BAB 2
LANDASAN TEORI

2.1 Citra (*Image)*

Menurut Oxford *Dictionaries* (oxforddictionaries.com), citra atau image adalah sebuah representasi dari bentuk eksternal dari seseorang atau sesuatu, sesuatu yang dapat dihasilkan dari kamera, teleskop, mikroskop, atau perangkat lain, yang dapat ditampilkan pada layar komputer atau video, atau penampilan optik yang dihasilkan oleh cahaya dari suatu objek tercermin dalam cermin atau yang dibiaskan melalui lensa [6].

2.1.1 Pengolahan Citra (*Image Processing*)

Pengolahan citra digital adalah metode yang digunakan untuk mengolah citra pada komputer sehingga sesuai dengan citra yang dibutuhkan. Secara umum, operasi pengolahan citra dapat diklasifikasikan dalam beberapa jenis sebagai berikut [7] :

1. *Image enhancement*

Operasi ini bertujuan untuk memperbaiki kualitas citra. Dengan operasi ini, ciri khusus pada citra dapat lebih ditonjolkan.

2. *Image restoration*

Operasi ini bertujuan untuk melakukan perbaikan pada citra dengan cara meminimumkan atau menghilangkan cacat pada citra.

3. *Image compression*

Operasi ini bertujuan untuk menghemat memori yang diperlukan untuk penyimpanan citra. Hal penting yang harus diperhatikan dalam kompresi citra ini adalah citra yang dikompresi harus tetap memiliki kualitas gambar yang baik.

4. *Image segmentation*

Operasi ini bertujuan untuk memecah suatu citra ke dalam beberapa segmen dengan suatu kriteria tertentu.

5. *Image analysis*

Operasi ini bertujuan untuk menghitung besaran kuantitif dari citra untuk menghasilkan deskripsinya. Teknik ini mengekstraksi ciri tertentu yang membantu mengidentifikasi objek.

6. *Image reconstruction*

Operasi ini bertujuan untuk membentuk ulang objek dari beberapa citra hasil proyeksi.

2.1.2 Konsep Dasar Warna

Warna-warna yang diterima indra penglihatan manusia merupakan hasil kombinasi cahaya dengan panjang gelombang berbeda. Kombinasi warna yang memberikan rentang warna yang paling lebar adalah *red* (R), *green* (G) dan *blue* (B). Ketiga warna tersebut dinamakan warna pokok dan sering disingkat dengan RGB. Setiap warna memiliki 3 buah atribut sebagai berikut [7] :

1. *Intensity/brightness/luminance*

Atribut yang menyatakan banyaknya cahaya yang diterima oleh mata tanpa mempedulikan warna. Kisaran nilainya antara gelap (hitam) dan terang (putih).

2. *Hue*

Atribut ini menyatakan warna sebenarnya, seperti merah, violet dan kuning. *Hue* digunakan untuk membedakan warna-warna dan menentukan kemerahan (*redness*), kehijauan (*greeness*) dan sebagainya dari cahaya.

3. *Saturation*

Atribut ini menyatakan tingkat kemurnian warna cahaya, yaitu mengindikasikan seberapa banyak warna putih yang diberikan pada warna. Sebagai contoh, warna merah adalah 100% warna jenuh (*saturated color*), sedangkan warna *pink* adalah warna merah dengan tingkat kejenuhan sangat rendah karena ada warna putih di dalamnya.

Dalam beberapa pemrosesan citra misalnya kompresi citra, penggunaan komponen warna RGB tidak disarankan karena keterkaitan warna pada komponen RGB sangat erat, sehingga jika dilakukan suatu perubahan terhadap salah satu komponennya akan mempengaruhi visual citra tersebut. Oleh karena itu pengkonversian RGB menjadi ruang warna (*color space*) yang lain biasa dilakukan.

Salah satu contoh ruang warna yang banyak digunakan adalah YcbCr atau *Luminance* (Y) dan *Chrominance blue and red* (Cb-Cr). Dalam model ini,

informasi *luminance* disimpan dalam komponen tunggal (Y) dan informasi *chrominance* disimpan dalam dua komponen warna (Cb dan Cr). Komponen *luminance* merepresentasikan terang gelapnya suatu citra dan komponen *chrominance* merepresentasikan corak warna dari citra tersebut.

Komponen Y atau *luminance* adalah komponen warna yang paling sensitif terhadap indra penglihatan manusia. Perubahan yang terjadi pada komponen *luminance* sangat mempengaruhi visual citra dibandingkan dengan komponen *chrominance*.

2.2 Teks

Teks adalah bahasa yang berfungsi, artinya adalah bahasa yang sedang melaksanakan tugas tertentu (menyampaikan pesan atau informasi) dalam konteks situasi, berlainan dengan kata-kata atau kalimat-kalimat lepas yang mungkin dituliskan di papan tulis. Bentuknya bisa percakapan dan tulisan (bentuk-bentuk yang digunakan untuk menyatakan apa saja yang dipikirkan). Sifat teks ialah meskipun teks itu bila dituliskan tampak seakan-akan terdiri dari kata-kata dan kalimat, namun sesungguhnya terdiri dari makna-makna. Makna-makna atau maksud yang ingin disampaikan kepada orang lain haruslah dikodekan dalam tuturan lisan atau kalimat-kalimat supaya dapat dikomunikasikan.

Teks merupakan produk dalam arti bahwa teks itu merupakan keluaran (output), sesuatu yang dapat direkam atau dipelajari (berwujud). Teks juga merupakan proses pemilihan makna yang terus-menerus. Ketika menerima atau memberi informasi dalam bentuk teks (lisan atau tulis) maka tentunya di dalam otak terjadi proses pemahaman (pemilihan makna) terhadap informasi tersebut, agar tidak terjadi kesalahpahaman. Adapun kriteria teks sebagai berikut:

Kriteria yang bersifat internal teks:

a. Kohesi: kesatuan makna
b. Koherensi: kepaduan kalimat (keterkaitan antarkalimat)
c. Kriteria yang bersifat eksternal teks:
d. Intertekstualitas: setiap teks saling berkaitan secara sinkronis atau diakronis.
e. Intensionalitas: cara-cara atau usaha-usaha untuk menyampaikan maksud atau pesan pembicaraan melalui sikap bicara, intonasi, dan

ekspresi wajah. Intensionalitas berkaitan dengan akseptabilitas (penerimaan informasi).

f. Informativitas: kuantitas dan kualitas informasi.
g. Situasionalitas: situasi tuturan.

2.3 Optical Character Recognition (OCR)

Optical Character Recognition (OCR) merupakan metode untuk mengubah teks pada *image* (citra) menjadi teks yang memungkinkan pengenalan karakter melalui mekanisme optik[1]. Optical Character Recognition (OCR) merupakan proses konversi bahan cetak ke dalam teks atau pengolah kata file yang dapat dengan mudah diedit dan disimpan. Teknologi OCR telah membuat terobosan besar pada cara penyimpanan informasi, berbagi dan editing. Dengan adanya OCR, citra yang dengan tulisan mesin ketik atau komputer dapat dimanipulasi. Text yang di-*scan* dengan OCR dapat dicari kata per kata atau per kalimat. Dan setiap text dapat dimanipulasi, diganti, atau diberikan barcode. Sejarah OCR dari masa ke masa yakni :

a. 1929 – Mesin Pengenal Digit (Digit recognition machine)
b. 1953 – Mesin Pengenal Alphanumerik (Alphanumeric recognition machine)
c. 1965 – US Mail sorting
d. 1965 – Sistem Perbankan Inggris (British)
e. 1976 – Mesin Pembaca Kurzweil
f. 1985 – Perangkat lunak PC Hardware-assisted
g. 1988 – Perangkat lunak PC Software-only
h. 1994-2000 – Industry consolidation

File citra yang akan dibaca di kirim ke komputer atau perangkat dengan OCR dan diproses menjadi teks. Berikut ini merupakan cara kerja OCR:

1. Meratakan gambar (*Auto Deskewing*)

Citra akan diatur kemiringannya, apabila hasil *scan* ada kemiringan, maka gambar akan dibuat sejajar atau lurus.

2. Menganalisa (*Analysis*)

OCR akan menganalisa dan memisahkan bagian teks dan bagian gambar.

3. Otomatis mengatur arah gambar (*Auto Orientation*)

OCR akan mengambil sebagian area pada citra dan mengidentifikasi arah teks yang benar. Citra akan diputar ke arah yang benar,pilihannya adalah 90^0, 180^0 atau 270^0.

4. Memisahkan setiap karakter huruf dan angka (*Separating single character*)

OCR akan memisahkan setiap karakter yang terdapat pada citra menjadi sebuah huruf atau angka.

5. Mengidentifikasi gambar (*Capturing the features of the characters and comparing*)

OCR akan mengidentifikasi satu persatu setiap gambar yang sudah dipisahkan, dan melakukan pengecekan terhadap database yang dimiliki oleh OCR, dan menetapkan huruf atau angka yang akan digunakan.

6. Menghasilkan file akhir (Recognition result output)

Setelah setiap citra sudah dirubah menjadi huruf atau angka (dalam format teks) maka OCR akan menghasilkan sebuah file dengan format teks, seperti Microsoft Word, Microsoft Excel, TXT, Rich Text dan Searchable PDF, tergantung dari pengaturan awal.

Metode-metode yang ada pada teknologi OCR yaitu, *matrix matching* dan *feature extraction*.

1. *Matrix Matching*

Metode pertama dan yang paling simpel adalah *matrix matching*. Lebih dikenal pula sebagai *pattern matching*, karena *scanner* akan mendeteksi karakter yang dipindai lalu menyamakan dengan database dari pola atau outline dari setiap karakter yang dianalisa dari bentuk citra.

Maka ketika *scanner* memindai sebuah karakter, sistem akan mendeteksi bentuk karakter tersebut dan menyamakan bentuk, sudut, dan pola karakter dari citra dengan yang ada di database. Jika karakter cocok maka karakter yang dipindai tersebut akan diidentifikasi sebagai karakter yang sama dalam database.

Hanya saja teknik ini memiliki keterbatasan, karena database ini memiliki keterbatasan akan jumlah variasi bentuk karakter. *Font* saat ini pun beraneka

ragam, pola dan bentuknya semakin kompleks. Metode ini merupakan metode yang paling banyak digunakan pada OCR saat ini.

2. *Feature Extraction*

Dikenal juga sebagai ICR atau *Intelligent Character Recognition* atau *topological feature analysis*. Ini adalah metode pengenalan karakter yang tidak tergantung dari template karakter yang dimiliki sistem.

Metode ini akan mendeteksi identitas sebuah karakter dari ruang kosong, bentuk yang berdekatan, garis diagonal, perpotongan dan sebagainya. Maka proses pengidentifikasiannya akan lebih kompleks.

Feature Extraction biasanya digunakan untuk mendeteksi karakter yang bentuknya tak standar, seperti tulisan tangan atau huruf-huruf ber-ornamen. *Matrix Matching* lebih cocok untuk dokumen yang menggunakan jenis huruf yang umum, seperti Times New Romans, Arial, atau *font* yang memiliki bentuk dasar karakter huruf lainnya. Database karakter pada *Feature Extraction* biasanya meliputi rangkaian dasar karakter angka dari 0-9 dan huruf a-z.

2.4 Leptonica

Leptonica [4] merupakan *library open source* yang dapat digunakan sebagai pengolah citra dengan merubah atau mengolah citra ke dalam bentuk *binary image* ataupun mengatur tingkat kontras, kecerahan dan *perspective* citra. Leptonica berisi perangkat lunak yang secara luas berguna untuk aplikasi pengolahan citra dan analisis citra.

Leptonica berasal dari bahasa Italia yang berarti "leptonic", yang merupakan kata sifat yang mengacu pada tiga *family* partikel fundamental yang sangat mirip, yang disebut *lepton*, yang lebih kita kenal dengan elektron yang terdiri dari neutrino.

Nama *lepton* berasal dari bahasa Yunani, yang berarti "cahaya" partikel. (Hal ini berbeda dengan partikel "berat", seperti proton.) Nama itu dipilih di pada tahun '40an oleh Abraham Pais, seorang ahli fisika partikel yang, selama 20 tahun terakhir telah menulis beberapa buku yang tentang sejarah fisika di abad ke-20, termasuk biografi Albert Einstein dan Niels Bohr.

Leptonica memiliki jumlah struktur data yang sangat kecil, dan jumlah operasi yang relatif besar. Leptonica menggunakan seluruh pendekatan

berorientasi objek. Struktur datanya melalui siklus hidup dimana mereka diciptakan, bertindak, dan dihancurkan. Implementasi yang dilakukan melalui fungsi yang biasanya namanya dimulai dengan nama struktur data primer yang terlibat. Developer berusaha untuk mengikuti seperangkat prinsip-prinsip desain untuk membuat kode yang aman, portabel dan transparan untuk digunakan sebaik mungkin (dengan C). Untuk menulis aplikasi sendiri, kita harus melihat ke dalam direktori program, yang ditulis untuk menguji fungsi *library*.

