BAB 2
LANDASAN TEORI

2.1 Definisi Kulit Manusia

Kulit merupakan lapisan terluar manusia yang bersentuhan langsung dengan lingkungan diluar tubuh. Untuk itu kulit berperan sebagai pelindung tubuh dari kerusakan atau pengaruh lingkungan yang buruk. Selain itu kulit mempunyai nilai estetika yang tinggi bagi manusia. Dengan peran tersebut, maka keberadaan kulit menjadi sangat penting bagi kesehatan dan penampilan seseorang.

2.2 Proses Penuaan Dini Pada Kulit Manusia

Proses kulit menua mempunyai dua fenomena yang saling berkaitan, yaitu:

1. Proses Penuaan Dini Intrinsik (*Intrinsic aging*; *true aging*; *chronologic aging*)
   Merupakan proses menua fisiologik yang berlangsung secara alamiah, disebabkan berbagai faktor dari dalam tubuh sendiri seperti genetika, hormonal dan rasial. Fenomena ini tidak dapat dicegah/dihindari dan mengakibatkan perubahan kulit yang menyeluruh sesuai dengan pertambahan usia.
2. Proses Penuaan Dini Ekstrinsik (*Extrinsic aging*)
   Terjadi akibat berbagai faktor dari luar tubuh. Faktor lingkungan seperti sinar matahari, kelembaban udara, suhu dan berbagai faktor eksternal lainnya dapat mempercepat proses menua kulit sehingga terjadi penuaan dini. Perubahan pada kulit terutama terjadi di daerah terpajan seperti kulit wajah sehingga wajah terlihat lebih tua, tidak sesuai dengan usia yang sebenarnya. Berbagai usaha dapat dilakukan untuk mencegah/memperlambat terjadinya penuaan dini ini.

2.3 Ciri Kulit Mengalami Penuaan Dini

Berikut beberapa tanda penuaan dini yang terlihat dikulit, yaitu:

1. Muncul Bercak Hitam atau *Age Spot/Liver Spot*
   Liver spot yang juga dikenal dengan *age spot* merupakan bercak-bercak hitam yang muncul pada penuaan kulit.

2. Tekstur Kulit Kasar

   Kulit kering dan kasar juga merupakan tanda umum yang dialami saat kulit mengalami penuaan dini. Ketika kulit terlalu sering terpapar matahari, kolagen dan elastin kulit akan rusak.

3. Pori-Pori Membesar

   Pembesaran pori-pori juga terkait dengan penuaan dini pada kulit. Seiring bertambahnya usia, pori-pori tumbuh lebih besar karena penumpukan sel kulit mati disekitar pori-pori.

4. Timbulnya Keriput Pada Kulit

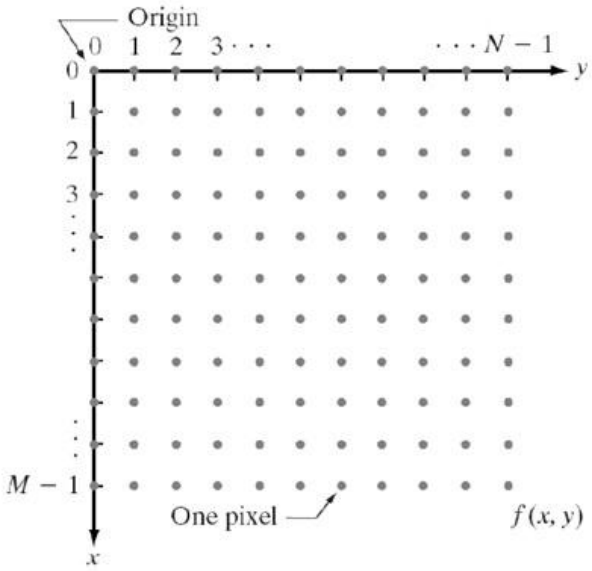
   Keriput disebabkan oleh paparan sinar matahari dan kerutan yang intensif oleh merokok. Paparan radikal bebas menyebabkan kerusakan DNA kulit dalam membran sel, akibatnya kolagen kulit dan elastin turun dari waktu ke waktu, hingga kulit mulai merosot dan kehilangan elastisitasnya.

2.4 Pengolahan Citra Digital

Secara umum, pengolahan citra digital menunjuk pada pemrosesan gambar dua dimensi menggunakan komputer. Dalam konteks yang lebih luas, pengolahan citra digital mengacu pada pemrosesan setiap data dua dimensi. Citra digital

merupakan sebuah larik (*array*) yang berisi nilai-nilai *real* maupun komplek yang direpresentasikan dengan deret *bit* tertentu.

Suatu citra dapat didefinisikan sebagai fungsi f(x,y) berukuran M baris dan N kolom, dengan x dan y adalah koordinat spasial, dan amplitudo f di titik koordinat (x,y) dinamakan intensitas suatu tingkat keabuan dari citra pada titik tersebut. Apabila nilai x, y dan nilai ampitudo f secara keseluruhan sehingga (*finite*) dan bernilai diskrit maka dapat dikatakan bahwa citra tersebut adalah citra digital. Gambar 2.12 menujukan posisi koordinat citra digital.



Gambar 2.1 Koordinat Citra Digital

Citra digital dapat ditulis dalam bentuk matrik sebagai berikut.

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,0) & \cdots & f(0,N-1) \\ f(0,0) & f(0,0) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix} \quad \text{.....(2-1)}$$

Nilai pada suatu irisan antara baris dan kolom (pada posisi x,y) disebut dengan *picture elements*, *image elements*, *pels*, atau *pixels*. Istialah terakhir (*pixel*) paling sering digunakan pada citra digital.

2.4.1 Teknik-Teknik Pengambilan Citra Digital

Ada beberapa teknik pengambilan citra digital yang bisa dilakukan, antara lain dengan menggunakan kamera digital (atau boleh juga *web-cam*) atau menggunakan scanner (bisa *scanner* umum, *finger-print scanner*, atau *bar-code scanner*). Hasil dari citra dengan penggunaan teknik kamera atau *scanner* berupa citra raster (atau citra dengan model matriks). Teknik pengambilan citra selain membutuhkan peralatan input, juga dibutuhkan suatu *card* yang disebut dengan *frame grabber* yang berupa rangkaian untuk mengolah citra secara *hardware*.

Selain itu, pengambilan citra dapat dilakukan dengan menggunakan teknik grafika komputer, yaitu dengan membentuk objek citra komputer sesuai dengan

teknik pengambilan citra. Teknik ini banyak digunakan untuk keperluan *entertainment* seperti pembuatan animasi atau untuk keperluan GIS *(Geographical Information System)*. Hasil dari citra dengan menggunakan teknik grafika ini bisa berupa citra raster atau citra vektor.

Baik citra raster maupun citra vektor mempunyai kelebihan dan kekurangannya masing-masing sehingga untuk memutuskan menggunakan model citra raster atau citra vektor diperlukan pengetahuan yang mendalam mengenai proyek yang sedang dikerjakan. Dalam pembahasan pengolahan citra digital lebih banyak dibahas mengenai model citra raster, baru pada pengolahan yang lebih lanjut, model citra vektor diperlukan.

Berikut ini beberapa contoh citra selain gambar atau foto yang sering dilihat baik dengan scanner atau kamera, yang dibedakan berdasarkan teknik pengambilan sinyalnya.

1. *Gamma Ray Imaging*, citra ini biasanya digunakan untuk citra-citra nuklir dan proses nuklir, juga digunakan untuk keperluan astronomi.
2. *X-Ray Imaging*, citra ini banyak digunakan untuk keperluan medis, rangkaian elektronik, dan astronomi.
3. *Image Ultraviolet*, citra ini banyak digunakan untuk keperluan citra biologi, mikroskopi, lithografi, laser, dan astronomi.
4. *Image Infrared*, citra ini banyak digunakan untuk keperluan farmasi mikrobiotik, dan mikro material.
5. *Image Ultrasound*, citra ini banyak digunakan untuk keperluan medis seperti USG dan diagnosa *thyroid*.

