada frame *kamera video*. Berdasarkan gambar dari kamera, algoritma yang berbeda bertugas untuk mendeteksi target baru atau marker dan mengevaluasi *virtual button*. Hasilnya akan disimpan dalam *state object* yang akan digunakan oleh *video background renderer* dan dapat diakses dari *application code*. *Tracker* dapat membuat beberapa dataset pada waktu yang sama dan mengaktifkannya.

Subtipe *Tracker* pada gambar 2.16 yang didefinisikan dalam Vuforia adalah sebagai berikut :

1) *ImageTarget*, objek dari *Trackable class* yang terdiri dari gambar.
2) *CylinderTarget*, objek dari *Trackable class* yang terdiri dari gambar yang diterapkan pada silinder atau permukaan kerucut.

3) *MultiTarget*, objek dari *Trackable class* yang terdiri dari beberapa gambar yang dikombinasikan dalam konfigurasi spasial yang diberikan.
4) *Marker*, objek dari *Trackable class* yang terdiri dari bingkai marker.
5) *Word*, objek dari *Trackable class* yang mewakili elemen teks.
6) *VirtualButton*, merupakan daerah persegi panjang dari *ImageTarget* dimana dapat memicu aktivitas jika disentuh atau terhalangi dalam tampilan kamera.

d. *Video Background Renderer*

Modul *Video Backgroud Renderer* melakukan *render* gambar dari kamera yang tersimpan di dalam *state object*. Kinerja dari *video background renderer* sangat bergantung pada *device* yang digunakan.

e. *Application Code*

Pengembang aplikasi harus mennginisialisasi semua komponen di atas dan melakukan tiga tahapan penting dalam *application code*, untuk setiap frame yang diproses *state object* akan diperbaharui dan memanggil method *appliction render*. Berikut adalah tahapan yang harus dilakukan, seperti:

1) Melakukan *query* terhadap *state object* pada target baru yang terdeteksi atau marker atau state baru dari unsur-unsur ini.
2) Melakukan *update* terhadap *aplication logic* dengan memasukan data baru.
3) Melakukan *render augmented* pada *graphics overlay*.

f. *Device Databases*

Database perangkat yang dibuat menggunaan *Online Target Manager*. Aset database target pada perangkat yang telah di-*download* mengandung file konfigurasi XML yang memungkinkan pengembang untuk mengkonfigurasi fitur *trackable* tertentu dan file biner yang berisi database *trackable*. Aset ini dikompilasi oleh pengembang aplikasi ke dalam paket installer aplikasi dan digunakan pada saat *runtime* oleh Vuforia SDK.

g. *Cloud Databases*

Database *Cloud* dapat dibuat dengan menggunakan *Target Manager* atau menggunakan *Vuforia Web Services API*. Dilakukan *query* pada *runtime* aplikasi menggunakan fitur *cloud recognition* dimana fitur ini akan melakukan pencarian visual dalam *cloud* menggunakan gambar kamera yang dikirim. Selain data target, target ditetapkan dapat berisi metadata yang kembali setelah dilakukan *query*.

h. *User-Defined Targets*

User-Defined Targets adalah target yang ditetapkan pengguna fitur yang didukung dengan suatu pendekatan dasar yang berbeda. Fitur ini memungkinkan untuk membuat target *on-the-fly* dari gambar kamera. Sebuah komponen pembangun dipanggil untuk menjadi *trigger* penciptaan *user-target* baru. Target yang dikembalikan ini diambil dari *cache,* tapi dipertahankan hanya untuk memberikan AR *session.*

i. *Word Targets*

Vuforia SDK dapat mengenali kata-kata dan melacaknya sama seperti target lain, dengan dua mode pengakuan yang tersedia, "*Words*" dan "*Characters*". Bila menggunakan " *Words* " mode, kata diakui (dan kemudian dilacak) jika termasuk dalam *Wordlist* yang disediakan, dimana *Wordlist* dapat disimpan pada perangkat dan dimuat oleh aplikasi pada saat *runtime*. *Wordlist* juga dapat diperpanjang dengan kata-kata tambahan, dan filter kata tertentu dapat dimuat dan diterapkan melalui SDK API. Selain itu, ketika menggunakan " *Characters* " mode, kata apapun yang dihasilkan dari urutan karakter yang sembarang (termasuk nomor) juga dapat terdeteksi.

2.9.4.2 *Trackable Base Class*

Konsep dari target dalam Vuforia SDK direpresentasikan dalam *two-fold manner*, melalui definisi dari dua kelas utama yaitu *Trackable* dan *TrackableResult*.

Trackable adalah kelas dasar yang digunakan untuk mengidentifikasi semua objek dunia nyata dimana Vuforia SDK dapat deteksi dan lacak dalam ruang 3D.

Trackable Class mendefinisikan sifat dasar untuk objek target yang diberikan, seperti nama, jenis, ID, dan data yang ditentukan pengguna. Subtipe *Trackable* yang spesifik akan menentukan atribut selanjutnya dimana akan berlaku bagi tipe tertentu dari objek. Subtipe *Trackable* yang didefinisikan saat ini dalam Vuforia adalah sebagai berikut :

a. *ImageTarget*, objek dari *Trackable class* yang terdiri dari gambar.
b. *CylinderTarget*, objek dari *Trackable class* yang terdiri dari gambar yang diterapkan pada silinder atau permukaan kerucut.
c. *MultiTarget*, objek dari *Trackable class* yang terdiri dari beberapa gambar yang dikombinasikan dalam konfigurasi spasial yang diberikan.
d. *Marker*, objek dari *Trackable class* yang terdiri dari bingkai marker.

Word, objek dari *Trackable class* yang mewakili elemen teks.

Tabel 2.3 berisi parameter-parameter yang dipakai di kelas *Trackable.*

Tabel 2.3 Parameter *Trackable*

Trackable Type	Enum mewakili jenis trackable, yaitu salah satu hal berikut: a. UNKNOWN_TYPE: trackable dari jenis yang tidak diketahui b. IMAGE_TARGET: trackable dari tipe ImageTarget c. CYLINDER_TARGET: trackable dari tipe CylinderTarget d. MULTI_TARGET: trackable dari tipe MultiTarget e. MARKER: trackable dari tipe FrameMarker f. WORD: trackable dari tipe Word
Trackable Name	a. Sebuah string yang secara unik mengidentifikasi trackable dalam database target. b. Max String Length : 64 karakter c. Set Character yang diperbolehkan : a-z, A-Z, 0-9, [-_.]
User Data	Properti yang mewakili *user-defined data* generik yang berhubungan dengan *trackable.*

Dengan menganalisis frame video yang diambil oleh kamera, Vuforia mencoba untuk mencocokkan elemen yang terlihat dalam frame video saat ini dengan satu atau lebih target dalam satu set yang telah ditetapkan. Ketika kecocokan ditemukan, objek *TrackableResult* dibuat dan diteruskan ke aplikasi melalui objek *state* di setiap frame.

Sebuah instansiasi *TrackableResult* berisi referensi ke target yang cocok, status, dan pose matriks yang mewakili posisi 3D saat ini dan orientasi dari *trackable* yang berkaitan dengan referensi frame kamera.