Fungsi yang ada dibagi menjadi dua *library*: satu level tingkat tinggi yang menggunakan serangkaian set struktur data seperti Pix, Box, dll, dan *library* level rendah yang hanya menggunakan tipe data C intrinsik. Ketika membangun *library*, kita juga membuat sebuah *library* tunggal yang menggabungkan keduanya. Nama file *library* level rendah terurut dibawah nama file *library* level tinggi. Nama fungsi juga mengikuti pola: nama fungsi level rendah yang dihasilkan dari fungsi level tinggi dengan menghapus awalan objek dan menambahkan "low". Sebagai contoh, pixScaleToGray4() panggilan scaleToGray4Low() untuk menjalankan eksekusi. Ketika struktur data yang dikumpulkan ke dalam sebuah array, nama struct ditambahkan huruf "a" diakhir nama filenya. Sebagai contoh, struct Pixa berisi array struct Pix.

Fungsi level atas adalah pixRead() dan pixWrite(). Encoders spesifik yang didukung oleh fungsi dalam file dinamakan dengan akhiran "io.c". Fungsi-fungsi ini, yang sebagian besar menggunakan aliran untuk I/O, merupakan hubungan antara struktur data citra, Pix dan kode level rendah untuk membaca dan menulis data citra. Ada juga fungsi-fungsi pada level tinggi yang khusus. Contohnya, tiffio.c memiliki fungsi untuk membaca dan menulis file yang berisi beberapa citra dan menulis file tiff khusus Tag yang tertanam dalam header. Juga, jpegio.c memiliki fungsi untuk membaca file jpeg, pilihan untuk mengkonversi RGB ke 8 bpp dengan colormap, mengurangi sebanyak dua tingkat, dan memberi peringatan jika data yang terkompresi rusak.

Selain itu, kita dapat menulis file PostScript dalam berbagai format, baik tingkat 1 (tak terkompresi) dan tingkat 2 (menggunakan DCT alias JPEG) dan kompresi CCITT-G4 dengan pilihan untuk menggambar melalui mask biner. Ini

diimplementaskan di psio.c, dimana akan ditemukan sejumlah fungsi yang mendukung untuk kedua format kompresi ini.

Operasi yang dapat dilakukan dengan leptonica yakni :

a. Rasterop (bitmap),
b. Transformasi affine (*scaling*, translasi, rotasi, geser) pada gambar dengan kedalaman pixel tertentu,
c. Biner dan *grayscale* morfologi,
d. Seedfill dan komponen yang terhubung
e. Transformasi citra dengan menggabungkan perubahan skala dan kedalaman pixel,
f. *Pixelwise masking, blending, enhancement*, dan aritmatika

1. Rasterop

   Rasterop adalah satu fungsi pengolahan citra yang paling berguna. Melakukan *arbitrary logical operation* antara dua *rectangle* citra sumber dan hasil, menyimpan hasilnya dalam citra hasil. Seperti mengolah komposisi citra, layar, biner morfologi, dll. Ia bekerja pada kedalaman citra. Antarmuka tingkat tinggi untuk rasterops diimplementasikan di rop.c. Fungsi umum dari pixRasterop() yakni terdiri dari 9 argumen: 5 untuk citra hasil dan *rectangle*, 3 untuk citra sumber dan sudut *rectangle* untuk digunakan, dan 1 untuk menentukan operasi yang sebenarnya. Terjemahan yang sederhana dapat dilakukan baik antara citra sumber dan hasil menggunakan fungsi umum, atau menggunakan pixRasteropIP().

2. Transformasi Affine

   Transformasi Affine adalah transformasi linear yang menghasilkan koordinat pada citra hasil (x', y') dari variabel sumber (x, y). Terdapat persamaan linear 3, dengan 6 koefisien. Set transformasi affine (rotasi, shear, penskalaan), dan set 6 koefisien tersebut setara dengan sebuah instance dari himpunan ini. Ini merupakan pekerjaan yang sulit untuk melakukan semua transformasi ini secara efisien pada citra dari berbagai kedalaman pixel, termasuk RGB.

3. Morfologi Biner

Ini adalah *source* untuk implementasi yang efisien dari morfologi biner dan morfologi *grayscale*. Biner morfologi dilakukan dengan dua cara:

a. Rasterops citra utuh secara berturut-turut untuk penataan elemen *arbitrary* (Sels)
b. Mengakumulasi kata tujuan (dwa) untuk Sels tertentu. Kode ini dihasilkan secara otomatis. Misalnya, kode dalam fmorphgen.1.c dan fmorphgenlow.1.c. File-file ini dihasilkan dengan menjalankan program prog/fmorphautogen.c. Hasil dapat diperiksa dengan membandingkan dwa dan rasterops citra utuh. Misalnya, prog/fmorphauto_reg.c.

Metode (b) jauh lebih cepat daripada metode (a), ini juga merupakan alasan upaya untuk mendukung penggunaan metode ini untuk semua Sels.

Ketika prog/fhmtautogen.c dikompilasi dan dijalankan, akan menghasilkan kode dwa C yakni fhmtgen.1.c dan fhmtgenlow.1.c. File-file ini kemudian dapat dikompilasi ke *library* atau ke program lain. Hasil dapat diperiksa dengan membandingkan hasil dwa dan rasterop. Misalnya, prog/fhmtauto_reg.c. Beberapa fungsi dengan parser sederhana disediakan untuk melakukan serangkaian operasi morfologi (ditambah biner dan ekspansi replikasi).

4. Penskalaan Citra

Penskalaan citra pada leptonica disediakan banyak implementasi sederhana dan relatif efisien. Warna dan grayscale gambar skala dengan menggunakan :

a. Sampling
b. Penyaringan lowpass diikuti oleh sampling
c. Pemetaan daerah
d. Interpolasi linear

Skala operasi dengan sampling antialias, pemetaan daerah dan interpolasi linier terbatas pada 2, 4 dan 8 bpp *gray*, 24 bpp warna RGB *full color*, dan 2, 4 dan 8 bpp *colormapped* (bpp == bit piksel). Operasi penskalaan sederhana dengan pengambilan sampel dapat dilakukan di bpp 1, 2, 4, 8, 16 dan 32. Interpolasi linear lebih lambat tetapi memberikan hasil yang

lebih baik, terutama untuk *upsampling*. Untuk *downsampling*, hasil terbaik yang diperoleh dengan pemetaan daerah skala. Pilihan *sharpening* setelah *resampling* disediakan untuk memperbaiki penampilan dengan mengurangi efek visual yang rata-rata melewati batas-batas ketajaman citra.

Untuk analisis cepat dari citra warna dan *grayscale*, sangat penting untuk memiliki integer *subsampling* dikombinasikan dengan pengurangan kedalaman pixel. Sehingga citra warna RGB dapat dikonversi ke citra biner dan *grayscale* dengan resolusi rendah.

5. Rotasi dan Geser Citra

   Menggeser citra dilakukan dengan rasterop dan interpolasi linier. Fungsi rasterop lebih cepat dan tidak mempunyai kendala pada kedalaman gambar. Terdapat fungsi untuk menggeser citra dengan titik-titik horisontal dan vertikal. Interpolasinya dapat digunakan pada citra 8 bpp dan 32 bpp untuk memberikan hasil yang halus.

   Ada tiga jenis rotator citra secara umum:

   1) Rotasi *grayscale* menggunakan pemetaan daerah
      a. pixRotateAM() untuk 8 bit citra abu-abu dan 24 bit
      b. pixRotateAMCorner() untuk 8 bit citra abu-abu
      c. pixRotateAMColorFast() untuk 24 bit berwarna yang lebih cepat
   2) Rotasi citra dengan *arbitrary bit depth*, menggunakan 2 atau 3 *shear*. Rotasi tersebut dapat dilakukan dengan titik *arbitrary*. Contoh : pixRotateShear(), pixRotateShearIP().
   3) Rotasi dengan sampling. Ini dapat digunakan pada citra *arbitrary depth*, dan dilakukan dengan titik *arbitrary*. *Colormaps* tetap dipertahankan. Rotasi geser jauh lebih cepat, dan bekerja pada citra piksel *arbitrary depth*.

6. Algoritma *Sequential*

   Developer menyediakan sejumlah algoritma *sequential* cepat, termasuk biner dan *grayscale seedfill*, dan fungsi jarak untuk citra biner. Biner yang paling efisien adalah pixSeedfill(), yang menggunakan algoritma Vincent

untuk perintah iterasi *raster* dan *antiraster*, dan dapat digunakan untuk mengisi baik 4 atau 8 *fill* yang terhubung. Serupa dengan algoritma *sequential raster*/*antiraster* yang biasanya digunakan untuk menghasilkan *distance map* dari citra biner, dan untuk *grayscale seedfill*. Developer juga menggunakan algoritma *stack-based filling* milik Heckbert untuk mengidentifikasi 4 dan 8 komponen terhubung dalam citra biner. Implementasi yang cepat dari *watershed transform*, menggunakan *priority queue*, juga disertakan.

7. *Image Enhancement*

   Beberapa implementasi *image enhancement* sederhana untuk citra warna dan grayscale telah disediakan. Termasuk *intensity mapping* dengan koreksi gamma dan *contrast enhancement, edge sharpening, smoothing, hue* dan modifikasi saturasi.

8. *Convolution* dan *Cousins*

   Sejumlah operasi pengolahan citra standar juga dimasukkan, seperti *block convolution, binary block rank filtering, rank order filtering grayscale* dan RGB, serta *edge* dan *local minimum/maximum extraction*. *Generic convolution* disertakan untuk kernel yang dapat dipisah maupun kernel yang tidak dapat dipisah, menggunakan float array di Pix.c.

9. Citra I/O

   Beberapa fasilitas telah disediakan untuk citra input dan output. Hal ini tentu saja diperlukan untuk membangun *executable* yang menangani citra. Fungsi telah disediakan untuk memungkinkan membaca dan menulis file dalam JPEG, PNG, TIFF, BMP, PNM GIF dan WEBP format. Format ini dipilih untuk alasan berikut:

   a. JFIF JPEG adalah metode standar untuk kompresi *lossy* citra warna dan *grayscale*. Format ini didukung secara *native* di semua browser, dan menggunakan kompresi *open source library* yang baik.
   b. PNG adalah metode standar untuk kompresi biner *lossless*, citra warna dan *grayscale*. Format ini didukung secara *native* di semua browser, dan menggunakan kompresi *open source library* yang

baik (zlib). Unggul dalam hampir setiap hal dibandingkan GIF (yang, sampai saat ini, terdapat hak paten kompresi LZW).

c. TIFF adalah format *interchange* umum, yang mendukung kedalaman piksel yang berbeda, *colormaps*, dll, dan juga memiliki format kompresi biner yang relatif baik dan digunakan secara luas (CCITT Group 4). TIFF juga mendukung multipage citra melalui struktur direktori.
d. BMP (sampai saat ini) belum memiliki kompresi. Format ini adalah format sederhana dengan *colormaps* yang tidak memerlukan adanya *library* eksternal. Format ini umum digunakan karena merupakan standar dari Microsoft.
e. PNM adalah format yang sangat sederhana yang masih digunakan secara luas di komunitas pengolahan citra. Tidak mendukung kompresi atau *colormaps*, tetapi mendukung citra biner, *grayscale* dan RGB. Seperti halnya BMP, implementasinya sederhana dan tidak memerlukan adanya *library* eksternal.
f. GIF masih banyak digunakan di dunia. Dengan berakhirnya paten LZW, sangat baik menambahkan dukungan untuk file GIF. *Library open source* GIF relatif tidak lengkap dan tidak didukung (karena sejarah paten Sperry-Rand-Burroughs-Univac).
g. WEBP adalah metode baru penyandian *wavelent* berasal dari libvpx, sebuah perpustakaan kompresi video. Leptonica menyediakan antarmuka yang mendasari webp ke codec. Namun perlu men-download libvpx, libwebp dan yasm sebagai pendukung.

Berikut adalah ringkasan dari dukungan kompresi dan keterbatasannya:

a) Semua format selain JPEG mendukung 1 bpp biner.
b) Mendukung semua format grayscale bpp 8 (GIF harus memiliki colormap).
c) Semua format kecuali GIF mendukung 24 bpp warna rgb.
d) Semua format kecuali PNM mendukung 8 bpp colormap.
e) PNG dan PNM dukungan 2 dan 4 bpp citra.

f) PNG mendukung 2 dan 4 bpp colormap, dan 16 bpp tanpa colormap.
g) PNG, JPEG, TIFF dan GIF dukungan gambar kompresi; PNM dan BMP tidak.
h) WEBP mendukung hanya 24 bpp warna rgb.

Menggunakan prog ioformats_reg untuk tes regresi pada semua kecuali GIF dan WEBP. Gunakan prog gifio_reg untuk pengujian GIF.

Versi terbaru dari Leptonica *library* telah dirilis beberapa kali dalam setahun, dan nomer versi yang disediakan untuk setiap perilisan terdapat pada file pembangun dan pada file allheaders.h. Semua versi termasuk versi 1.42 sampai versi 1.60 telah diarsipkan di http://code.google.com/p/leptonica, hingga versi terbaru setelah versi 1.60. Bagaimanapun, code.google.com tidak lagi mensupport upload distribusi terbaru, yang bisa didapatkan di website leptonica.org.

