BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1. Tinjauan Pustaka

Berikut beberapa penelitian atau karya yang berkaitan dengan pengolahan citra untuk pengukuran luas daun dan pengolahan citra berbasis *Android* :

Mahdi M. Ali, Ahmed Al – Ani, Derek Eamus dan Daniel K.Y. Tan (2012) dalam penelitiannya *A New Image-Processing-Based Technique for Measuring Leaf Dimension* memaparkan bahwa untuk mencari teknik baru dalam pengukuran luas daun dengan memanfaatkan pengolahan citra. Metode yang digunakan adalah dengan menggunakan metode konvensional yaitu metode Non – Digital dan metode Digital sebagai pendekatannya sedangkan untuk metode yang digunakan adalah sebuah *software* yang diintegrasikan dengan alat digital yang ada. Hasilnya menunjukkan perbedaan atau error yang kecil pada pengukuran luas daun.

Sapan Thakker dan Prof. Harsh Kapadia (2015) dalam penelitiannya *Image Processing on Embedded Platfrom Android* memaparkan bahwa pendekatan inovatif terhadap desain, perkembangan, dan implementasi dari pengolahan citra berbasis aplikasi *embedded vision platform*. Dengan menggunakan library OpenCV, pengolahan citra berbasis algoritma dapat diaplikasikan pada perangkat *android*. Dengan menggunakan OpenCV dan *android* yang berbasis sistem *embedded vision* dapat dikembangkan yang mana dapat menggantikan sistem berbasis *machine vision*. *Android* berbasis sistem *embedded vision* mengurangi ukuran dari sistem dan juga solusi biaya efektif untuk industri. Aplikasi *android* yang diberikan pada penelitian ini menampilkan operasi dasar seperti perubahan warna, deteksi tepi, operasi morfologi, dll.

Piyush Chaudhary, Sharda Godara, A.N. Cheeran, dan Anand K. Chaudhari (2012) dalam penelitiannya *Fast and Accurate Method for Leaf Area Measurement* memaparkan bahwa implementasikan pengolahan citra secara sederhana, cepat, dan algoritma yang akurat untuk menghitung luas daun. Citra diambil menggunakan kamera digital dan disimpan dalam format JPEG. Citra RGB diubah warnanya ke dalam ruang warna CIELAB. Citra yang telah diubah

warnanya disegmentasi menggunakan teknik *threshold*. *Threshold* dihitung menggunakan metode OTSU. Lubang di dalam area daun diisi menggunakan *region filling technique*. Jumlah piksel di dalam objek kotak dan area daun dihitung dan luas daun diukur dengan menggunakan statistik jumlah pikselnya. Keakuratan dari algoritmanya diatas 99% yang mana dibuktikan dengan menbandingkan hasil dari algoritma yang digunakan dengan metode hitungan *grid*.

Parmar D. K., Ghodasara Y.R., Patel K. P., Patel K. V., dan Kathiriya D. R. (2015) dalam penelitiannya *Estimation of Plant Leaf Area using Java Image Processing Techniques* memaparkan bahwa luas daun merepresentasikan jumlah dari material daun dalam ekosistem dan mengendalikan hubungan antara biosfer dan atmosfer melalui berbagai macam proses seperti fotosintesis, respirasi, transpirasi dan penangkapan air hujan. Hal tersebut juga membantu parameter dalam mengevaluasi kerusakan yang disebabkan penyakit daun, kekurangan mikronutrisi, air dan tekanan lingkungan, kebutuhan fertilisasi untuk pengelolaan yang efektif dan pemeliharaan. Produksi ketelitian pertanian mengadopsi metode kecepatan dan keakuratan untuk menghitung luas daun tanaman. Pada saat ini, penghitungan *grid* dan *Leaf Area Meter* (Li3100) digunakan sebagai metode konvensional untuk estimasi luas daun. Metode ini sederhana dalam prinsip dan sangat akurat tetapi cenderung menghabiskan waktu. Permasalahan di atas dapat diatasi dengan *Leaf Area Meter* yang rancang; Aplikasi Pengolahan Citra Berbasis *Java*. Aplikasi tersebut memiliki banyak keuntungan seperti *User Friendly*, cepat, akurat dan *reusable*. Jurnal ini mendiskusikan perbedaan metode estimasi luas daun yang menggunakan Mesin Li 3100, Metode Manual Grafir dan Metode berbasis Software. Hasil menunjukkan bahwa software luas daun memberikan hasil 99.96% sampai 95.31% keakurasiannya untuk daun yang berbeda kecuali Jasud. Diantara format dua citra, .jpg menunjukkan komparasi yang lebih sedikit eror dibanding format .png.