Vuforia SDK mendefinisikan subtipe *TrackableResult* untuk masing-masing subtipe *trackable* :

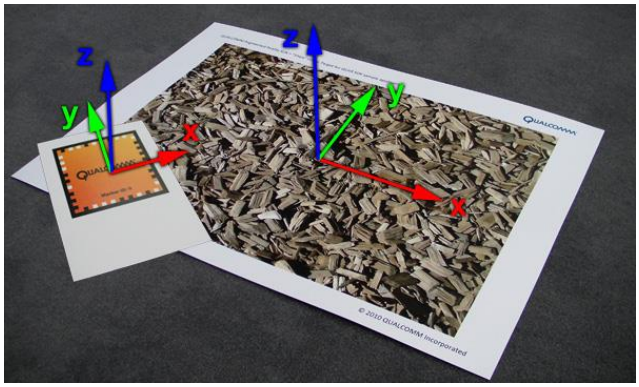
a. ImageTargetResult
b. CylinderTargetResult
c. MultiTargetResult
d. MarkerResult
e. WordResult

Tabel 2.4 berisi parameter-parameter yang dipakai di kelas *TrackableResult.*

Tabel 2.4 Parameter *TrackableResult*

Trackable Result Type	Enum mewakili jenis hasil trackable; setiap jenis sebenarnya cocok dengan jenis yang sesuai dari kelas trackable : a. UNKNOWN_TYPE: trackable dari jenis yang tidak diketahui b. IMAGE_TARGET: trackable dari tipe ImageTarget c. CYLINDER_TARGET: trackable dari tipe CylinderTarget d. MULTI_TARGET: trackable dari tipe MultiTarget e. MARKER: trackable dari tipe FrameMarker f. WORD: trackable dari tipe Word
Trackable	*Trackable* yang mengacu pada hasil *TrackableResult.*
Trackable Result Status	Setiap hasil *trackable* memiliki informasi status yang terkait dengannya dalam objek *state* dimana *state* diperbarui saat setiap frame kamera diproses. Status ini ditandai dengan enum: a. UNKNOWN - Keadaan *trackable* tidak diketahui. Hal ini biasanya dikembalikan sebelum inisialisasi *tracker*. b. UNDEFINED - Keadaan *trackable* tidak didefinisikan. c. NOT_FOUND - *Trackable* tidak ditemukan, misalnya, *trackable* direferensikan bukan dari bagian database. d. DETECTED - *Trackable* tersebut terdeteksi pada frame ini. e. TRACKED - *Trackable* tersebut dilacak dalam bingkai ini. f. EXTENDED_TRACKED - Trackable tersebut dilacak dalam bingkai ini menggunakan fitur *extended tracking*
Trackable Result Pose	Pose yang masih berlaku dari DETECTED, TRACKED atau EXTENDED_TRACKED *trackable* dikembalikan sebagai matriks 3x4 dalam urutan *row-major*. Vuforia SDK menyediakan fungsi *tool* sederhana untuk mengkonversi Vuforia Pose matriks menjadi GL model-view matrix dan memproyeksikan poin 3D dari adegan 3D ke layar perangkat.

Vuforia SDK menggunakan sistem koordinat tangan kanan. Setiap target gambar dan *frame marker* mendefinisikan sistem koordinat lokal dengan (0,0,0) di pusat (tengah) dari target. + X pergi ke kanan, + Y naik, dan + Z poin dari trackable (arah dari mana dapat dilihat), seperti pada gambar 2.17.



Gambar 2.17 Sistem Koordinat Vuforia SDK

Asal dari sistem koordinat lokal dari *multi-target* didefinisikan oleh komponen-komponennya. Bagian target gambar *multi-target* diubah menjadi relatif terhadap asal koordinat. Pose yang dilaporkan *MultiTargetResult* adalah posisi koordinat asal, terlepas dari bagian individual dimana dilacak dalam *multi-target*. Fitur ini memungkinkan objek geometris (misalnya, kotak) untuk dilacak terus menerus dengan koordinat yang sama, bahkan jika bagian sasaran gambar lainnya yang terlihat dalam tampilan kamera.

2.9.4.3 *Word Targets*

Word Targets merupakan elemen textual dimana Vuforia dapat mengenali dan melacak kata yang valid. Berikut adalah kriteria kata yang dapat dikenali oleh Vuforia :

a. Sebuah kata yang memiliki panjang hingga 24 karakter
b. Sebuah kata yang mungkin berisi *hypens* (kata majemuk tertutup, misalnya : “Mother-in-law”)
c. Sebuah kata yang mungkin berisi ruang (kata majemuk terbuka, misalnya : “Hot dog”
d. Sebuah kata yang tidak mengandung angka

Vuforia SDK dapat mengenali kata-kata dan melacaknya yang sama seperti jenis dari target lain, dengan dua mode pengakuan yang tersedia yaitu "*Words*" dan "*Characters*." Apabila menggunakan "*Words*" mode, kata tersebut diakui (dan kemudian dilacak) jika termasuk dalam *wordlist* yang diberikan, dimana *wordlist* dapat disimpan pada perangkat dan dimuat oleh aplikasi pada saat *runtime*.

Wordlist juga dapat diperpanjang dengan kata-kata tambahan, dan filter kata-kata tertentu yang dapat dimuat dan diterapkan melalui SDK API. Selain itu, ketika menggunakan " *Characters* " mode, setiap kata yang dihasilkan dari urutan karakter sembarang (termasuk nomor) juga dapat terdeteksi.

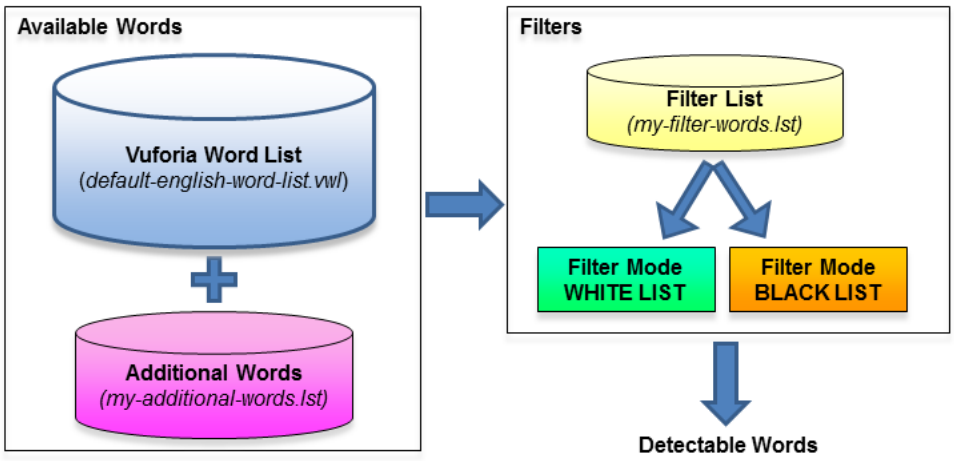
Berikut adalah kebutuhan-kebutuhan untuk aplikasi yang menggunakan fitur *Text Recognition* (word targets) :

a. Database perangkat berisi *wordlist*, yaitu satu set referensi kata dimana aplikasi harus mengenali dan melacak kata. SDK ini dilengkapi dengan *default wordlist* untuk bahasa inggris yang dapat dimuat pada *runtime* aplikasi. Wordlist ini berisi lebih dari 100.000 kata bahasa inggris dan akan memenuhi sebagian besar kasus penggunaan aplikasi untuk teks *recognition*.

b. Sebuah *wordlist* tambahan disimpan dalam teks biasa ".lst" yang dapat ditambahkan ke *default wordlist. Wordlist* ini adalah kata yang tidak ditemukan dalam *default wordlist* yang disediakan dengan SDK, tetapi *wordlist* ini juga harus diakui oleh aplikasi.

c. *Wordlist* juga dapat disaring dengan menggunakan *filter list* untuk mengecualikan kata-kata tertentu yang terdeteksi, atau untuk memungkinkan hanya kata-kata tertentu untuk dideteksi.