Jumlah download Leptonica meningkat dari versi awal 1.69 sampai saat ini, semenjak adanya bundling dengan Tesseract dan kerjasama di Ubuntu 12-04. Terdapat 2046 daftar kategori dari semua fungsi yang ada di Leptonica yang disusun dan dapat dicari di halaman website http://tpgit.github.io/UnOfficialLeptDocs/leptonica/functions.html. Ada juga ringkasan file di direktori src dan prog dengan deskripsi singkat dari setiap file. Pastikan versi yang digunakan sesuai dengan yang dibutuhkan untuk *package* debian. Versi-versi Leptonica yang telah dirilis yakni :

a. 1.69 : 3.0.0
b. 1.70 : 4.0.0
c. 1.71 : 4.2.0
d. 1.72 : 4.3.0
e. 1.73 : 4.4.0

Cara menggunakan Leptonica sebagai *source* di Android Studio yakni :

1. Unduh *open source* Leptonica di situs Google Code http://code.google.com/p/leptonica.
2. Buat direktori *library* pada Android Studio
3. Salin direktori file yang telah didownload kedalam direktori *library*.

4. Edit setting.gradle dengan meng-*add library* ke include Gunakan *library* pada file class Java dengan meng-*import library* dan menginisialisasi method yang digunakan. Contoh hasil *add* pada setting.gradle:

```
include ':app', ':Leptonica'
project(':Leptonica').projectDir = new File('libs/Leptonica')
```

5. Jika mengalami error pada *Default Configuration*, ganti coding pada tahap 4 dengan :

```
include ':app'
include ':libs:Leptonica'
```

6. Pada folder app/build.gradle, tambahkan proyek *library* sebagai *dependency* :

```
dependencies {
   compile fileTree(dir: 'libs', include: ['*.jar'])
   compile 'com.android.support:appcompat-v7:21.0.3'
   compile project(":Leptonica")
     //compile project(":libs:Leptonica") jika melalui tahap 5

}
```

7. Kemudian tambahkan global config pada gradle.properties untuk menampung semua versi SDK dan versi *build tools* untuk menjaga konsistensi setiap module. Edit gradle.properties,tambahkan :

```
ANDROID_BUILD_MIN_SDK_VERSION=14
ANDROID_BUILD_TARGET_SDK_VERSION=21
ANDROID_BUILD_TOOLS_VERSION=21.1.3
ANDROID_BUILD_SDK_VERSION=21
```

8. Klik sync project dengan icon gradle. Maka library akan muncul dan siap digunakan dalam proyek.

Leptonica memiliki lebih dari 2000 fungsi, berikut ini merupakan daftar fungsi yang ada berdasarkan kategori :

Tabel 2.1 Daftar Fungsi Berdasarkan Kategori

Kategori	Jumlah Fungsi
Auto-gen'd Destination Word Accumulation (DWA) Morphology Code	9
Basic Box, Boxa and Boxaa Functions	104
Basic image operations	102
Basic Pix Array Functions	90
Basic Pix Functions	192
Colormap Utilities and Related Operations	83

Connected Components in Binary Images	47
Float-Valued Images	76
Formatted I/O	175
Fundamental Data Structures for Computation	350
Image Morphology	135
Image Operations with Filling	32
Image Quantization/Depth Conversion	145
Image Scaling	83
Line Graphics and Special Output	45
Low-Level Pixel Access	21
Maze	4
Miscellaneous Utilities	104
Other Geometric Image Transforms	104
Specialized Document Image Processing	117
Specialized Image Filters	28
21 Kategori	2046

Terdapat total 2046 fungsi yang masing-masing penjelasan tiap fungsinya dapat dilihat pada situs:
http://tpgit.github.io/UnOfficialLeptDocs/leptonica/functions.html

Library Leptonica yang telah di *wrap* ke dalam bahasa Java (dengan bantuan library tess-two untuk representasi Java dari native Leptonica object), terdiri dari 17 class yakni :

1. AdaptiveMap.java
2. Binarize.java
3. Box.java
4. Boxa.java
5. Clip.java
6. Constans.java
7. Convert.java
8. Enhance.java
9. JpegIO.java
10. Pix.java
11. Pixa.java
12. Projective.java
13. ReadFile.java

14. Rotate.java
15. Scale.java
16. Skew.java

Class yang digunakan pada proyek ini yakni :

1. Pix.java

Merupakan representasi Java dari native Leptonica PIX. Dapat digunakan untuk mendeteksi blur, menampilkan citra, mengolah penskalaan citra dan gabungan dari fungsi – fungsi yang ada di object class PIX pada Leptonica. Berikut ini tabel fungsi dari tiap class object PIX :

Nama Class	Kategori	Fungsi
pix1.c	Basic Pix Functions	Implementasi Pix (citra) class – fundamental
pix2.c	Basic Pix Functions	Pixel poking pada citra, pad dan border pixels
pix3.c	Basic Pix Functions	Masking dan operasi logical, counting, mirrored tiling
pix4.c	Basic Pix Functions	Histogram, estimasi fg/bg, rectangle extraction
pix5.c	Basic Pix Functions	Property measurement untuk citra 1bbp, rectangular region extraction, property measurement along lines, crop untuk foreground

2. Pixa.java

Merupakan representasi Java dari native Leptonica PIXa. Berikut ini tabel fungsi dari tiap class object PIXa :

Nama Class	Kategori	Fungsi
pixacc.c	Basic image operations	Antarmuka sederhana untuk *grayscale image arithmetic*
pixabasic.c	Basic Pix Array Functions	Implementasi class Pixa (array dari Pix) dan Pixaa (2d array dari Pix)
pixafunc1.c	Basic Pix Array Functions	Binary Pixa (array of Pix) filtering, sorting, clipping.
pixafunc2.c	Basic Pix Array Functions	Rendering Pixa atau Pixaa pada Pix

3. Box.java

Merupakan representasi Java dari native Leptonica BOX. Berikut ini tabel fungsi dari tiap class object BOX :

Nama Class	Kategori	Fungsi
boxbasic.c	Basic Box, Boxa and Boxaa Functions	Implementasi dari class Box, Boxa (Box array), dan Boxaa
boxfunc1.c	Basic Box, Boxa and Boxaa Functions	Box intersection, union, containment. Boxa union, extent, permutation
boxfunc2.c	Basic Box, Boxa and Boxaa Functions	Boxa/Box transforms. Boxa/Boxaa sorting. Boxaa utilities
boxfunc3.c	Basic Box, Boxa and Boxaa Functions	Mask <-> Boxa; painting into and drawing Boxa

4. ReadFile.java

Merupakan representasi Java dari native Leptonica readfile. Berikut ini tabel fungsi dari tiap class object readfile :

Nama Class	Kategori	Fungsi
readfile.c	Formatted I/O	Pix dari memori dan disk; mendeteksi format file

5. WriteFile.java

Merupakan representasi Java dari native Leptonica writefile. Berikut ini tabel fungsi dari tiap class object writefile :

Nama Class	Kategori	Fungsi
writefile.c	Formatted I/O	Pix dari memori dan disk; memilih format file

6. Rotate.java

Merupakan representasi Java dari native Leptonica rotate. Berikut ini tabel fungsi dari tiap class object rotate :

Nama Class	Kategori	Fungsi
rotate.c	Other Geometric Image Transforms	Top-level image rotation dengan interpolasi dan sampling

rotateam.c	Other Geometric Image Transforms	Interpolated image rotation
rotateamlow.c	Other Geometric Image Transforms	Low-level helpers untuk interpolated rotation
rotateorth.c	Other Geometric Image Transforms	Orthogonal image rotation and flips
rotateorthlow.c	Other Geometric Image Transforms	Low-level helpers untuk orthogonal rotation
rotateshear.c	Other Geometric Image Transforms	Rotasi citra dengan shear

7. Projective.java

Merupakan representasi Java dari native Leptonica projective. Berikut ini tabel fungsi dari tiap class object projective :

Nama Class	Kategori	Fungsi
projective.c	Other Geometric Image Transforms	4-pt projective transforms pada citra dan koordinat, dengan sampling dan interpolation

Berikut ini penjelasan setiap method yang ada pada *library* class yang digunakan dalam bahasa Java :

1. Box.java

Method	Parameter	Keterangan
Box()	long mNativeBox	Pointer untuk native Box object. Method ini digunakan dalam native code untuk membuat sebuah Box wrapper baru untuk specified native BOX
Box()	int x, int y, int w, int h	Parameter x X-koordinat pojok atas kiri dari box. Parameter y Y- koordinat pojok atas kiri dari box. Parameter w Width dari box. Parameter h Height dari box. Membuat sebuah box dengan geometri yang spesifik.
getCenter()		Mereturn geometri

getX()		Me-return x-koordinat box pada pixel.
getY()		Me-return y-koordinat box pada pixel.
getWidth()		Me-return width box pada pixel.
getHeight()		Me-return height box pada pixel.
getGeometry()		Me-return sebuah array yang berisi koordinat dari box.
getGeometry()	int[] geometry	Mengisi array berisi koordinat dari box
recycle()		Melepaskan resource dan membebaskan beberapa memori yang terasosiasi dengan Box.

2. Boxa.java

Method	Parameter	Keterangan
Boxa()	long nativeBoxa	Method ini digunakan dalam native code untuk membuat sebuah Box wrapper baru untuk specified native BOXa
getCount()		Me-return hasil perhitungan
getGeometry()	int index	Me-return sebuah array yang berisi koordinat dari box.
getGeometry()	int index, int[] geometry	Mengisi array berisi koordinat dari box
recycle()		Melepaskan resource dan membebaskan beberapa memori yang terasosiasi dengan Box.

3. Clip.java

Method	Parameter	Keterangan
clipRectangle ()	Pix source, Box box	Box tidak akan di-clipped jika berada di luar pix dimension.

4. Pix.java

Method	Parameter	Keterangan
Pix()	long nativePix	Method ini digunakan dalam native code untuk membuat sebuah Pix wrapper baru untuk specified native PIX
Pix()	int width, int height, int depth	INDEX_W adalah index width dari citra diantara dimension array. INDEX_H adalah index height dari citra diantara dimension array. INDEX_D adalah index bit-depth dari citra diantara dimension array.
getNativePix()		Me-return sebuah pointer native Pix object. Digunakan dalam native code.
getData()		Me-return raw bytes dari native PIX object.
getDimensions()		Me-return sebuah array dari dimensi citra.
getDimensions()	int[] dimensions	Mengisi sebuah array dengan dimensi citra.
clone()		Me-return sebuah clone dari Pix.
copy()		Me-return sebuah deep copy

		dari Pix yang dapat dimodifikasi tanpa mempengaruhi Pix asli.
invert()		Melakukan invert pada Pix in-place.
recycle()		Melepaskan resource dan membebaskan beberapa memori yang terasosiasi dengan Pix.
createDariPix()	byte[] pixData, int width, int height, int depth	Method ini digunakan dalam native code pada Pix dari data raw Pix yang didapat dari method getData().
getRect()		Me-return sebuah Rect dengan width dan height dari Pix.
getWidth()		Me-return width dari Pix.
getDepth()		Me-return depth dari Pix.
getRefCount()		Me-return hasil perhitungan rect
getPixel()	int x, int y	Me-return hasil perhitungan piksel
setPixel()	int x, int y, int color	Parameter x, koordinat x (0...width-1) dari pixel untuk di return. * Parameter y, koordinat y (0...height-1) dari pixel untuk di return.
nativeGetRefCount()	long nativePix	Parameter x, koordinat x (0...width-1) dari pixel ke set. * Parameter y, koordinat y (0...height-1) dari pixel ke set.

5. Pixa.java

Method	Parameter	Keterangan
createPixa()	int size	Sebuah pointer untuk native PIXA object. Digunakan dalam native code
createPixa()	int size, int width, int height	Sebuah pointer untuk native PIXA object. Digunakan dalam native code
Pixa()	long nativePixa, int width, int height	Sebuah pointer untuk native PIXA object. Digunakan dalam native code
getNativePixa()		Me-return pemanggilan native Pixa
copy()		Melakukan copy Pixa
sort()	int field, int order	Melakukan pengurutan field
size()		Me-return hasil perhitungan size
recycle()		Melakukan recycle
finalize()		Akan meng-attemp untuk melepaskan native memory dua kali.
join()	Pixa otherPixa	Melakukan penggabungan Pixa dengan Pixa lain
addPix()	Pix pix, int mode	Menambahkan Pixa
addBox()		Menambahkan box pada Pixa
add()	Pix pix, Box box, int mode	Menambahkan pix dan box pada Pixa
getBox()	int index	Me-return box
getPix()	int index	Me-return pix
getWidth()		Me-return width
getHeight()		Me-return height
getRect()		Me-return rect

getBoxRect()	int index	Me-return rect dari box
getBoxGeometry()	int index	Me-return geometri dari box
getBoxGeometry()	int index, int[] dimensions	Me-return geometri array dari box
getBoxRects()		Me-return rect array dari box
replacePix()	int index, Pix pix, Box box	Mengganti pix
mergeAndReplacePix()	int indexA, int indexB	Menggabungkan dan mengganti Pix
writeToFileRandomCmap( )	File file	Method ini digunakan dalam native code untuk membuat sebuah Pixa dengan kapasitas specified minimum.
iterator()		Membuat sebuah wrapper untuk specified native Pixa pointer.
PixIterator() hasNext() next() remove()	int mIndex;	Me-return sebuah pointer pada native PIXA object. Digunakan dalam native code.
nativeCreate()	int size	Membuat sebuah shallow copy dari Pixa. Berisi Pix yang di cloning, dan menghasilkan Pixa yang mungkin akan di recycle terpisah dengan original.
nativeCopy()	long nativePixa	Mengurutkan Pixa mengacu pada specified field dan urutan.
nativeSort()	long nativePixa, int field, int order	Me-return jumlah dari element pada Pixa.
nativeJoin()	long nativePixa, long otherPixa	Me-recycle Pixa dan

		membebaskan alokasi memori native.
nativeGetCount()	long nativePixa	Me-return hasil perhitungan native
nativeDestroy()	long nativePixa	Me-merge konten dari Pixa lain dengan Pixa ini
nativeAddPix()	long nativePixa, long nativePix, int mode	Menambahkan sebuah Pix ke dalam Pixa ini.
nativeAddBox()	long nativePixa, long nativeBox, int mode	Menambahkan sebuah Box pada Pixa.
nativeAdd()	long nativePixa, long nativePix, long nativeBox, int mode	Menambahkan sebuah Pix dan associated Box pada Pixa ini.
nativeWriteToFileRandomCmap()	long nativePixa, String fileName, int width, int height	Me-return Box pada specified index, atau null jika error.
nativeReplacePix()	long nativePixa, int index, long nativePix, long nativeBox	Me-return Pix pada specified index, atau null jika error.
nativeMergeAndReplacePix()	long nativePixa, int indexA, int indexB	Me-return width dari Pixa, atau 0 jika tidak di set saat di create.
nativeGetBox()	long nativePix, int index	Me-return height dari Pixa, atau 0 jika tidak di set saat di create.
nativeGetPix()	long nativePix, int index	Me-return sebuah bounding Rect pada Pixa, dan me-return (0,0,0,0) jika width dan height tidak di set saat di create.

