BAB 2
LANDASAN TEORI

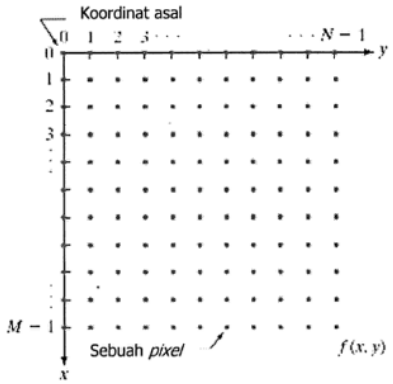
2.1 Citra

Definisi dari Citra adalah suatu representasi (gambaran), kemiripan, atau imitasi dari suatu objek dalam bentuk 2 dimensi (2D) yang mempunyai panjang dan lebar. Citra terdapat dua macam yaitu citra diskrit dan citra kontinu. Citra kontinu dihasilkan dari sistem optik yang menerima sinyal analog, misalnya mata manusia dan kamera analog. Citra diskrit dihasilkan melalui proses digitalisasi terhadap citra kontinu [3].

2.1.1 Pengolahan Citra Digital

Secara umum, pengolahan citra digital menunjuk pada pemrosesan gambar 2 dimensi menggunakan komputer. Dalam konteks yang lebih luas, pengolahan citra digital mengacu pada pemrosesan setiap data 2 dimensi. Citra digital merupakan sebuah larik (*array*) yang berisi nilai riil maupun kompleks yang direpresentasikan dengan deretan bit tertentu.

Suatu citra dapat didefinisikan sebagai fungsi tertentu f(x,y) berukuran M baris dan N kolom, dengan x dan y adalah koordinat spasial, dan amplitudo f dititik koordinat (x,y) dinamakan intensitas atau tingkat keabuan dari citra pada titik tersebut. Apabila nilai x, y, dan nilai amplitudo f secara keseluruhan berhingga (*finite*) dan bernilai diskrit maka dapat dikatakan bahwa citra tersebut adalah citra digital.



Gambar 2.1 Koordinat citra digital

Citra digital dapat dituliskan dalam bentuk matriks sebagai berikut:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{bmatrix} \quad \dots (2.1)$$

2.1.2 Jenis Citra

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

2.1.2.1 Citra Biner

Citra biner adalah citra digital yang hanya memiliki dua kemungkinan nilai *pixel* yaitu hitam dan putih. Citra biner juga sering disebut dengan citra *black and white* atau citra monokrom. Hanya dibutuhkan 1 bit untuk mewakili nilai setiap *pixel* dari citra biner

2.1.2.2 Citra Grayscale

Citra *grayscale* merupakan citra digital yang hanya memiliki satu nilai kanal pada setiap *pixel*nya, dengan kata lain nilai bagian *RED* = *GREEN* = *BLUE*. Nilai tersebut digunakan untuk menunjukkan tingkat intensitas. Warna yang dimiliki

adalah warna dari hitam, keabuan, dan putih. Tingkat keabuan di sini merupakan warna abu dengan berbagai tingkatan dari hitam hingga mendekati putih. Citra *grayscale* memiliki kedalaman warna 8 bit (256 kombinasi warna keabuan).

2.1.2.3 Citra Warna (8 bit)

Setiap *pixel* dari citra wana (8 bit) hanya diwakili oleh 8 bit dengan jumlah maksimum yang dapat digunakan adalah 256 warna. Ada dua jenis citra warna 8 bit. Pertama, citra warna 8 bit dengan menggunakan palet warna 256 dengan setiap paletnya memiliki pemetaan nilai (*colormap*) RGB tertentu. Model ini lebih sering digunakan. Kedua, setiap *pixel* memiliki format 8 bit sebagai berikut.