Gambar 2.18 menunjukan gambaran dari unsur-unsur yang terlibat dalam mempersiapkan *word targets.*



Gambar 2.18 Unsur-Unsur Yang Terlibat Dalam *Word Target*

Default Wordlist untuk bahasa Inggris, bersama dengan kemampuan filter, harus lebih dari cukup untuk menutupi sebagian dari kata yang umum digunakan pada sebagian besar aplikasi pengenalan teks. Namun, dalam beberapa kasus, seperti aplikasi yang perlu mengenali hanya terminologi medis (bukan dalam penggunaan sehari-hari), daftar kata bawaan tidak akan cukup dan harus diganti dengan daftar kata kustom.

Untuk menambahkan daftar kata dari sebuah file, file tersebut harus file teks biasa mengikuti konvensi berikut :

a. Nama file harus diakhiri dengan ".lst" ekstensi (misalnya, "tambahan-words-saya.lst").
b. Isi file harus dimulai dengan "UTF-8" header.
c. File harus berisi satu kata per baris. Lebih khusus, kata-kata dalam daftar filter file harus dipisahkan oleh karakter *line feed* ("\ n", kode ASCII: 0x0A).

Mesin pengenalan teks Vuforia bergantung pada UTF-8 encoding karakter; Namun, hanya karakter tertentu yang diterima, yaitu, kata berlaku jika terdiri dari karakter yang diizinkan dalam tabel berikut :

Tabel 2.5 Tabel Karakter yang Diizinkan pada Vuforia

Unicode code point	Character	UTF-8 (hex.)	UTF-8 (decimal)	Name
U+000A	*[LF]*	0A	10	Linefeed
U+0020	*[SPACE]*	20	32	Space
U+0027	**'**	27	39	Apostrophe
U+002D	**-**	2d	45	Hyphen-Minus
U+0041	**A**	41	65	Latin Capital Letter A
U+0042	**B**	42	66	Latin Capital Letter B
U+0043	**C**	43	67	Latin Capital Letter C
U+0044	**D**	44	68	Latin Capital Letter D
U+0045	**E**	45	69	Latin Capital Letter E
U+0046	**F**	46	70	Latin Capital Letter F
U+0047	**G**	47	71	Latin Capital Letter G
U+0048	**H**	48	72	Latin Capital Letter H
U+0049	**I**	49	73	Latin Capital Letter I
U+004A	**J**	4a	74	Latin Capital Letter J
U+004B	**K**	4b	75	Latin Capital Letter K
U+004C	**L**	4c	76	Latin Capital Letter L

U+004D	**M**	4d	77	Latin Capital Letter M
U+004E	**N**	4e	78	Latin Capital Letter N
U+004F	**O**	4f	79	Latin Capital Letter O
U+0050	**P**	50	80	Latin Capital Letter P
U+0051	**Q**	51	81	Latin Capital Letter Q
U+0052	**R**	52	82	Latin Capital Letter R
U+0053	**S**	53	83	Latin Capital Letter S
U+0054	**T**	54	84	Latin Capital Letter T
U+0055	**U**	55	85	Latin Capital Letter U
U+0056	**V**	56	86	Latin Capital Letter V
U+0057	**W**	57	87	Latin Capital Letter W
U+0058	**X**	58	88	Latin Capital Letter X
U+0059	**Y**	59	89	Latin Capital Letter Y
U+005A	**Z**	5a	90	Latin Capital Letter Z
U+0061	**a**	61	97	Latin Small Letter A
U+0062	**b**	62	98	Latin Small Letter B
U+0063	**c**	63	99	Latin Small Letter C
U+0064	**d**	64	100	Latin Small Letter D
U+0065	**e**	65	101	Latin Small Letter E
U+0066	**f**	66	102	Latin Small Letter F
U+0067	**g**	67	103	Latin Small Letter G
U+0068	**h**	68	104	Latin Small Letter H
U+0069	**i**	69	105	Latin Small Letter I
U+006A	**j**	6a	106	Latin Small Letter J
U+006B	**k**	6b	107	Latin Small Letter K
U+006C	**l**	6c	108	Latin Small Letter L
U+006D	**m**	6d	109	Latin Small Letter M
U+006E	**n**	6e	110	Latin Small Letter N
U+006F	**o**	6f	111	Latin Small Letter O
U+0070	**p**	70	112	Latin Small Letter P
U+0071	**q**	71	113	Latin Small Letter Q
U+0072	**r**	72	114	Latin Small Letter R
U+0073	**s**	73	115	Latin Small Letter S
U+0074	**t**	74	116	Latin Small Letter T
U+0075	**u**	75	117	Latin Small Letter U
U+0076	**v**	76	118	Latin Small Letter V
U+0077	**w**	77	119	Latin Small Letter W
U+0078	**x**	78	120	Latin Small Letter X
U+0079	**y**	79	121	Latin Small Letter Y
U+007A	**z**	7a	122	Latin Small Letter Z

2.9.4.4 Mengatur *Environment* Pengembangan Android

Vuforia SDK membutuhkan SDK Android dan NDK. Android NDK merupakan perpanjangan dari SDK Android yang memungkinkan pengembang Android membangun kinerja bagian *performance-critical* dari aplikasi mereka dalam *native code*. SDK dan NDK terhubung melalui Java Native Interface (JNI). Untuk mengatur e*nvironment* pengembangan, dibutuhkan komponen-komponen ini dalam urutan sebagai berikut, dengan menggunakan versi terbaru dari perangkat dengan Vuforia SDK:

a. JDK (Java SE)
b. Android ADT (Android Developer Tools) Bundle, yang mencakup :
    1) Eclipse IDE with the ADT plugin
    2) Android SDK Tools
    3) Android Platform Tools
    4) Android SDK platform terbaru
c. Cygwin environment
d. Android NDK

2.9.4.4.1 JDK *(Java SE)*

Download Java SE Development Kit (JDK) dari situs ini : http://www.oracle.com/technetwork/java/javase/downloads/



Gambar 2.19 Halaman Java SE Downloads

Klik Download dari bagian JDK dari 'Java Platform, Standard Edition' tabel. Instal lingkungan JDK dengan pengaturan default. Sedangkan untuk Mac, JDK sudah terintegrasi ke dalam sistem operasi Mac OS X.