6. Projective.java

Method	Parameter	Keterangan
projectiveTransform()	Pix pixs, float[] dest, float[] src	Input: pixs (32 bpp), ptad (4 pts of final koordinat

		space), ptas (4 pts of initial koordinat space), me-return: pixd, atau null jika error
nativeProjectivePtaColor()	long pix , float[] dest, float[] src	Me-return hasil dari native code

7. ReadFile.java

Method	Parameter	Keterangan
readMem()	byte[] encodedData, Context ctx	Membuat sebuah 32bpp Pix object dari encoded data. Hanya mendukung format BMP dan JPEG.
readBytes8()	byte[] pixelData, int width, int height	Membuat sebuah 8bpp Pix object dari raw 8bpp grayscale pixel.
replaceBytes8()	Pix pixs, byte[] pixelData, int width, int height	Mengganti bytes pada sebuah 8bpp Pix object dengan raw grayscale 8bpp pixel.
readFile()	File file	Me-return file dari pembacaan
readFile()	Context context, File file	Membuat sebuah Pix object dari encoded file data Hanya mendukung format BMP dan JPEG.
loadDenganPicasso()	Context context, File file	Me-return load konten dengan Picasso
loadDenganBitmapFactory()	File file	Me-return load konten dengan Bitmap
readBitmap()	Bitmap bmp	Membuat sebuah Pix object dari data Bitmap.

		Hanya mendukung ARGB_8888-formatted bitmap.

8. Rotate.java

Method	Parameter	Keterangan
rotate()	Pix pixs, float degrees	Menyajikan rotasi dengan default parameter.
rotate()	Pix pixs, float degrees, boolean quality	Menyajikan rotasi dengan resizing menggunakan default parameter.
rotate()	Pix pixs, float degrees, boolean quality, boolean resize	Menyajikan basic image rotation mengenai center.
rotateOrth()	Pix pixs, int quads	Me-return rotasi quads citra

8. WriteFile.java

Method	Parameter	Keterangan
writeBytes8()	Pix pixs	Menulis sebuah 8bpp Pix ke dalam flat byte array.
writeBytes8()	Pix pixs, byte[] data	Menulis sebuah 8bpp Pix ke dalam flat byte array. Me-return jumlah data byte yang ditulis pada data
writeFiles()	Pixa pixas, File path, String prefix, int format	Menulis seluruh citra ke dalam Pixa array ke dalam file individu menggunakan format spesifik.
writeMem()	Pix pixs, int format	Menulis sebuah Pix ke dalam byte array using menggunakan format spesifik encoding dari Constants.IFF
writeImpliedFormat()	Pix pixs, File file	Menulis sebuah Pix ke dalam file menggunakan file extension sebagai output format.

writeImpliedFormat()		Menulis sebuah Pix ke dalam file menggunakan file extension sebagai output format, mendukung formats.jpg atau .jpeg untuk JPEG dan.bmp untuk bitmap.
writeBitmap()	Pix pixs	Menulis sebuah Pix ke dalam Android Bitmap object. Bitmap keluaran selalu ARGB_8888, tetapi input Pixs dapat berupa bit-depth.

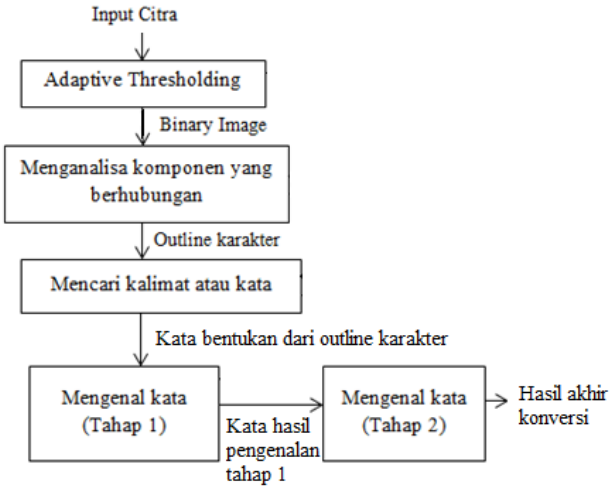
2.5 Tesseract Library

Tesseract [2] merupakan salah satu *library open source* yang dapat digunakan pada OCR sebagai *library* bahasa atau *language* agar OCR dapat mengenali karakter. Tesseract pada awalnya dikembangkan sebagai perangkat lunak berpemilik di Hewlett-Packard antara tahun 1985 hingga 1995. Pada saat ini, untuk pertama kalinya, detail arsitektur dan algoritma dipublikasikan.

Tesseract dimulai sebagai sebuah proyek penelitian seorang PhD di HP Labs, Bristol, dan mendapatkan momentum sebagai perangkat lunak dan *add-on* perangkat keras untuk lini HP *scanner flatbed*. Proyek ini termotivasi oleh fakta bahwa mesin OCR komersial pada saat itu berada di tahap awal, gagal total dan hanya mementingkan kualitas cetak terbaik. Setelah proyek bersama antara HP Labs Bristol, dan Divisi scanner HP di Colorado, Tesseract memimpin secara signifikan dalam akurasi yang lebih baik dibandingkan dengan mesin OCR komersial, tetapi Tesseract tetap tidak menjadi suatu produk.

Tahap selanjutnya dari perkembangannya adalah kembali ke HP Labs Bristol untuk investigasi OCR sebagai alat kompresi. Pengembangan lebih berkonsentrasi pada peningkatan efisiensi pada tingkat akurasi. Pada akhir proyek ini, pada akhir tahun 1994, pembangunan berhenti sepenuhnya. Mesin OCR dikirim ke UNLV untuk melakukan tes akurasi pada tahun 1995, di mana mesin tersebut membuktikan bahwa kualitas Tesseract lebih baik dibandingkan mesin OCR komersial. Setelah sepuluh tahun tanpa perkembangan apapun, Hewlett Packard dan UNLV merilis Tesseract sebagai *open source* pada tahun 2005.

Tesseract saat ini sedang dikembangkan oleh Google dan dirilis di bawah Lisensi Apache, Versi 2.0. Tesseract dianggap salah satu perangkat lunak mesin OCR bebas yang paling akurat yang tersedia saat ini. Berikut ini merupakan arsitektur dari mesin Tesseract OCR, yakni :



Gambar 2.1 Arsitektur Pada Tesseract OCR

1. *Adaptive Thresholding*

Proses ini digunakan untuk merubah citra warna yang menjadi citra masukan menjadi citra *grayscale* atau citra biner. Tujuannya adalah untuk memudahkan dalam menganalisa citra karena citra biner hanya memiliki dua bit yakni 0 dan 1.

2. Menganalisa komponen terhubung *(Connected Component Analysis)*

Proses ini digunakan untuk melakukan proses segmentasi teks yang ada pada citra menjadi karakter yang nantinya akan dikenali.

3. Mencari kalimat atau kata *(Find Lines and Words)*

Proses ini digunakan untuk mencari baris teks pada suatu citra.

4. Mengenal kata tahap 1 *(Recognize Word Pass 1)*

Proses ini digunakan untuk proses pengenalan tahap awal, yaitu membandingkan karakter yang didapat dari citra dengan data yang sudah ada.

5. Mengenal kata tahap 2 *(Recognize Word Pass 2)*

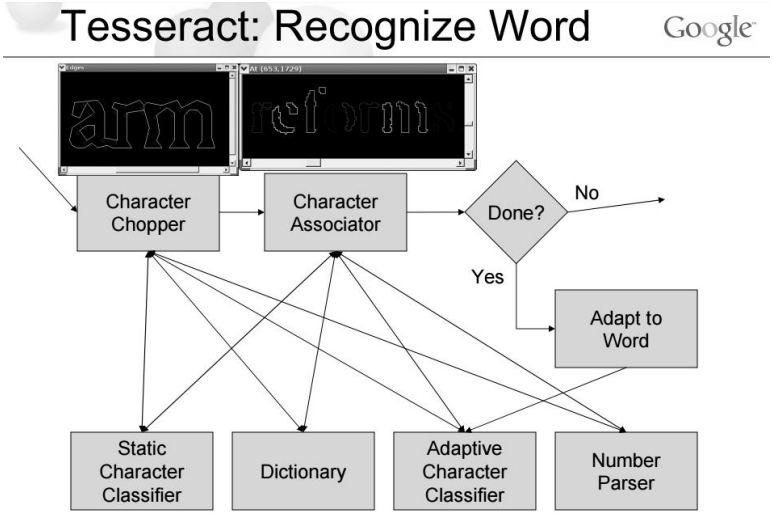
Proses terakhir adalah proses pengenalan kembali dari keseluruhan halaman pada citra untuk memperbaiki hasil pengenalan tahap pertama.

2.5.1 Pengenalan Karakter Pada Tesseract

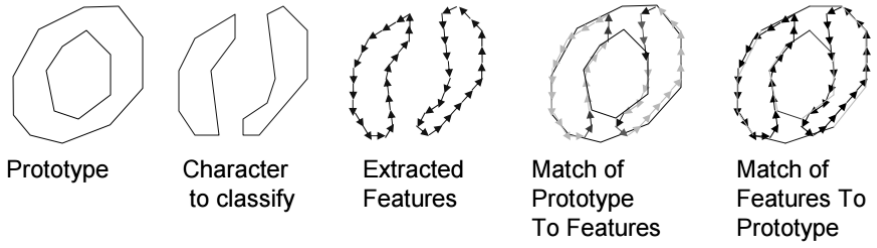
Proses pengenalan karakter pada Tesseract yakni :

1. Pemisahan karakter-karakter termasuk didalamnya mengklasifikasi karakter, *parsing number,* dan pencocokan data pada kamus *library.*
2. Mengasosiasikan setiap karakter dengan kembali mengklasifikasi karakter, dan mencocokkan kembali pola-pola karakter dengan data pada *library.*
3. Jika karakter yang dibaca per baris telah habis maka setiap karakter yang terbaca akan diadaptasikan ke dalam bentuk kata. Jika baris belum habis maka sistem akan memproses baris selanjutnya dengan proses yang sama.

Proses pengenalan kata dan pengenalan pola karakter, dirujuk dari Tesseract OSCON (tesseract-ocr.googlecode.com) ditunjukkan pada Gambar 2.2 dan Gambar 2.3 berikut :



Gambar 2.2 Pengenalan Kata Pada Tesseract



Gambar 2.3 Pengenalan Dan Pencocokan Karakter Pada Tesseract

2.5.2 Cara Penggunaan Tesseract Pada OCR

Cara penggunaan Tesseract pada OCR dengan Android Studio yakni :

1. Unduh *open source* Tesseract di situs Google Code http://code.google.com/p/tesseract-ocr.
2. Buat direktori *library* pada Android Studio
3. Salin direktori tessdata dan tessdll.dll kedalam direktori.
4. Edit setting.gradle dengan meng-*add library* ke include. Gunakan *library* pada file class Java dengan meng-*import library* dan menginisialisasi method yang digunakan. Contoh hasil *add* pada setting.gradle:

```
include ':app', ':Tesseract'
project(':Leptonica').projectDir = new File('libs/ Tesseract ')
```

5. Jika mengalami error pada *Default Configuration,* ganti coding pada tahap 4 dengan :

```
include ':app'
include ':libs: Tesseract'
```

6. Pada folder app/build.gradle, tambahkan proyek *library* sebagai *dependency* :

```
dependencies {
   compile fileTree(dir: 'libs', include: ['*.jar'])
   compile 'com.android.support:appcompat-v7:21.0.3'
   compile project(":Tesseract")
     //compile project(":libs: Tesseract") jika melalui tahap 5

}
```

7. Kemudian tambahkan global config pada gradle.properties untuk menampung semua versi SDK dan versi *build tools* untuk menjaga konsistensi setiap module. Edit gradle.properties,tambahkan :

```
ANDROID_BUILD_MIN_SDK_VERSION=14
ANDROID_BUILD_TARGET_SDK_VERSION=21
ANDROID_BUILD_TOOLS_VERSION=21.1.3
ANDROID_BUILD_SDK_VERSION=21
```

8. Klik *sync project* dengan icon gradle. Maka library akan muncul dan siap digunakan dalam proyek.

2.5.3 Cara Kerja Tesseract

Tesseract bekerja dengan langkah seperti berikut:

1) Membaca Input

Garis-garis dibaca dari citra yang di-*scan*, pada deteksi tepi.