Tabel 2.1 Konstruksi Citra Warna 8 bit

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
R	R	R	G	G	G	B	B

2.1.2.4 Citra Warna (16 bit)

Citra warna 16 bit (biasanya disebut citra *highcolor*) dengan setiap *pixel*nya diwakili dengan 2 *byte memory* (16 bit). Warna 16 bit memiliki 65.536 warna. Dalam formasi bitnya, nilai merah dan biru mengambil tempat di 5 bit di kanan dan kiri. Komponen hijau memiliki 5 bit ditambah 1 bit ekstra . pemilihan komponen hijau dengan deret bit dikarenakan penglihatan manusia lebih sensitif terhadap warna hijau.

Tabel 2.2 Konstruksi Citra Warna 16 bit

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	R	R	R	R	G	G	G	G	G	G	B	B	B	B	B

2.1.2.5 Citra Warna (24 bit)

Setiap *pixel* dari citra warna 24 bit diwakili dengan 24 bit sehingga total 16.777.216 variasi warna. Variasi ini sudah lebih dari cukup untuk memvisualisasikan seluruh warna yang dapat dilihat penglihatan manusia. Penglihatan manusia dipercaya hanya dapat membedakan hingga 10 juta warna saja.

Setiap poin informasi *pixel* (RGB) disimpan ke dalam 1 *byte* data. 8 bit pertama menyimpan nilai biru, kemudian diikuti dengan nilai hijau pada 8 bit kedua dan pada 8 bit terakhir merupakan warna merah.

2.1.3 Elemen Citra

Citra dibentuk dari beberapa elemen. Elemen dasar tersebut di manipulasi dalam pengolahan citra, elemen tersebut adalah [3]:

1. Kecerahan

   Kecerahan atau intensitas cahaya. Kecerahan pada sebuah titik (*pixel*) di dalam citra bukanlah intensitas yang riil, tetapi sebenarnya adalah intensitas rata-rata dari suatu area yang melingkupinya.

2. Kontras

   Kontras menyatakan sebaran terang dan gelap di dalam sebuah gambar. Citra dengan kontras rendah dicirikan oleh sebagian besar komposisi citranya adalah terang atau sebagian besar gelap. Pada citra dengan kontras yang baik, komposisi gelap dan terang tersebar secara merata.

3. Kontur

   Kontur adalah keadaan yang ditimbulkan oleh perubahan intensitas pada *pixel* yang bertetangga. Karena adanya perubahan intensitas, mata manusia dapat mendeteksi tepi objek di dalam citra.

4. Warna

   Warna adalah persepsi yang dirasakan oleh sistem visual manusia terhadap panjang gelombang cahaya yang dipantulkan oleh objek. Setiap warna mempunyai panjang gelombang. Warna yang diterima

oleh mata merupakan hasil kombinasi cahaya dengan panjang gelombang berbeda. Kombinasi warna yang memberikan rentang warna yang paling lebar adalah *Red* (R), *Green* (G), *Blue* (B).

5. Bentuk

   Bentuk adalah properti intrinsik dari objek tiga dimensi, dengan pengertian bahwa bentuk merupakan properti intrinsik utama untuk sistem visual manusia. Pada umumnya citra yang dibentuk oleh mata merupakan citra dwimatra (dua dimensi), sedangkan objek yang dilihat umumnya berbentuk trimatra (tiga dimensi). Informasi bentuk objek dapat diekstraksi dari citra pada permulaan pra-pengolahan dan segmentasi citra.

6. Tekstur

   Tekstur diartikan sebagai distribusi spasial dari derajat keabuan di dalam sekumpulan *pixel-pixel* yang bertetangga. Jadi tekstur tidak dapat didefinisikan untuk sebuah *pixel*. Sistem visual manusia menerima informasi citra sebagai suatu kesatuan. Resolusi citra yang diamati ditentukan oleh skala dimana tekstur tersebut dipersepsi.