Wang Jingwen dan Liu Hong (2012) dalam penelitiannya *Measurement and Analysis of Plant Leaf Area Based on Image Processing* memaparkan bahwa untuk meningkatkan keakuratan luas daun tanaman, metode baru penghitungan

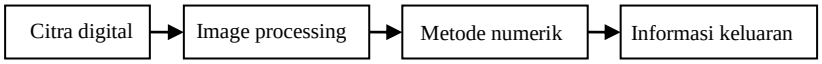
dari luas daun tanaman berbasis dari *Snake Model* diadaptasi dalam studi ini. Dalam jurnal ini, kita mengambil kontur daun dengan peningkatan *Snake Model*, membuat tabel kode berantai 8 arah, menghitung luas daun dengan piksel di kontur daun dan menvisualkan hasilnya secara statistik dengan histogram area. Metode ini membuktikan lebih akurat dari metode pada umumnya, yang mana memiliki potensi yang baik untuk aplikasi praktis.

Hsien Ming Easlon dan Arnold J. Bloom (2014) dalam penelitiannya *Easy Leaf Area : Automated Digital Image Analysis for Rapid and Accurate Measurement of Leaf Area* memaparkan bahwa pengukuran luas daun dari foto digital membutuhkan masukan dari pengguna secara signifikan kecuali latar dari gambar di hilangkan. Easy Leaf Area dikembangkan untuk sejumlah proses dari *Arabidopsis* dalam hitungan menit, menghilangkan latar dari gambar daun dan menyimpan hasilnya ke dalam sebuah *file* CSV. Easy Leaf Area menggunakan rasio warna dari setiap piksel untuk membedakan daun dan area kalibrasi dari latar gambar dan membandingkan jumlah piksel daun dengan kalibrasi warna merah untuk mengeliminasi kebutuhan jarak kamera dari latarnya atau skala manual dengan penggaris yang mana *software* lainnya membutuhkannya. Leaf Area menaksir dengan software ini gambar yang di ambil menggunakan ponsel kamera lebih akurat dari pada ImageJ yang menggunkan flatbed scanner. Easy Leaf Area menyediakan *easy-to-use* metode untuk pengukuran cepat dari luas daun dan penaksiran non destruktif dari daerah *canopy* dari gambar digital.

2.2 Landasan Teori

2.2.1 Umum

Pengolahan citra (*image processing*) merupakan proses mengolah piksel-piksel yang terdapat didalam citra digital (Ir. Balza Achmad, M.Sc.E dan Kartika Firdausy, S.T.,M.T, 2005). Dengan adanya informasi yang bisa diperoleh dari suatu citra digital dan telah dikembangkannya beberapa metode numerik yang diformulasikan untuk perhitungan integral, maka hal tersebut dapat digunakan untuk tujuan tertentu. Dapat digambarkan pada blok diagram berikut ini:



Gambar 2.1 Pemrosesan citra digital

2.2.2 Model Citra

Citra ada dua macam, yaitu citra kontinyu dan citra diskrit. Citra kontinyu dihasilkan dari sistem optik yang menerima sinyal analog, misalnya mata manusia, dan kamera analog. Citra diskrit dihasilkan melalui proses digitalisasi terhadap citra kontinyu. Beberapa sistem optik dilengkapi dengan fungsi digitalisasi sehingga ia mampu menghasilkan citra diskrit, misalnya kamera digital dan *scanner*. Citra diskrit disebut juga citra digital.

Citra agar dapat diolah dengan komputer, maka harus direpresentasikan secara numerik dengan nilai-nilai diskrit. Representasi citra dari fungsi kontinyu menjadi diskrit disebut digitalisasi. Citra yang dihasilkan inilah yang disebut citra digital (*digital image*). Pada umumnya citra digital berbentuk empat persegi panjang, dan dimensi ukurannya dinyatakan sebagai *lebar x tinggi* (Rinaldi Munir, 2004).