2) Deteksi Tepi/Outline

   Pixel berwarna hitam terbagi menjadi *blob*. Kemudian *blob* tersebut diproses untuk mengekstrak outline, pada deteksi tepi.

3) Garis/Skew

   Garis-garis berasal dari untaian *blob* dengan outline. Gradien/rotasi dari halaman kemudian dihitung. Lalu garis-garis disesuaikan dengan *skew*. Sentuhan akhirnya dengan menetapkan *blob*, sehingga garis-garis tersebut dapat dikenali.

4) Kata/Segmenter
5) Klasifikasi
6) Kualitas

   Kualitas kata dan huruf dicentang

7) Menulis Output

2.5.4 Versi Release Tesseract

Berikut ini adalah versi-versi dari Tesseract yang di-*release*:

a. Tesseract *release* Mei 15 2007 – V1.04
b. Tesseract *release* Juli 18 2007 – V2.00
c. Tesseract *release* Agustus 30 2007 – V2.01
d. Tesseract *release* April 21 2008 – V2.02
e. Tesseract *release* April 22 2008 – V2.03
f. Tesseract *release* Juni 30 2009 – V2.04
g. Tesseract *release* September 30 2010 – V3.00
h. Tesseract *release* Oktober 21 2011 – V3.01
i. Tesseract *release* Oktober 23 2012 – V3.02.02
j. Tesseract *release* Februari 4 2014 – V3.03(rcl)
k. Tesseract *release* Juli 11 2015 – V3.04.00
l. Tesseract *release* Februari 16 2016 – V3.04.01

Terdapat 457 file class yang ada di Tesseract library. Fungsi dan nama file class selengkapnya dapat dilihat di: http://tesseract-ocr.repairfaq.org/main.html. Library Tesseract yang telah di *wrap* ke dalam bahasa Java (dengan bantuan library tess-two untuk representasi Java dari native Tesseract object), terdiri dari 3 class yakni :

1. PageIterator.java

Method	Parameter	Keterangan
begin()		Me-reset iterator dari poin ke awal halaman.
next()	int level	Pindah ke start dari next object pada given level di page hierarchy, dan me-return false jika mencapai page terakhir
getBoundingBox()	int level	Mengambil bounding box: x, y, w, h * Koordinat sistem: * Integer koordinats merupakan crack diantara pixel.

2. ResultIterator.java

Method	Parameter	Keterangan
getUTF8Text()	int level	Me-return string teks untuk object pada given level
confidence()	int level	Me-return mean confidence dari current object pada given level.

3. TessBaseAPI.java

Method	Parameter	Keterangan
PageIteratorLevel()	int RIL_BLOCK, int RIL_PAR, int RIL_TEXTLINE, int RIL_WORD, int RIL_SYMBOL	Element dari page hierarchy, digunakan untuk menyediakan fungsi yang mengoperasikan setiap level tanpa harus memiliki sebanyak 5x fungsi.
TessBaseAPI()		Konstruksi dari TessBaseAPI
Finalize()		Dipanggil dari GC untuk

		membersihkan native data yang telah di set saat mengkonstruksi objek
init()	String datapath, String language	Menginisialisasi Tesseract engine dengan sebuah specified language model. Me-return true jika success.
init()	String datapath, String language, int ocrEngineMode	Menginisialisasi Tesseract engine dengan sebuah specified language model. Me-return true jika success. Parameter ocrEngineMode yakni OCR engine mode untuk di set
getInitLanguagesAsString()		Me-return languages string yang digunakan pada last valid initialization.
clear()		Membebaskan hasil dari recognition dan semua jenis data citra, tanpa harus membebaskan data yang akan memakan waktu untuk di-reload.
end()		Menutup tesseract dan membebaskan seluruh memori.
setVariable()	String var, String value	Mengatur value dari sebuah internal "variable"
setPageSegMode()	int mode	Mengatur mode page segmentation. Ini mengatur seberapa banyak processing OCR engine yang akan dilakukan sebelum pengenalan karakter.
setDebug()	boolean enabled	Mengatur debug mode.
setRectangle()	Rect rect	Menhindari recognition untuk meng-sub-rectangle citra.
setRectangle()	int left, int top, int width, int height	Menhindari recognition untuk meng-sub-rectangle citra dengan

		parameter disesuaikan
setImage()	File file	Menyediakan citra untuk dikenali oleh Tesseract.
setImage()	Bitmap bmp	Menyediakan citra untuk dikenali oleh Tesseract. *Parameter bmp bitmap representasi dari citra
setImage()	Pix image	Menyediakan citra dengan format Leptonica pix untuk dikenali oleh Tesseract
setImage()	byte[] imagedata, int width, int height, int bpp, int bpl	Menyediakan citra untuk dikenali oleh Tesseract.
getUTF8Text()		Karakter yang dikenali dari citra dikembalikan dalam bentuk string dengan pengkodean UTF8.
getHtmlText()		Me-returnkan karakter yang dikenali dari citra dalam bentuk html yang dapat dipergunakan
meanConfidence()		Me-returnkan mean confidence dari pengenalan teks
wordConfidences()		Me-return seluruh kata confidences (antara 0 dan 100) pada sebuah array. =
getThresholdedImage()		Me-return sebuah copy dari internal citra thresholded dari Tesseract.
getRegions()		Me-return hasil dari page layout analysis sebagai sebuah Pixa,
getTextlines()		Me-return textlines sebagai Pixa.
getStrips()		Me-return strips sebagai Pixa.
getWords()		Me-return kata bounding boxes sebagai Pixa.
getResultIterator()		Me-return load konten dengan

		Picasso
getHOCRText()	int page	Membuat sebuah HTML-formatted string dengan hOCR markup dari data internal.
setInputName()	String name	Mengatur nama dari input file. Hanya untuk training dan reading pada UNLV zone file.
setOutputName()	String name	Mengatur nama dari output file. Hanya untuk debugging.
ReadConfigFile()	String filename	Membaca sebuah "config" file berisi satu set variable, value pairs.
getBoxText()	int page	Me-returnkan karakter yang dikenali dari citra dalam bentuk kode yang sama dengan UTF8
onProgressValues()	final int percent, final int left, final int right, final int top, final int bottom, final int left2, final int right2, final int top2, final int bottom2	Memanggil dari native code untuk menampilkan progress value

2.6 Library Tess-Two

Tess-two adalah sebuah *fork* dari *tools* Tesseract untuk Android yang menyediakan kemampuan untuk memanfaatkan mesin OCR pada perangkat Android. *Tools* Tesseract untuk Android terdiri dari tiga fitur, yakni Android API, mesin Tesseract OCR, dan Leptonica Image Processing Library. Library ini dapat diunduh dan dikembangkan secara gratis di https://github.com/rmtheis/tess-two.

Tess-two memiliki alat-alat untuk mengkompilasi dan menjalankan Tesseract maupun Leptonica *Image Processing Library* di Android OS. *Library* tess-two menambahkan beberapa fungsi tambahan. Proyek ini bekerja dengan Tesseract 3.05.00 dev, Leptonica 1,73, libjpeg 9b, dan libpng 1.6.20. Kode asli dari *library* juga disertakan dalam folder tess-two/jni.

Dalam modul tess-two terdapat modul eyes-two yang berisi tambahan kode dari proyek eyes-free. Modul eyes-two tidak dibutuhkan untuk mengakses

Tesseract dan Leptonica API. Modul tess-two-test berisi *instrumented unit test* untuk tess-two.

2.6.1 Mengkonfigurasi tess-two dengan gradle

Ada beberapa langkah yang perlu diikuti jika mengintegrasikan tess-two dengan proyek Android gradle.

a. Buat folder *library* di bawah direktori utama proyek. Misalnya, jika proyek bernama **Project1**, maka buat folder **Project1/libraries**

b. Sekarang salin seluruh direktori tess-two ke folder yang baru saja dibuat.

c. Hapus *libs* folder dalam direktori tess-two. Jika perlu, hapus project.properties, build.xml, .classpath, dan .project. file pada tess-two. Karena ini tidak diperlukan untuk proyek.

d. Buat file build.gradle dalam direktori tess-two, isi dengan script berikut :

```
buildscript {
    repositories {
        mavenCentral()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:0.9.+'
    }
}

apply plugin: 'android-library'

android {
    compileSdkVersion 19
    buildToolsVersion "19.0.3"

    defaultConfig {
        minSdkVersion 8
        targetSdkVersion 19
    }

    sourceSets.main {
        manifest.srcFile 'AndroidManifest.xml'
        Java.srcDirs = ['src']
        resources.srcDirs = ['src']
```

```
        res.srcDirs = ['res']
        jniLibs.srcDirs = ['libs']
    }
}
```

d. Edit file settings.gradle di direktori utama proyek (Project1/settings.gradle) dan tambahkan baris ini : include ':libraries:tess-two'

e. Kemudian sync proyek di Android Studio dan tambahkan tess-two sebagai modul *dependency* untuk proyek utama (setelah sinkronisasi tess-two harus muncul sebagai modul)

2.7 Java

Java [8] adalah bahasa pemrograman berorientasi objek murni yang dibuat berdasarkan kemampuan-kemampuan terbaik bahasa pemrograman objek sebelumnya (C++, Ada, Simula). Bahasa pemrograman Java terlahir dari *The Green Project*, yang berjalan selama 18 bulan, dari awal tahun 1991 hingga musim panas 1992. Proyek tersebut belum menggunakan versi yang dinamakan “Oak”. Proyek ini dimotori oleh Patrick Naughton, Mike Sheridan, dan James Gosling, beserta sembilan pemrogram lainnya dari Sun Microsystems. Salah satu hasil proyek ini adalah maskot Duke yang dibuat oleh Joe Palrang.

Pertemuan proyek berlangsung di sebuah gedung perkantoran Sand Hill Road di Menlo Park. Sekitar musim panas 1992 proyek ini ditutup dengan menghasilkan sebuah program Java Oak pertama, yang ditujukan sebagai pengendali sebuah peralatan dengan teknologi layar sentuh (*touch screen*), seperti pada PDA sekarang ini. Teknologi baru ini dinamai "*7" (*Star Seven*).

Setelah era *Star Seven* selesai, sebuah anak perusahaan televisi kabel tertarik ditambah beberapa orang dari proyek *The Green Project*. Mereka memusatkan kegiatannya pada sebuah ruangan kantor di 100 Hamilton Avenue, Palo Alto. Perusahaan baru ini bertambah maju, jumlah karyawan meningkat dalam waktu singkat dari 13 menjadi 70 orang. Pada rentang waktu ini juga ditetapkan pemakaian internet sebagai medium yang menjembatani kerja dan ide di antara mereka. Pada awal tahun 1990-an, internet masih merupakan rintisan, yang dipakai hanya di kalangan akademisi dan militer.

Mereka menjadikan perambah (*browser*) Mosaic sebagai landasan awal untuk membuat perambah Java pertama yang dinamai “Web Runner”, terinsipirasi dari film 1980-an, Blade Runner. Pada perkembangan rilis pertama, Web Runner berganti nama menjadi “Hot Java”. Pada sekitar bulan Maret 1995, untuk pertama kali kode sumber Java versi 1.0 dibuka. Kesuksesan mereka diikuti dengan pemberitaan pertama kali pada surat kabar *San Jose Mercury News* pada tanggal 23 Mei 1995. Tetapi terjadi perpecahan di antara mereka suatu hari pada pukul 04.00 di sebuah ruangan hotel Sheraton Palace. Tiga dari pimpinan utama proyek, Eric Schmidt dan George Paolini dari Sun Microsystems bersama Marc Andreessen, membentuk Netscape.

Nama Oak, diambil dari pohon oak yang tumbuh di depan jendela ruangan kerja "Bapak Java", James Gosling. Nama Oak ini tidak dipakai untuk versi release Java karena sebuah perangkat lunak lain sudah terdaftar dengan merek dagang tersebut, sehingga diambil nama penggantinya menjadi "Java". Nama ini diambil dari kopi murni yang digiling langsung dari biji (kopi tubruk) kesukaan Gosling. Konon kopi ini berasal dari Pulau Jawa. Jadi nama bahasa pemrograman Java tidak lain berasal dari kata Jawa (bahasa Inggris untuk Jawa adalah Java).

Sebagai bahasa pemrograman, Java dirancang menjadi handal dan aman. Java juga dirancang agar dapat dijalankan di semua *platform*. Java mudah dipelajari, terutama bagi programmer yang telah mengenal C/C++. Java merupakan bahasa pemrograman berorientasi objek yang merupakan paradigma pemrograman masa depan.