2.2 Pengertian Robot

Kata robot pertama kali diperkenalkan oleh Karel Capek pada sebuah pentas sandiwara yang diadakan pada tahun 1921 dalam pertunjukkannya *R.U.R. (Rossum's Universal Robots)*. Kata robot itu sendiri berasal dari kata *robota* dalam bahasa Ceko yang berarti budak, pekerja, atau kuli. Pengertian lain dari robot adalah suatu alat yang terdiri dari sistem mekanik, sistem elektronik, dan sistem kendali yang berperilaku seperti manusia yang memudahkan pekerjaan manusia. Dari berbagai definisi yang ada, definisi paling dapat diterima adalah dari "Robot Institute of America", yang mendefinisikan robot sebagai manipulator yang berfungsi jamak yang dapat diprogram ulang dan dirancang untuk memindahkan benda kerja, komponen-komponen, peralatan atau perangkat khusus dengan berbagai macam tugas [4].

2.3 Klasifikasi Umum Robot

Secara umum robot dapat diklasifikasikan berdasarkan penggunaan aktuator, berdasarkan kebutuhan akan operator robot, dan berdasarkan kegunaannya [5].

2.3.1 Klasifikasi Robot Berdasarkan Penggunaan Aktuator

Dalam klarifikasi ini terdapat robot manipulator dan *robot mobile*. Jenis *robot mobile* ada 2 (dua) yaitu robot beroda dan robot berkaki [5].

2.3.1.1 Robot Manipulator

Robot ini hanya memiliki satu tangan seperti tangan manusia yang fungsinya untuk memegang atau memindahkan barang. Robot manipulator merupakan sebuah rangkaian benda kaku (*rigid bodies*) terbuka yang terdiri atas sendi dan terhubung dengan *link* dimana setiap posisi sendi ditentukan dengan variabel tunggal sehingga jumlah sendi sama dengan nilai derajat kebebasan. Robot manipulator yang sering dipakai sebagai robot industri pada dasarnya terdiri atas struktur mekanik, penggerak (aktuator), sensor dan sistem kontrol. Dasar (*base*) manipulator sering disebut kerangka dasar (*base frame*) dan ujung dari manipulator biasanya dilengkapi dengan *end-efector* yang salah satu jenisnya adalah *gripper*.

Terdapat banyak jenis sendi. Jenis sendi yang paling banyak digunakan terdiri atas dua macam, yaitu sendi *revolute joint* dan *prismatic joint*. *Revolute joint* sering digunakan sebagai pinggang (*waist*), bahu (*shoulder*) dan siku (*elbow*), dan pergerakan *revolute joint* akan menghasilkan satu derajat kebebasan [5].



Gambar 2.2 Robot *Manipulator*

Bagian dasar manipulator bisa kaku terpasang pada lantai area kerja ataupun terpasang pada rel. Rel berfungsi sebagai *path* atau alur sehingga memungkinkan robot untuk bergerak dari satu lokasi ke lokasi lainnya dalam satu area kerja. Bagian tambahan merupakan perluasan dari bagian dasar, bisa disebut juga lengan atau *arm*. Bagian ujungnya terpasang pada *efector* yang berfungsi untuk mengambil atau mencekam material. Manipulator digerakkan oleh aktuator atau disebut sistem *drive* yang menyebabkan gerakan yang bervariasi dari manipulator. Aktuator bisa menggunakan elektrik, hidrolik ataupun *pneumatik*.

Aktuator elektrik adalah aktuator yang menggunakan listrik sebagai tenaga penggeraknya. Beberapa aktuator elektrik yang biasa digunakan antara lain solenoid, motor DC, motor *stepper*, *servomotor*, dan motor AC. Sedangkan aktuator hidrolik adalah aktuator yang menggunakan fluida dalam bentuk cairan sebagai pemacu gerakannya. Aktuator ini memiliki torsi yang besar tetapi responsnya cukup lambat. Prinsip kerja hidrolik menggunakan perbedaan volume cairan yang ditekan atau dimampatkan untuk membangkitkan tekanan pada piston.