Citra digital yang berukuran N *x* M lazim dinyatakan dengan matrik yang berukuran N baris dan M kolom sebagai berikut:

$$f(x,y)=\begin{pmatrix} f(0,0) & f(0,1) & \dots & f(0,M) \\ f(1,0) & f(1,1) & \dots & f(1,M) \\ \dots & \dots & \dots & \dots \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{pmatrix}$$

Gambar 2.2 Matrik Citra Digital

2.2.3 Citra Digital

1. Pengertian Citra digital

   Citra digital adalah gambar pada bidang dua dimensi, merupakan representasi keadaan visual yang disimpan secara elektronik dengan bit data

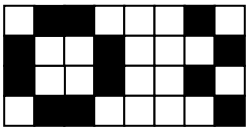
yang merepresentasikan warna. Seringkali citra digital mengalami penurunan mutu (degradasi), misalnya mengandung cacat atau derau (*noise*) sehingga sulit diinterpretasi karena informasi yang disampaikan oleh citra tersebut menjadi berkurang. Agar citra yang mengalami gangguan mudah diinterpretasi baik oleh manusia maupun mesin, maka citra digital perlu dimanipulasi menjadi citra digital lain yang kualitasnya lebih baik atau disebut pengolahan citra (*image processing*).

2. Format Citra digital

Format citra digital yang banyak dipakai adalah:

1. Citra Biner

Pada citra biner, setiap titik bernilai 0 atau 1, masing-masing merepresentasikan warna tertentu. Contoh yang paling lazim: warna hitam bernilai 0 dan warna putih bernilai 1. Setiap titik pada citra hanya membutuhkan 1 bit, sehingga seitap *byte* dapat menampung informasi 8 titik. Gambar berikut menunjukkan contoh representasi citra biner kedalam data digital.



Gambar 2.3 Citra biner dan representasinya dalam data digital

2. Citra skala keabuan (*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 dan nilai maksimumnya bergantung pada jumlah bit yang digunakan. Contohnya untuk skala keabuan 4 bit, maka jumlah kemungkinan nilainya adalah $2^4 = 16$, dan nilai maksimumnya adalah $2^4-1 = 15$; sedangkan untuk skala keabuan 8 bit, maka jumlah kemungkinan nilainya adalah $2^8 = 256$, dan nilai maksimumnya adalah $2^8-1 = 255$.

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

3. Citra warna (*true color*)

Pada citra warna, setiap titik mempunyai warna yang spesifik yang merupakan kombinasi dari 3 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). Jumlah kombinasi warna yang mungkin untuk format citra ini adalah 2^{24} atau lebih dari 16 juta warna, dengan demikian bisa dianggap mencakup semua warna yang ada, inilah sebabnya format ini dinamakan *true color*.

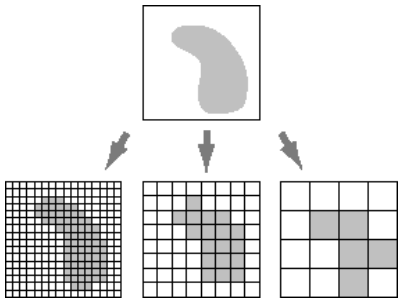
4. Citra warna berindeks

Pada format ini, informasi setiap titik merupakan indeks dari suatu tabel yang berisi informasi warna yang tersedia, yang disebut palet warna (*color map*). Jumah bit yang dibutuhkan oleh setiap titik pada citra bergantung pada jumlah warna yang tersedia dalam palet warna. Sebagai contoh, untuk palet berukuran 16 warna, setiap titik membutuhkan 4 bit; dan untuk palet berukuran 256 warna, setiap titik membutuhkan 8 bit atau 1 *byte*. Palet warna merupakan bagian dari cira warna berindeks, sehingga pada saat menyimpan citra ini kedalam *file*, informasi palet warna juga harus disertakan (Ir. Balza Achmad, M.Sc.E dan Kartika Firdausy, S.t.,MT, 2005).

2.2.4 Piksel

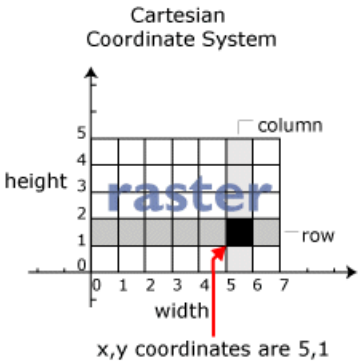
Piksel adalah unsur citra atau representasi sebuah titik terkecil dalam sebuah citra digital yang dihitung perinci. Piksel berasal dari akronim bahasa inggris yaitu *Picture Element* yang disingkat menjadi *Pixel*.

