BAB II
LANDASAN TEORI

2.1 Tunanetra

Tunanetra adalah seseorang yang memiliki hambatan dalam penglihatan/tidak berfungsinya indera penglihatan [2]. Berdasarkan tingkat gangguannya tunanetra dibagi dua yaitu buta total (*Total Blind*) dan yang masih mempunyai sisa penglihatan (*Low Vision*).

2.1.1 Klasifikasi Tunanetra

Penyandang tunanetra memiliki beberapa klasifikasi berdasarkan hal berikut:

2.1.1.1 Berdasarkan waktu terjadinya ketunanetraan

Tunanetra dapat diklasifikasikan berdasarkan waktu terjadinya ketunanetraannya seperti berikut:

a. Tunanetra sebelum dan sejak lahir, yakni mereka yang sama sekali tidak memiliki pengalaman penglihatan.
b. Tunanetra setelah lahir dan atau pada usia kecil, yakni mereka telah memiliki kesan – kesan visual serta pengalaman visual tetapi belum kuat dan mudah terlupakan.
c. Tunanetra pada usia sekolah atau pada masa remaja, mereka telah memiliki kesan – kesan visual dan meninggalkan pengaruh yang mendalam terhadap proses perkembangan pribadi.
d. Tunanetra pada usia dewasa, yakni mereka yang pada umumnya dengan segala kesadaran mampu melakukan latihan – latihan penyesuaian diri.
e. Tunanetra dalam usia lanjut, sebagian besar sudah sulit mengikuti latihan – latihan penyesuaian diri.

2.1.1.2 Berdasarkan kemampuan daya penglihatan

Tunanetra dapat diklasifikasikan berdasarkan kemampuan daya penglihatan terhadap objek maupun cahaya di depannya seperti berikut:

a. Tunanetra ringan, yakni mereka yang memiliki hambatan dalam penglihatan akan tetapi mereka masih dapat mengikuti program – program pendidikan dan mampu melakukan pekerjaan/kegiatan yang menggunakan fungsi penglihatan.
b. Tunanetra setengah berat, yakni mereka yang kehilangan sebagian daya penglihatan, hanya dengan menggunakan kaca pembesar mampu mengikuti pendidikan biasa atau mampu membaca tulisan yang bercetak tebal.
c. Tunanetra berat, yakni mereka yang sama sekali tidak dapat melihat.

2.1.1.3 Berdasarkan kelainan – kelainan pada mata

Tunanetra juga dapat disebabkan oleh kelainan yang terjadi pada mata penderita seperti berikut:

a. Myopia, adalah penglihatan jarak dekat, bayangan tidak terfokus dan jatuh di belakang retina.
b. Hyperopia, adalah penglihatan jarak jauh, bayangan tidak terfokus dan jatuh di depan retina.
c. Astigmatisme, adalah penyimpangan atau penglihatan kabur yang disebabkan karena ketidakberesan pada kornea mata.

2.2 Tongkat Tunanetra

Alat yang digunakan tunanetra untuk membantu berjalan saat ini ialah sebuah tongkat khusus. Tidak ada nama atau istilah khusus untuk tongkat yang digunakan seorang tunanetra. Namun tongkat yang digunakan bukan merupakan tongkat biasa melainkan tongkat yang didesain khusus untuk seorang tunanetra. Ada dua ciri khusus tongkat yang digunakan tunanetra ialah terdapat tali pegangan agar tidak mudah lepas dan memiliki garis merah di bagian batang tongkatnya.

Fungsi tongkat bagi seorang tunanetra tidak hanya untuk membantu berjalan namun menjadi sebuah identitas tersendiri agar orang lain dapat mengenali. Kelebihan tongkat yang dirasakan seorang tunanetra ialah untuk mengenali tempat – tempat yang baru pertama kali dilewatinya. Biasanya untuk beberapa tunanetra tidak lagi menggunakan tongkat jika jalan yang dilaluinya merupakan kebiasaan sehari – hari.



Sumber Gambar: Gesunda Medical

Gambar 2.1 Tongkat Tunanetra

2.3 Hambatan

Menurut KBBI hambatan atau *obstacle* adalah sebuah halangan atau rintangan. Hambatan mengacu pada sesuatu yang mengganggu atau mencegah tindakan atau kemajuan. Hambatan adalah sesuatu materi atau nonmaterial yang menjadi sebuah gangguan atau halangan dalam sebuah proses.

Ketika seseorang tunanetra berjalan tentu akan menemui beberapa hambatan di depannya. Hambatan yang biasa ditemui tunanetra ialah parit, pohon, dahan pohon, tiang listrik, lubang, gorong – gorong dan lain – lain. Hambatan ini dapat membahayakan seorang tunanetra ketika berjalan.

2.3.1 Contoh – contoh hambatan

Terdapat beberapa contoh hambatan berdasarkan lingkungannya seperti berikut:

2.3.1.1 Hambatan Fisik

Terdapat beberapa hambatan secara fisik yang berada di sekitar lingkungan sehari – hari seperti berikut:

a. Hambatan arsitektur, dapat menghalangi seseorang dalam berjalan.

b. Pintu, gerbang dan akses masuk lainnya, dibuat untuk menghalangi orang yang tidak berkenan masuk.
c. Objek besar, pohon – pohon tumbang atau runtuh yang menghalangi, jalan khusus, rel kereta, saluran air atau gorong – gorong, dapat menghalangi ketika berjalan.
d. Gundukan pasir, batu atau terumbu karang dapat menghalangi seseorang berjalan.
e. Bukit, gunung, fenomena cuaca dapat menghalangi laju kendaraan atau pesawat terbang.
f. Meteor, meteorit, mikro meteroit, debu angkasa, komet dan lainnya dapat mengganggu pesawat luar angkasa.

2.3.1.2 Hambatan dalam Olahraga

Dalam olahraga terdapat beberapa hambatan yang dikenali khusus sesuai aturan dalam kompetisi untuk menciptakan kesulitan dalam pertandingan:

a. Dalam atletik, terdapat hambatan untuk pelari 110 dan 3000 meter seperti lompat tinggi.
b. Dalam kompetisi berkuda terdapat hambatan untuk dilompati.
c. Dalam tenis dan voly terdapat sebuah net yang berdiri sebagai hambatan dan pemisah lapangan.
d. Dalam balap sepeda, motor dan mobil, sirkuit didesain untuk menghindari hambatan berupa batas jalan untuk menambah kesulitan berkompetisi.
e. Dalam olahraga tim, seperti sepakbola dan basket, pemain penyerang diberi hambatan pemain bertahan agar sulit dilewati.
f. Dalam olahraga lain, seperti *parkour*, diharuskan melalui beberapa titik khusus sebagai hambatan.

2.3.1.3 Hambatan dalam Ekonomi

Dapat didefinisikan sebagai unsur – unsur material kekurangan yang orang mungkin miliki untuk mencapai tujuan tertentu, seperti:

a. Kurangnya uang merupakan hambatan untuk suatu pembangunan.

b. Kurangnya air sebagai hambatan dalam kapasitas manusia untuk memproduksi lahan perkebunan untuk dapat bertahan hidup.
c. Kurangnya cahaya sebagai hambatan untuk beraktifitas di malam hari.
d. Kurangnya listrik sebagai hambatan untuk menggunakan alat elektronik dan mesin listrik.
e. Kurangnya sekolah dan tenaga pengajar sebagai hambatan dalam keutuhan kemasyarakatan.
f. Kurangnya rumah sakit dan tenaga medis merupakan hambatan dalam memenuhi kebutuhan kesehatan publik.
g. Kurangnya infrastruktur transportasi sebagai hambatan dalam berdagang, industri dan aktifitas wisata dan pembangunan ekonomi.

2.3.1.4 Hambatan dalam Budaya dan Sosial

Orang – orang dicegah untuk mencapai tujuan tertentu dengan hambatan biologis, psikologis, social atau budaya, seperti:

a. Penyakit, sebagai hambatan bagi manusia seutuhnya.
b. Cacat fisik sebagai hambatan untuk mobilitas penyandang cacat, yang dapat difasilitasi oleh sumber daya aksesibilitas.
c. Rasa malu merupakan suatu hambatan untuk berkomunikasi dan bersosial.
d. Ketakutan merupakan hambatan yang dapat mencegah berhadapan dengan musuh atau lawan social politik atau menghadapi hambatan ekonomi.
e. Pengucilan social atau penangkapan individu sebagai hambatan untuk integrasi sosial – budaya menjadi sebuah komunitas.
f. Kurangnya koordinasi psikomotor sebagai hambatan untuk pengembangan kemampuan yang berkualitas.
g. Tingkat penguasaan idiom yang dapat diucapkan atau perbedaan antara bahasa yang digunakan, sebagai hambatan untuk hubungan sosial nasional atau internasional.
h. Agama – agama yang berbeda sebagai hambatan untuk saling pemahaman moral atau dialog antaragama nasional maupun internasional.