Aktuator *pneumatik* adalah aktuator yang menggunakan udara sebagai pemicu gerakannya. Sifatnya sukar dikendalikan tetapi memiliki respons yang lebih cepat. Prinsip kerja *pneumatik* menggunakan perbedaan volume udara yang ditekan atau dimampatkan untuk membangkitkan tekanan pada piston [5].

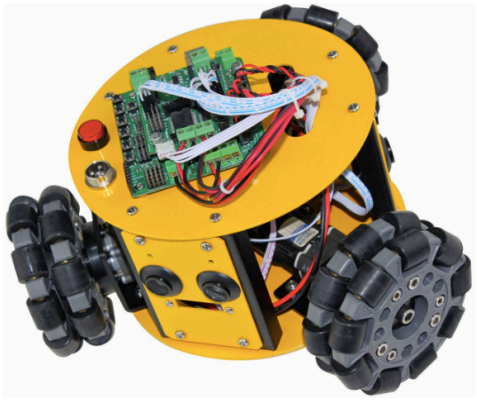
2.3.1.2 Mobile Robot

Mobile robot adalah konstruksi robot yang ciri khasnya adalah mempunyai aktuator untuk menggerakkan keseluruhan badan robot tersebut, sehingga robot tersebut dapat melakukan perpindahan posisi dari satu titik ke titik yang lain. Secara umum *robot mobile* dibagi menjadi dua yaitu robot beroda dan robot berkaki [5].

a. Robot Beroda

   Robot beroda ini sangat disukai bagi orang yang mulai mempelajari robot. Hal ini karena membuat robot ini tidak memerlukan kerja fisik yang berat. Roda sering kali dipilih, karena memberikan *traction* yang bagus, mudah diperoleh dan dipakai, dan juga mudah untuk memasangnya pada robot. Untuk dapat membuat sebuah robot beroda

minimal diperlukan pengetahuan tentang mikrokontroler dan sensor-sensor elektronik. Base robot beroda dapat dengan mudah dibuat dengan menggunakan *polywood* atau tripleks, akrilik dan menggunakan logam. Robot ini dapat dibuat sebagai pengikut garis (*line follower*) atau pengikut dinding (*wall follower*) ataupun pengikut cahaya.



Gambar 2.3 Robot Beroda

b. Robot Berkaki

Untuk melewati medan yang tidak menentu, robot berkaki sangat baik digunakan karena robot berkaki lebih mudah beradaptasi bila dibandingkan menggunakan roda seperti untuk menaiki tangga. Bentuk robot berkaki banyak diadopsi dari bentuk makhluk hidup atau manusia.



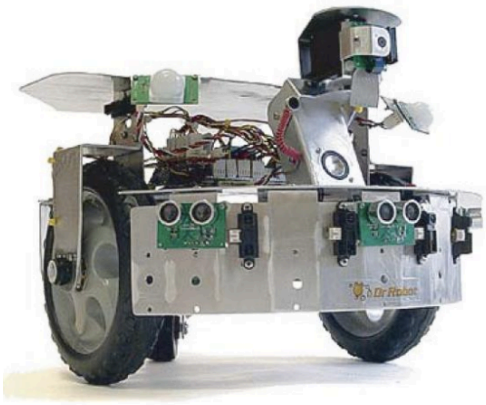
Gambar 2.4 Robot Berkaki

2.3.2 Klasifikasi Robot Berdasarkan Kebutuhan akan Operator Robot

Klarifikasi ini terdiri atas *autonomous* robot, *teleoperated* robot, dan *semi autonomous* robot. Dari ketiga jenis robot tersebut dapat dijelaskan sebagai berikut .

2.3.2.1 Autonomous Robot

Robot *autonomous* adalah robot yang dapat melakukan tugas-tugas yang diinginkan dalam lingkungan yang tidak terstruktur tanpa bimbingan manusia secara terus menerus. Banyak jenis robot memiliki beberapa tingkat otonomi. Tingkatan otonomi sangat dibutuhkan dalam bidang-bidang, seperti eksplorasi ruang angkasa, membersihkan lantai, memotong rumput, dan pengolahan air limbah [5].