Dimensi dari setiap piksel dapat ditentukan ukurannya sesuai dengan kebutuhan. Ukuran piksel menentukan bagaimana kasar atau halusnya citra yang akan di representasikan. Semakin banyak piksel yang digunakan, maka akan semakin halus atau lebih detail citra yang ditampilkan dan akan berpengaruh terhadap penyimpanan dan kecepatan proses. Apabila semakin sedikit piksel yang digunakan, maka akan tejadi kehilangan informasi atau kehalusan citra akan terlihat lebih kasar.



Gambar 2.4 Representasi citra digital dalam berbagai macam ukuran piksel

Lokasi dalam setiap piksel di definisikan dalam bentuk baris dan kolom dimana didalamnya terdapat informasi mengenai posisi piksel. Apabila piksel memuat sistem koordinat kartesian, maka setiap baris merupakan paralel dengan sumbu x (*x-axis*), dan kolom paralel dengan sumbu y (*y-axis*).



Gambar 2.5 Lokasi dalam Setiap Piksel

Luas suatu citra digital direpresentasikan dalam setiap piksel dengan lebar dan panjang yang sama. Sebagai contoh : 92 x 92 piksel, berarti 92 titik pada

bidang horizontal dan 92 titik pada bidang vertikal, atau sama dengan 8464 piksel persegi (www.ilmukomputer.com).

DPI (*dots per inch*) menunjukkan banyaknya titik per inchi, merupakan ukuran penskalaan resolusi citra yang digunakan sebagai patokan untuk kualitas pencetakan suatu citra digital dari scanner atau printer. Tiap *dots* dari *scanner* akan menjadi piksel pada citra digital. *Scanner* adalah alat yang digunakan untuk membuat citra digital dari sumber asli, sehingga diperlukan penskalaan resolusi citra sebagai patokan untuk kualitas pencetakan atau sering disebut dpi (*dots per inch*). Formula hubungannya adalah sebagai berikut:

Detail akurasi (dpi) = dimensi citra digital (piksel) ÷ dimensi asli (*inch*)

(http://fred.dsimprove.be).

Piksel tidak memiliki ukuran dalam sebuah komputer dan tidak juga memiliki ukuran digital. Komputer mengkonversi piksel kedalam angka, dan susunan angka ini dikenali sebagai citra digital. (http://202.46.4.53/ Suplemen Petunjuk Teknis Pencitraan Digital 20 JIKN-2007.pdf).

2.2.5 Smartphone

Ponsel pintar atau yang lebih populer dengan *smartphone* merupakan sebuah ponsel yang memiliki OS (*Operating System*) yang terdiri dari atas kombinasi dari fitur sebuah PC (*Personal Computer*) dan fitur yang terdapat pada ponsel juga tentunya. Sebagian besar *smartphone* dapat digunakan untuk mengakses internet, dengan layar sentuh, dan dilengkapi dengan kamera.

Mobile Operating System yang digunakan untuk *smartphone* terdiri dari beberapa jenis, yaitu:

1. Android
2. iOS
3. Windows Phone
4. Blackberry
5. Firefox OS
6. Sailfish OS
7. Ubuntu Touch

2.2.5.1 Android

Android adalah sebuah operating sistem yang digunakan untuk menjalankan sebuah *smartphone*. Android pada awalnya dikembangkan oleh Android Inc., yang kemudian dibeli oleh *google* pada tahun 2005 setelah diberikan dukungan secara finansial oleh *google*. Sistem operasi Android tersebut resmi dirilis pada tahun 2007, bersamaan dengan didirikannya sebuah perusahaan Open Handset Alliance, konsorsium dari beberapa perusahaan – perusahaan perangkat keras, perangkat lunak, serta telekomunikasi yang memiliki tujuan untuk memajukan standar terbuka dari perangkat seluler.

Android adalah sistem operasi yang bersifat *open source*, yang berarti, bahwa perangkat lunaknya dapat dimodifikasi dan dikembangkan secara bebas. Saat ini sudah ada banyak sekali komunitas pengembang aplikasi (apps) yang memperluas fungsionalitas perangkat, umumnya ditulis dalam versi kustomisasi bahasa pemrograman *Java*. Pada bulan Oktober 2013, ada lebih dari satu juta aplikasi yang tersedia untuk Android, dan sekitar 50 miliar aplikasi telah diunduh dari *Google Play*, toko aplikasi Android.