2.3.1.5 Hambatan dalam Politik

Hambatan atau kesulitan suatu masyarakat, perwakilan politik yang representatif, partai politik atau negara menempatkan satu sama lain untuk menghambat tindakan tertentu melawan mereka, seperti:

a. Pencegahan kelompok minoritas politik untuk mencapai aspirasi mereka diparlemen oleh mayoritas suara yang dominan secara politis, dalam prosedur legislatif.
b. Represi ideologis, penganiayaan dan hukuman penjara karena alasan politik.
c. Pemblokiran pengaruh politik dan ekonomi internasional suatu negara dengan suatu perjanjian multilateral atau aliansi antar negara menentang pengaruh tersebut.

2.3.1.6 Hambatan dalam Teknologi

Perbaikan kondisi hidup untuk setiap kalangan manusia selalu dihadapi tantangan kebutuhan teknologi yang masih tidak terjangkau dan tidak tersedia. Dimana teknologi ada yang dapat dikembangkan negara sendiri atau diperoleh dari negara lain. Hal itu harus mengatasi hambatan seperti:

a. Dalam transfer teknologi, pertukaran dan keahlian diplomasi negosiasi dengan negara yang memiliki teknologi baru.
b. Dalam pembangunan internal, tingkat pendidikan di suatu negara, informasi yang dapat diakses, dasar teknologi industry, tingkat institusi penelitian dan pengembangan teknologi, dan tingkat kerjasama internasional.

2.3.1.7 Hambatan dalam Militer

Dalam suatu kenegaraan terdapat batas – batas antar negara yang telah disetujui secara bersama. Tentunya memiliki pertahanan militer untuk menjaga perbatasan tersebut. Hambatan dalam militer yaitu:

a. Memblok atau menghancurkan sumber daya dan jalur logistik seperti jembatan, jalan tol, pelabuhan atau bandara, membuat sebuah pertahanan kepada imigrasi, turis dan lainnya.

b. Kunjungan dari daerah lawan, diharuskan diblok untuk menghindari ancaman yang mungkin terjadi.

2.3.2 Hambatan bagi Tunanetra

Secara lebih spesifik terdapat beberapa hambatan yang seringkali menjadi ancaman bagi tunanetra, seperti:

a. Pohon

Pohon yang berada di pinggir jalan atau diatas trotoar menjadi salah satu ancaman bagi tunanetra ketika berjalan.



Sumber Gambar: Tribun Jateng

Gambar 2.2 Pohon di trotoar

b. Portal

Bagi tunanetra yang sering beraktifitas di sekitar perumahan biasanya sering menemui portal yang tertutup sehingga menjadi ancaman ketika berjalan.



Sumber Gambar: Lampost.co

Gambar 2.3 Portal jalan

c. Jalan berlubang

Tidak jarang kita temui beberapa jalan yang rusak karena genangan air, itu dapat menjadi ancaman bagi tunanetra.



Sumber Gambar: beritatanjungpinang.com

Gambar 2.4 Jalan Berlubang

d. Parit

Ketika seorang tunanetra melalui jalan – jalan kecil atau gang biasanya ditemui parit – parit kecil atau parit yang menyatu dengan jalan yang sangat berbahaya bagi tunanetra.



Sumber Gambar: panoramio.com

Gambar 2.5 Parit

e. Polisi tidur

Seiring peningkatan pengguna sepeda motor yang melintasi daerah pemukiman yang biasa dipakai anak – anak bermain maka dipasang polisi tidur agar pengendara motor bisa mengurangi kecepetan kendaraannya. Hal itu malah memberikan ancaman tersendiri bagi seorang tunanetra ketika berjalan.



Sumber Gambar: panmas.depok.go.id

Gambar 2.6 Polisi Tidur

f. Tiang listrik

Tiang listrik menjadi salah satu ancaman bagi tunanetra karena jumlahnya yang semakin banyak dan penempatannya yang dapat menghalangi jalan.



Sumber Gambar: Ryan Mustamin

Gambar 2.7 Tiang Listrik di trotoar

2.4 Computer Vision

Computer Vision disebut juga *image understanding* ialah disiplin ilmu yang mempelajari bagaimana untuk merekonstruksi, menginterprestasi dan memahami 3D *scene* dari citra 2D dalam hal sifat – sifat struktur data citra yang tersedia dalam suatu *scene* [6].

2.5 Definisi Citra

Secara harfiah, citra adalah gambar pada bidang dwimatra (dua dimensi). Ditinjau dari sudut pandang matematis, citra merupakan fungsi menerus (*continue*) dari intensitas cahaya pada bidang dwimatra. Sumber cahaya menerangi objek, objek memantulkan kembali sebagian dari berkas cahaya tersebut. Pantulan cahaya ini ditangkap oleh alat – alat optik, misalnya mata pada manusia, kamera, pemindai (*scanner*), dan sebagainya, sehingga bayangan objek yang disebut citra tersebut terekam [7].

Citra bergerak (*moving images*) adalah rangkaian citra diam yang ditampilkan secara beruntun (sekuensial) sehingga memberi kesan pada mata kita sebagai gambar yang bergerak. Setiap citra di dalam rangkaian itu disebut *frame*.

Gambar – gambar yang tampak pada film layar lebar atau televisi pada hakikatnya terdiri atas ratusan sampai ribuan frame.

2.5.1 Citra Biner

Citra biner adalah citra yang hanya memiliki dua kemungkinan nilai piksel yaitu hitam dan putih. Citra biner juga disebut sebagai citra B&W (*black and white*) atau citra monokrom. Hanya dibutuhkan 1 bit untuk mewakili nilai setiap piksel dari citra biner. Citra biner sering kali muncul sebagai hasil dari proses pengolahan seperti segmentasi, pengembangan morfologi, ataupun *dhitering* [8].

2.5.2 Citra Grayscale

Citra *grayscale* merupakan citra digital yang hanya memiliki satu nilai kanal pada setiap pikselnya, dengan kata lain nilai bagian RED = GREEN = BLUE. Nilai tersebut digunakan untuk menunjukan tingkat intensitas. Warna yang dimiliki adalah warna dari hitam, keabuan, dan putih. Tingkatan keabuan disini merupakan warna abu dengan berbagai tingkatan dari hitam hingga mendekati putih [8].

2.5.3 Citra Berwarna

Citra berwarna adalah citra digital yang setiap pikselnya mengandung informasi warna. Informasi warna ini biasanya dibentuk dari paling sedikit 3 sampel (saluran warna). Saluran warna yang paling umum digunakan adalah RGB [8].

2.6 Pengolahan Citra Digital

Pengolahan citra digital menyatakan “pemrosesan gambar berdimensi-dua melalui komputer digital (Jain, 1989). Foto adalah contoh gambar berdimensi dua yang dapat diolah dengan mudah. Setiap foto dalam bentuk citra digital (misalnya berasal dari kamera digital) dapat diolah melalui perangkat lunak tertentu. Sebagai contoh, apabila hasil bidikan kamera terlihat agak gelap, citra dapat diolah menjadi lebih terang. Dimungkinkan pula untuk memisahkan foto orang dari latar belakangnya. Gambaran tersebut menunjukkan hal sederhana yang dapat dilakukan melalui pengolahan citra digital [8].

2.7 Representasi Ciri (*Feature*)

Sebuah ciri (*feature*) pada pengolahan citra didefinisikan sebagai sebuah struktur spesifik yang terdapat dalam data citra. Sebagai contoh, sebuah tepi dapat direpresentasikan sebagai variabel boolean di setiap titik citra yang dideskripsikan apakah terdapat sebuah tepi pada titik citra tersebut. Atau kita juga dapat merepresentasikan tepi dengan ukuran kepastian dan menggabungkan ini dengan informasi tentang orientasi tepi. Demikian juga dengan warna pada wilayah tertentu dapat direpresentasikan dalam hal warna rata – rata (tiga skalar) atau histogram warna (tiga fungsi).

Ketika sistem *computer vision* atau algoritma *computer vision* dirancang dengan berbagai pilihan representasi fitur dapat menjadi masalah. Dalam beberapa kasus, citra dengan tingkat detail tinggi dalam mendeskripsikan fitur mungkin diperlukan untuk memecahkan masalah, tetapi hal ini akan menyebabkan biaya yang tinggi untuk mendapatkan banyak data dan membutuhkan banyak proses.



Sumber Gambar: Scholarpedia [9]

Gambar 2.8 Representasi Fitur pada Citra

2.8 Deskriptor Citra

Pada *computer vision*, deskriptor citra ialah keterangan ciri visual pada citra, citra bergerak, algoritma atau aplikasi yang memproduksinya. Deskriptor ini mendiskripsikan karakter dasar seperti bentuk, warna, tekstur, gerakan dan lainnya.

2.8 Deteksi Titik Kunci (*Interests Points Detection*)

Pada langkah awal pengolahan citra, diperlukan pendeteksian titik kunci terlebih dahulu. Deskripsi titik kunci ini diperoleh ketika suatu citra akan diolah.

Untuk mengetahui fitur – fitur yang terdapat pada suatu citra maka diperlukan proses pendeteksian titik kunci.

2.9 ORB: Oriented FAST and Rotated BRIEF

ORB merupakan sebuah algoritma yang dikenal dalam *computer vision* yang digunakan untuk mendeteksi dan mendeskripsikan fitur. ORB dibentuk berdasarkan algoritma FAST untuk mendeteksi fitur dan algoritma BRIEF untuk mendeskripsi fitur. Algoritma ini diklaim dapat menjadi alternatif metode yang kuat dan cepat selain menggunakan SIFT dan SURF [10].