Petunjuk instalasi rinci dan persyaratan sistem dapat ditemukan di http://www.oracle.com/technetwork/java/javase/index-137561.html

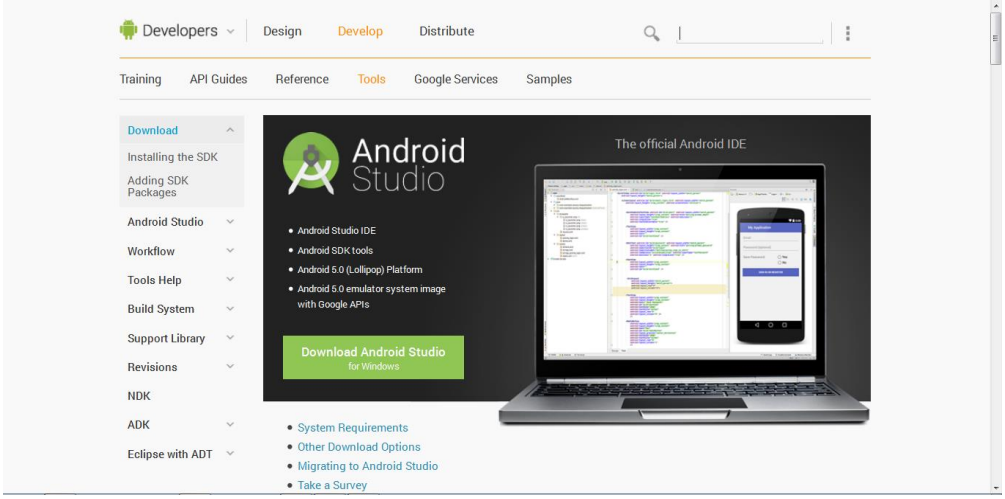
2.9.4.4.2 Android ADT *(Android Developer Tools)*

Android Development Tools Bundle meliputi:

a. Eclipse IDE + ADT plugin
b. Android SDK Tools
c. Android Platform-tools
d. The latest Android platform

Cara menginstal ADT Bundle

a. download versi terbaru dari situs berikut:
   http://developer.android.com/sdk/index.html/



Gambar 2.20 Halaman ADT Bundle Download

b. Unpack ZIP yang telah didownload, dimana berisi konten berikut :
   1) Folder "/eclipse", yang berisi Eclipse IDE (dengan ADT plugin)
   2) Folder "/sdk", yang berisi Android SDK dan SDK/platform tools
   3) "SDK_Manager.exe"

c. Salin folder dan file diatas ke dalam direktori pengembangan, seperti berikut :

C:\Development\Android\

d. Setelah langlah diatas, IDE dapat dijalankan dengan Eclipse.exe yang terletak di folder "/eclipse".

Untuk Mac OS X disarankan menggunakan versi 32-bit dari ADT Bundle.

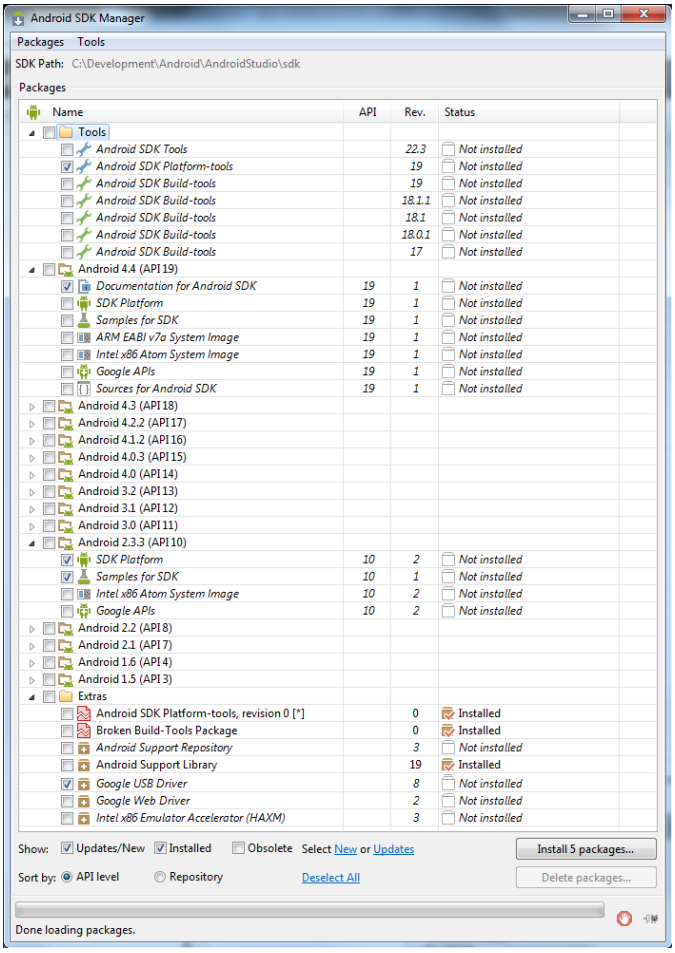
2.9.4.4.3 SDK *Manager*

SDK Manager memungkinkan untuk menginstal atau memperbarui Android Platform Tools yang terbaru. Ketika men-download dan menginstal ADT Bundle, tools terbaru harus sudah tersedia dengan bundel.

a. Dalam Eclipse, pilih menu Window-> Android SDK Manager. Jika lokasi Android SDK tidak diatur dengan benar dalam Eclipse, pergi ke Windows-> Preferences> Android, dan atur field lokasi SDK ke root dari SDK install directory.

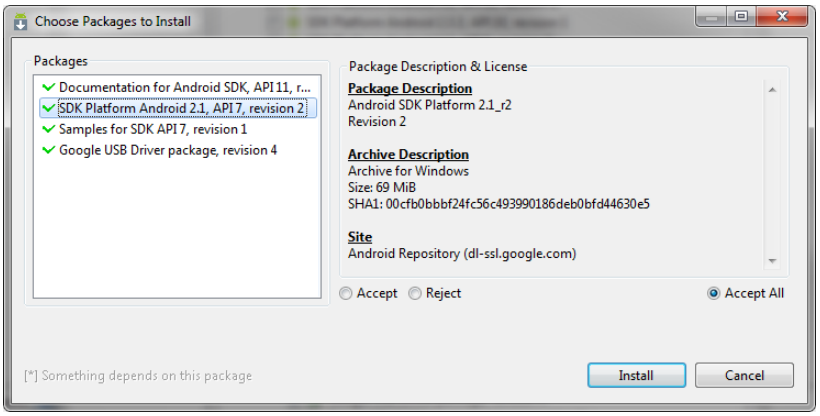
b. Pada window manager Android SDK, urutkan berdasarkan tingkat API, klik Deselect All dan periksa kotak berikut:

Tools:	Android SDK Platform-tools
Android 4.4 (API 19):	Documentation for Android SDK
Android 2.3.3 (API 10):	SDK Platform
	Samples for SDK (optional)
Extras:	Google USB Driver package (only required on Windows)



Gambar 2.21 Android SDK Component Selection

c. Untuk menginstal opsi yang dipilih, klik Install 5 packages ... dan menerima semua izin pada halaman berikutnya. Gunakan Menerima Semua sebagai jalan pintas dan klik Install.