2.4.2 Teknik-Teknik Pengolahan Citra Digital

Secara umum, teknik pengolahan citra digital dibagi menjadi tiga tingkat pengolahan, yakni sebagai berikut:

a) Tahap 1, yang dinamakan dengan *low-level processing* (pengolahan tingkat rendah). Pengolahan ini merupakan operasi dasar dalam pengolahan citra, seperti pengurangan noise *(noise reduction)*, perbaikan citra *(image enhancement)*, dan restorasi citra *(image restoration)*.

b) Tahap 2, yang dinamakan dengan *mid-level processing* (pengolahan tingkat menengah). Pengolahan ini meliputi segmentasi pada citra, deskripsi objek, dan klasifikasi objek secara terpisah.
c) Tahap 3, yang dinamakan dengan *high-level processing* (pengolahan tingkat tinggi). Pengolahan ini meliputi analisis citra *(image analysis)*.

Dari ketiga tahap pengolahan citra digital diatas, dapat dinyatakan suatu gambaran mengenai teknik-teknik pengolahan citra digital dan macam-macamnya, antara lain:

1. Perbaikan kualitas citra *(image enhancement)*

   Jenis operasi ini bertujuan untuk memperbaiki kualitas citra dengan cara memanipulasi parameter-parameter citra seperti kontras dan kecerahannya. Dengan operasi ini, ciri-ciri khusus yang terdapat di dalam citra lebih ditonjolkan.
2. Pemugaran citra *(image restoration)*

   Jenis operasi ini bertujuan untuk memperbaiki model citra, biasanya berhubungan dengan bentuk citra yang sesuai.
3. Pemampatan citra *(image compression)*

   Jenis operasi ini dilakukan agar citra dapat direpresentasikan dalam bentuk yang lebih *compact* sehingga memerlukan memori yang lebih sedikit. Hal penting yang harus diperhatikan dalam pemampatan adalah citra yang telah dimampatkan harus tetap mempunyai kualitas gambar yang bagus.
4. Segmentasi citra *(image segmentation)*

   Jenis operasi ini bertujuan untuk memecah suatu citra ke dalam beberapa segmen dengan suatu kriteria tertentu. Jenis operasi berkaitan erat dengan pengenalan pola.
5. Analisis citra *(image analysis)*

   Jenis operasi ini bertujuan untuk menghitung besaran kuantitatif dari citra untuk menghasilkan deskripsinya. Teknik analisis citra mengekstrasi ciri-ciri tertentu yang membantu dalam identifikasi objek.
6. Rekonstruksi citra *(image reconstruction)*

Jenis operasi ini bertujuan untuk membentuk ulang objek dari beberapa citra hasil proyeksi. Operasi rekonstruksi citra banyak digunakan dalam bidang medis.

2.5 Jenis Citra

Nilai suatu pixel memiliki nilai dalam rentang tertentu, dari nilai minimum sampai nilai maksimum. Jangkauan yang digunakan berbeda-beda tergantung dari jenis warnanya. Namun secara umum jangkauannya adalah 0-255. Citra dengan penggambaran seperti ini digolongkan ke dalam citra integer. Berikut adalah jenis-jenis citra berdasarkan nilai pixelnya.

2.5.1 Citra Warna

Pada citra warna, setiap titik mempunyai warna spesifik yang merupakan kombinasi dari tiga warna dasar, yaitu merah, hijau, dan biru. Format citra ini sering disebut sebagai citra RGB (red-green-blue). Setiap warna dasar mempunyai intensitas sendiri dengan nilai maksimum 255 (8 bit). Misalnya warna kuning merupakan kombinasi warna merah dan hijau sehingga nilai RGB-nya adalah 255 255 0, sedangkan warna ungu muda nilai RBG-nya adalah 150 0 150. Dengan demikian, setiap titik pada citra warna membutuhkan data 3 byte atau 24 bit.

Jumlah kombinasi warna yang mungkin untuk format citra ini adalah 224 atau lebih dari 16 juta warna, dengan demikian bisa dianggap mencakup semua warna yang ada. Inilah sebabnya format ini dinamakan *true color*.



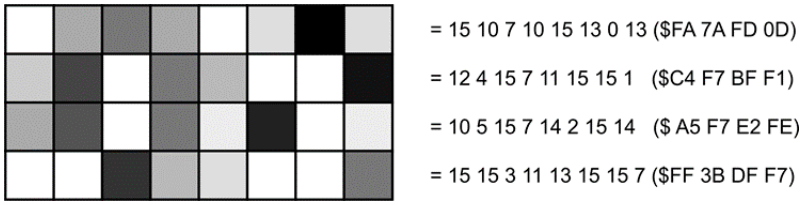
Gambar 2.2 Citra Warna dan Representasinya dalam Data Digital

2.5.2 Citra Grayscale

Citra skala keabuan memberi kemungkinan warna yang lebih banyak daripada citra biner, karena ada nilai-nilai lain diantara nilai minimum (biasanya

0) dan nilai maksimumnya. Banyaknya kemungkinan nilai minimum dan nilai maksimumnya bergantung pada jumlah *bit* yang digunakan. Contohnya untuk skala keabuan 4 *bit*, maka jumlah kemungkinan nilainya adalah 24 = 16, dan nilai maksimumnya adalah 24 – 1 = 15. Sedangkan untuk skala keabuan 8 *bit*, maka jumlah kemungkinan nilainya adalah 28 = 256, dan nilai maksimumnya adalah 28 – 1 = 255.

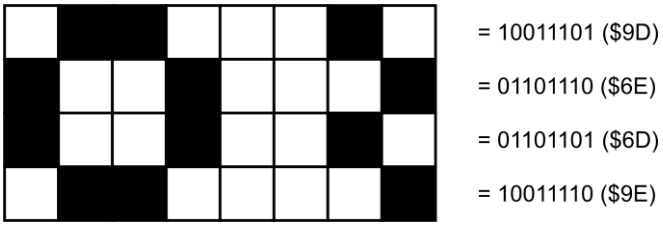
Format citra ini disebut skala keabuan karena pada umumnya warna yang dipakai adalah antara hitam sebagai warna minimal dan warna putih sebagai warna maksimalnya. Namun pada prakteknya, warna yang dipakai tidak terbatas pada warna abu-abu. Sebagai contoh, jika warna minimalnya adalah putih dan warna maksimalnya adalah merah, maka semakin besar nilainya, semakin besar pula intensitas warna merahnya.



Gambar 2.3 Citra *Grayscale* 4 *bit* dan Representasinya dalam Data Digital

2.5.3 Citra Biner

Pada citra biner, setiap titik bernilai 0 atau 1, masing-masing merepresentasikan warna tertentu. Warna hitam bernilai 0 dan warna putih bernilai 1. Pada standar citra untuk ditampilkan di layar komputer, nilai biner ini berhubungan dengan ada tidaknya cahaya yang ditembakkan oleh *electron gun* yang terdapat di dalam monitor komputer. Angka 0 menyatakan tidak ada cahaya, dengan demikian warna yang direpresentasikan adalah hitam. Untuk angka 1 terdapat cahaya, sehingga warna yang direpresentasikan adalah putih. Setiap titik pada citra hanya membutuhkan 1 bit, sehingga setiap *byte* dapat menampung informasi 8 titik.