Algoritma ORB merupakan algoritma yang berbasis pada nilai biner. Dimana terdapat dua bagian utama yaitu FAST *keypoint detector* dan BRIEF *rotation* yang disempurnakan sedemikian rupa sehingga dapat meningkatkan kemampuannya dalam mendeteksi dan mendeskripsi fitur.

2.9.1 FAST Detector

FAST *detector* banyak digunakan dalam implementasi *computer vision*. Algoritma ini bertujuan untuk mendeteksi fitur dengan cara mendeteksi tepi. Kelebihan dari metode ini ialah memiliki komputasi yang efisien. Sesuai dengan namanya FAST *detector* merupakan salah satu metode deteksi fitur yang cepat dibandingkan metode lain. Metode ini sangat cocok digunakan untuk mendeteksi citra berbasis *realtime* yaitu pada pengambilan gambar video.

Di dalam ORB, FAST menggunakan satu parameter yaitu intensitas *threshold* diantara piksel pusat dan sekelilingnya. Pendekatan FAST-9 yang memiliki perfomansi yang baik yang digunakan dalam ORB. FAST tidak menghitung sudut maka digunakanlah perhitungan Harris Corner untuk menentukan *keypoint* FAST. Untuk mendapatkan sejumlah N *keypoint*, pertama diatur *threshold* seminim mungkin untuk mendapatkan *keypoint* yang lebih besar dari N, lalu diurut sesuai dengan Harris Corner dan mengambil titik N teratas. FAST tidak memproduksi fitur dengan multiskala. Maka dibangunlah skala piramid pada citra dan menghitung FAST pada setiap level piramid.

Berikut merupakan langkah – langkah yang dilakukan metode ORB untuk mendeteksi fitur yang terdapat pada citra:

1. Melakukan *segment test algorithm* di titik p pada koordinat (Xp, Yp) dalam citra dengan menggunakan lingkaran Bresenham 16 piksel.
2. Menerapkan *decision tree* pada setiap titik piksel tetangga dengan atribut:

$$1, jika\ intensitas\ piksel\ tetangga\ >\ (p\ +\ threshold)$$
$$-1, jika\ intensitas\ piksel\ tetangga\ <\ (p\ -\ threshold)$$
$$0, jika\ intensitas\ piksel\ tetangga\ =\ titik\ p.$$

3. Jika lebih dari 9 (karena FAST-9) piksel tetangga terhadap titik pusat p yang lebih cerah atau gelap maka titik p disebut sebuah fitur.
4. Lakukan klasifikasi dan ekstrak atribut untuk semua piksel pada citra.
5. Untuk setiap piksel, pilih atribut seperti pada tahap 2 menjadi sebuah sublist.
6. Pada setiap titik yang dijadikan sublist, lakukan sublist lagi ke dalam 3 bagian yang lebih mendalam.
7. Ulangi langkah 3 hingga sublist berisi hanya bagian titik piksel yang dinyatakan sebagai sebuah fitur

2.9.2 Orientation by Intensity Centroid

Pendekatan yang digunakan untuk menghitung orientasi ialah dengan *intensity centroid*. Asumsinya ialah intensitas sudut seimbang dari titik pusat dan vector ini digunakan untuk menghitung orientasi.

Definisi *moments*:

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y)$$

Dengan *moments* tersebut mungkin ditemukan *centroid*:

$$C = (\frac{m_{10}}{m_{00}} \frac{m_{01}}{m_{00}})$$

Lalu membangun vector dari sudut pusat O ke *centroid*, OC. Orientasi patch sebagai berikut:

$$\theta = atan2(m_{01}, m_{10})$$

Dimana atan2 merupakan quadran arctan.

2.9.3 BRIEF

BRIEF merupakan sebuah metode yang digunakan untuk mendeskripsikan fitur yang dikombinasi oleh *feature detector* lain. BRIEF merupakan deskriptor yang menggunakan *binary test* diantara Pada deskripsi atau ekstraksi fitur, ORB mengadopsi metode BRIEF yang didesain ulang agar dapat mendeskripsi fitur pada citra. Berikut merupakan langkah – langkah yang dilakukan metode ORB untuk mendeskripsi fitur pada citra:

1. Melakukan *binary test* pada $\tau\ (p: x, y)$, lalu memberi nilai dengan ketentuan:

   $$1, jika\ p(x)\ <\ p(y)$$
   $$0, untuk\ kemungkinan\ lain.$$

   Dimana $p(x)$ merupakan intensitas piksel yang dilakukan *smoothing* dari p pada titik x.
2. Lalu terapkan rumus BRIEF pada dimensi nd:

   $$f_{nd}(p) \coloneqq \sum_{1 \le i \le nd} 2^{i-1}\ \tau(p: x_i, y_i)$$

3. Untuk setiap $S * S$, piksel di *smoothing* dengan Gaussian dan pilih piksel menggunakan *binary test*.
4. Melakukan *spatial arrangement of the binary test* untuk setiap titik piksel. piksel citra yang di *smoothing*.

2.10 Definisi Tepi

Yang dimaksud dengan tepi (*edge*) adalah perubahan nilai intensitas derajat keabuan yang mendadak (besar) dalam jarak yang singkat. Perbedaan intensitas inilah yang menampakkan rincian pada gambar. Tepi biasanya terdapat pada batas antara dua daerah berbeda pada suatu citra. Tepi dapat diorientasikan dengan suatu arah, dan arah ini berbeda – beda pada bergantung pada perubahan intensitas [7].

2.11 OpenCV

OpenCV merupakan sebuah *library* pemrograman yang berfungsi untuk mengimplementasikan ilmu *computer vision*. Lisensi yang digunakan ialah *BSD License* dan karenanya *library* ini dapat digunakan secara bebas untuk kegiatan akademik ataupun kegiatan komersil. *Library* ini mendukung beberapa bahasa pemrograman populer diantaranya C/C++, Python dan Java(Android) serta mendukung berbagai platform seperti Windows, Linux, Android, iOS dan MacOS.Algoritma yang tersedia didalam OpenCV telah mencapai lebih dari 2500.

OpenCV dikembangkan oleh pusat penelitian Intel Rusia di Nizhny Novgorod, Rusia, lalu didukung oleh Willow Garage dan saat ini dikelola oleh Itseez. Hal yang dapat dilakukan oleh OpenCV diantaranya meningkatkan kualitas citra(*enhance)*, memotong citra(*cropping*), melakukan segmentasi, mendeteksi fitur pada citra dan hal lainnya yang berhubungan dengan pengolahan citra.

2.11.1 Modul OpenCV

Untuk dapat melakukan pengolahan citra pada platform yang kita gunakan, OpenCV menyediakan modul yang dapat digunakan. Berikut merupakan modul utama yang terdapat dalam OpenCV:

Tabel 1.1 Modul OpenCV

Modul	Fungsi
Core	Struktur data inti, tipe data, dan manajemen memori.
ImgProc	Filter citra, transformasi citra, dan analisis citra
HighGui	GUI, menulis dan membaca citra dan video.
Video	Analisis *motion* dan deteksi objek pada video.
Calib3d	Kalibrasi kamera dan rekonstruksi 3D.
Features2d	Ekstraksi fitur, deskripsi dan pencocokan fitur.
ObjDetect	Deteksi objek menggunakan *cascade* dan histogram.
ML	Model statistic dan klasifikasi algoritma.
Flann	Pencarian cepat dalam ruang dimensi citra.
GPU	Algoritma yang dioptimasi terhadap GPU.

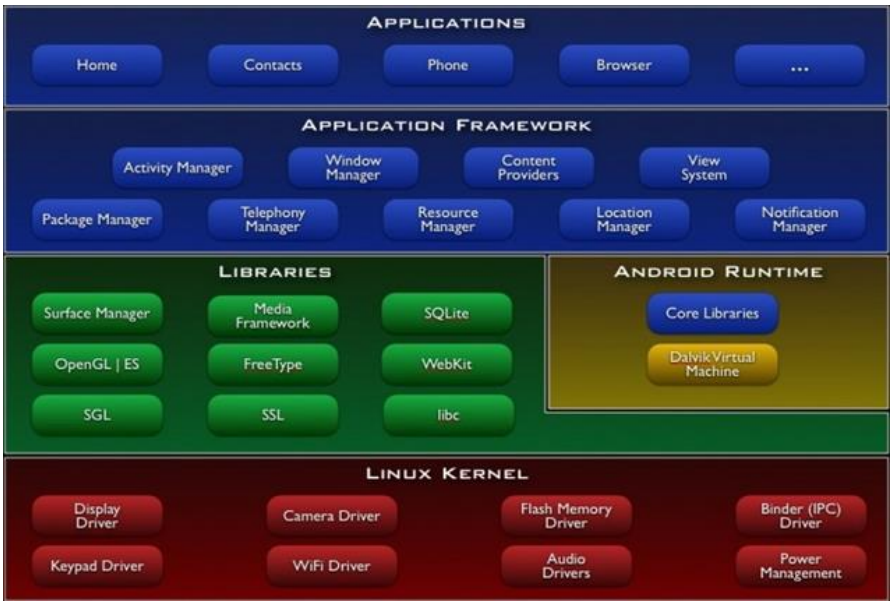
Stitching	*Warping, blending,* dan *bundle* untuk *stitching* citra.
Nonfree	Algoritma yang telah dipatenkan dan digunakan terbatas

2.12 Android

Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis linux yang mencakup sistem operasi, *middleware* dan aplikasi. Android juga menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri yang akan digunakan untuk berbagai macam piranti gerak. Awalnya, Google Inc membeli Android Inc., pendatang baru yang membuat piranti lunak untuk ponsel. Kemudian dalam pengembangan Android, dibentuklah Open Handset Aliance, konsorsium dari 34 perusahaan piranti keras, piranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia [11].