Sejak tahun 2008, Android mulai secara bertahap melakukan sejumlah pembaruan atau update untuk meningkatkan kinerja dari sistem operasi tersebut dengan menambahkan fitur baru, memperbaiki *bug* pada versi android yang sebelumnya. Setiap vesi yang dirilis dinamakan secara alfabetis dengan berdasarkan nama sebuah makanan pencuci mulut, seperti cupcake, donut, dan sebagainya. Berikut nama – nama versi android:

a. Android (1.0)
b. Cupcake (1.2 – 1.5)
c. Donut (1.6)
d. Èclair (2.0 – 2.1)
e. Froyo (2.2 – 2.2.3)
f. Gingerbread (2.3 – 2.3.7)
g. Honeycomb (3.0 – 3.2.6)
h. Ice Cream Sandwich (4.0 – 4.0.4)
i. Jelly Bean (4.1 – 4.3)

j. Kit Kat (4.4+)

k. Lollipop (5.0)

l. Marshmallow (6.0)

2.2.5.2 Device Android

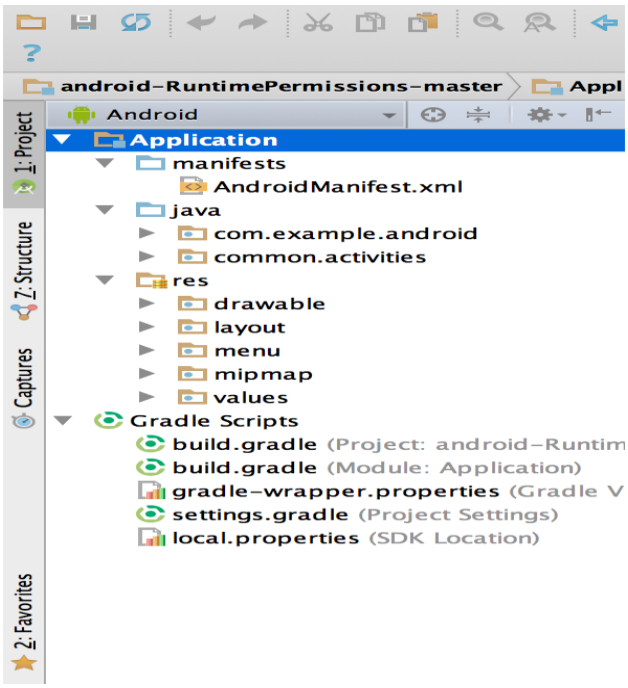
Device android yang digunakan untuk penelitian skripsi adalah ponsel pintar dengan merk *Sony* dengan tipe *ST27i* dan menggunakan OS android dengan versi *JellyBean*.

2.2.5.3 Android Studio

Android Studio adalah lingkungan pengembangan terpadu – *Integrated Development Environment (IDE)* untuk pengembangan aplikasi *Android*, berdasarkan *IntelliJ IDEA*. Selain merupakan editor kode IntelliJ dan alat pengembang yang berdaya guna, *Android Studio* menawarkan fitur lebih banyak untuk meningkatkan produktivitas saat membuat aplikasi *Android*., misalnya :

1. Sistem pembuatan berbasis *Gradle* yang fleksibel
2. Emulator yang cepat dan kaya fitur
3. Lingkungan yang menyatu untuk pengembangan bagi semua perangkat *Android*
4. *Instan Run* untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat *APK* baru
5. Template kode dan integrasi *Github* untuk membuat fitur aplikasi yang sama dan mengimpor kode contoh.
6. Alat penguji dan kerangka kerja yang ekstensif
7. Dukungan C++ dan NDK

Struktur Proyek



Gambar 2.6 Struktur Proyek *Android Studio*

Setiap proyek di *Android Studio* berisi satu atau beberapa modul dengan file kode sumber dan file sumber daya. Jenis – jenis modul mencakup :