Gambar 2.4 Citra Biner dan Representasinya Dalam Data Digital

2.6 Peningkatan Kualitas Citra

Perbaikan citra biasanya dilakukan pada awal proses pengolahan citra. Proses ini dilakukan guna menghasilkan kualitas citra yang lebih baik untuk diolah pada tahap selanjutnya. Kualitas suatu citra dipengaruhi oleh empat faktor utama yaitu:

1. Pencahayaan

   Pencahayaan berhubungan dengan cahaya atau sumber cahaya yang terdapat disekitar lingkungan sistem. Secara langsung masalah pencahayaan mempengaruhi kualitas citra yang diperoleh. Teknik pencahayaan yang baik akan menghasilkan citra yang baik atau tidak mengandung derau yang berlebihan. Hal ini penting untuk pemrosesan selanjutnya di bagian pengolahan citra, sehingga dihasilkan informasi yang akurat mengenai obyek dan lingkungan sistem pada umumnya.

2. Pemfokusan obyek

   Pemfokusan obyek terkait erat dengan lensa kamera yang digunakan sebagaimana diketahui, lensa merupakan peralatan optik yang berfungsi mengumpulkan cahaya yang dipantulkan oleh obyek dan juga daerah sekitarnya. Pemfokusan obyek akan menentukan ketajaman atau kejelasan citra yang diperoleh.

3. Kuantitas pendeteksi obyek

   Pada sebuah kamera, pendeteksi obyek berupa dioda – dioda yang peka terhadap cahaya. Dioda – dioda ini memiliki kemampuan mendeteksi muatan cahaya dan mengkonversikannya menjadi muatan listrik. Disamping itu dioda – dioda akan membentuk array citra. Semakin banyak dioda yang menyusun

array citra, resolusi citra yang mampu dihasilkan akan semakin besar, berarti kualitas citra semakin baik pula.

4. Frekuensi sampling

   Faktor ini berhubungan dengan periode dan frekuensi sampling itu sendiri. Periode sampling merupakan waktu yang diperlukan untuk proses pengambilan sample citra analog. Sedangkan frekuensi sampling adalah banyaknya sample citra yang diambil dalam satu periode sampling. Frekuensi sampling untuk satu periode harus tepat agar citra yang dihasilkan dapat diidentifikasi dengan baik . frekuensi sampling sangat menentukan kualitas citra yang terbentuk pada proses digitalisasi ini. Frekuensi sampling yang berlebihan atau sebaliknya sangat kurang akan berdampak pada kejelasan citra.

Disamping keempat faktor umum diatas, ada beberapa faktor lain yang menyebabkan suatu citra banyak mengandung derau yaitu permukaan obyek, latar belakang obyek yang tidak rata, sudut pandang pengambilan citra dan kekontrasan antar obyek dengan latar belakangnya. Derau yang terlalu besar menyebabkan unsur – unsur obyek dalam suatu citra sulit untuk diperoleh.

Terkait dengan perangkat lunak yang digunakan, kualitas citra dapat ditingkatkan dengan dua metoda yang sederhana yaitu mengatur nilai *contras* dan *brigthness*. Kombinasi yang tepat antara kedua variabel tersebut dapat meningkatkan kualitas citra yang terbentuk, sehingga proses identifikasi unsur – unsur obyek dapat dilakukan dengan baik.

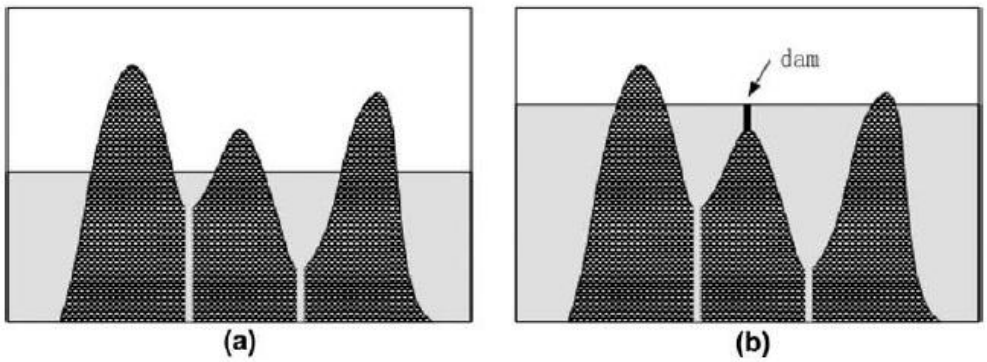
2.7 Watershed

Konsep *watershed* sebagai sebuah metode untuk melakukan segmentasi terhadap citra diperkenalkan oleh C.Lantuejoul dan Sergei Beucher (1979) dan dikembangkan oleh Jean Serra (1982). Transformasi *watershed* memandang citra sebagai sebuah relief topografi dimana intensitas setiap *pixel* mempresentasikan ketinggian topografinya[15]. Dalam sebuah permukaan topografi, apabila air hujan jatuh diatasnya, sesuai dengan hukum gravitasi maka air tersebut akan mengalir melewati jalur yang lebih rendah sampai ia mencapai ketinggian yang paling rendah atau minima dimana ia tidak dapat mengalir kemana-mana lagi.

Himpunan titik-titik pada permukaan topografi citra dimana aliran air yang melewatinya menuju ke minima tertentu yang sama, menjadi sebuah catchment basin (cekungan yang terisi air) yang berasosiasi dengan minima tersebut dan membentuk sebuah *region* citra[16].

Watershed terbentuk dilokasi dimana air dari kedua *catchment* basin yang berdekatan bertemu dan merupakan batas dari dua buah *catchment* basin tersebut. Keseluruhan watershed yang terbentuk menghasilkan seluruh kontur tertutup yang ada pada citra dan mempresentasikan obyek-obyek dalam citra yang telah tersegmentasi. Gambar 2.5 menunjukkan ilustrasi algoritma watershed dalam satu dimensi. permukaan air pada Gambar 2.5 (a) akan terus bertambah. Untuk memisahkan dua catchment basins yang berbeda setelah permukaan air terus bertambah, dibentuk dam antara dua garis sebagaimana Gambar 2.5 (b).

Tujuan utama dari algoritma segmentasi berdasarkan konsep ini adalah mencari garis *watershed* (Beusher S). Ide dasarnya sangat sederhana: andaikan sebuah lubang dilubangi di setiap bagian minimumnya dan seluruh topografi memenuhi dari bawah dengan membiarkan air sampai ke lubang pada tingkat yang seragam. Ketika air yang naik di kolam penangkapan maka air akan bergabung, oleh karena itu sebuah bendungan (dam) dibangun untuk mencegah penggabungan. Banjir akhirnya mencapai tahap di mana hanya bagian atas bendungan yang bisa dilihat di atas garis air. Batas bendungan ini berhubungan dengan pembagian garis dalam *watershed*.

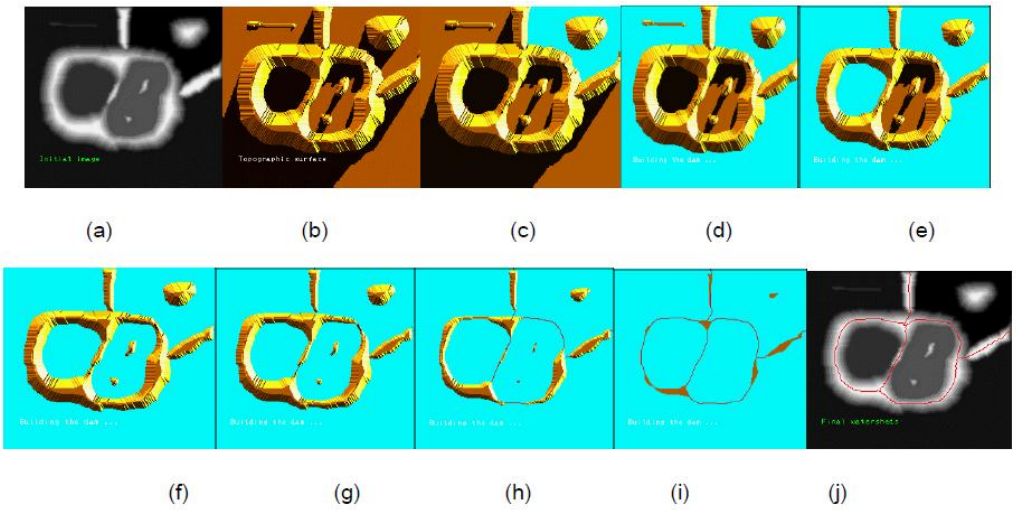


Gambar 2.5 *Watershed* pada Satu Dimensi. (a) Dam Belum Terbentuk pada Waktu t, (b) Dam Terbentuk pada Waktu t+b.