2.12.1 Arsitektur Android

Secara garis besar Arsitektur Android dapat dijelaskan dan digambarkan sebagai berikut [11]:



Sumber Gambar: Safaat [11]

Gambar 2.9 Arsitektur Android

1. *Applications and Widgets*

   Adalah sebuah *layer* dimana kita berhubungan dengan aplikasi saja, di mana biasanya kita *download* aplikasi kemudian kita lakukan instalasi dan jalankan aplikasi tersebut. Di *layer* terdapat aplikasi inti termasuk klien email, program SMS, kalender, peta, browser, kontak, dan lain – lain. Semua aplikasi ditulis menggunakan bahasa pemrograman Java.

2. *Applications Frameworks*

   Android adalah *"Open Development Platform*" yaitu Android menawarkan kepada pengembang, atau memberi kemampuan kepada pembimbing untuk membangun aplikasi yang bagus dan inovatif. Pengembang bebas untuk mengakses perangkat keras, akses informasi *resources*, menjalankan *services background*, mengatur alarm, dan menambahkan status *notifications*, dan sebagainya. Pengembang memiliki akses penuh menuju API *framework* seperti yang dilakukan oleh aplikasi yang kategori inti/ arsitektur aplikasi dirancang supaya kita dengan mudah dapat menggunakan kembali komponen yang salah digunakan (*reuse*).

   Sehingga bisa kita simpulkan *Applications frameworks* ini adalah layer dimana para pembuat aplikasi melakukan pengembangan/pembuatan aplikasi yang akan dijalankan sistem operasi Android, karena pada layer inilah aplikasi dapat dirancang dan dibuat, seperti *content-provider* yang berupa sms dan panggilan telepon.

   Komponen – komponen yang termasuk di dalam layer ini adalah sebagai berikut:

   a. *Views*
   b. *Content Provider*
   c. *Resource Manager*
   d. *Notification Manager*
   e. *Activity Manager*

3. *Libraries*

Ini adalah *layer* dimana fitur – fitur Android berada, biasanya para pembuat aplikasi mengakses *library* untuk menjalankan aplikasinya. Berjalan diatas kernel, *layer* ini meliputi berbagai *library* C/C++ inti seperi Libe dan SSl, serta:

a. Libraries media untuk pemutaran media audio dan video
b. Libraries untuk manajemen tampilan
c. Libraries Graphics mencakup SGL dan OpenGL untuk grafis 2D dan 3D
d. Libraries SQLite untuk dukungan database
e. Libraries SSL dan WebKit terintegrasi dengan web browser dan security
f. Libraries LiveWebcore mencakup modern web browser dengan engine embedded web view
g. Libraries 3D yang mencakup implementasi OpenGL ES 1.0 API's

4. *Android Run Time*

   Layer yang membuat aplikasi Android dapat dijalankan dimana dalam prosesnya menggunakan implementasi Linux. *Dalvik Virtual Machine* (DVM) merupakan mesin yang membentuk dasar kerangka aplikasi Android. Di dalam Android Run Time dibagi menjadi dua bagian yaitu:

   a. Core Libraries: Aplikasi Android dibangun dalam bahasa Java, sementara Dalvik sebagai virtual mesinnya bukan *Virtual Machine Java*, sehingga diperlukan sebuah libraries yang berfungsi untuk menterjemahkan bahasa java/c yang ditangani oleh *Core libraries*.
   b. *Dalvik Virtual Machine*: Virtual mesin berbasis register yang dioptimalkan untuk menjalankan fungsi – fungsi secara efisien, dimana merupakan pengembangan yang mampu membuat linux kernel untuk melakukan threading dan manajemen tingkat rendah

5. *Linux Kernel*

   Linux Kernel adalah layer dimana inti dari sistem operasi dari Android itu berada. Berisi file – file system yang mengatur sistem *processing, memory, resource, drivers,* dan sistem – sistem operasi Android lainnya.

2.12.2 Versi Android

Sejak dirilis perdana, 5 November 2007, Android memiliki beberapa versi diantaranya [11]:

1. Android versi 1.1

   Pada 9 Maret 2009, Google merilis Android versi 1.1. Android versi ini dilengkapi dengan pembaruan estetis pada aplikasi, jam alarm, voice search (pencarian suara), pengiriman pesan dengan Gmail, dan pemberitahuan email.
2. Android versi 1.5 (*Cupcake*)

   Pada pertengahan Mei 2009, Google kembali merilis telepon seluler dengan menggunakan Android dan SDK (Software Development Kit) dengan versi 1.5 (Cupcake). Terdapat beberapa pembaruan termasuk juga penambahan beberapa fitur dalam seluler versi ini yakni kemampuan merekam dan menonton video dengan modus kamera, mengunggah video ke Youtube dan gambar ke Picasa langsung dari telepon, dukungan Bluetooth A2DP, kemampuan terhubung secara otomatis ke headset Bluetooth, animasi layar, dan keyboard pada layar yang dapat disesuaikan
   dengan sistem.
3. Android versi 1.6 (*Donut*)

   Donut (versi 1.6) dirilis pada September dengan menampilkan proses pencarian yang lebih baik dibanding sebelumnya, penggunaan baterai indikator dan kontrol applet VPN. Fitur lainnya adalah galeri yang memungkinkan pengguna untuk memilih foto yang akan dihapus pada kamera, camcorder dan galeri yang dintegrasikan pada CDMA / EVDO, 802.1x, VPN, Ge dan Text-to-speech engine.
4. Android versi 2.0/2.1 (*éclair*)

   Pada 3 Desember 2009 kembali diluncurkan ponsel Android dengan versi 2.0/2.1 (Eclair), perubahan yang dilakukan adalah pengoptimalan hardware, peningkatan Google Maps 3.1.2, perubahan UI dengan browser baru dan dukungan HTML5, daftar kontak yang baru, dukungan flash untuk kamera 3,2 MP, digital Zoom, dan Bluetooth 2.1. untuk bergerak cepat dalam persaingan perangkat generasi berikut, Google melakukan investasi dengan mengadakan

kompetisi aplikasi Mobile terbaik (killer apps- aplikasi unggulan). Kompetisi ini berhadiah $25,000 bagi setiap pengembang aplikasi terpilih. Kompetisi diadakan selama dua tahap yang tiap tahapnya dipilih 50 aplikasi terbaik. Dengan semakin berkembangnya dan semakin bertambahnya jumlah handset Android, semakin banyak pihak ketiga yang berminat untuk menyalurkan aplikasi mereka kepada sistem operasi Android. Aplikasi terkenal yang diubah ke dalam sistem operasi Android adalah Shazam, Backgrounds, dan WeatherBug. Sistem operasi Android dalam situs internet juga dianggap penting untuk menciptakan aplikasi Android asli, contohnya oleh MySpace dan Facebook.

5. Android versi 2.2 (*Froyo: Frozen Yoghurt*)

   Pada 20 Mei 2010, Android versi 2.2 (Froyo) diluncurkan. Perubahan - perubahan umumnya terhadap versi-versi sebelumnya antara lain dukungan Adobe Flash 10.1, kecepatan kinerja dan aplikasi 2 sampai 5 kali lebih cepat, intergrasi V8 JavaScript engine yang dipakai Google Chrome yang mempercepat kemampuan rendering pada browser, pemasangan aplikasi dalam SD Card, kemampuan WiFi Hotspot portabel, dan kemampuan auto update dalam aplikasi Android Market.

6. Android versi 2.3 (*Gingerbread*)

   Pada 6 Desember 2010, Android versi 2.3 (Gingerbread) diluncurkan. Perubahan-perubahan umum yang didapat dari Android versi ini antara lain peningkatan kemampuan permainan (gaming), peningkatan fungsi copy paste layar antar muka (User Interface) didesain ulang, dukungan format video VP8 dan WebM, efek audio baru (reverb, equalization, headphone virtualization, dan bass boost), dukungan kemampuan Near Field Communication (NFC), dan dukungan jumlah kamera yang lebih dari satu.

7. Android versi 3.0 (*Honeycomb*)

   Android Honeycomb dirancang khusus untuk tablet. Android versi ini mendukung ukuran layar yang lebih besar. User Interface pada Honeycomb juga berbeda karena sudah didesain untuk tablet Honeycomb juga mendukung multi prosesor dan juga akselerasi perangkat keras (hardware) untuk grafis.