Java juga dirancang untuk menghasilkan aplikasi–aplikasi dengan performansi yang terbaik, seperti aplikasi database Oracle 8i/9i yang *core*-nya dibangun menggunakan bahasa pemrograman Java. Sedangkan Java bersifat *neutral architecture*, karena *Java Compiler* yang digunakan untuk mengkompilasi kode program Java dirancang untuk menghasilkan kode yang netral terhadap semua arsitektur perangkat keras yang disebut sebagai *Java Bytecode*. Karakteristik Java antara lain :

a. Sederhana (Simple)
b. Berorientasi Objek (*Object Oriented*)
c. Terdistribusi (*Distributed*)

d. *Interpreted*
e. *Robust*
f. Aman (*Secure*)
g. *Architecture Neutral*
h. *Portable*
i. *Performance*
j. *Multithreaded*
k. Dinamis

Sebagai sebuah platform, Java terdiri atas dua bagian utama, yaitu:

1. *Java Virtual Machine* (JVM).
2. *Java Application Programming Interface* (Java API).

Sun membagi arsitektur Java membagi tiga bagian, yaitu:

1. *Enterprise Java* (J2EE) untuk aplikasi berbasis web, aplikasi sistem tersebar dengan beraneka ragam klien dengan kompleksitas yang tinggi. Merupakan superset dari *Standard Java*.
2. *Standard Java* (J2SE), ini adalah yang biasa dikenal sebagai bahasa Java.
3. *Micro Java* (J2ME) merupakan subset dari J2SE dan salah satu aplikasinya yang banyak dipakai adalah untuk *wireless device/ mobile device*.

Macam-macam Java 2 *Software Developer Kit* (J2SDK) antara lain :

a. J2SE (Java 2 *Standard Edition*)
   Ini adalah yang biasa dikenal sebagai bahasa Java.
b. J2EE (Java 2 *Enterprise Edition*)
   Untuk aplikasi berbasis web, aplikasi sistem tersebar dengan beraneka ragam klien dengan kompleksitas yang tinggi. Merupakan *superset* dari *Standard Java.*
c. J2ME (Java 2 *Micro Edition*)
   Merupakan *subset* dari J2SE dan salah satu aplikasinya yang banyak dipakai adalah untuk *wireless device* atau *mobile device.*

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

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

Contoh kode program sederhana “Hello World” yang ditulis menggunakan bahasa pemrograman Java :

```
// Outputs "Hello, world!" and then exits
public class HelloWorld {
  public static void main(String args[]) {
    System.out.println("Hello, world!");
  }
}
```

2.8 Android

Android adalah sebuah sistem operasi untuk perangkat mobile berbasis linux yang mencakup sistem operasi, middleware dan aplikasi. Android merupakan platform atau aplikasi yang bebas untuk dikembangkan [10]. Android meyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Awalnya, Google Inc. membeli Android Inc. yang merupakan pendatang baru yang membuat peranti lunak untuk *smartphone*. Kemudian untuk mengembangkan Android, dibentuklah Open Handset Alliance, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia.

Pada saat perilisan perdana Android, 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 Service* (GMS) dan kedua adalah yang benar-benar bebas distribusinya tanpa dukungan langsung Google atau dikenal sebagai *Open Handset Distribution* (OHD). Pada 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 modifikasi kernel Linux 2.6.

Sejak Android dirilis telah dilakukan berbagai pembaharuan berupa perbaikan bug dan penambahan fitur baru. Pada masa saat ini sebagian besar vendor-vendor *smartphone* sudah memproduksi *smartphone* berbasis android, vendor-vendor itu antara lain HTC, Motorola, Samsung, LG, HKC, Huawei, Archos, Webstation Camangi, Dell, Nexus, SciPhone, WayteQ, Sony Ericsson, Acer, Philips, T-Mobile, Nexian, IMO, Asus dan masih banyak lagi vendor *smartphone* di dunia yang memproduksi Android. Hal ini, karena Android adalah sistem operasi yang *open source* sehingga bebas didistribusikan dan dipakai oleh vendor manapun. Tidak hanya menjadi sistem operasi di *smartphone,* saat ini Android menjadi pesaing utama dari Apple pada sistem operasi Tablet dan 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 *tools* untuk pengembangan, *market* aplikasi android serta dukungan yang sangat tinggi dari 15 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 Android SDK

Android SDK adalah *tools* API (Application Programming Interface) yang diperlukan untuk mulai mengembangkan aplikasi pada platform Android menggunakan bahasa pemrograman Java. Android merupakan *subset* perangkat lunak untuk ponsel yang meliputi sistem operasi, middleware dan aplikasi kunci yang dikembangkan oleh Google. Saat ini disediakan Android SDK (*Software Development Kit*) sebagai alat bantu dan API untuk mulai mengembangkan aplikasi pada platform Android menggunakan bahasa pemrograman Java.

Sebagai platform aplikasi-netral, Android memberikan kesempatan untuk membuat Aplikasi yang dibutuhkan yang bukan merupakan aplikasi bawaan *smartphone*. Beberapa fitur-fitur Android yang paling penting adalah :

1. *Framework* yang mendukung penggantian komponen dan *reusable*.
2. Mesin Virtual Dalvik dioptimalkan untuk perangkat mobile.
3. *Integrated browser* berdasarkan *engine open source* Webkit.
4. Grafis yang dioptimalkan dan didukung oleh libraries grafis 2D, grafis 3D berdasarkan spesifikasi opengl ES 1,0 (Opsional akselerasi hardware).
5. SQLite untuk penyimpanan data.
6. Media *support* yang mendukung audio, video, dan gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF), GSM (tergantung *hardware*).
7. Bluetooth, EDGE, 3G, dan WiFi (tergantung *hardware*).
8. Kamera, GPS, kompas, dan accelerometer (tergantung *hardware*).
9. Lingkungan *development* yang lengkap dan kaya termasuk perangkat emulator, perangkat untuk *debugging*, profil dan kinerja memori, dan *plug-in* untuk IDE Eclipse.

2.10 Android Native Development Kit (NDK)

Native Development Kit (NDK) [12] adalah satu set alat yang disediakan oleh Android yang memungkinkan untuk menambahkan dan mengeksekusi kode dalam bahasa C dan C ++ ke dalam aplikasi Android yang dikembangkan.
NDK dalam beberapa kasus akan meningkatkan peforma komputasi dan pemrosesan data yang kompleks. NDK akan memudahkan penulisan kode untuk terintegrasi langsung dengan kernel dan hardware *device* pada Android.

NDK digunakan oleh SDK android sebagai abstraksi layer presentasi ke layer hardware. Sehingga memudahkan developer untuk berinteraksi langsung dengan hardware tanpa menggunakan bahasa Java. NDK dapat digunakan untuk membangun dari *source code* sendiri atau mengambil *library prebuilt* yang sudah ada. NDK dapat berguna dalam kasus seperti:

a. Ketika bertemu dengan algoritma dengan kompleksitas yang tinggi atau kalkulasi kompleks pada permodelan matematika
b. Ketika aplikasi membutuhkan koneksi langsung ke layer device seperti membaca signal radio dari bluetooth, koneksi ke VOIP, implementasi *physic computation* pada *game* atau bahkan akses untuk me-render augmentasi kamera dengan penambahan object secara real time.
c. Ketika membutuhkan kinerja ekstra dari perangkat untuk mesin aplikasi seperti aplikasi permainan atau simulasi fisika.
d. Me-*reuse library* C atau C++ buatan sendiri ataupun buatan pengembang lain.

2.10.1 Manfaat NDK

Manfaat menggunakan NDK diantaranya adalah:

a. Dapat menggunakan kode-kode / *library* yang ditulis dalam bahasa C/C++ ke dalam aplikasi yang dikembangkan
b. Memudahkan interaksi langsung dengan hardware yang dibutuhkan jika ada *custom usage* yang dibutuhkan
c. Dalam beberapa kasus C/C++ memberikan performa aplikasi yang bagus untuk melakukan proses komputasi untuk kalkulasi yang besar
d. Sebagai solusi jika memang proses-proses tertentu hanya bisa atau hanya terdapat dukungan dalam bahasa C/C++

2.10.2 Cara Menggunakan NDK

Sebelum menggunakan NDK, pengembang harus terlebih dahulu familiar dengan bahasa C/C++, JNI (Java Native Interface), penggunaan cygwin (harus terinstal terlebih dahulu untuk pengguna windows, untuk mac dan linux sudah ter-*default*) dan harus familiar dengan Android Application Development baik itu SDK maupun IDE nya. Berikut langkah dalam menggunakan NDK:

1. Install NDK terlebih dahulu. NDK dapat di-*download* di: https://developer.android.com/ndk/downloads/index.html
2. Setelah selesai install NDK, pastikan Cygwin (untuk Windows) sudah terinstall
3. Buat C/C++ code dalam bentuk JNI
4. Buat file Android.mk
5. Letakkan di folder jni
6. Buatlah Java Code
7. Klik Kanan di project, lalu pilih add native support
8. Klik Finish pada dialog yang muncul bertuliskan NDK enable
9. Klik kanan di project, pilih Make Targets kemudian pilih Create
10. Masukkan nama "Target Name" pada dialog dan klik Ok. Hasil *compile* code C/C++ akan menjadi lib.so
11. Build .so dengan cara pilih Make Targets lalu pilih Build. Pilih target yang sudah dibuat sebelumnya kemudian Build pada dialog yang muncul

Kelemahan dalam menggunakan NDK adalah:

1. Kode dan project kita akan semakin kompleks
2. Dibutuhkan pemahaman terhadap C/C++ dan JNI (Java Native Interface) bagi yang belum familiar
3. Kebutuhan resource yang besar bagi pengguna windows untuk mengunduh cygwin dan Android NDK yang ukurannya cukup besar

2.10.3 Versi Release NDK

Versi-versi dari Android NDK yang telah di-*release* yaitu sebagai berikut:

1) Android NDK Revision 1 release Juni 2009
2) Android NDK Revision 2 release September 2009

3) Android NDK Revision 3 release Maret 2010
4) Android NDK Revision 4b release Juni 2010
5) Android NDK Revision 5 release Desember 2010
6) Android NDK Revision 5b release Januari 2011
7) Android NDK Revision 5c release Juni 2011
8) Android NDK Revision 6 release Juli 2011
9) Android NDK Revision 6b release Agustus 2011
10) Android NDK Revision 7 release November 2011
11) Android NDK Revision 7b release Februari 2012
12) Android NDK Revision 7c release April 2012
13) Android NDK Revision 8 release Mei 2012
14) Android NDK Revision 8b release Juli 2012
15) Android NDK Revision 8c release November 2012
16) Android NDK Revision 8d release Desember 2012
17) Android NDK Revision 8e release Maret 2013
18) Android NDK Revision 9 release Juli 2013
19) Android NDK Revision 9b release Oktober 2013
20) Android NDK Revision 9 release Juli 2013
21) Android NDK Revision 9b release Oktober 2013
22) Android NDK Revision 9c release Desember 2013
23) Android NDK Revision 9d release Maret 2014
24) Android NDK Revision 10 release Juli 2014
25) Android NDK Revision 10b release September 2014
26) Android NDK Revision 10c release Oktober 2014
27) Android NDK Revision 10d release Desember 2014
28) Android NDK Revision 10e release Mei 2015
29) Android NDK Revision 11 release Maret 2016
30) Android NDK Revision 11b release Maret 2016

2.11 SQLite

Menurut Jay A. Kreibich (2010,12), SQLite merupakan paket perangkat lunak yang bersifat *public domain* yang menyediakan sistem manajemen basis data relasional atau RDBMS. Sistem basis data relasional digunakan untuk

menyimpan *record* yang didefinisikan oleh pengguna pada ukuran tabel yang besar dan memproses perintah *query* yang kompleks dan menggabungkan data dari berbagai tabel untuk menghasilkan laporan dan rangkuman data.

Kata ‘Lite’ pada SQLite tidak menunjuk pada kemampuannya, melainkan menunjuk pada sifat dari SQLite, yaitu ringan ketika dihubungkan dengan kompleksitas pengaturan, *administrative overhead*, dan pemakaian sumber. SQLite memiliki fitur-fitur sebagai berikut :

1. Tidak Memerlukan *Server*

   Arsitektur SQLite tidak memiliki arsitektur *client server*. Kebanyakan sistem *database* skala besar memiliki paket *server* yang besar yang membentuk mesin *database*.

2. *Single File Database*

   SQLite mengemas seluruh *database* ke dalam suatu *single file*. *Single file* tersebut berisi *layout database* dan data aktual yang berada pada tabel dan indeks yang berbeda. Format *file* dapat digunakan pada banyak *platform* dan dapat diakses pada mesin manapun tanpa memperhatikan *native byte order* ataupun ukuran kata. Pengemasan *database* kedalam suatu *file* tunggal memudahkan pengguna untuk membuat, menyalin, ataupun mem-*backup image database* yang berada di dalam media penyimpanan.

3. *Zero Configuration*

   SQLite tidak membutuhkan apapun untuk melakukan instalasi dan konfigurasi. Dengan mengeliminasi *server* dan menggabungkan *database* secara langsung ke dalam aplikasi, maka pengguna tidak perlu mengetahui bahwa mereka sedang menggunakan *database.*

4. *Embedded Device Support*

   Ukuran *code* dari SQLite bersifat kecil dan penggunaan sumber daya yang konservatif membuatnya cocok digunakan untuk *embedded system* yang berjalan terbatas pada sistem operasi.