Salah satu aplikasi dasar dari segmentasi *watershed* adalah ekstraksi dari seragam objek yang dekat dari *background*. Bagian citra yang mempunyai sifat variasi kecil di tingkat keabuan mempunyai nilai *gradien* yang kecil. Akan tetapi, pada kenyataannya kita sering melihat segmentasi *watershed* diaplikasikan ke *gradient* dari sebuah citra dari pada citra itu sendiri. Pada perumusan ini, *regional* minima dari kolam penangkapan berhubungan dengan nilai kecil dari *gradient* yang berhubungan ke objek yang diamati.

Contoh dapat dilihat pada Gambar 2.6 pada bagian (a) adalah citra asli. Pada bagian (b) adalah citra topografi. Citra topografi adalah bentuk citra 3 dimensi jika dilihat dari atas. Pada bagian(c)-(g) adalah tahap *flooding*. Pada bagian (h) dan (i) tampak dibangun dam agar dua buah *catchment basins* tidak bergabung. Bagian (j) adalah hasil akhir garis *watershed*.

Ada dua metode utama yang terdapat dalam *transformasi watershed* terhadap citra yang merupakan representasi ulang dari simulasi *watershed* dalam konsep topografi. Kedua metode tersebut mengacu pada konsep *flooding* dalam yang menggambarkan perilaku air dalam suatu *relief* topografi. Metode pertama disebut sebagai *watershed* dengan simulasi perendaman (*immersion simulation*), dan metode kedua adalah dengan simulasi air hujan (*rainfall simulation*).



Gambar 2.6 Pembentukan *Watershed* (a) Citra asli. (b) Pemandangan topografik. (c)-(g) Lima Tahap *Flooding* (h) Awal Penggabungan Dua Kolam.(i) Pembentukan Dam. (j) Garis *Watershed* pada Citra

2.8 Pengenalan Pola

Pola adalah suatu entitas yang terdefinisi (mungkin secara samar) dan dapat diidentifikasi serta diberi nama. Pola bisa merupakan kumpulan hasil pengukuran atau pemantauan dan bisa dinyatakan dalam notasi vektor. Contoh : sidik jari, raut wajah, gelombang suara, tulisan tangan dan lain sebagainya. Dalam pengenalan pola data yang akan dikenali biasanya dalam bentuk citra atau gambar, akan tetapi ada pula yang berupa suara [10].

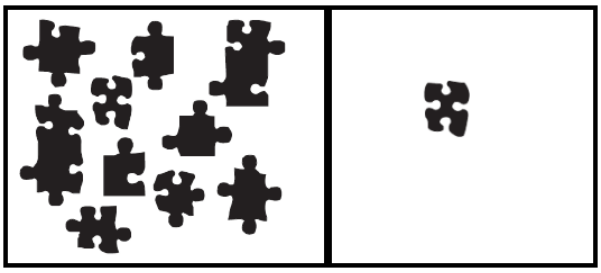
Secara umum pengenalan pola (*pattern recognition*) adalah suatu ilmu untuk mengklasifikasikan atau menggambarkan sesuatu berdasarkan pengukuran kuantitatif fitur (ciri) atau sifat utama dari suatu obyek.

Menurut Theodoridis dan Koutroumbas [11], pengenalan pola (*pattern recognition*) dapat diartikan sebagai proses klasifikasi dari objek atau pola menjadi beberapa kategori atau kelas yang bertujuan untuk pengambilan keputusan.

Pengenalan pola (*pattern recognition*) merupakan teknik yang bertujuan untuk mengklasifikasikan citra yang telah diolah sebelumnya berdasarkan kesamaan atau kemiripan ciri yang dimilikinya. Bagian terpenting dari teknik pengenalan pola adalah bagaimana memperoleh informasi atau ciri penting yang terdapat dalam sinyal [10].

2.8.1 Template Matching

Pada dasarnya *template matching* adalah proses yang sederhana. Suatu citra masukan yang mengandung *template* tertentu dibandingkan dengan template pada basis data. Template ditempatkan pada pusat bagian citra yang akan dibandingkan dan dihitung seberapa banyak titik yang paling sesuai dengan template. Langkah ini diulangi terhadap keseluruhan citra masukan yang akan dibandingkan. Nilai kesesuaian titik yang paling besar antara citra masukan dan citra template menandakan bahwa template tersebut merupakan citra template yang paling sesuai dengan citra masukan.



Gambar 2.7 Ilustrasi Template Matching

Gambar 2.7 bagian kiri merupakan citra yang mengandung objek yang sama dengan objek pada *template* yang ada di sebelah kanan. *Template* diposisikan pada citra yang akan dibandingkan dan dihitung derajat kesesuaian pola pada citra masukan dengan pola pada citra template.

Tingkat kesesuaian antara citra masukan dan *citra template* bisa dihitung berdasarkan nilai eror terkecil [12] dengan menggunakan persamaan 2-2.

$$\min e = \sum_{(x,y)\epsilon W} (I_{x,y} - T_{x,y})^2 \text{ .....(2-2)}$$

I adalah pola *pixel* citra masukan yang akan dibandingkan. T adala pola *pixel citra template*. *Template* dengan nilai eror paling kecil adalah *template* yang paling sesuai dengan citra masukan yang akan dibandingkan. Ukuran objek yang beragam bisa diatasi dengan menggunakan *template* berbagai ukuran. Namun hal ini membutuhkan tambahan ruang penyimpanan. Penambahan *template* dengan berbagai ukuran akan membutuhkan komputasi yang besar. Jika suatu *template* berukuran persegi dengan ukuran m x m dan sesuai dengan *citra* yang berukuran N x N, dan dimisalkan *pixel* m2 sesuai dengan semua titik *citra*, maka komputasi yang harus dilakukan adalah O(N2m2). Komputasi tersebut harus dilakukan dengan *template* yang tidak beragam. Jika parameter *template* bertambah, seperti ukuran *template* yang beragam, maka komputasi yang dilakukan juga akan bertambah. Hal ini yang menyebabkan metode *template matching* menjadi lamban.

2.9 Android

Android adalah sebuah sistem operasi untuk peranggkat *mobile* berbasis linux yang mencakup sistem operasi, *middleware* dan aplikasi. Android

menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Perilisan pertama android pada tanggal 5 November 2007, android bersama *Open Handset Alliance* menyatakan mendukung pengembangan *open source* pada perangkat *mobile*. Di lain pihak, *Google* merilis kode-kode android di bawah lisensi *apache*, sebuah lisensi perangkat lunak dan *open platform* perangkat seluler.