Tablet pertama yang dibuat dengan menjalankan Honeycomb adalah Motorola Xoom.

8. Android versi 4.0 (*Ice Cream Sandwich*)

   Diumumkan pada tanggal 19 Oktober 2011, membawa fitur Honeycomb untuk smartphone dan menambahkan fitur baru termasuk membuka kunci dengan pengenalan wajah, jaringan data pemantauan penggunaan dan kontrol, terpadu kontak jaringan sosial, perangkat tambahan fotografi, mencari email secara offline, dan berbagi informasi dengan menggunakan NFC.

9. Android versi 4.1 (Jelly Bean)

   Diumumkan pada tanggal 27 Juni 2012, android versi Jelly Bean merupakan peningkatan dari versi sebelumnya, dimana pada versi ini fungsi dan kinerja antarmuka pengguna sudah ditingkatkan menjadi lebih baik lagi seperti antisipasi sentuh, triple buffering, perpanjangan waktu sync, dan peningkatan frame rate hingga 60fps untuk menciptakan tampilan yang lebih halus.

10. Android versi 4.4 Kitkat

    Diumumkan pada tanggal 3 September 2013 android dengan versi KitKat memiliki beberapa pembaruan antara lain Pembaruan antarmuka dengan bar status dan navigasi transparan pada layar depan, Optimasi kinerja pada perangkat dengan spesifikasi yang lebih rendah, Kerangka kerja pencetakan, NFC Host Card Emulation sebagai emulator kartu pintar, WebViews berbasis Chromium, Perluasan fungsionalitas bagi layanan pendengar notifikasi, API umum untuk mengembangkan dan mengelola klien pesan teks, kemampuan untuk menentukan aplikasi SMS standar, Kerangka kerja baru untuk transisi UI, Kerangka kerja akses penyimpanan untuk mengambil konten dan dokumen dari sumber lain, Peningkatan tampilan mode layar penuh, tombol perangkat lunak dan status bar bisa diakses dari tepi dengan cara menggesek, Penyeimbang audio, pemantauan audio, dan peningkatan suara audio, Perekam aktivitas layar yang terintegrasi.

2.13 IDE

IDE atau *Integrated Development Environment* ialah program computer yang memiliki beberapa fasilitas yang diperlukan dalam pembangunan perangkat

lunak. Tujuan dari IDE adalah untuk menyediakan semua utilitas yang diperlukan dalam membangun perangkat lunak.

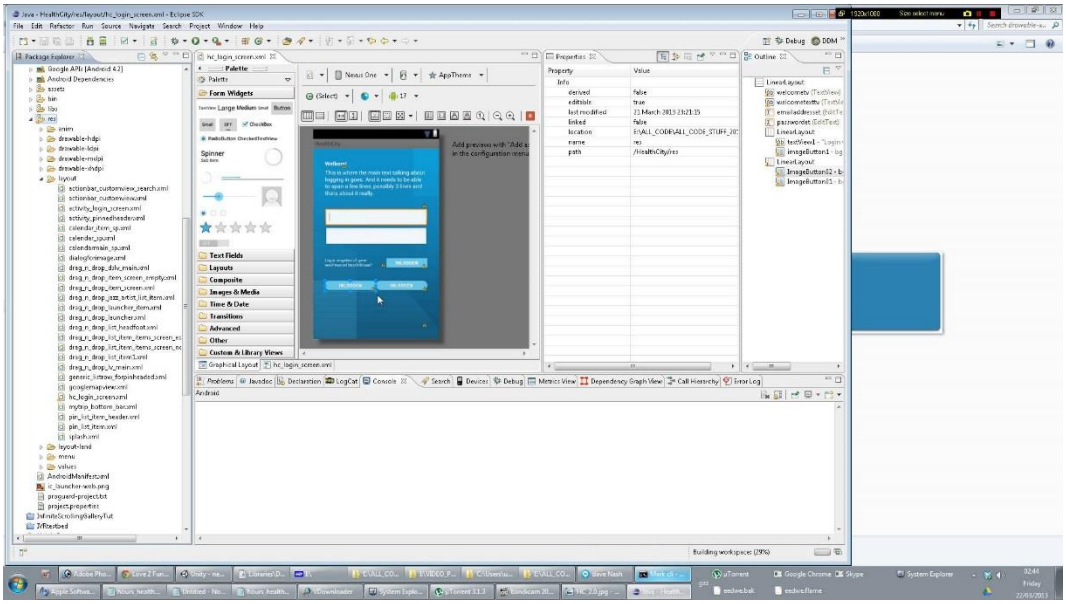
Sebuah IDE, atau secara bebas dapat diterjemahkan sebagai Lingkungan Pengembangan Terpadu, setidaknya memiliki fasilitas:

a. Editor, yaitu fasilitas untuk menuliskan kode sumber dari perangkat lunak.
b. Compiler, yaitu fasilitas untuk mengecek sintaks dari kode sumber kemudian mengubah dalam bentuk binary yang sesuai dengan bahasa mesin.
c. Linker, yaitu fasilitas untuk menyatukan data binary yang beberapa kode sumber yang dihasilkan compiler sehingga data – data binary tersebut menjadi satu kesatuan dan menjadi suatu program komputer yang siap dieksekusi.
d. Debugger, yaitu fasilitas untuk mengetes jalannya program untuk mencari bug kesalahan yang terdapat dalam program.

Sampai tahap tertentu IDE modern dapat membantu memberikan saran yang mempercepat penulisan. Pada saat penulisan kode, IDE juga dapat menunjukkan bagian – bagian yang jelas mengandung kesalahan atau keraguan.

2.14 Eclipse

Dalam membangun sebuah perangkat lunak untuk sistem operasi Android diperlukan sebuah tools IDE. Eclipse merupakan salah satu IDE yang dapat digunakan untuk pembangunan aplikasi Android.



Sumber Gambar: youtube.com

Gambar 2.10 Tampilan IDE Eclipse

2.15 Basis Data

Basis data terdiri atas 2 kata yaitu Basis dan Data. Basis kurang lebih dapat diartikan sebagai markas atau gudang, tempat bersarang/berkumpul. Sedangkan data adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia (pegawai, siswa, pembeli, pelanggan), barang, hewan, peristiwa, konsep, keadaan, dan sebagainya, yang diwujudkan dalam bentuk angka, huruf, symbol, teks, gambar, bunyi atau kombinasinya [12].

Basis Data (*Database*) sendiri dapat didefinisikan dalam sejumlah sudut pandang seperti:

a. Himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasikan sedemikian rupa agar kelak dimanfaatkan kembali lebih cepat dan mudah.
b. Kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (*redundansi*) yang tidak perlu untuk memenuhi berbagai kebutuhan.

c. Kumpulan *file/table* atau arsip yang saling berhubungan yang disimpan dalam media penyimpanan elektronis.

Basis data dan lemari arsip sesungguhnya memiliki prinsip kerja dan tujuan yang sama. Prinsip utamanya adalah pengaturan data/arsip. Dan tujuan utamanya adalah kemudahan dan kecepatan dalam pengambilan kembali data/arsip. Perbedaannya hanya terletak pada media penyimpanan yang digunakan. Jika lemari arsip menggunakan lemari dari besi atau kayu sebagai media penyimpanan, maka basis data menggunakan media penyimpanan elektronis seperti *disk* (disket atau harddisk).

Basis data bukan hanya sekedar penyimpanan data secara elektronis (dengan bantuan komputer). Tidak semua bentuk penyimpanan data secara elektronis bisa disebut basis data. Yang sangat diutamakan dalam basis data adalah pengaturan/pemilihan/pengelompokkan/pengorganisasian data yang akan disimpan sesuai fungsi/jenisnya. Pemilihan, pengelompokkan, pengorganisasian ini dapat berbentuk sejumlah *file/table* terpisah atau dalam bentuk pendefinisian kolom – kolom (*field*) data dalam setiap *file/table*.

2.16 Sistem Manajemen Basis Data (*Database Management System*)

Pengelolaan basis data secara fisik tidak dilakukan oleh pemakai secara langsung, tapi ditangani oleh sebuah perangkat lunak (sistem) yang khusus. DBMS diartikan sebagai suatu program computer yang digunakan untuk memasukkan, mengubah, menghapus, memanipulasi, dan memperoleh data/informasi dengan praktis dan efisien [13].

Dibandingkan dengan sistem yang berbasis kertas, DBMS memiliki 4 keunggulan:

a. Kepraktisan: Sistem yang berbasis kertas akan menggunakan kertas yang sangat banyak untuk menyimpan informasi, sedangkan DBMS menggunakan media penyimpanan sekunder yang berukuran kecil tapi padat informasi.

b. Kecepatan: Mesin dapat mengambil atau mengubah data jauh lebih cepat daripada manusia.
c. Mengurangi Kejemuan: Orang cenderung menjadi bosan kalua melakukan tindakan – tindakan berulang yang menggunakan tangan (misalnya harus mengganti suatu informasi).
d. Kekinian: Informasi yang tersedia pada DBMS akan bersifat mutakhir dan akurat setiap saat.