Gambar 2.22 Android Component License Acceptance

d. Tambahkan platform-tools \ direktori path Windows :

```
;C:\Development\Android\android-sdk-windows\platform-tools\
```

Update variabel *path* Mac untuk menunjuk ke Android SDK Platform-tools di /etc/rc.common file or ~/.bash_profile :

```
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/libexec:/System/Library/CoreServices:/Developer/usr/bin:~/Development/Android/android-sdk-macosx/tools:~/Development/Android/android-sdk-macosx/platform-tools:export PATH
```

Update *path* Linux untuk menunjuk ke Android SDK Platform-tools. Jika menggunakan bash shell tambahkan *path* berikut ke ~/.bashrc :

```
export PATH=/opt/android-sdk-linux/tools/:/opt/android-sdk-linux/platform-tools/:$PATH
```

2.9.4.4.4 Cygwin *Enironment*

Cygwin *Environmet* tidak relevan bagi pengguna Linux yang memiliki GNU yang terinstal dan berjalan pada platform ini. Sama dengan Linux, Mac yang memiliki Apple Developer Tools (Xcode) didalamnya tidak relevan dengan Cygwin *Environmet.* Jika membutuhkan Xcode dapat diinstal dari situs berikut : http://developer.apple.com/xcode/.

Sejak versi 2.8, Vuforia SDK untuk platform Android mencakup Java API dan C++ API. Java API memungkinkan rangkaian lengkap dari fitur Vuforia dan fungsi yang memungkinkan pengembang untuk membangun aplikasi Android tanpa harus menulis *native C++ code*.

C++ API dimaksudkan untuk digunakan oleh pengembang dengan persyaratan khusus, seperti kemampuan untuk mengintegrasikan C++ berbasis *third party libraries* atau kustom C++ komponen. Bila hal ini tidak diperlukan, umumnya disarankan untuk menggunakan Java API.

Jika membutuhkan C++ API asli, diperlukan compiler C++ GNU, seperti yang akan dijelaskan dalam paragraf selanjutnya. Jika tidak, untuk pengembangan berbasis Java murni dapat melewati bagian ini.

Sebuah kompiler GNU diperlukan untuk mengkompilasi aplikasi dinamis sebagai *library* bersama untuk NDK Android. Android membuat file dirancang untuk berjalan dengan gcc4. Pada windows, cara mudah untuk memiliki environment yang lengkap adalah dengan menginstal Cygwin.

Cygwin menggunakan *installer* pembantu untuk mengelola proses instalasi.

a. Download dari http://www.cygwin.com/ dan jalankan installer (setup.exe) dan pilih "Install from the Internet". Disarankan untuk tidak mengubah direktori root di halaman berikutnya dan meninggalkannya di "C:\cygwin".
b. Pilih paket berikut untuk didownload: `All -> Devel -> "make: The GNU version of the 'make' utility"`
c. Pilih skip untuk mengubahnya ke nomor versi yang sebenarnya, saat ini adalah 3,82-90.
d. Selesaikan instalasi dengan klik nect.

Cygwin *Environmet* sepenuhnya diatur untuk bekerja dengan SDK QCAR. Jika memiliki *Environmet* lain yang sejenis untuk dipasang, pastikan untuk mengatur variabel *path* windows untuk menunjuk ke "C:\cygwin\bin" sehingga *bash* menggunakan versi GNU's make.exe.

2.9.4.4.5 Android NDK

Jika menggunakan C++ API dari Vuforia SDK, maka perlu menginstal Android NDK yang memungkinkan *native C++ programming* di Android. Jika hanya menggunakan Java API dapat melewati bagian ini.

Android NDK merupakan perpanjangan dari SDK android yang memungkinkan pengembang Android membangun bagian *performance-critical* dari aplikasi dalam *native code*.

a. Download NDK *package* dari :
   http://developer.android.com/sdk/ndk/index.html
b. Unzip arsip dan salin konten ke direktori. Agar konsisten dengan setup sebelumnya, disarankan menempatkan isi di :
   "C:\Development\Android\android-ndk-rxy\."

Dengan demikian, Android SDK dan NDK berbagi direktori induk yang sama.

NDK membutuhkan direktori diatas yang akan ditambahkan ke *windows path.*

c. Klik kanan *My Computer* pada dekstop dan pilih *properties.*
d. Pada tab *Advanced*, pilih variabel *environment* dan cari variable "*path*" di jendela *System variables.*
e. Setelah menekan *edit*, *scroll* ke akhir dari *Variable value* dan menambahkan *path* lengkap ke direktori ke bagian akhir *path*, dipisahkan oleh titik koma dari path sebelumnya. Dalam contoh diatas harus ditambahkan bagian berikut :
C:\Development\Android\android-ndk-rxy

Untuk menguji instalasi, kompilasi salah satu contoh aplikasi NDK. Menggunkan *bash shell cygwin,* arahkan ke direktori root dari aplikasi demo.

2.9.4.5 Instalasi Vuforia SDK

Vuforia SDK didistribusikan sebagai paket ZIP untuk platform berikut: Windows, Mac OS X, dan Linux. Untuk mulai pengembangan dengan Vuforia SDK:

a. Download SDK Vuforia
b. Ekstrak isi dari paket ZIP dan memasukkannya ke dalam <DEVELOPMENT_ROOT>
c. Sesuaikan pengaturan Vuforia Lingkungan di Eclipse

Setelah diekstrak, paket SDK menunjukan struktur direktori dalam *environment* pengembangan Android. Struktur ini memastikan bahwa aplikasi *sampel* dapat dengan mudah dibangun dan disebarkan menggunakan NDK Android dan *environment* pengembang Eclipse Java.

Ketika menginstal versi terbaru dari Vuforia SDK, disarankan agar menginstalnya di bawah pohon direktori baru. Untuk mengkompilasi set baru dari contoh aplikasi diperlukan untuk menyesuaikan pengaturan variabel Eclipse workspace.

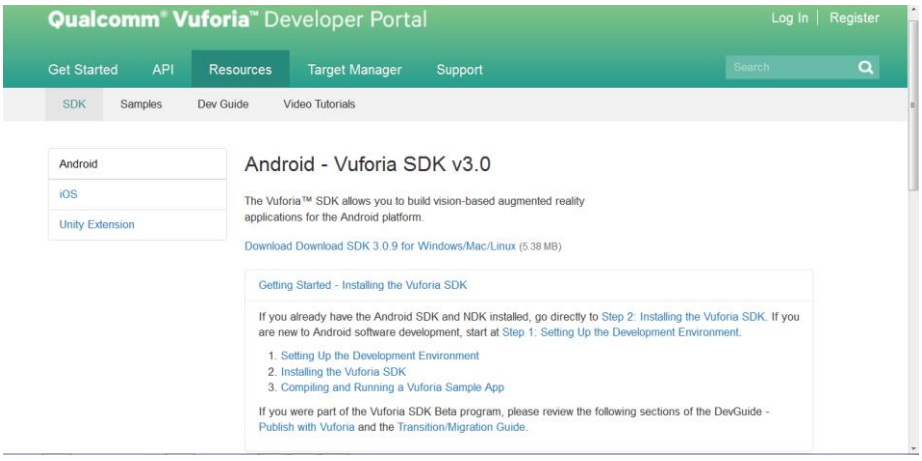
Untuk merampingkan pengembangan, telah dirancang struktur direktori yang membuat Vuforia SDK dan aplikasi di pohon-pohon yang terpisah. Ini akan memastikan update mudah untuk SDK. Dalam langkah-langkah sebelumnya digunakan direktori awal untuk instalasi SDK dan NDK yang kita disebut <DEVELOPMENT_ROOT> = C: \ Pembangunan \ Android

Arsip yang telah didownload lalu diekstrak, akan menunjukan struktur direktori yang akan diintegrasikan ke dalam *environment* pengembangan Android. Setelah lokasi instalasi direkomendasikan *environment* pengembangan akhirnya akan menghasilkan struktur berikut. Di sini xx-yy-zz ditujukan untuk nomor versi dari Vuforia SDK.

```
<DEVELOPMENT_ROOT>\
android-ndk-rxy\
android-sdk-windows\
vuforia-sdk-android-xx-yy-zz\
    build\          Vuforia Augmented Reality SDK
    licenses\       License Agreements
    samples\        Sample applications with full source code
    assets\         Additional assets required to use Vuforia SDK
    readme.txt              Starting read-me document
```

2.9.4.5.1 Install Vuforia SDK

Vuforia SDK didistribusikan melalui Vuforia Portal Developer. Berikut adalah langkah-langkah untuk mengakses installer. Meskipun mendistribusikan Vuforia SDK untuk platform yang tercantum di bawah ini, dukungan untuk pengembangan terbatas pada Windows 7 Platform 32/64-bit.