Gambar 2.5 *Autonomous Robot*

2.3.2.2 Teleoperated Robot

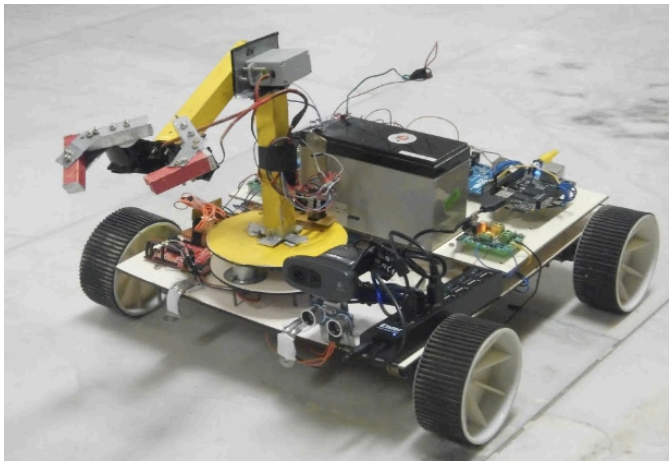
Robot ini dalam pengoperasian mesinnya dikendalikan dari kejauhan. Hal ini mirip dalam arti untuk frase "*remote control*", dikendalikan oleh operator (manusia) menggunakan *remote control* [5].



Gambar 2.6 ***Teleoperated Robot***

2.3.2.3 Semi Autonomous Robot

Robot semi *autonomous* adalah robot yang pengendaliannya secara otomatis untuk pengerjaan tertentu dan pengendalian jarak jauh dengan menggunakan *remote control*. Hal ini bertujuan robot dapat melewati lingkungan atau lintasan yang berbahaya bagi manusia [5].



Gambar 2.7 ***Semi Autonomous Robot***

2.4 Komponen Robot

Dalam pembuatan robot, diperlukan beberapa komponen yang mendukung kerja suatu robot. Terdapat dua komponen dasar yaitu Perangkat Keras (*Hardware*) dan Perangkat Lunak (*Software*).

2.4.1 Perangkat Keras

2.4.1.1 Catu Daya

Catu daya mempunya peran yang sangat penting dalam perancangan sebuah robot. Tanpa bagian ini robot tidak akan berfungsi. Begitu juga bila pemilihan catu daya tidak tepat, maka robot tidak akan bisa berjalan dengan baik.

bekerja dengan baik. Penentuan sistem catu daya yang akan digunakan ditentukan oleh banyak faktor, di antaranya:

1. Tegangan

   Setiap catu daya tidak memiliki tegangan yang sama. Hal ini akan berpengaruh terhadap desain catu daya. Tegangan tertinggi dari salah satu catu daya akan menentukan nilai tegangan catu daya.

2. Arus

   Setiap catu daya satuan Ah (*Ampere Hour*). Semakin besar Ah, semakin lama daya tahan catu daya bila digunakan pada beban yang sama.

3. Teknologi Catu Daya

   Baterai isi ulang ada yang dapat diisi hanya apabila benar-benar kosong, dan ada pula yang dapat diisi ulang kapan saja tanpa harus menunggu baterai tersebut benar-benar kosong.



Gambar 2.8 Catu daya *Lithium polymer*

2.4.1.2 Motor DC

Motor DC adalah jenis motor listrik yang bekerja menggunakan sumber tegangan DC. Motor DC atau motor arus searah sebagaimana namanya, menggunakan arus langsung dan tidak langsung/*direct-unidirectional*. Motor DC digunakan pada penggunaan khusus dimana diperlukan penyalaan *torque* yang tinggi atau percepatan yang tetap untuk kisaran kecepatan yang luas.