2.16.1 Bahasa Basis Data (*Database Language*)

DBMS merupakan perantara bagi pemakai dengan basis data dalam *disk*. Cara berinteraksi/berkomunikasi antara pemakai dengan basis data tersebut diatur dalam suatu bahasa khusus yang ditetapkan oleh perusahaan pembuat DBMS. Bahasa itu dapat kita sebut sebagai Bahasa Basis Data yang terdiri atas sejumlah perintah (*statement*) yang diformulasikan dan dapat diberikan *user* dan dikenali/diproses oleh DBMS untuk melakukan suatu aksi/pekerjaan tertentu. Contoh – contoh bahasa basis data adalah SQL, dBase, QUEL dan sebagainya.

Sebuah Bahasa Basis Data biasanya dapat dipilah ke dalam 2 bentuk yaitu:

a. Data Definition Language (DDL) dan
b. *Data Manipulation Language* (DML).

2.16.1.1 *Data Definition Language* (DDL)

Struktur/skema basis data yang menggambarkan/mewakili desain basis data secara keseluruhan dispesifikasikan dengan bahasa khusus yang disebut *Data Definition Language* (DDL). Dengan bahasa inilah kita dapat membuat tabel baru, membuat indeks, mengubah tabel, menentukan struktur penyimpanan tabel, dan sebagainya. Hasil dari kompilasi perintah DDL adalah kumpulan tabel yang disimpan dalam file khusus yang disebut Kamus Sata (*Data Dictionary*).

Kamus Data merupakan suatu metadata (superdata) yaitu data yang mendeskripsikan data sesungguhnya. Kamus Data ini akan selalu diakses dalam suatu operasi basis data sebelum suatu file data yang sesungguhnya diakses.

2.16.1.2 *Data Manipulation Language* (DML)

Merupakan bentuk Bahasa Basis Data yang berguna untuk melakukan manipulasi dan pengambilan data pada suatu basis data. Manipulasi data dapat berupa:

a. Penyisipan/penambahan data baru ke suatu basis data.
b. Penghapusan data dari suatu basis data.
c. Pengubahan data di suatu basis data.

Pada level fisik, kita harus mendefinisikan algoritma yang memungkinkan pengaksesan yang efisien terhadap data. Pada level yang lebih tinggi, yang dipentingkan bukan hanya efisiensi akses, tetapi juga efisiensi interaksi manusia (pemakai) dengan sistem (kemudahan permintaan akses).

Data Manipulation Language (DML) merupakan bahasa yang bertujuan memudahkan pemakai untuk mengakses data sebagaimana direpresentasikan oleh model data. Ada 2 jenis DML, yaitu:

1. Prosedural, yang mensyaratkan agar pemakai menentukan, data apa yang diinginkan serta bagaimana cara mendapatkannya.
2. Nonprosedural, yang membuat pemakai dapat menentukan data apa yang diinginkan tanpa menyebutkan bagaimana cara mendapatkannya.

2.17 SQLite

SQLite adalah sebuah *embedded database* relasional yang berbasis *open source*. Awalnya dirilis pada tahun 2000, dirancang untuk menyediakan cara yang nyaman untuk aplikasi dapat mengelola data tanpa *overhead* dengan sistem manajemen basis data relasional yang terdedikasi. SQLite memiliki kelebihan sangat portabel, mudah digunakan, kompak, efisien, dan dapat diandalkan [14].

2.17.1 Sejarah SQLite

SQLite terlahir di kapal perang. Pengembang SQL, D. Richard Hipp, pernah bekerja untuk General Dynamics yang mengembangkan perangkat lunak bagi U.S.

Navy untuk digunakan sebagai pengontrol arah tembak misil. Perangkat lunak dijalankan pada Hewlett-Packard Unix (HPUX) dan menggunakan Informix *database* sebagai *back-end*. Sebagai perangkat lunak primer yang mereka gunakan, Informix dinilai berlebihan. Bagi *Database Administrator* (DBA) yang berpengalaman, itu bisa menghabiskan sepanjang hari untuk menginstall atau mengupgrade. Apa yang benar-benar dibutuhkan adalah *database* mandiri yang mudah digunakan dan yang bisa berpergian dengan program dan dijalan dimana saja terlepas dari terinstall atau tidaknya perangkat lunak [14].

Pada Januari 2000, Hipp dan koleganya mendiskusikan ide untuk membuat sebuah *embedded database* yang simpel yang akan menggunakan *library* GNU DBM B-Tree (GDBM) sebagai *back-end*, yang tidak akan memerlukan instalasi atau dukungan administrasi apapun. Kemudian, ketika memiliki waktu luang, Hipp mulai mengerjakan proyek tersebut, dan pada Agustus 2000, SQLite 1.0 dirilis.

Sesuai rencana, SQLite 1.0 menggunakan GDBM sebagai *storage manager*. Namun, Hipp menggantinya segera dengan implementasi B-Tree buatan sendiri yang mendukung transaksi dan menyimpan rekaman dalam kunci. Dengan upgrade *major* pertama, SQLite mulai evolusi yang stabil, tumbuh di kedua fitur dan pengguna. Pada pertengahan 2001 banyak proyek komersial dan *open source* yang mulai menggunakannya. Dalam tahun berikutnya, anggota lain dari komunitas *open source* mulai menulis ekstensi SQLite sebagai bahasa skrip dan *library* favorit. Satu per satu ekstensi baru antarmuka *Open Database Connectivity* (ODBC) diikuti dengan ekstensi untuk Perl, Phyton, Ruby, Java dan bahasa andalan lain mulai memberikan tempat khusus bagi SQLite.

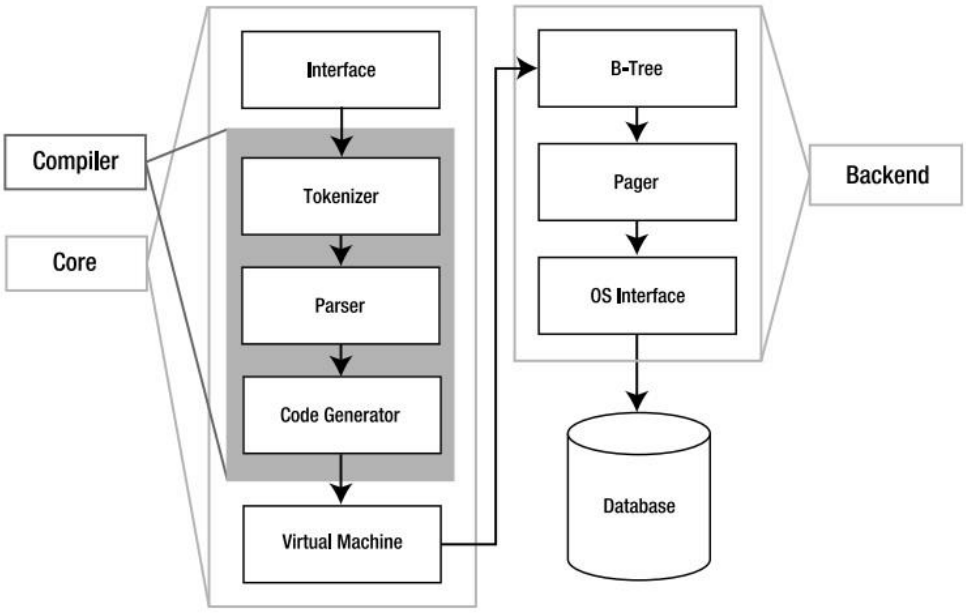
SQLite mulai upgrade besar dari versi 2 ke 3 pada tahun 2004. Tujuan utama ditingkatan agar mendukung teks UTF-8 dan UTF-16. Sementara 3.0 awalnya dijadwalkan untuk rilis pada musim panas 2005, America Online menyediakan dana yang dibutuhkan untuk melihat bahwa itu selesai pada bulan Juli 2004. Selain internasiolisasi, versi 3 membawa banyak fitur baru lainnya seperti perubahan C API, format yang lebih kompak untuk file database (pengurangan ukuran 25

persen), mengetik manifest, dukungan *Binary Large Object* (BLOB), 64-bit ROWIDs, *autovacuum*, dan meningkatkan konkurensi. Terlepas dari banyaknya fitur baru, secara keseluruhan *library footprint* masih kurang dari 240 *kilobyte*. Perbaikan lain di versi 3 adalah kode pembersihan yang baik, meninjau kembali dan menulis ulang, atau membuang barang asing di akumulasi di versi 2.x.

SQLite terus tumbuh dengan fitur yang masih tetap setia pada tujuan desain awal: kesederhanaan, fleksibilitas, kekompakan, kecepatan, dan kemudahan keseluruhan penggunaan. Pada saat buku ini masuk percetakan, SQLite menambahkan CHECK *constraints*. Selanjutnya menambahkan *recursive triggers* dan *foreign keys*.

2.17.2 Arsitektur SQLite

SQLite memiliki arsitektur modular elegan yang mengambil beberapa pendekatan agak unik untuk manajemen database relasional. Ini terdiri dari delapan modul terpisah dikelompokkan dalam tiga subsistem utama. Modul ini membagi pemrosesan *query* ke dalam tugas – tugas diskrit yang bekerjanya seperti jalur perakitan. Bagian atas tumpukan mengkompilasi *query*, tengah mengeksekusinya, dan bagian bawah menangani penyimpanan dan berinteraksi dengan sistem operasi.