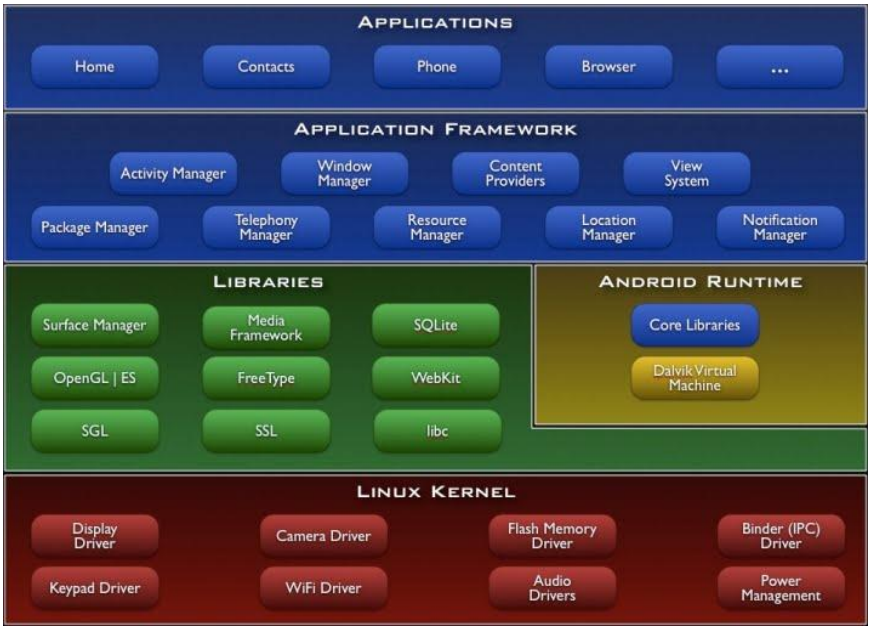
Di dunia ini terdapat dua jenis distributor sistem operasi android. Pertama yang mendapat dukungan penuh dari *Google* atau *Google Mail Services* (GMS) dan kedua adalah yang benar-benar bebas distribusinya tanpa dukungan langsung *google* atau dikenal sebagai *Open Handset Distribution* (OHD). Di dunia ini terdapat dua jenis distributor sistem operasi android. Pertama yang mendapat dukungan penuh dari *Google* atau *Google Mail Services* (GMS) dan kedua adalah yang benar-benar bebas distribusinya tanpa dukungan langsung *google* atau dikenal sebagai *Open Handset Distribution* (OHD).

Sekitar September 2007 Google mengenalkan Nexus one, salah satu jenis smartphone yang menggunakan android sebagai sistem operasinya. Telepon seluler ini diproduksi oleh HTC Corporation dan tersedia di pasaran pada 5 Januari 2010. Pada 9 Desember 2008, diumumkan anggota baru yang bergabung dalam program kerja Android ARM Holdings, Atheros Communications, diproduksi oleh Asustek Computer Inc, Garmin Ltd, Softbank, Sony Ericsson, Toshiba Corp, dan Vodafone Group Plc. Seiring pembentukan Open Handset Alliance, OHA mengumumkan produk perdana mereka, android perangkat mobile yang merupakan memodifikasi kernel linux 2.6. Sejak android dirilis telah dilakukan berbagai pembaruan berupa perbaikan bug dan penambahan fitur baru.

Tidak hanya menjadi sistem operasi di smartphone, saat ini android menjadi pesaing utama dari apple pada sistem operasi Table PC. Pesatnya pertumbuhan android selain faktor yang disebutkan di atas adalah karena android itu sendiri adalah platform yang sangat lengkap baik itu sistem operasinya, aplikasi dan tool pengembangan, market aplikasi android serta dukungan yang sangat tinggi dari komunitas open source di dunia, sehingga android terus berkembang pesat baik dari segi teknologi maupun dari segi jumlah device yang ada di dunia.

2.9.1 Arsitektur Android

Sistem operasi Android memiliki 4 lapisan (layer) yang merupakan komponen system Android. Gambar berikut merupakan lapisan arsitektur system operasi android.



Gambar 2.8 Diagram Arsitektur Sistem Operasi Android

Google mengibaratkan Android sebagai sebuah tumpukan software. Setiap lapisan dari tumpukan ini menghimpun beberapa program yang mendukung fungsi-fungsi spesifik dari system operasi. Berikut ini susunan dari lapisan – lapisan tersebut dilihat dari lapisan dasar hingga lapisan teratas.

2.9.1.1 Linux Kernel

Android dibangun di atas kernel Linux. 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*, *WiFi*, *Flash Memory*, audio, dan IPC (*Interprocess Communication*) untuk mengatur aplikasi dan lubang keamanan.

2.9.1.2 Libraries

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

1. Media Library untuk memutar dan merekam berbagai macam format audio dan video.
2. Surface Manager untuk mengatur hak akses layer dari berbagai aplikasi.
3. Graphic Library termasuk didalamnya SGL dan OpenGL, untuk tampilan 2D dan 3D.
4. SQLite untuk mengatur relasi database yang digunakan pada aplikasi.
5. SSl dan WebKit untuk browser dan keamanan internet.

Pustaka-pustaka tersebut bukanlah aplikasi yang berjalan sendiri, namun hanya dapat digunakan oleh program yang berada di level atasnya. Sejak versi Android 1.5, pengembang dapat membuat dan menggunakan pustaka sendiri menggunakan Native Development Toolkit (NDK).

2.9.1.3 Android Runtime

Pada android tertanam paket pustaka 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 pustaka yang telah ada. Dalam Android Runtime terdapat 2 bagian utama, diantaranya:

1. Pustaka Inti, android dikembangkan melalui bahasa pemrograman Java, tapi Android Runtime bukanlah mesin virtual Java. Pustaka inti android menyediakan hampir semua fungsi yang terdapat pada pustaka Java serta beberapa pustaka khusus android.
2. Mesin Virtual Dalvik, Dalvik merupakan sebuah mesin virtual yang dikembangkan oleh Dan Bornstein yang terinspirasi dari nama sebuah perkampungan yang berada di Iceland. Dalvik hanyalah interpreter mesin virtual yang mengeksekusi file dalam format *Dalvik Executable* (*.dex).

Dengan format ini Dalvik akan mengoptimalkan efisiensi penyimpanan dan pengalamatan memori pada file yang dieksekusi. Dalvik berjalan di atas kernel Linux 2.6, dengan fungsi dasar seperti *threading* dan manajemen memori yang terbatas.

2.9.1.4 Aplication Framework

Kerangka aplikasi menyediakan kelas-kelas yang dapat digunakan untuk mengembangkan aplikasi android. Selain itu, juga menyediakan abstraksi untuk mengakses perangkat, serta mengatur tampilan *User Interface* dan sumber daya aplikasi. Bagian terpenting dalam kerangka aplikasi android adalah sebagai berikut:

1. Activity Manager, berfungsi untuk mengontrol siklus hidup aplikasi dan menjaga keadaan ”*Backstack*“ untuk navigasi penggunaan.
2. Content Providers, berfungsi untuk merangkum data yang memungkinkan digunakan oleh aplikasi lainnya, seperti daftar nama.
3. Resource Manager berfungsi untuk mengatur sumber daya yang ada dalam program. Seperti karakter, grafik dan *file* layout.
4. Location Manager, berfungsi untuk memberikan informasi detail mengenail lokasi perangkat android berada.
5. Notification Manager, mencakup berbagai macam peringatan seperti, pesan masuk, janji, dan lain sebagainya yang akan ditampilkan pada status bar.

2.9.1.5 Aplication

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.

2.9.2 XML Pada Android

XML (*Extensible Markup Language*) adalah bahasa markup yang mendifinisikan seperangkat aturan untuk encoding dalam dokumen yang bersifat human-readable dan machine-readable. Pada android sendiri berkas berjenis XML memiliki banyak fungsi, diantaranya:

1. Berkas

   Android Manifest Setiap aplikasi android harus memiliki berkas ini (dengan nama AndroidManifest.xml) pada direktori root. Pada berkas ini berisi informasi penting tentang aplikasi untuk sistem android.

2. Layout

   Android menyediakan kosakata XML sederhana yang sesuai dengan kelas View dan subclass-nya, seperti untuk widget dan layout.

3. Database

   Database XML adalah persistensi data perangkat lunak sistem yang memungkinkan data yang akan disimpan dalam format XML. Data ini kemudian dapat dilihat, diekspor dan serialisasi ke dalam format yang diinginkan. XML database biasanya berhubungan dengan dokumen berorientasi database.

2.10 Java

Di antara sekian banyak perangkat lunak yang sudah ada, kita sering mendengar istilah Java, atau disebut juga Java J2ME (Java 2 *Micro Edition*), yang merupakan turunan dari bahasa Java, di mana fungsi-fungsinya difokuskan untuk pembuatan aplikasi pada lingkungan telepon seluler.