Gambar 2.23 Halaman Download Vuvoria SDK

a. Windows

Meskipun telah disarankan mengembangkan di Windows 7 32/64-bit, Vuforia SDK juga telah berhasil berjalan pada Windows XP 32/64-bit. Namun, saat ini tidak menawarkan dukungan khusus untuk platform ini.

1) Download file arsip dari halaman berikut : https://developer.vuforia.com/resources/sdk/android .
2) Ekstrak isi arsip dan simpan di dalam <DEVELOPMENT_ROOT>.

b. Mac OS X

Instalasi telah diuji pada Mac OS X 10.6 dan OS X 10.7.

1) Download file arsip dari halaman berikut : https://developer.vuforia.com/resources/sdk/android.
2) Ekstrak isi arsip.

c. Linux

Instalasi telah diuji pada Ubuntu 11.10.

1) Pastikan JVM diinstal.
2) Download file arsip dari halaman berikut : https://developer.vuforia.com/resources/sdk/android.

Ekstrak isi arsip.

2.9.4.5.2 Mengatur QCAR *Environment Variable*

Struktur direktori yang direkomendasikan memungkinkan untuk upgrade SDK independen dari pengembangan aplikasi. Proyek AR akan berada dalam struktur yang direkomendasikan di bawah <DEVELOPMENT_ROOT>. Satu variabel workspace harus diatur di Eclipse sehingga menyadari hirarki ini :

a. Dalam Eclipse, buka jendela-> *Preferences*.
b. Arahkan ke *Java-> Build Path-> Variabel Classpath* menggunakan browser hirarki.
c. Buat variabel baru dengan memilih *New* ...

d. Tambahkan :

   QCAR_SDK_ROOT

   dalam Name : field, dan navigate gunakan `Folder`... ke direktori <DEVELOPMENT_ROOT> \ vuforia-sdk-android-xx-yy-zz didefinisikan dalam bagian pengaturan Vuforia SDK. Dalam contoh di atas, nilai variabel:

   ```
   C:/Development/Android/vuforia-sdk-android-xx-yy-zz
   ```

   di mana xx-yy-zz menunjukkan nomor versi Vuforia SDK.

Pengaturan ini penting karena pengaturan *classpath* di file sampel menggunakan referensi ini untuk menyertakan file JAR umum yang di-*share*. Preferensi menu pada Mac OS X berada di Eclipse-> *Preferences*.

2.9.4.5.3 Mengatur Pembangunan *Path*

Arahkan pada *package properties* (klik kanan pada *package -> Properties*) untuk "*Order and Eksport*" tab setelah memilih halaman "*Build Java Path*". Verifikasi apakah tanda centang disertakan sebelum QCAR_SDK_ROOT / build / java / vuforia / Vuforia.jar. Jika tidak, atur untuk memastikan Vuforia.jar dikemas dengan aplikasi.

2.9.5 Microsoft Bing Translator

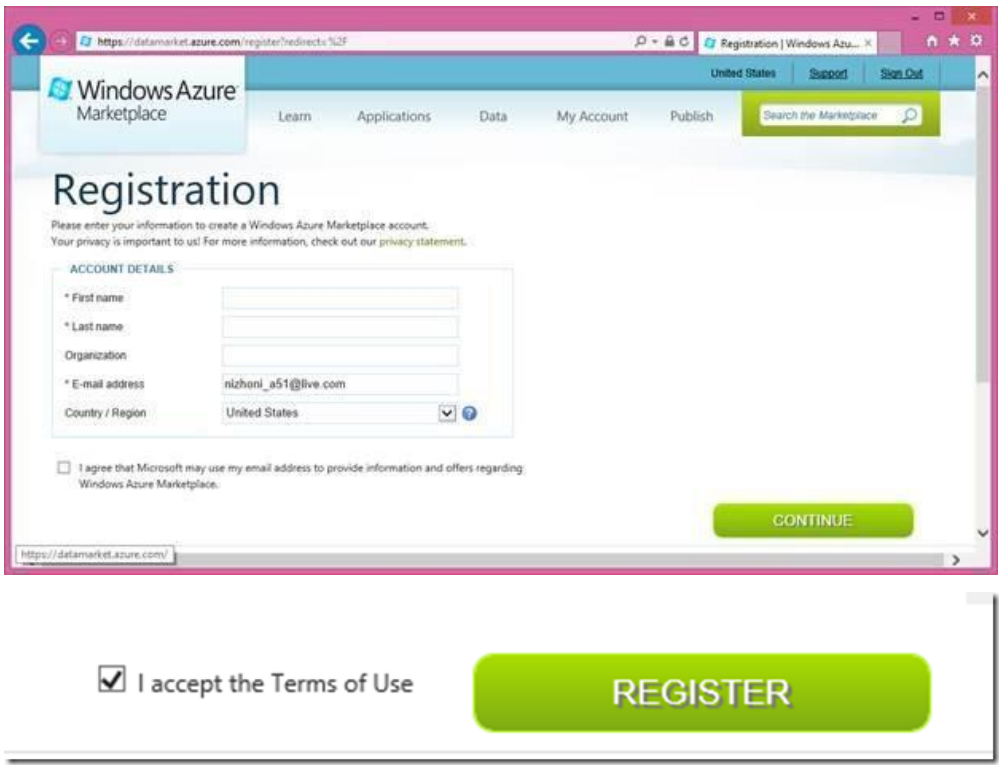
Microsoft Translator API atau *Bing Translator API* adalah sebuah layanan yang disediakan untuk menerjemahkan teks ke bahasa tertentu. Saat ini *Microsoft Translator API* menjadi alternatif yang banyak digunakan *developer* dalam mengebangkan aplikasi mengingat Google tidak lagi menggratiskan layanan *Google Translate API*[22].

Microsoft Translator API merupakan sebuah layanan *online* yang tersedia pada *Windows Azure Marketplace*. *Microsoft Translator API* menjual layanannya berdasarkan jumlah karakter teks yang dikirimkan ke API per bulan dan tersedia secara gratis untuk penggunaan sampai dengan 2 juta karakter per bulan.