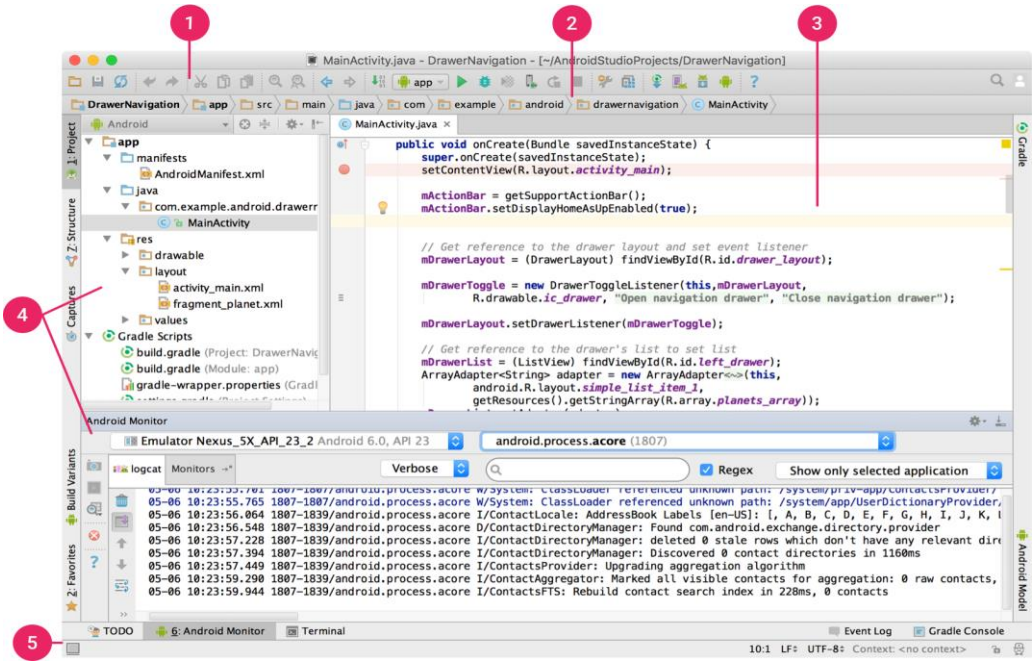
- Modul Aplikasi *Android*
- Modul Perpustakaan
- Modul *Google App Engine*

Secara default, *Android Studio* menampilkan file proyek dalam tampilan proyek *Android* seperti yang ditunjukkan dalam gambar 1. Tampilan ini diatur menurut modul untuk memberi akses cepat ke file sumber kunci proyek.

Semua file versi terlihat di bagian atas di bawah *Gradle Scripts* dan masing - masing modul aplikasi berisi folder berikut :

- *Manifest* : berisi file *AndroidManifest.xml*.
- *Java* : berisi file kode sumber *Java*, termasuk kode pengujian *Junit*.
- *Res* : berisi semua sumber daya bukan kode, seperti tata letak *XML*, string *UI*, dan gambar bitmap.

Antarmuka Pengguna



Gambar 2.7 Antarmuka *Android Studio*

1. Bilah alat : Untuk melakukan berbagai jenis tindakan, termasuk menjalankan aplikasi dan meluncurkan alat *Android*.
2. Bilah Navigasi : Untuk menavigasi di antara proyek dan file yang dibuka untuk pengeditan. Di sini tampilan struktur yang terlihat tampak lebih ringkas dari pada jendela *Project*.
3. Jendela editor : Untuk membuat dan mengubah kode. Bergantung pada jenis file saat ini, editor dapat berubah. Misalnya, ketika melihat file tata letak, editor akan menampilkan *Layout Editor*.
4. Jendela alat : Untuk memberi akses ke tugas – tugas spesifik pengelolaan proyek, penelusuran, kontrol versi, dan banyak lagi.
5. Bilah status : menampilkan status proyek dan *IDE* itu sendiri, serta setiap peringatan pesan.

2.2.6 Pustaka OpenCV

OpenCV adalah singkatan dari *Open Source Computer Vision Library* merupakan sebuah pustaka perangkat lunak yang ditujukan untuk pengolahan citra

dinamis secara *real time* yang dibuat oleh Intel dan saat ini didukung oleh Willow Garage and Itseez. Program ini bebas dan berada dalam naungan sumber terbuka dari lisensi BSD. Pustaka ini merupakan pustaka lintas platform. Program ini didedikasikan sebagian besar untuk pengolah citra secara *real time*. OpenCV pertama kali diluncurkan secara resmi pada tahun 1999 oleh Inter Research sebagai lanjutan dari bagian proyek bertajuk aplikasi intensif berbasis CPU, *real tim ray tracking*, dan tembok penampil 3D. Para kontributor utama dalam proyek ini termasuk mereka yang berkecimpung dalam bidang optimasi di Intel Rusia dan juga tim pustaka performasi intel. Pada awalnya tujuan utama dari proyek OpenCV ini dideskripsikan sebagai berikut :