Java merupakan suatu teknologi perangkat lunak yang di dalamnya mencakup bahasa pemrograman. Selain itu Java juga merupakan suatu *platform* yang memiliki *virtual machine* dan *library* yang diperlukan untuk menulis dan menjalankan suatu program. Pada awalnya java dikembangkan pada lingkungan komputer oleh *Sun Microsystem Inc*. dengan tujuan untuk menghasilkan suatu bahasa komputer sederhana tanpa harus terikat pada arsitektur tertentu.

Pada tahun 1995 Sun meluncurkan sebuah browser berbasis Java dengan julukan Hot Java, kemudian diikuti Netscape yang memutuskan untuk membuat browser dengan dilengkapi bahasa Java. Setelahnya, ikut bergabung pula berbagai pengembang ternama diantaranya IBM dan Microsoft. Pada tahun berikutnya, *Sun Microsystem Inc*. merilis *Java Software Development Kit* (JDK) pertamanya, yaitu JDK 1.1. Kemudian terus berkembang dari pemrograman applet yang berjalan di browser menjadi pemrograman kelas dunia yang banyak digunakan untuk pengembangan aneka ragam aplikasi. Bahasa pemrograman Java sendiri secara garis besar dikelompokan menjadi 3, yaitu:

1. J2SE (Java 2 Standar Edition), digunakan untuk mengembangkan aplikasiaplikasi desktop dan applet.
2. J2EE (Java 2 Enterprise Edition) dipergunakan untuk mengembangkan aplikasi-aplikasi berbasis client/server berskala enterprise.
3. J2ME (Java 2 Micro Edition). J2ME diaplikasikan pada berbagai perangkat kecil dengan jumlah memori, kapasitas penyimpan dan user interface terbatas, seperti ponsel dan PDA.

Hingga saat ini ada dua versi *platform* Java J2ME yang umum digunakan pada telepon genggam yaitu MIDP 1.0 yang masih memiliki spesifikasi sederhana dan menyediakan fungsi dasar untuk aplikasi mobile, di antaranya basic user interface dan keamanan jaringan. Dan MIDP 2.0 yang sudah ditambahkan berbagai fasilitas seperti game, multimedia, dukungan berbagai jenis konektivitas, maupun OTA.

2.10.1 Kelebihan dan Kekurangan Java

Kelebihan Java:

a. Multiplatform. Kelebihan utama dari java ialah dapat dijalankan di beberapa platform/sistem operasi komputer, sesuai dengan prinsip tulis sekali, jalankan dimana saja. Dengan kelebihan ini pemrogram cukup menulis sebuah program java dan dikompilasi (diubah, dari bahasa yang dimengerti manusia menjadi bahasa mesin/bytecode) sekali lalu hasilnya dapat dijalankan di atas beberapa platform tanpa perubahan. Kelebihan ini memungkinkan sebuah program

berbasis java dikerjakan diatas operating sistem linux tetapi dijalankan dengan baik di atas Microsoft Windows. Platform yang didukung sampai saat ini adalah Microsoft Windows, linux, Mrac OS dan Sun Solaris. Penyebabnya adalah setiap sistem operasi menggunakan programnya sendiri-sendiri untuk menginterpretasikan bytecode tersebut.

b. OOP (Object Oriented Programming) yang artinya semua aspek yang terdapat di java adalah objek. Java merupakan salah satu bahasan pemrograman berbasis objek secara murni. Semua tipe data diturunkan dari kelas dasar yang disebut Object. Hal ini sangat memudahkan pemrogram untuk mendesain, membuat, mengembangkan, dan mengalokasi kesalahan sebuah program dengan basis java secara cepat, tepat, mudah dan terorganisir. Kelebihan ini menjadikan java sebagai salah satu bahasa pemrograman termudah, bahkan untuk fungsi yang advance seperti komunikasi antara komputer sekalipun
c. Library yang lengkap, dimana java terkenal dengan kelengkapan library/perpustakaan (kumpulan program-program yang disertakan dalam pemrogram java) yang sangat memudahkan dalam penggunaan oleh para pemrogram untuk membangun aplikasinya. Kelengkapan perpustakaan ini ditambah dengan keberadaan komunitas java yang besar yang terus menerus membuat perpustakaan-perpustakaan baru untuk melingkupi seluruh kebutuhan pembangunan aplikasi.
d. Bergaya C++ , memiliki sintaks seperti bahasa pemrograman C++ sehingga menarik banyak pemrogram C++ untuk pindah ke java. Saat ini pengguna java sangat banyak, sebagian besar adalah pemrogram C++ yang pindah ke java.
e. Pengumpulan sampah otomatis, memiliki fasilitas pengaturan penggunaan memori sehingga para pemrogram tidak perlu melakukan pengaturan memori secara langsung (seperti halnya dalam bahasa C++ yang dipakai secara luas).

Kekurangan Java:

a. Mudah didekompilasi. Dekompilasi adalah proses membalikkan dari kode jadi menjadi kode sumber. Ini memungkinkan karena kode jadi java merupakan bytecode yang menyimpan banyak atribut bahasa tingkat tinggi seperti

namanama kelas, metode dan tipe data. Hal yang sama juga terjadi pada Microsoft.Net Platform. Dengan demikian, algoritma yang digunakan program akan lebih sulit disembunyikan dan mudah dibajak/di-reverseengineer.

b. Penggunaan memori yang banyak. Penggunaan memori untuk program berbasis java jauh lebih besar daripada bahasa tingkat tinggi generasi sebelumnyaseperti C/C++ dan Pascal (lebih spesifik lagi delphi dan Object Pascal). Biasanya ini bukan merupakan masalah bagi pihak yang menggunakan teknologi terbaru, tetapi menjadi masalah bagi mereka yang masih harus berkutat dengan mesin komputer berumur lebih dari 4 tahun.

2.10.2 Tahap Kompilasi Java

Tulis/ubah. Pemrogram menulis program dan menyimpannya di media dalam bentuk berkas “Java”.

a. Kompilasi. Pengkompilasi membentuk bytecodes dari program menjadi bentuk berkas 'Class'
b. Muat. Pemuat kelas memuat bytecodes ke memori.
c. Verifikasi. Peng-verifikasi memastikan bytecodes tidak menganggu sistem keamanan java.
d. Jalankan. Gantt Chart. Apabila tidak melibatkan.

2.11 UML (Unified Modeling Language)

UML adalah kependekan dari *Unified Modeling Language* yang merupakan suatu cara untuk menyelesaikan suatu masalah dengan mendeskripsikannya yang telah menjadi standar dalam dunia industri untuk memvisualisasikan, merancang dan mendokumentasikan sistem perangkat lunak. Dengan menggunakan UML kita dapat membangun model untuk segala bentuk dan jenis aplikasi perangkat lunak, yang mana aplikasi yang dibangun dapat berjalan pada perangkat lunak dengan sistem operasi dan jaringan apapun. Selain itu, dengan menggunakan UML, programmer dapat mengerti, memahami dengan jelas maksud, tujuan dan arah rancangan sistem, sehingga sistem perngakat lunak yang dirancang dapat dibangun dengan bahasa program apapun. Namun karena UML merupakan suatu model pengembangan sistem perangkat lunak yang berbasis object oriented

sehingga menggnuakan bentuk class dan operation dalam konsep dasarnya, maka lebih cocok dalam pemrogramannya dengan menggunakan bahasa-bahasa pemrograman yang berbasis OOP (*Object Oriented Programming*) seperti C, C++, Java, VB dan lain sebagainya.