Sumber gambar: Owens [14].

Gambar 2.11 Arsitektur SQLite

1. The Interface

The Interface merupakan tumpukan paling atas dalam SQLite C API. Itu merupakan bagian dimana program, bahasa skrip dan *library* berinteraksi dengan SQLite

2. The Compiler

Proses kompilasi dimulai dengan *tokenizer* dan *parser*. Mereka bekerja bersama untuk membentuk pernyataan Structured Query Language (SQL) dalam form teks, dan kemudian mengubahnya ke hierarki struktur data ke layer yang lebih rendah dapat dengan mudah bekerja. *Tokenizer* SQLite di *handcoded*. Parser di-*generate* oleh *custom parser generator* milik SQLite, yang dinamakan Lemon. Lemon didesain untuk performansi tinggi dan dapat mengambil tindakan pencegahan terhadap kebocoran memori. Setelah *statement* dipecah menjadi token, dievaluasi, dan disusun kembali dalam bentuk pohon parsing, parser melewati seluruh dahan pohon ke kode generator.

Kode generator menerjemahkan pohon parsing menjadi semacam bahasa assembly khusus untuk SQLite. Bahasa assembly ini terdiri dari instruksi yang dieksekusi oleh mesin virtual. Kode generator memiliki pekerjaan untuk mengkonversi pohon parsing menjadi sebuah mini-program yang lengkap ditulis dalam perakitan ini dan menyerahkan pada mesin virtual untuk pengolahan.

3. The Virtual Machine

Di tengah tumpukan terdapat *virtual machine, Virtual Database Engine* (VDBE). VDBE bekerja pada kode bit seperti mesin virtual Java atau bahasa scripting interpreter. Kode bit VDBE terdiri dari 128 opcodes, yang semuanya berpusat di sekitar operasi database. VDBE adalah mesin virtual yang dirancang khusus untuk pengolahan data. Setiap instruksi dalam instruksi yang ditetapkan baik menyelesaikan operasi *database* tertentu (seperti membuka kursor tabel, membuat *record*, mengekstrak kolom, atau memulai transaksi) atau memanipulasi

stack dalam beberapa cara untuk mempersiapkan operasi semacam itu. Bersama dan dalam urutan yang benar, kumpulan instruksi VDBE ini dapat memenuhi perintah SQL, walaupun kompleks. Setiap pernyataan SQL dalam SQLite untuk memilih dan merubah baris untuk membuat tabel, *view* dan indeks yang pertama dikompilasi ke dalam bahasa mesin virtual ini, membentuk sebuah program *standalone* yang mendefinisikan bagaimana melakukan perintah yang diberikan.

4. The Back-end

Back-end terbuat dari *B-tree, page cache,* dan *OS interface*. *B-tree* dan *page cache* bekerja bersama sebagai *information brokers*. Transaksi mereka ialah halaman *database*, yang memiliki blok data yang seragam, seperti mobil angkutan, yang dibuat untuk transportasi. Di dalam halaman adalah barangnya: bit informasi yang lebih menarik seperti *record*, kolom, dan indeks masukan. Baik *B-tree* dan *page cache* tidak memiliki pengetahuan tentang konten. Mereka hanya memindahkan dan mengatur halaman, tidak peduli isinya.

5. Utilities and Test Code

Macam – macam utilitas dan *services* yang banyak digunakan seperti alokasi memori, perbandingan string, dan konversi *Unicode* secara rutin disimpan dalam modul utilitas. Pada dasarnya menangkap semua modul untuk melayani beberapa modul yang ingin digunakan atau dibagikan. Modul pengujian berisi segudang tes regresi yang dirancang untuk memeriksa setiap sudut kecil dari kode *database*. Modul ini menjadi satu alasan mengapa SQLite sangat handal, ia dapat melakukan banyak pengujian regresi

2.18 UML (*Unified Modelling Language*)

Unified Modelling Language (UML) merupakan sebuah notasi grafik untuk menggambarkan diagram konsep perangkat lunak. Dapat digunakan untuk menggambar diagram dari domain masalah, membuat desain perangkat lunak, atau implementasi perangkat lunak yang telah lengkap [14]. *Fowler* menjelaskan terdapat tiga perbedaan yaitu Konseptual, Spesifikasi, dan Implementasi.

Pada tahap Spesifikasi dan Impelementasi memiliki hubungan yang kuat dengan kode program. Itu merupakan kebutuhannya mengubah tingkat *Spesifikasi* menjadi kode sumber. Dengan demikian ada aturan dan semantik diagram pada tingkat tersebut harus diikuti. Diagram tersebut memiliki sedikit ambiguitas dan banyak formalitas.

UML merupakan kesatuan dari bahasa pemodelan yang dikembangkan oleh Booch, *Object Modelling Technique* (OMT) dan *Object Oriented Software Engineering* (OOSE) [15]. Metode Booch dari Grady Booch sangat terkenal dengan nama metode *Design Object Oriented*. Metode ini menjadikan proses analisis dan desain ke dalam empat tahapan iteratif, yaitu: identifikasi kelas – kelas dan obyek – obyek, identifikasi semantic dari hubungan obyek dan kelas terbut, perincian interface dan implementasi. Keunggulan metode Booch adalah pada detil dan kayanya dengan notasi dan elemen. Pemodelan OMT yang dikembangkan oleh Rumbaugh didasarkan pada analisis terstruktur dan pemodelan entity-relationship. Tahapan utama dalam metodologi ini adalah analisis, desain sistem, desain obyek dan implementasi. Keunggulan metode ini adalah dalam penotasian yang mendukung semua konsep OO. Metode OOSE dari Jacobson lebih memberi penekanan pada use case. OOSE memiliki tiga tahapan yaitu membuat model requirement dan analisis, desain, implementasi, dan model pengujian (test model). Keunggulan metode ini adalah mudah dipelajari karena memiliki notasi yang sederhana namun mencakup seluruh tahapan dalam rekayasa perangkat lunak.

Dengan UML, metode Booch, OMT dan OOSE digabungkan dengan membuang elemen – elemen yang tidak praktis ditambah dengan elemen – elemen dari metode lain yang lebih efektif dan elemen – elemen baru yang belum ada pada metode terdahulu sehingga UML lebih ekspresif dan seragam daripada metode lainnya. Gambar berikut adalah unsur – unsur yang membentuk UML.



Sumber Gambar: Munawar [15]

Gambar 2.12 Unsur Pembentuk UML

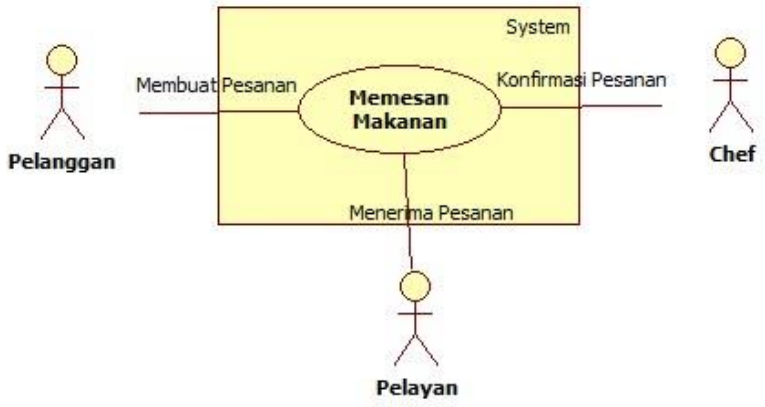
2.18.1 Diagram UML

UML mendefinisikan diagram – diagram berikut ini:

a. *Use Case Diagram*

Use Case Diagram membantu dalam menentukan fungsi dan fitur dari perangkat lunak dari perspektif pengguna. Sebuah *use case* mendeskripsikan bagaimana *user* berinteraksi dengan sistem dengan mendefinisikan langkah – langkah yang dibutuhkan untuk mencapai tujuan yang spesifik [5].

Sebuah diagram *use case* merupakan gambaran dari semua *use case* dan bagaimana mereka terkait. Dalam diagram *use case*, *use case* ditampilkan sebagai oval. Para aktor dihubungkan oleh garis kepada *use case* yang berhubungan. Perhatikan bahwa tidak ada rincian *use case* pada diagram dan harus dijelaskan secara terpisah. Perhatikan juga bahwa *use case* ditempatkan dalam persegi panjang tetapi aktor tidak. Persegi panjang ini adalah pengingat visual dari batas – batas sistem dan bahwa pelaku berada di luar sistem.



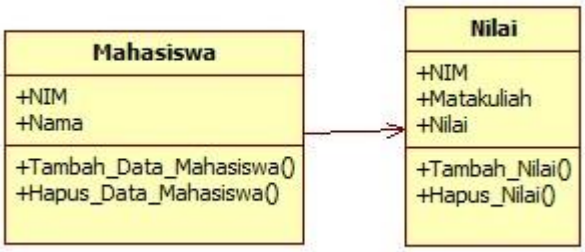
Gambar 2.13 Contoh Diagram Use Case

Gambar 2.17 menunjukan Ruang Lingkup Diagram *use case*. Sebuah persegi besar merupakan ruang lingkup sistem. Semua yang terdapat dalam persegi tersebut merupakan sistem yang dibangun. Diluar persegi tersebut kita dapat melihat aktor yang mempunyai aksi terhadap sistem. Sederhananya aktor ialah user pengguna. Aktor juga dapat menjadi sistem lain, dapat juga berupa perangkat seperti jam [14].