5. Fitur-Fitur Yang Unik

   SQLite menggunakan sistem dengan tipe dinamis untuk tabel-tabel. SQLite memungkinkan pengguna untuk memasukkan nilai ke dalam kolom tanpa memperhatikan tipe data. Pada beberapa cara pemakaiannya,

sistem yang bertipe dinamis pada SQLite mirip dengan sistem yang ditemukan pada bahasa *scripting* yang populer, yang sering memiliki sebuah tipe skalar yang dapat menerima semua tipe data dari *integer* sampai *string*. Fitur lainnya adalah kemampuan untuk memanipulasi lebih dari satu basis data pada satu waktu. SQLite mempunyai kemampuan dalam menghubungkan sebuah koneksi *database* tunggal dengan banyak file basis data secara bersamaan.

Hal ini memungkinkan SQLite untuk memproses SQL *statement* yang menjembatani beberapa basis data sekaligus.

6. *Compatible License*

   SQLite dan SQLite code tidak memiliki lisensi pengguna dan tidak dilindungi oleh GNU's Not Unix (GNU) General Public License (GPL) atau lisensi *open source* sejenisnya. Hal ini berarti pengguna dapat melakukan apapun dengan *source code* SQLite, sehingga *library code* dapat digunakan dengan berbagai cara, dimodifikasi dengan berbagai cara dan didistribusikan dengan berbagai cara.

7. *Highly Reliable*

   Sejumlah tes telah dilakukan sebelum *library* SQLite masing-masing dirilis. Hal ini dilakukan untuk mempertahankan tingkat kehandalan yang tinggi.

2.12 Unified Modelling Language (UML)

Sampai era tahun 1990 puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: metodologi booch, metodologi coad, metodologi OOSe, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock, dan sebagainya. Masa itu terkenal dengan masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan kelompok atau perusahaan lain yang menggunakan metodologi yang berlainan.

Dimulai pada bulan Oktober 1994, Grady Booch, Jim Rumbaugh dan Ivar Jacobson, yang merupakan tiga tokoh yang boleh dikata metodologinya banyak

digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 di-*release* draft pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh *Object Management Group,* sebuah organisasi yang telah mengembangkan model, teknologi, dan standar OOP sejak tahun 1980-an.

UML [10] adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem. UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek (OOP). Penggunaan UML dalam industri terus meningkat. Ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri peranti lunak dan pengembangan sistem.

Berikut beberapa tujuan atau fungsi dari penggunaan UML, diantaranya:

1. Dapat memberikan bahasa permodelan visual kepada pengguna dari berbagai macam pemrograman maupun proses rekayasa.
2. Dapat menyatukan praktek-praktek terbaik yang ada dalam permodelan.
3. Dapat memberikan model yang siap untuk digunakan, merupakan bahasa permodelan visual yang ekspresif untuk mengembangkan sistem dan untuk saling menukar model secara mudah.
4. Dapat berguna sebagai *blue print*, sebab sangat lengkap dan detail dalam perancangannya yang nantinya akan diketahui informasi yang detail mengenai *coding* suatu program.
5. Dapat memodelkan sistem yang berkonsep berorientasi objek, jadi tidak hanya digunakan untuk memodelkan perangkat lunak (*software*) saja.
6. Dapat menciptakan suatu bahasa permodelan yang nantinya dapat dipergunakan oleh manusia maupun oleh mesin.

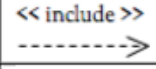
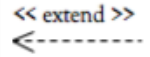
Berikut contoh macam-macam diagram yang paling sering digunakan dalam pembangunan aplikasi berorientasi *object* atau *object oriented programming* (OOP), yaitu *use case* diagram, diagram *activity, class* diagram, dan *sequence* diagram.

1. *Use Case* Diagram

Use case diagram digunakan untuk memodelkan bisnis proses berdasarkan perspektif pengguna sistem. *Use case* diagram terdiri atas diagram untuk *use case* dan aktor. Aktor merepresentasikan orang yang akan mengoperasikan atau orang yang berinteraksi dengan sistem aplikasi. *Use case* merepresentasikan operasi-operasi yang dilakukan oleh aktor. *Use case* digambarkan berbentuk elips dengan nama operasi dituliskan di dalamnya. Aktor yang melakukan operasi dihubungkan dengan garis lurus ke *use case*.

Simbol-simbol yang ada pada *use case* diagram dijelaskan pada tabel berikut:

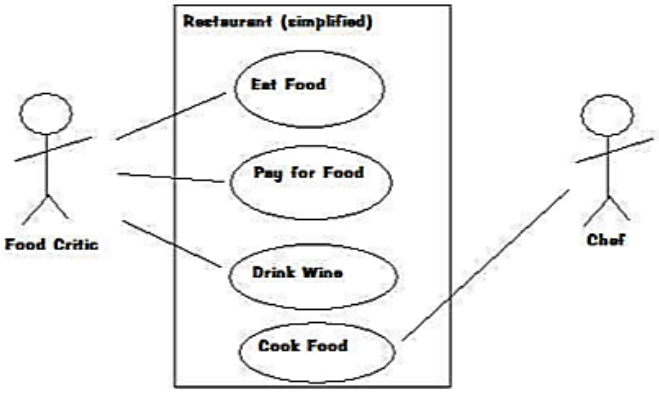
Tabel 2.2 Simbol-simbol *Use Case* Diagram

NO	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
2	<< include >> 	*Include*	Menspesifikasikan *use case* sumber secara *eksplisit*.
3	<< extend >> 	*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber
4		*Association*	Menghubungkan antara aktor dengan *use case*
5		*System*	Menspesifikasikan batasan *use case* yang ada pada sistem
6		*Use Case*	Deskripsi dari urutan aksi-aksi atau proses yang dilakukan oleh aktor atau yang ditampilkan sistem

Menurut James Rumbaugh (1999) terdapat 4 tipe relasi yang digunakan pada *use case* diagram, yaitu asosiasi, generalisasi, extend dan include. Berikut adalah penjelsan dari masing-masing relasi tersebut.

1. Asosiasi

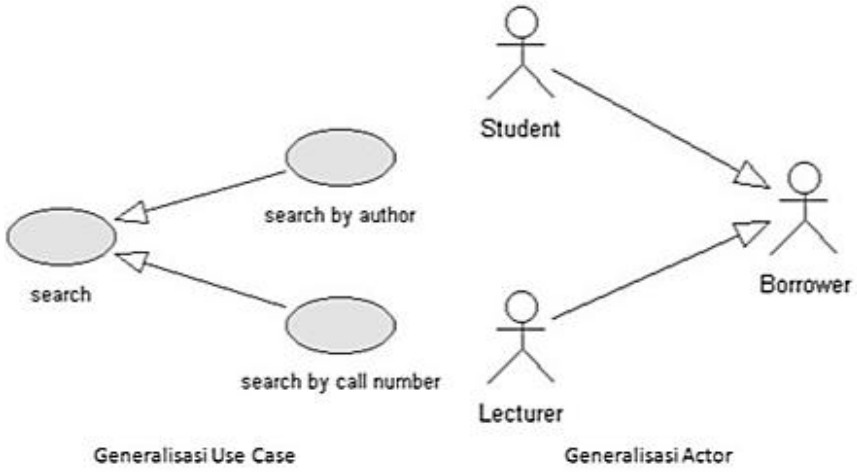
   Asosiasi digunakan untuk menggambarkan interaksi antara aktor dengan setiap *use case* tertentu. Relasi ini digambarkan sebagai garis penghubung aktor terhadap *use case* yang berelasi dengan aktor tersebut. Contoh relasi asosiasi digambarkan pada Gambar 2.4.



Gambar 2.4 Relasi Asosiasi Aktor Dan Use Case

2. Generalisasi

   Suatu relasi antara *use case* umum (induk) dan *use case* yang lebih spesifik (anak). Relasi Generalisasi memungkinkan *use case* atau aktor yang lebih spesifik memiliki perilaku (*behaviour*) yang sama dengan yang lebih umum atau bisa disebut induk. Relasi generalisasi digambarkan dengan anak panah segitiga. *Use case* atau aktor yang terletak di kepala anak panah adalah *use case* atau aktor induk dan yang terletak di sisi lainnya adalah yang mewarisi perilaku *use case* atau actor induk. Contoh digambarkan pada Gambar 2.5.

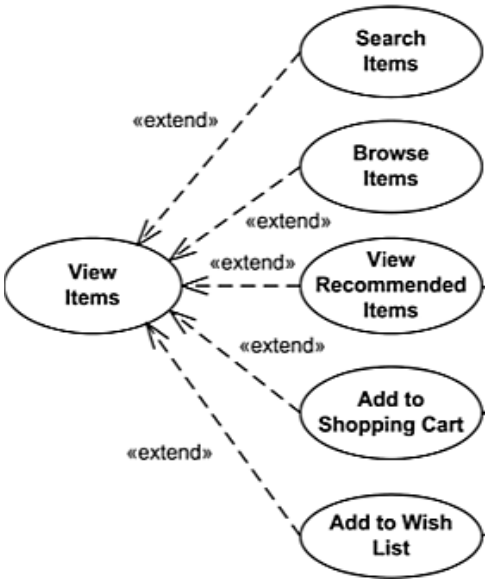


Gambar 2.5 Relasi Generalisasi *Use Case*

3. *Extend*

   Relasi *extend* memungkinkan terjadinya penambahan beberapa *behaviour* (tingkah laku) ke dalam *use case* awal yang pada dasarnya

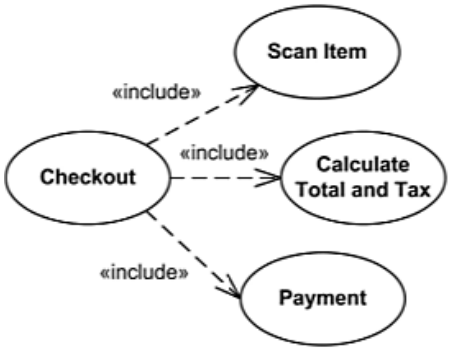
use case tersebut sudah dapat berdiri sendiri tanpa adanya penambahan. *Extend* digambarkan dengan anak panah yang mempunyai garis putus-putus. *Use case* yang berada pada kepala anak panah adalah *use case* awal dan yang berada di lain sisi adalah *use case* tambahan. Contoh digambarkan pada Gambar 2.6.



Gambar 2.6 Contoh *Use Case* dengan *Extend*

4. *Include*

   Relasi *include* memungkinkan terjadinya penambahan perilaku ke dalam *use case* awal yang pada dasarnya tidak dapat berdiri sendiri tanpa adanya penambahan *use case*, dan *use case* awal tidak akan lengkap tanpa adanya *use case* tambahan ini. *Use case* yang berada pada kepala anak panah adalah *use case* penambah, dan pada sisi lain adalah *use case* awal digambarkan pada Gambar 2.7.



Gambar 2.7 Contoh *Use Case* dengan *Include*

Berikut ini contoh *use case* diagram pada suatu kasus sederhana :

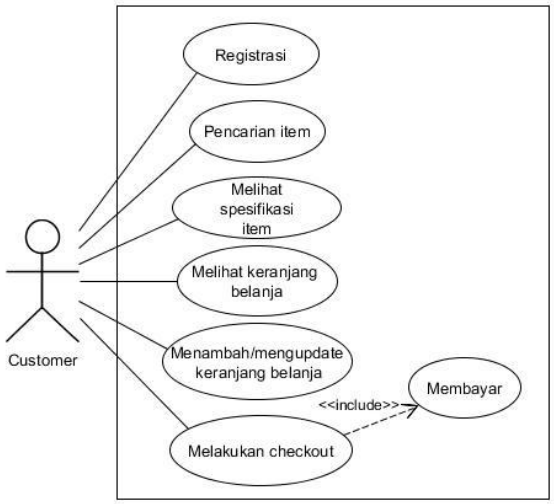
Pada sebuah *online shop* seorang customer dapat melakukan pencarian item, melihat spesifikasi item, menambah barang ke keranjang belanja, melihat dan mengupdate keranjang belanja, membayar dan melakukan *checkout*. Untuk melakukan *checkout* customer harus melakukan registrasi *user* terlebih dahulu.

Aktor :

1. Customer

Use case :

1. Pencarian item
2. Melihat spesifikasi item
3. Melihat keranjang belanja
4. Menambah atau mengupdate keranjang belanja
5. Membayar
6. Melakukan checkout

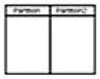


Gambar 2.8 Contoh *use case* diagram

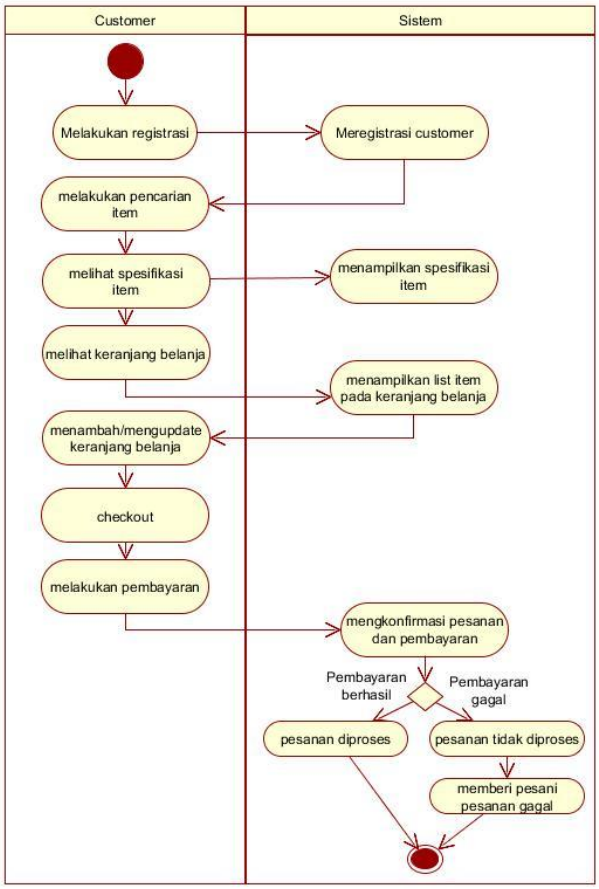
2. *Activity* Diagram

Diagram *activity* dipakai untuk memodelkan interaksi antar objek di dalam sistem. Berbeda dengan *sequence* diagram yang lebih menonjolkan kronologis dari operasi-operasi yang dilakukan, diagram *activity* lebih fokus pada pemahaman atas aktifitas keseluruhan operasi yang dilakukan oleh objek.