a. Penelitian penginderaan citra lanjutan tidak hanya melalui kode program terbuka, tetapi juga kode yang telah teroptimasi untuk infrastruktur penginderaan citra
b. Menyebarluaskan ilmu penginderaan citra dengan menyediakan infrastruktur bersama di mana para pengembang dapat menggunakan secara bersama – sama sehingga kode akan tampak lebih mudah dibaca dan ditransfer
c. Membuat aplikasi komersial berbasiskan penginderaan citra, di mana kode yang telah teroptimasi tersedia secara bebas dengan lisensi yang tersedia secara bebas yang tidak mensyaratkan program harus terbuka atau gratis.

OpenCV dioptimasikan dengan 2.500 lebih pustaka algoritma OpenCV menyediakan fitur *Integrated Performance Primitive* (IPP) Intel sehingga bisa lebih mengoptimalisasikan aplikasi *vision* jika menggunakan *processor* Intel.

OpenCV terdiri dari lima pustaka yaitu :

- CV : pustaka untuk algoritma pengolahan citra dan penglihatan
- ML : pustaka untuk pembelajaran mesin
- Highgui : pustaka untuk GUI, gambar, video *input/output*
- CXCORE : pustaka untuk struktur data, mendukung XML, dan fungsi – fungsi grafis

2.2.6.1 Segmentasi Citra

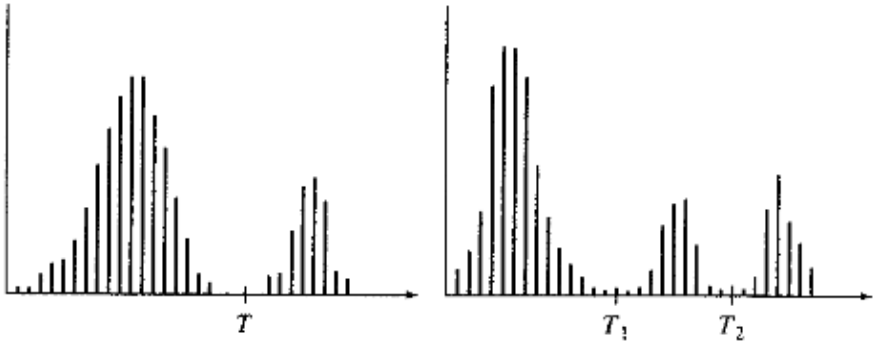
Segmentasi dikaitkan dengan pemilah – pemilah atau pengelompokkan suatu objek di dalam citra ke dalam segmen - segmen (area atau *region*) di mana masing – masing segmen mempunyai sifat – sifat yang berbeda dengan segmen yang menjadi tetangganya. Ini merupakan bagian dasar di dalam analisis *scene* – di dalam menjawab pertanyaan – pertanyaan seperti di mana dan sebagaimana besarkah objek tersebut, di manakah latar belakangnya, berapa banyakkah objek yang berada di dalamnya, berapa banyakkah permukaan yang meliputinya, dan lain – lain. Segmentasi merupakan kebutuhan yang sangat mendasar untuk pengidentifikasian dan pengklasifikasian objek di dalam *scene*, dan memfokuskan tugasnya pada algoritma – algoritma subsekuensial pada bentuk, tekstur, atau warna dari suatu area yang homogen.

2.2.6.2 Thresholding

2.2.6.2.1 Dasar *Thresholding*

Dimisalkan bahwa *histogram* derajat keabuan yang ditampilkan pada gambar 2.8 sesuai dengan gambar, *f(x,y),* yang terdiri dari benda yang terpapar cahaya pada latar yang gelap, yang sedemikian rupa pada piksel objek dan latarnya memiliki pengelompokkan tingkat keabuan pada dua model yang dominan. Satu cara yang pasti untuk mengekstrak objek dari latarnya adalah dengan memilih *threshold T* yang mana memisahkan dua model tersebut. Yang kemudian titik (*x*,*y*) untuk setiap $f(x,y) > T$ disebut sebagai titik objek.