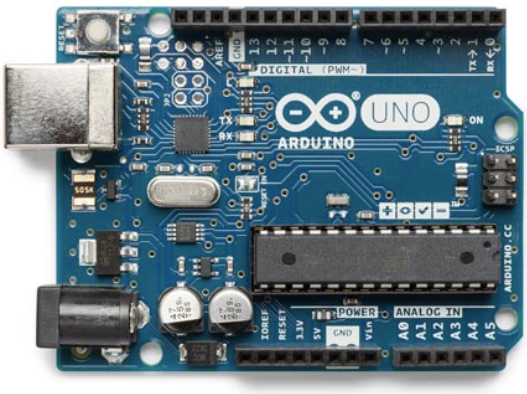


Gambar 2.9 Motor DC

2.4.1.3 Arduino

Arduino adalah salah satu mikrokontroler *single-board* yang bersifat *open source*. Arduino dikembangkan oleh sebuah tim yang beranggotakan orang-orang dari berbagai belahan dunia. Anggota inti dari tim ini adalah Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, David Mellis, dan Nicholas Zambetti [6]. Arduino dirancang agar bisa memudahkan penggunaannya pembuatan *prototype* maupun penggunaan elektronik dalam berbagai bidang.

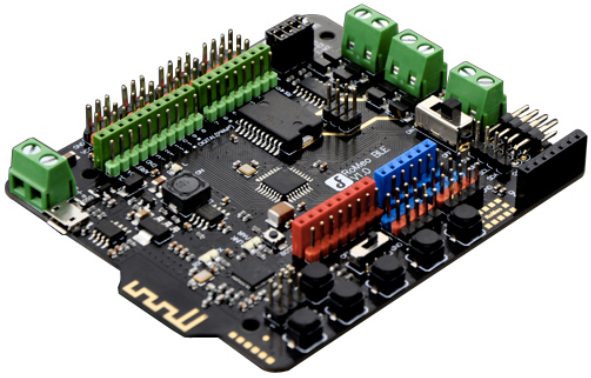
Pada *board* Arduino terdapat komponen utama yaitu sebuah *chip* mikrokontroler dengan jenis AVR. Mikrokontroler itu sendiri adalah *chip* atau IC (*integrated circuit*) yang bisa diprogram langsung melalui komputer dari Arduino IDE. Tujuan menanamkan program pada mikrokontroler adalah agar rangkaian elektronik dapat membaca *input*, memproses *input* tersebut dan kemudian menghasilkan *output* seperti yang diinginkan.



Gambar 2.10 Arduino UNO *Board*

2.4.1.4 DF Robot RoMeo BLE

DF Robot RoMeo BLE adalah *All-in-One board* yang berbasis Arduino yang dirancang untuk keperluan robotika ataupun. Papan mikrokontroler ini, selain memiliki fitur-fitur seperti yang dimiliki pada papan Arduino juga dilengkapi dengan *driver* motor DC L293D dan modul *Bluetooth* 4.0 [7].



Gambar 2.11 DF Robot RoMeo BLE

2.4.1.5 Driver Motor DC L293D

Driver motor DC adalah IC (*Integrated Circuit*) yang memiliki fungsi umum sebagai penggerak motor DC. Sebuah motor dapat diatur kecepatan dan arah putarannya dengan metode PWM (*Pulse Width Modulation*) yang secara umum adalah sebuah cara untuk memanipulasi lebar sinyal yang dinyatakan dengan pulsa dalam satu periode. Dalam *driver* motor DC, PWM digunakan untuk mengatur tegangan listrik yang akan dialirkan ke motor DC. Salah satu *driver* motor yang

tersedia di pasaran yaitu L293D yang di dalamnya terdapat rangkaian *H-Bridge* yang dapat mengatur arah putaran motor DC. Sehingga IC ini dapat menggerakkan motor DC agar robot dapat bergerak maju-mundur.