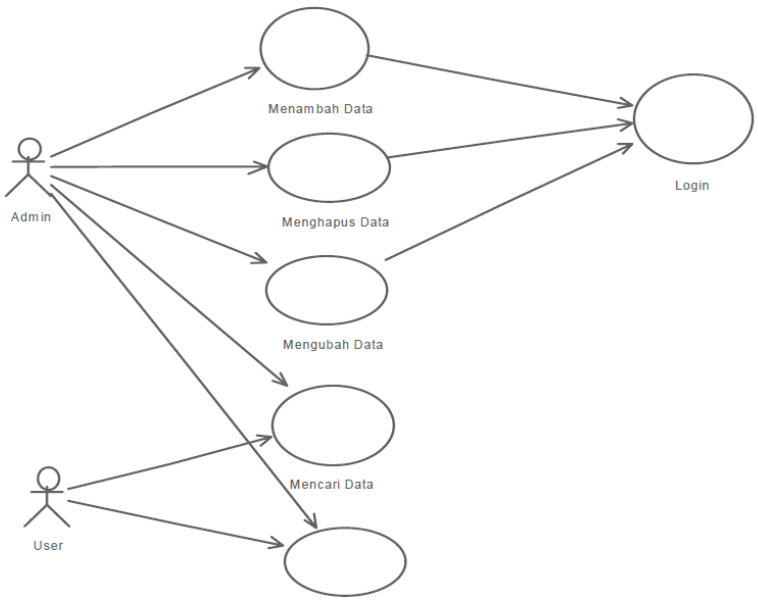
Secara lebih mendalam, UML lebih dari sekedar sebuah standard dan penemuan dari suatu notasi-notasi yang disatukan, tetapi juga berisi konsep-konsep baru yang menarik yang tidak ditemukan secara umum dalam komunitas object oriented. Dalam membangun suatu model perangkat lunak dengan UML, digunakan bentuk-bentuk diagram atau symbol untuk merepresentasikan elemen-elemen dalam sistem. Bentuk diagram yang digunakan untuk merepresentasikannya adalah sebagai berikut:

1. Use-case Diagram
2. Class Diagram
3. Sequence diagram
4. Collaboration Diagram
5. Activity Diagram
6. Component Diagram
7. Deployment Diagram

2.11.1 Use-case Diagram

Use-case diagram merupakan suatu bentuk diagram yang menggambarkan fungsi-fungsi yang diharapkan dari sebuah sistem yang dirancang. Dalam *Use-case diagram* penekanannya adalah "apa" yang diperbuat oleh sistem, dan bukan "*bagaimana*". Sebuah *use-case* akan merepresentasikan sebuah interaksi antara pelaku atau *actor* dengan sistem.

Use-case diagram yang digunakan dalam mercancang suatu sistem dapat sangat membantu pada saat menyusun *requirement* sebuah sistem, mengomunikasikannya dengan klien, dan merancang pengujian untuk semua fitur yang terdapat dalam sistem. Dalam suatu sistem aplikasi database, *use-case diagram* sangat membantu *requierement* apa saja yang diperlukan.



Gambar 2.9 Contoh Use-case Diagram

2.11.2 *Class* Diagram

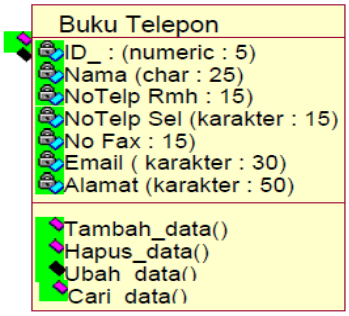
Sebuah *Class* diagram menunjukkan struktur yang statis dari beberapa class dalam suatu sistem. *Class-class* merepresentasikan suatu keadaan (atribut/properti) dan yang akan dikerjakan oleh sistem (metoda/fungsi). Class memiliki tiga area pokok:

1. Nama (dan *stereotype*)
2. Atribut
3. Metoda

Atribut dan metoda dalam class diagram dapat memiliki salah satu sifat seperti berikut di bawah ini:

1. Private, hanya dapat diakses oleh class itu sendiri.
2. Protected, hanya dapat diakses oleh class itu sendiri dan turunan dari class tersebut.
3. Public, dapat diakses oleh class selain dari class yang bersangkutan.

   Class dapat direpresentasikan dalam sebuah *interface* atau sebaliknya merupakan implementasi dari sebuah *interface* yang berupa class abstrak yang hanya tidak memiliki *attribute* dan hanya memiliki metoda. Berikut merupakan bentuk class diagram secara umum:

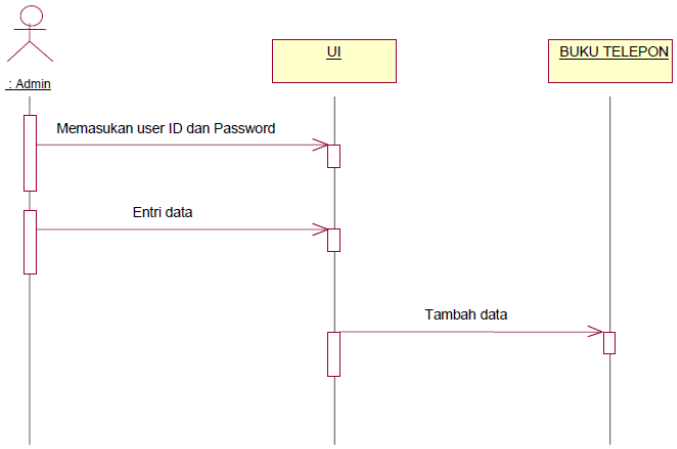


Gambar 2.10 Contoh Class Diagram

2.11.3 Sequance Diagram

Sequence diagram merupakan diagram yang mengambarkan kolaborasi yang dinamis antara obyek satu dengan yang lain. Kolaborasi ini ditunjukkan dengan adanya interaksi antar obyek di dalam dan di sekitar sistem yang berupa pesan atau instruksi yang berurutan.

Sequence diagram umumnya digunakan untuk menggambarkan suatu skenario atau urutan langkah-langkah yang dilakukan baik oleh *actor* maupun sistem yang merupakan respon dari sebuah kejadian untuk mendapatkan hasil atau *output*.

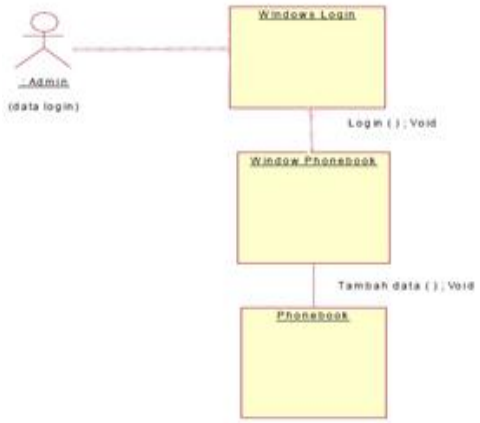


Gambar 2.11 Contoh Sequance Diagram

2.11.4 *Collaboration* Diagram

Sebuah *collaboration* diagram menunjukkan kolaborasi yang dinamis yang mirip dengan *sequence* diagram. *Collaboration* diagram digambarkan sebagai

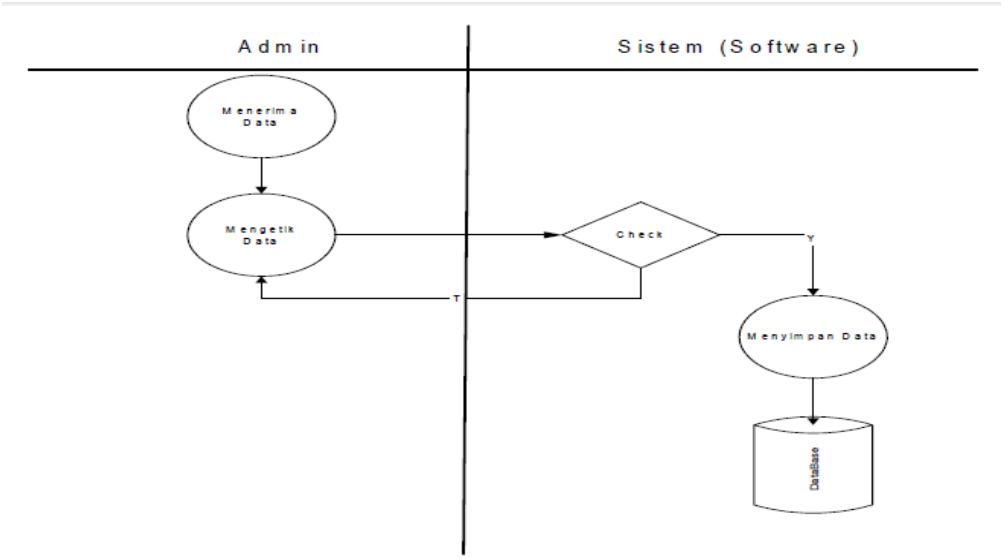
sebuah *object* diagram dimana sejumlah obyek ditunjukkan disekitarnya dengan hubungan-hubungannya.