Gambar 2.8(b) menunjukkan contoh yang sedikit lebih umum, dimana tiga model dominan mencirikan *histogram* gambar(untuk contoh, dua tipe dari objek cahaya pada sebuah latar gelap).



(a) (b)

Gambar 2.8 *Histogram* derajat keabuan yang dapat dibagi oleh (a) *single threshold*, dan (b) *multiple threshold*.

Dalam hal ini, *multilevel thresholding* mengklasifikasikan sebuah titik(x,y) sebagai milik salah satu objek jika $T< (x,y) < T$, ke objek yang lain jika $f(x,y) > T$, dan ke latar jika $f(x,y) < T$. *Thresholding* dapat ditampilkan sebagai operasi yang melibatkan fungsi T pada formula :

$$T = T[x, y, p(x, y), f(x, y)]$$

Gambar 2.9 Gambar Formula *Threshold*

Dimana $f(x,y)$ adalah tingkat keabuan dari titik (x,y) dan $p(x,y)$ menandakan beberapa ciri pada titik ini, sebagai contoh, tingkat rata – rata keabuan dari *neighborhood* terpusat pada (x,y). Citra yang sudah di *threshold* g(x,y) didefiniskan sebagai

$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{if } f(x, y) \leq T. \end{cases}$$

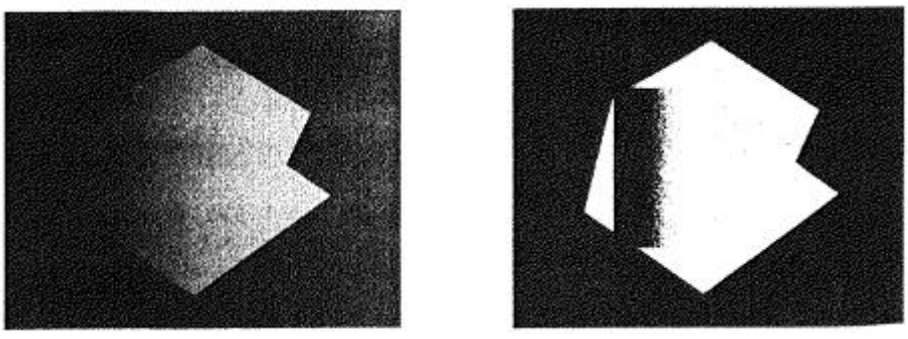
Gambar 2.10 Gambar Formula Citra Setelah *Threshold*

Sehinggapiksel berlabel 1sesuai dengan objek dimana piksel berlabel 0 sesuai dengan latarnya. Ketika T bergantung pada koordinat spasial x dan y, *threshold*nya disebut dengan dinamis atau *adaptive.*

2.2.6.2.2 Adaptive Thresholding

Untuk membagi gambar asli ke dalam subimage dan kemudian memanfaatkan ambang batas yang berbeda untuk segmen masing-masing *subimage*. Isu-isu kunci dalam pendekatan ini adalah bagaimana membagi gambar dan bagaimana memperkirakan ambang batas untuk setiap subimage dihasilkan. Sejak ambang digunakan untuk setiap pixel tergantung pada lokasi piksel dalam hal *subimage*, jenis *thresholding*nya adalah *adapative thresholding*. Dalam hal ini penggambarkan *thresholding* adaptif dengan menggunakan contoh sederhana. Sebuah contoh yang lebih komprehensif diberikan pada bagian berikutnya.

Semua *subimage* yang tidak mengandung batas antara objek dan latar belakang memiliki variasi yang kurang dari 75. Semua *subimage* berisi batas-batas yang memiliki variasi lebih dari 100. Setiap *subimage* dengan variasi lebih besar dari 100 yang tersegmentasi dengan ambang batas dihitung untuk setiap subimage menggunakan algoritma pada dasar *thresholding*. Awal nilai *T* dalam setiap kasus terpilih sebagai titik tengah antara minimum dan tingkat keabua - abuan maksimum dalam *subimage* tersebut. Semua *subimage* dengan variasi kurang dari 100 diperlakukan sebagai satu gambar komposit, yang tersegmentasi menggunakan *threshold* tunggal. Hasil segmentasi menggunakan prosedur ini ditunjukkan pada gambar 2.11.



(a) (b)

Gambar 2.11 Hasil Segmentasi Menggunakan *Adaptive Thresholding*. (a) Gambar Asli. (b) Gambar Setelah Proses *Adaptive Thresholding*.