Untuk dapat menggunakan layanan *Microsoft Translator API*, *developer* diharuskan terdaftar pada *Windows Azure Marketplace* untuk dapat memiliki *Client ID* dan *Client Secret*. *Client ID* dan *Client Secret* digunakan Microsoft untuk mengetahui aplikasi pihak ketiga yang sedang menggunakan layanannya tersebut. Berikut langkah-langkah untuk mendapatkan *account* pada *Windows Azure Marketplace*:

a. *Sign-In* dan *Register*

Sign-in ke *Windows Azure Marketplace*. Jika ini pertama kali, maka akan tampil form registrasi. Isi kolom yang tersedia kemudian klik "*Continue*".



Gambar 2.24 Form Registrasi Windows Azure Marketplace

Kemudian akan tampil halaman *Terms of Use*. Cek pada "*I accept the Terms of Use*" dan klik "*Register*".

b. Berlangganan ke *Microsoft Translator API*

1) Cari layanan *Micrososft Translator API* di *Windows Azure Marketplace*

Setelah menyelesaikan registrasi *Windows Azure Marketplace* (langkah 1), kemudian kunjungi https://datamarket.azure.com, pada kotak pencarian ketikan "*Translator*" dan tekan *enter*.



Gambar 2.25 Kotak Pencarian Windows Azure Marketplace

Pada hasil pencarian akan ditemukan *Microsoft Translator API*.



Gambar 2.26 Hasil Pencarian

2) Berlangganan ke *Microsoft Translator API*

Klik *Microsoft Translator*, kemudian akan tampil halaman tawaran yang diberikan oleh *Microsoft Translator API*. Di samping kanan layar akan terlihat berbagai paket pemakaian perbulan. Pilihlah salah satu paket yang diberikan, mulai dari yang gratis ataupun yang berbayar. Pada penelitian ini penulis memilih paket gratis dengan *volume* pemakaian 2 juta karakter per bulan.



Gambar 2.27 Paket Gratis Layanan Microsoft Translator

Klik "*Sign Up*".

c. Mendapatkan *Developer Credentials*.

Pada tahap ini diasumsikan telah menyelesaikan langkah 1 dan 2, telah terdaftar pada *Windows Azure Marketplace*, dan telah menggunakan *account* azure marketplace untuk berlangganan layanan *Microsoft Translator API*.

1) Mendaftarkan Aplikasi pada *Windows Azure Marketplace*

Sign-in ke https://datamarket.azure.com/, pada bagian bawah halaman akan terlihat beberapa link. Klik "*Register Your Application*".



Gambar 2.28 Register Your Appplication

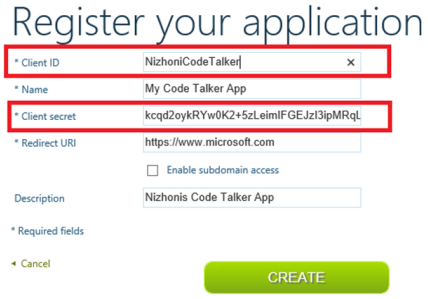
2) Mendapatkan *Client ID* dan *Client Secret*

Client ID dan *Client Secret* digunakan untuk mengotentikasi aplikasi pada saat menggunakan layanan *Microsoft Translator API*.

a) Isi *field Client ID* dan *Name*

b) *Field Client Secret* akan terisi otomatis dan jangan diubah.

c) Isi "*Redirect URL*" dengan URL https yang *valid* seperti https://microsoft.com



Gambar 2.29 *Client ID* dan *Client Secret*

d) Klik "*Create*"

Untuk dapat menggunakan fungsi *Microsoft Translator API* pada pemrograman android, *developer* android memutuhkan suatu *library* yaitu *microsoft-translator-java-api-0.6.2-jar-with-dependencies.jar* yang merupakan *library* yang dibangun oleh pihak ketiga dengan *licensed* dibawah *Apache License, Version* 2.0 yang dibuat oleh Jonathan Griggs pada tahun 2013 lalu guna menjembatani *developer* android agar dapat mengakses *Microsoft Translator API* yang pada pemrogramannya menggunakan bahasa pemrograman ASP.

Dukungan bahasa yang terdapat pada *library microsoft-translator-java-api-0.6.2-jar-with-dependencies.jar* adalah 41 bahasa, yaitu: *Arabic*, *Bulgarian*, *Catalan*, *Chinese Simplified*, *Chinese Traditional*, *Czech*, *Danish*, *Dutch*, *English*, *Estonian*, *Finnish*, *French*, *German*, *Greek*, *Haitian Creole*, *Hebrew*, *Hindi*, *Hmong Daw*, *Hungarian*, *Indonesian*, *Italian*, *Japanese*, *Korean*, *Latvian*, *Lithuanian*, *Malay*, *Norwegian*, *Persian*, *Polish*, *Portuguese*, *Romanian*, *Russian*, *Slovak*, *Slovenian*, *Spanish*, *Swedish*, *Thai*, *Turkish*, *Ukrainian*, *Urdu*, dan *Vietnamese*.

Pada program, untuk dapat memanfaatkan fitur *Microsoft Translator API* diperlukan beberapa langkah, diantaranya mengimpor *class* `com.memetix.mst.language.Language` untuk mengambil bahasa dan `com.memetix.mst.translate.Translate` untuk menerjemahkan percakapan.

2.10 *Object Oriented Programming*

Pemograman terstruktur adalah arus utama pada hari-hari sebelumnya dalam rekayasa perangkat lunak. Programmer mulai mengembangkan blok standar kode untuk melakukan operasiseperti pencetakan, dan kemudian menyalin dan menyisipkan kode tersebut ke setiap aplikasi yang mereka tulis. Sementara hal ini mengurangi waktu pengembangan untuk aplikasi baru, cukup sulit jika perubahan yang diperlukan dalam blok kode, karena pengembang harus membuat perubahan di mana-mana. Dalam pemrograman terstruktur terdapat beberapa hal yang mana pemrograman berorientasi obyek dirancang untuk memecahkan hal tersebut.

Paradigma berorientasi objek adalah cara yang berbeda dalam memandang aplikasi. Pendekatan berorientasi objek, membagi aplikasi ke dalam banyak potongan kecil, atau benda, yang cukup independen satu sama lain. Kemudian aplikasi dibangun dengan menggabungkan semua benda atau potongan kecil ini

bersama-sama. Beberapa prinsip orientasi objek diantaranya enkapsulasi, pewarisan, dan polimorfisme.

a. Enkapsulasi (*Encapsulation*)

Sistem berorientasi objek menggabungkan sepotong informasi dengan perilaku tertentu yang bertindak atas informasi tersebut. Lalu dirangkum menjadi sebuah objek. Hal ini disebut dengan enkapsulasi. Cara lain untuk melihat enkapsulasi adalah dengan membagi aplikasi menjadi bagian-bagian kecil dari fungsi terkait. Sebagai contoh, informasi yang berkaitan dengan rekening bank, seperti nomor rekening, saldo, nama pelanggan, alamat, jenis account, tingkat bunga, dan tanggal pembukaan. Lalu perilaku untuk rekening bank: membuka, menutup, deposit, withdraw, jenis perubahan, perubahan pelanggan, dan perubahan alamat. Setelah itu, informasi dan perilaku ini dirangkum menjadi sebuah objek akun. Akibatnya, setiap perubahan ke sistem perbankan mengenai rekening hanya dapat diimplementasikan dalam objek akun. Enkapsulasi bekerja seperti *one-stop shop* untuk semua informasi account dan perilaku.