Gambar 2.12 Contoh Collaboration Diagram

2.11.5 *Activity* Diagram

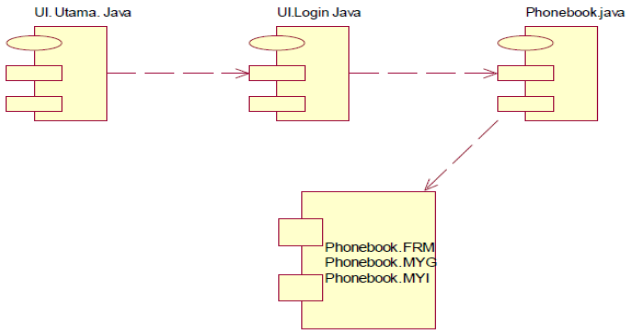
Sebuah *Activity* Diagram menunjukkan suatu alur kegiatan secara berurutan. *Activity* Diagram digunakan untuk mendiskripsikan kegiatan-kegiatan dalam sebuah operasi meskipun juga dapat digunakan untuk mendeskripsikan alur kegiatan yang lainnya seperti *use case* atau suatu interaksi.



Gambar 2.13 Contoh Activity Diagram

2.11.6 *Component* Diagram

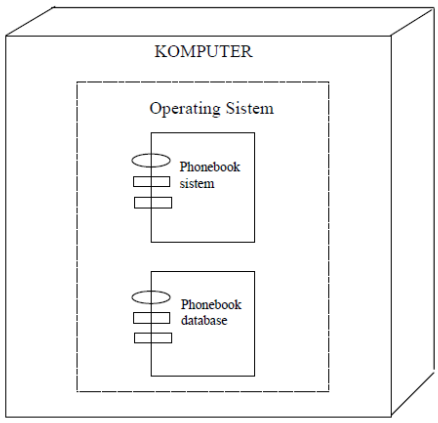
Component Diagram menunjukkan struktur dan hubungan antar komponen *software* termasuk ketergantungan (*dependency*) diantara komponen-komponen tersebut. Komponen pada piranti lunak adalah berupa modul-modul yang berisikan *code*, baik *library* maupun *executable*. Umumnya komponen yang terbentuk dari beberapa class dan/atau *package*, atau juga dapat dari komponen-komponen yang lebih kecil.



Gambar 2.14 Component Diagram

2.11.7 *Deployment* Diagram

Deployment Diagram menunjukkan arsitektur fisik pada *hardware* dan *software* pada suatu sistem yang dirancang. *Deployment* diagram juga dapat menunjukkan perngkat-perangkat dan *nodes* diantara hubungan yang dimilikinya antar komponen.



Gambar 2.15 Contoh Deployment Diagram

2.12 Tool Pedukung Pembuatan Aplikasi Android

Adapun tool pendukung dalam pembangunan aplikasi pengecekan umur kulit secara otomastis yaitu:

2.12.1 Java Development Kit (JDK)

JDK adalah *software development kit* merupakan seperangkat kode, *library*, form dan paket yang berguna untuk membuat aplikasi atau *applet java*. Di dalam JDK terdapat *Java Runtime Environment* (JRE) yang akan memproses *byte code* yang dihasilkan dari interpenter *java*. Tugas dari *interpenter java* ini adalah untuk mengcompile *source.java* menjadi *source.class* yang selanjutnya *source.class* inilah yang akan dijalankan oleh JRE.

2.12.2 Software Development Kit (SDK)

SDK adalah suatu kumpulan dari *library* dan *toolkit* seperti emulator untuk mengembangkan atau menciptakan aplikasi untuk suatu perangkat lunak, perangkat *computer*, *system* operasi atau *platform*. Di dalam SDK terdapat *tools* yang dibutuhkan dalam pengembangan android, diantaranya yaitu *adb shell* yang merupakan singkatan dari android *development bridge* yang dapat menjalankan terminal android seperti terminal pada *system* operasi linux, dan *command* yang terdapat dalam *adb shell* sendiri seperti *command linux* pada umumnya.

2.12.3 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*: Target sistem operasi *Eclipse* adalah *Microsoft Windows*, *Linux*, *Solaris*, *AIX*, *HP-UX* dan *Mac OS X*.
b. *Mulit-language*: *Eclipse* dikembangkan dengan bahasa pemrograman *Java*, akan tetapi *Eclipse* mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti *C/C++*, *Cobol*, *Python*, *Perl*, *PHP*, dan lain sebagainya.

c. *Multi-role*: Selain sebagai IDE untuk pengembangan aplikasi, *Eclipse* pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembangan web, dan lain sebagainya.

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*. *Eclipse* dibuat dari kerja sama antara perusahaan-perusahaan anggota '*Eclipse Foundation*' (beserta individu-individu lain). Banyak nama besar yang ikut dalam '*Eclipse Foundation*', termasuk *IBM*, *BEA*, *Intel*, *Nokia*, *Borland*. *Eclipse* bersaing langsung dengan Netbeans IDE. *Plugin* tambahan pada *Eclipse* jauh lebih banyak dan bervariasi dibandingkan IDE lainnya.

1. IntelliJ IDEA (commercial, free 30-day trial)
2. Oracle JDeveloper (free)
3. Xinox JCreator (ada versi berbayar maupun free)

Eclipse awalnya dikembangkan oleh *IBM* untuk menggantikan perangkat lunak *IBM Visual Age for Java* 4.0. Produk ini diluncurkan oleh *IBM* pada tanggal 5 November 2001, yang menginvestasikan sebanyak US$ 40 juta untuk pengembangannya. Semenjak itu konsursium *Eclipse Foundation* mengambil alih untuk pengembangan *Eclipse* lebih lanjut dan pengaturan organisasinya. 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:

1. *Core platform*
2. *OSGi*
3. *SWT (Standard Widget Toolkit)*
4. *Jface*
5. *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. *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 *Eclipse*, *plug-in* UML2 tersedia untuk membuat Diagram UML. Dengan menggunakan PDE setiap orang bisa membuat *plugin* sesuai dengan keinginannya. Salah satu situs yang menawarkan *plug-in* secara gratis seperti *Eclipse downloads by project*.

Sejak tahun 2006, *Eclipse Foundation* mengkoordinasikan peluncuran *Eclipse* secara rutin dan simultan yang dikenal dengan nama *Simultaneous Release*. Setiap versi peluncuran terdiri dari *Eclipse Platform* dan juga sejumlah proyek yang terlibat dalam proyek *Eclipse*. Tujuan dari sistem ini adalah untuk menyediakan distribusi *Eclipse* dengan fitur-fitur dan versi yang terstandarisasi. Hal ini juga dimaksudkan untuk mempermudah *Deployment* dan *maintenance* untuk sistem *enterprise*, serta untuk kenyamanan. Peluncuran simultan dijadwalkan pada bulan Juni setiap tahunnya.