2.4.1.6 Bluetooth 4.0

Bluetooth adalah sebuah teknologi nirkabel dengan gelombang radio yang bekerja pada frekuensi 2.4 GHz. *Bluetooth* 4.0 merupakan teknologi *Bluetooth* versi terbaru dengan berbagai keunggulan yang salah satunya adalah hemat energi. *Bluetooth* 4.0 memiliki kemampuan dalam penggunaan daya yang minimum sehingga perangkat dapat berfungsi dalam jangka waktu yang sangat lama. *Bluetooth* 4.0 dihadirkan dalam rangka memenuhi tantangan teknologi tanpa kabel pada perangkat energi rendah yaitu arloji, pedometer, alat kesehatan, maupun hiburan. Dibandingkan dengan generasi sebelumnya, *Bluetooth* 4.0 memiliki tingkat kecepatan transfer data sekitar 1 Mbps dan mampu menjangkau lebih dari 100 meter.

2.4.2 Perangkat Lunak

2.4.2.1 Arduino IDE

Arduino IDE adalah sebuah editor yang digunakan untuk menulis program, mengompilasi ke mikrokontroler. Arduino IDE ini dikembangkan khususnya untuk mengembangkan program yang akan dimuat pada papan mikrokontroler berbasis Arduino. Bahasa pemrograman yang digunakan yaitu C/C++ [8].



Gambar 2.12 Tampilan jendela *editor* Arduino IDE

2.5 Kecerdasan Buatan

Kecerdasan buatan berasal dari bahasa Inggris "*Artificial Intelligence*" atau disingkat AI, yaitu *intelligence* adalah kata sifat yang berarti cerdas, sedangkan *artificial* artinya buatan. Kecerdasan yang dimaksud di sini merajuk pada mesin yang mampu berpikir, menimbang tindakan yang diambil, dan mampu mengambil keputusan seperti yang dilakukan oleh manusia.

Menurut Alan Turing, ahli matematika berkebangsaan Inggris yang dijuluki bapak komputer modern dan pembongkar sandi Nazi dalam era Perang Dunia II 1950, menetapkan definisi *Artificial Intelligence*: "Jika komputer tidak dapat dibedakan dengan manusia saat berbicara melalui terminal komputer, maka bisa dikatakan komputer itu cerdas, mempunyai kecerdasan".

Menurut John Carthy dari Stanford mendefinisikan kecerdasan sebagai "Kemampuan untuk mencapai sukses dalam menyelesaikan suatu permasalahan".

Menurut Herbert Alexander Simon (15 Juni 1916-9 Februari 2001): "Kecerdasan buatan (*artificial intelligence*) merupakan kawasan penelitian, aplikasi, dan instruksi yang terkait dengan pemrograman komputer untuk melakukan sesuatu hal yang dalam pandangan manusia adalah cerdas".

Menurut Rich and Knight (1991): “ Kecerdasan buatan (AI) merupakan sebuah studi tentang bagaimana membuat komputer melakukan hal-hal yang pada saat ini dapat dilakukan lebih baik oleh manusia”

2.5.1 Lingkup Utama Kecerdasan Buatan

Lingkup utama kecerdasan buatan antara lain [9]:

1. Sistem pakar

   Komputer digunakan sebagai saran untuk menyimpan pengetahuan para pakar. Dengan demikian komputer akan memiliki keahlian untuk menyelesaikan masalah dengan meniru keahlian yang dimiliki para pakar.
2. Pengolahan bahasa alami

   Dengan pengolahan bahasa alami ini diharapkan pengguna mampu berkomunikasi dengan komputer dengan menggunakan bahasa sehari-hari.
3. Pengenalan ucapan.