b. Pewarisan (*Inheritance*)

Pewarisan adalah prinsip dasar berorientasi objek yang kedua. Dalam sistem berorientasi objek, pewarisan adalah mekanisme yang memungkinkan untuk membuat objek baru berdasarkan objek lama: Objek anak mewarisi sifat-sifat dari objek induk. Salah satu manfaat utama dari pewarisan adalah kemudahan dalam pemeliharaan. Ketika suatu perubahan yang mempengaruhi objek anak, hanya perlu mengubah objek orangtua maka objek anak secara otomatis akan mewarisi perubahannya. Contohnya jika mamalia tiba-tiba menjadi berdarah dingin, hanya objek mamalia akan perlu diubah. Kucing, anjing, manusia, ikan paus, dan benda-benda lainnya secara otomatis objek anak akan mewarisi karakteristik baru, yaitu karakteristik berdarah dingin dari mamalia .

c. Polimorfisme (*Polymorphism*)

Prinsip ketiga orientasi objek adalah polimorfisme. Dari arti katanya, polimorfismemerupakan terjadinya berbagai bentuk, tahapan, atau jenis. Polimorfisme berarti memiliki banyak bentuk atau implementasi dari fungsi tertentu. Dalam hal sistem berorientasi objek, ini berarti bahwa perancang sistem dapat memliliki banyak implementasi dari fungsi tertentu.

2.11UML

Unified Modeling Language (UML) adalah bahasa yang telah menjadi standar untuk visualisasi, pemodelan, perancangan dan mendokumentasikan artifak suatu sistem perangkat lunak berbasis objek yang terdiri dari kumpulan diagram, dikembangkan untuk membantu para pengembang sistem[23]. Dikembangkan oleh Grady Booch, Ivar Jacobson dan James Rumbaugh dari Rational Software pada 1990-an. UML telah diadopsi oleh Object Management Group (OMG) pada tahun 2007 dan telah dikelola organisasi ini sejak saat itu. Pada tahun 2000 UML diterima oleh International Organization for Standardization (ISO) sebagai standar untuk pemodelan sistem perangkat lunak intensif.

Hingga saat ini, UML telah berkembang hingga versi 2.5. UML memiliki banyak sekali diagram, terutama versi terbaru UML yaitu UML 2.5. Berikut akan dijelaskan macam-macam diagram yang paling sering digunakan dalam pembangunan aplikasi berbasis objek.

a. *Business Use Case* Diagram

Business use case digunakan untuk mewakilifungsi yang disediakan oleh organisasi secara keseluruhan. *Business use case* juga digunakan secara luas selama kegiatan pemodelan bisnis untuk mengatur konteks sistem dan untuk membentuk sebuah fondasi untuk menciptakan use case.

b. *Use Case* Diagram

Use case digunakan untuk memodelkan proses bisnis berdasarkan perspektif pengguna sistem. *Use case* diagram terdiri atas diagram untuk use case dan actor. *Actor* merepresentasikan orang yang akan mengoperasikan atau orang yang berinteraksi dengan sistem aplikasi.

c. *Activity* Diagram

Activity diagram menggambarkan berbagai aliran aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing aliran berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity* diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity* diagram merupakan *state* diagram khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internalprocessing*). Oleh karena itu *activity* diagram tidak menggambarkan *behaviour* internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

d. *Sequence* Diagram

Sequence diagram menjelaskan secara detail urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari *use case*, diantaranya:

1) Interaksi yang terjadi antar *class*.
2) Operasi apa saja yang terlibat.
3) Urutan antar operasi.
4) Informasi yang diperlukan oleh masing-masing operasi.

e. *Class* Diagram

Class diagram merupakan diagram yang selalu ada di permodelan sistem berorientasi objek. *Class* diagram menunjukkan hubungan antar *class* dalam sistem yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan. Selain itu, *class* diagram menunjukan juga atribut dan *method* yang ada pada setiap *class*.

2.12*Java*

Inovasi bahasa komputer dimotivasi oleh dua faktor: perbaikan dalam seni pemrograman dan perubahan dalam lingkungan komputasi, tidak terkecuali Java[24]. Dibangun di atas warisan yang kaya dariC dan C++, Java menambahkan perbaikan dan fitur yang mencerminkan keadaan seni dalam pemrograman saat ini. Menanggapi munculnya lingkungan online, Java menawarkan fitur yang merampingkan pemrograman untuk arsitektur yang sangat terdistribusi.

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di Sun Microsystems saat ini merupakan bagian dari Oracle dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana serta dukungan rutin-rutin aras bawah yang minimal. Aplikasi-aplikasi berbasis java umumnya dikompilasi ke dalam p-code (bytecode) dan dapat dijalankan pada berbagai Mesin Virtual Java (JVM).

Versi awal Java ditahun 1996 sudah merupakan versi rilis sehingga dinamakan Java Versi 1.0. Java versi ini menyertakan banyak paket standar awal yang terus dikembangkan pada versi selanjutnya:

a. java.lang: Peruntukan kelas elemen-elemen dasar.
b. java.io: Peruntukan kelas input dan output, termasuk penggunaan berkas.
c. java.util: Peruntukan kelas pelengkap seperti kelas struktur data dan kelas kelas penanggalan.
d. java.net: Peruntukan kelas TCP/IP, yang memungkinkan berkomunikasi dengan komputer lain menggunakan jaringan TCP/IP.
e. java.awt: Kelas dasar untuk aplikasi antarmuka dengan pengguna (GUI)
f. java.applet: Kelas dasar aplikasi antar muka untuk diterapkan pada penjelajah web.

Seperti telah dibahas sebelumnya, banyak jenis komputer dan sistem operasi yang terhubung ke Internet. Untuk program-program untuk secara dinamis didownload ke semua berbagai jenis platform, beberapa sarana untuk menghasilkan kode dieksekusi portabel diperlukan. Mekanisme yang sama yang

membantu menjamin keamanan juga membantu menciptakan portabilitas karena fungsionalitasnya yang memungkinkan aplikasi java mampu berjalan di beberapa platform sistem operasi yang berbeda. Saat ini java merupakan bahasa pemrograman yang populer digunakan dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi berbasis web.

Java memliki beberapa kelebihan dibandingkan dengan bahasa pemograman lain, diantaranya:

a. Multiplatform.
b. OOP (*Object Oriented Programming*).
c. *LibraryClass*yang Lengkap.
d. Mewarisi kekayaan C/C++.
e. Pengumpulan sampah otomatis

Selain dari beberapa kelebihan yang dimiliki oleh java, terdapatpula beberapa kekurangan yang dimiliki java, diantaranya:

a. Mudah didekompilasi.

   Dekompilasi adalah proses membalikkan hasil kompilasi menjadi kode sumber. Ini dimungkinkan karena kode jadi Java merupakan bytecode yang menyimpan banyak atribut bahasa tingkat tinggi. Dengan demikian, algoritma yang digunakan program akan lebih sulit disembunyikan dan mudah dibajak/direverse-engineer.

b. Penggunaan memori yang banyak.

Penggunaan memori untuk program berbasis Java jauh lebih besar daripada bahasa tingkat tinggi generasi sebelumnya seperti C/C++ dan Pascal.