Tabel 2.3 Simbol-simbol *Activity* Diagram

NO	GAMBAR	NAMA	KETERANGAN
1		*Swimlane*	Menunjukkan siapa yang bertanggung jawab dalam melakukan aktivitas dalam suatu diagram.
2		*Action*	Langkah-langkah dalam sebuah activity. Action bisa terjadi saat memasuki activity, meninggalkan activity, atau pada event yang spesifik.
3		*Initial State*	Menunjukkan dimana aliran kerja dimulai.
4		*Activity Final Node*	Menunjukkan dimana aliran kerja diakhiri.
5		*Decision Node*	Menunjukkan suatu keputusan yang mempunyai satu atau lebih transisi dan dua atau lebih transisi sesuai dengan suatu kondisi.
6		*Control Flow*	Menunjukkan bagaimana kendali suatu aktivitas terjadi pada aliran kerja dalam tindakan tertentu.

Berikut ini contoh *activity* diagram pembelian item oleh customer pada kasus yang sama dengan contoh kasus *use case* diagram sebelumnya :

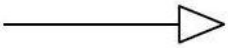


Gambar 2.9 Contoh *Activity* Diagram

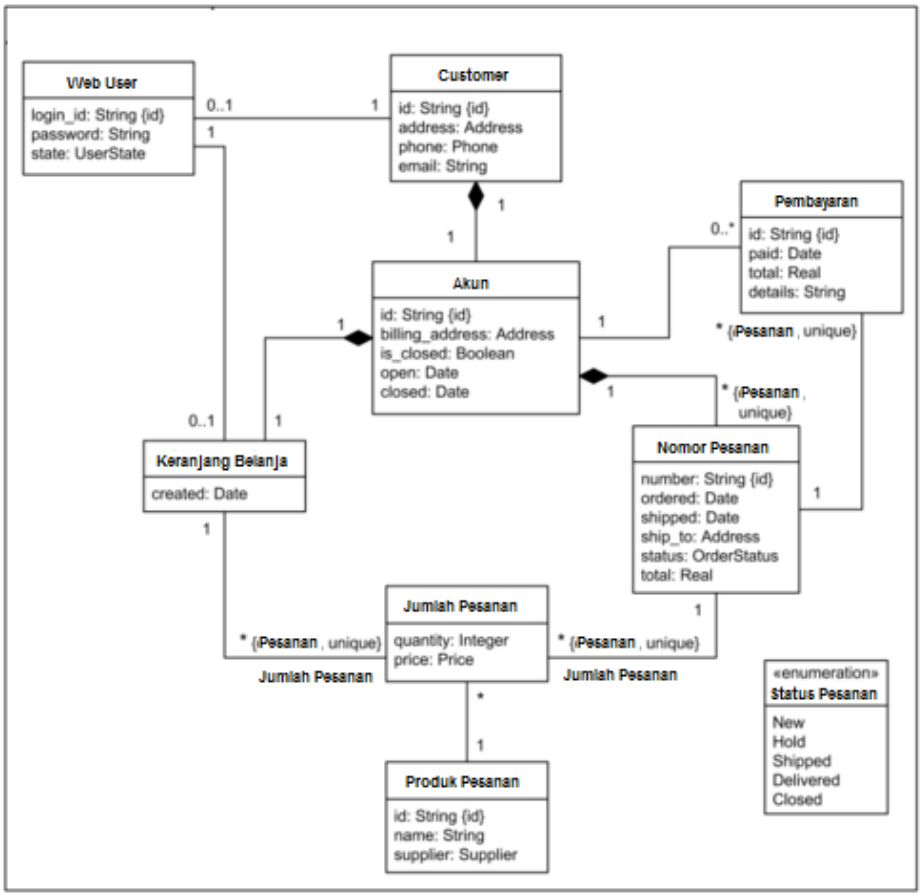
3. *Class* Diagram

Class diagram merupakan diagram yang selalu ada di permodelan sistem berorientasi objek. *Class* diagram menggambarkan struktur statis *class* di dalam sistem. *Class* merepresentasikan sesuatu yang ditangani oleh sistem. Diagram dapat berhubungan dengan yang lain melalui berbagai cara: *associated* (terhubung satu sama lain), *dependent* (satu *class* tergantung atau menggunakan *class* yang lain), *specialed (*satu *class* merupakan spesialisasi dari *class* lainnya), atau *package* (grup bersama sebagai satu unit). Sebuah sistem biasanya mempunyai beberapa *class* diagram. Simbol-simbol yang ada pada *class* diagram dijelaskan pada tabel berikut:

Tabel 2.4 Simbol-simbol *Class* Diagram

Simbol	Keterangan
asosiasi / *association*	relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*
asosiasi berarah / *directed association* 	relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*
generalisasi 	relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus)
kebergantungan / *dependency* 	relasi antar kelas dengan makna kebergantungan antar kelas
agregasi / *aggregation* 	relasi antar kelas dengan makna semua-bagian (*whole-part*)

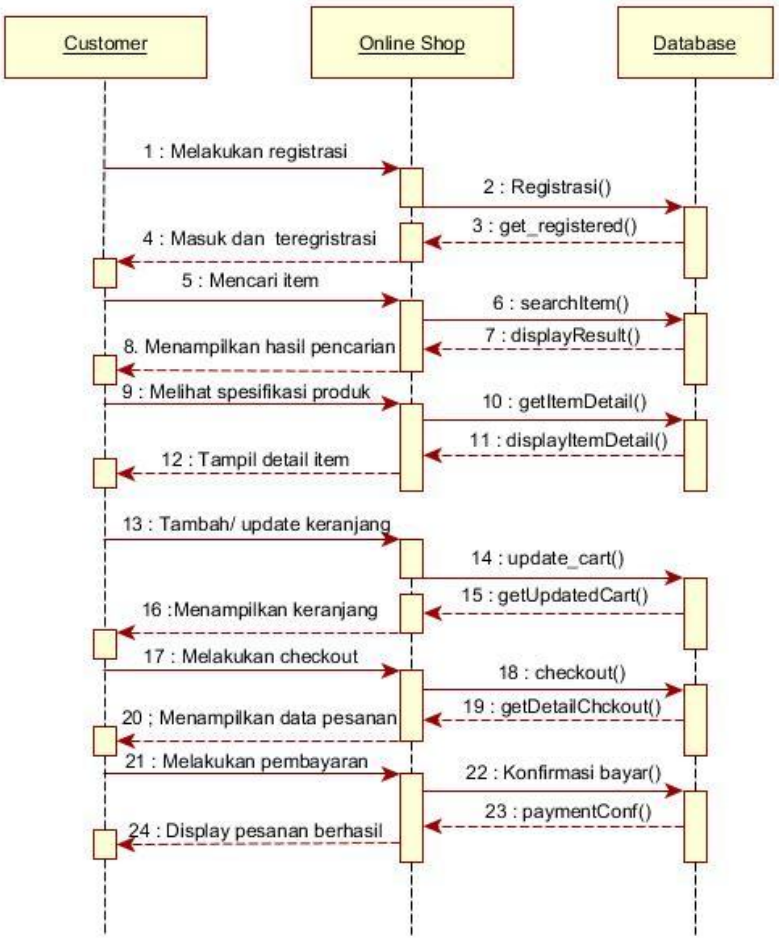
Berikut ini contoh *class* diagram pada kasus yang sama dengan kasus pada contoh *use case* sebelumnya :



Gambar 2.10 Contoh *Class* Diagram

4. *Sequence* Diagram

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. Diagram ini diatur berdasarkan waktu. Objek-objek yang berkaitan dengan proses berjalannya operasi diurutkan dari kiri ke kanan berdasarkan waktu terjadinya dalam pesan yang terurut. Berikut ini contoh *sequence* diagram pada kasus sebelumnya :



Gambar 2.11 Contoh *sequence* diagram

2.13 Android Studio

Android Studio (10) adalah sebuah IDE untuk Android Development yang diperkenalkan google pada acara Google I/O 2013. Android Studio merupakan pengembangkan dari Eclipse IDE, dan dibuat berdasarkan IDE Java populer, yaitu IntelliJ IDEA. Android Studio merupakan IDE resmi untuk pengembangan aplikasi Android.

Sebagai pengembangan dari Eclipse, Android Studio mempunyai banyak fitur-fitur baru dibandingkan dengan Eclipse IDE. Berbeda dengan Eclipse yang menggunakan Ant, Android Studio menggunakan Gradle sebagai *build environment*. Beberapa fitur yang ada pada Android Studio yakni sebagai berikut :

a. Menggunakan Gradle-based build system yang fleksibel.
b. Dapat membangun *multiple* APK

c. *Template support* untuk Google Services dan berbagai macam tipe perangkat.
d. Layout editor yang lebih bagus.
e. Built-in support untuk Google Cloud Platform, sehingga mudah untuk integrasi dengan Google Cloud Messaging dan App Engine.
f. Import library langsung dari Maven repository

Jika dibandingkan, Android Studio memang dari sisi build lebih baik dibandingkan Eclipse, karena Android Studio menggunakan Gradle. Ditambah lagi berbeda dengan Eclipse, kita tidak perlu lagi dipusingkan dengan dependencies package pada Android Studio. Satu hal tambahan lagi yang membuat Android Studio unggul adalah dukungan layout xml editor secara visual yang jauh lebih baik daripada Eclipse. Walaupun begitu, Android Studio saat ini masih dalam tahap beta dan belum mempunyai dukungan untuk NDK/Native Development Kit.

Android Studio dianggap sebagai code editor cerdas. Inti dari Android Studio adalah editor kode cerdas mampu code completion dengan cerdas, refactoring, dan analisis code. Editor yang baik membantu Anda menjadi pengembang aplikasi Android lebih produktif.

Android Studio dapat digunakan untuk membangun aplikasi untuk ponsel Android, tablet, Android Wear, Android TV, Android Auto dan Google Glass. Dengan Project Android View baru dan modul dukungan di Android Studio, lebih mudah untuk mengelola proyek aplikasi dan *resource*. Android Studio hadir dengan emulator gambar yang optimal.

Virtual Device Manager yang telah diperbarui dan efisien memberikan profil perangkat yang telah ditetapkan untuk perangkat Android umum. Selain itu, Android Studio dapat membuat beberapa APK untuk aplikasi Android dengan fitur yang berbeda menggunakan proyek yang sama.

Beberapa perbedaan tampilan pada Android Studio dibandingkan dengan Eclipse IDE yakni :

1. Layout

Sama seperti Eclipse, layout di Android dibedakan menjadi 2, yaitu design dan text, secara default pada saat dibuka layout xml di android studio

menggunakan mode design, tapi mode ini bisa diubah ke mode text, sehingga terlihat *source code* dari text XML-nya. Android Studio sudah memiliki Live Preview, jadi tidak perlu bolak-balik ke mode design hanya untuk melihat hasil dari kode XML nya.

2. Struktur Folder

Struktur folder pada Android Studio terdapat folder .idea, app, gradle dan file – file yang ada diroot folder. Folder app/ adalah folder utama pada aplikasi Android. Jika menggunakan library dalam bentuk .jar maka tempatkan difolder libs. Selanjutnya folder src/. src adalah folder dimana tersimpan *source* Java dan layout dalam bentuk XML, secara default Android Studio akan men-*generate* dua folder yaitu, androidtest dan main. Folder androidtest adalah folder khusus UnitTest, jika belum paham, fokus saja pada folder main, karena source dan resource terletak difolder tersebut.

3. Gradle

Yang baru di Android Studio adalah gradle, sebuah featured build automation. Selengkapnya dapat lihat di situs berikut ini http://www.gradle.org/. File Gradle berisi *library* yang digunakan, versi aplikasi, *signed key properties*, lokasi repository dll. File yang akan sering kita ubah adalah file build.gradle yang berada dalam folder app/. File ini berisi pengaturan untuk versi sdk yang di *compile, build version* yang digunakan, nama paket aplikasi, minimal SDK yang digunakan, *versioncode*, *versionname* dan *dependencies* yang digunakan.

System requirements Android Studio untuk Windows :

a. Microsoft® Windows® 10/8/7/Vista (32 or 64-bit)
b. Minimal RAM 2 GB, disarankan RAM 4 GB
c. 400 MB hard disk space
d. Minimal 1 GB for Android SDK, emulator system images, and caches
e. Minimal resolusi layar 1280 x 800
f. Java Development Kit (JDK) 7
g. Pilihan untuk akselerasi emulator : Intel® processor with support for Intel® VT-x, Intel® EM64T (Intel® 64), dan Execute Disable (XD) Bit functionality