Di dalam ruang persegi terlihat *use case*. Dengan bentuk oval dan nama didalamnya. Garis terhubung antara aktor dan *use case* yang memiliki relasi. Hindari menggunakan panah, tidak ada yang tahu kemana arah itu mengacu.

b. *Class Diagram*

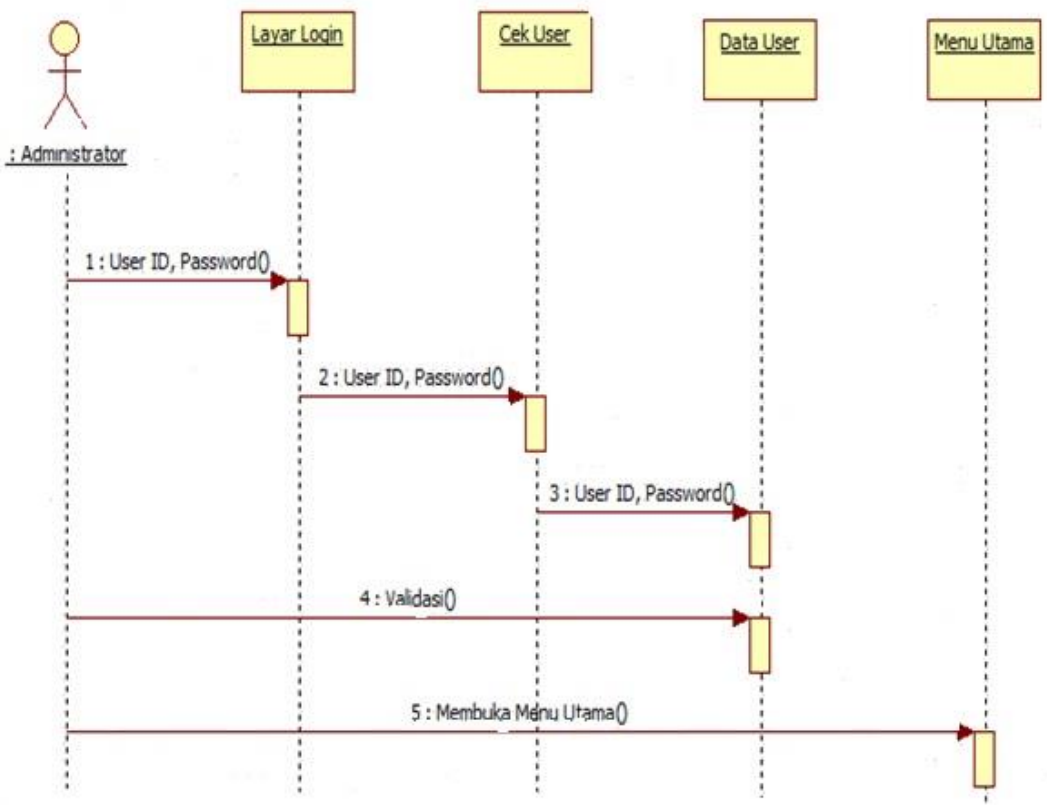
Class Diagram dalam UML memungkinkan kita untuk menunjukan konten yang statis dan hubungan antara *class*. Dalam *class diagram*, kita dapat memunculkan anggota *variable* dan kumpulan fungsi yang terdapat dalam *class*. Dapat juga memunculkan apakah suatu *class* merupakan turunan dari yang lain atau apakah itu menjadi *reference* bagi *class* lain. Singkatnya, *class diagram* menggambarkan semua kode sumber beserta ketergantungannya terhadap *class* [14].



Gambar 2.14 Contoh *Class Diagram*

c. *Sequence Diagram*

Sequence Diagram merupakan model dinamis yang sering digunakan dalam UML. Sesuai perkiraan, UML menyediakan banyak sekali komponen yang dapat membantu menggambar diagram yang sesuai kebutuhan [14]. *Sequence Diagram* menjelaskan secara detail urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari *use case* interaksi yang terjadi antar *class*, operasi apa saja yang terlibat, urutan antar operasi, dan informasi yang diperlukan oleh masing – masing operasi. Pembuatan *sequence diagram* merupakan aktifitas yang paling penting dari proses desain karena artifak inilah yang akan menjadi pedoman dalam proses pemrograman nantinya dan berisi aliran control dari program. Oleh karena itu diperlukan waktu yang lebih lama di pembuatan *sequence diagram* ini untuk menghasilkan diagram yang terdesain dengan baik. *Sequence diagram* biasanya tersusun dari elemen Obyek, *Interaction* dan *Message*. *Interaction* menghubungkan 2 obyek dengan pesannya.

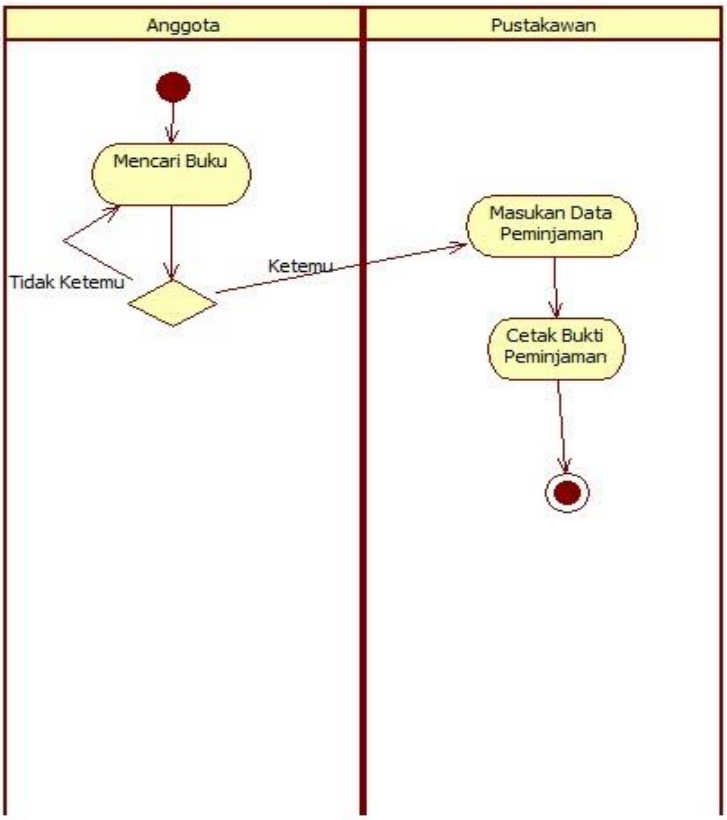


Gambar 2.15 Contoh *Sequence Diagram*

d. *Activity Diagram*

Activity Diagram menggambarkan berbagai alir aktifitas dalam sistem yang sedang dirancang, bagaimana masing – masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity* diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

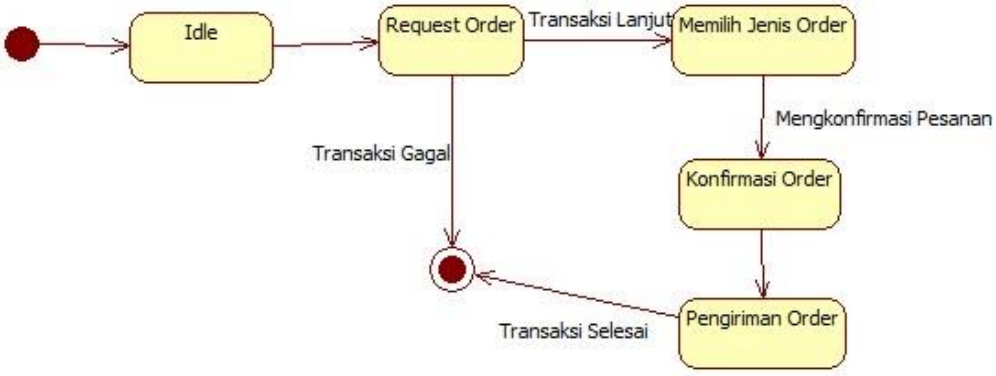
Setiap *activity diagram* selalu mempunyai satu *initial state*. *Initial node* yang digambarkan dengan simbol lingkaran padat merupakan titik yang mengawali *activity diagram*. *Activity diagram* dapat diakhiri dengan memberikan *activity final diagram* yang digambarkan dengan lingkaran padat dengan mempunyai cincin dibagian luarnya.



Gambar 2.16 Contoh *Activity Diagram*

e. *Statechart Diagram*

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu state ke state lainnya) suatu objek pada sistem sebagai akibat dari stimuli yang diterima. Jika diagram sekuen digunakan untuk interaksi antar objek maka diagram status digunakan untuk interaksi dalam sebuah objek. Perubahan tersebut digambarkan dalam suatu graf berarah.



Gambar 2.17 Contoh *Statechart Diagram*