   Melalui pengenalan ucapan, diharapkan manusia mampu berkomunikasi dengan komputer menggunakan suara.
4. *Computer Vision*

   Menginterpretasikan gambar atau objek-objek tampak melalui komputer.
5. *Intelligent Computer aid Instruction*

   Komputer dapat digunakan sebagai tutor yang dapat melatih dan mengajar

2.5.2 Keuntungan Kecerdasan Buatan

Keuntungan dari kecerdasan buatan yaitu:

1. Kecerdasan buatan lebih bersifat permanen. Kecerdasan alami akan cepat mengalami perubahan. Hal ini dimungkinkan karena sifat manusia yang pelupa. Kecerdasan buatan tidak akan berubah sepanjang sistem komputer dan program tidak mengubahnya.

2. Kecerdasan buatan lebih mudah diduplikasi dan disebarkan. Mentransfer pengetahuan manusia dari satu orang ke orang lain butuh proses dan waktu lama. Di samping itu suatu keahlian tidak akan pernah bisa diduplikasi secara lengkap. Sedangkan jika pengetahuan terletak pada suatu sistem komputer, pengetahuan tersebut dapat ditransfer atau disalin dengan mudah dan cepat dari satu komputer ke komputer lain.
3. Kecerdasan buatan lebih murah dibanding dengan kecerdasan alami. Menyediakan layanan komputer akan lebih mudah dan lebih murah dibanding dengan harus mendatangkan seseorang untuk mengerjakan sejumlah pekerjaan dalam jangka waktu yang sangat lama.
4. Kecerdasan buatan bersifat konsisten. Hal ini disebabkan karena kecerdasan buatan adalah bagian dari teknologi komputer. Sedangkan kecerdasan alami senantiasa berubah-ubah.
5. Kecerdasan buatan dapat didokumentasikan. Keputusan yang dibuat komputer dapat didokumentasikan dengan mudah dengan melacak setiap aktivitas dari sistem tersebut. Kecerdasan alami sangat sulit untuk direproduksi.
6. Kecerdasan buatan dapat mengerjakan pekerjaan lebih cepat dibanding dengan kecerdasan alami.
7. Kecerdasan buatan dapat mengerjakan pekerjaan lebih baik dibanding dengan kecerdasan alami.

2.5.3 Implementasi Kecerdasan Buatan

Kecerdasan buatan dapat diimplementasikan pada berbagai bidang, di antaranya:

a. Visualisasi Komputer

   Kecerdasan buatan pada bidang visualisasi komputer ini memungkinkan sebuah sistem komputer mengenali gambar sebagai masukan. Contohnya mengenali sebuah pola pada suatu gambar.

b. Pengenalan Suara

Kecerdasan buatan pada pengenalan suara ini dapat mengenali suara manusia. Cara mengenali suara ini dengan mencocokkan pada acuan yang telah diprogram terlebih dahulu. Contohnya perintah komputer dengan menggunakan suara pengguna.

c. Sistem Pakar

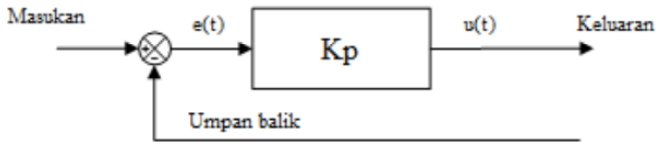
Kecerdasan buatan pada sistem pakar ini memungkinkan sebuah sistem komputer memiliki cara berpikir dan penalaran seorang ahli dalam mengambil keputusan, untuk memecahkan masalah.

d. Robotika dan Sistem Sensor

Kecerdasan buatan pada robotika dan sistem sensor ini memungkinkan sebuah sistem robot memiliki kecerdasan buatan dalam pergerakan robot.

2.6 Kendali Proporsional

Kendali proporsional berfungsi untuk memperkuat nilai kesalahan penggerak (nilai *error*). Kendali proporsional akan mengeluarkan nilai keluaran yang berbanding lurus dengan nilai kesalahan (selisih antara nilai aktual dengan nilai yang diinginkan) [10].



Gambar 2.13 Diagram blok Kendali Proporsional

Hubungan antara nilai keluaran kendali proporsional u(t) dengan sinyal *error* e(t) dibuat oleh persamaan:

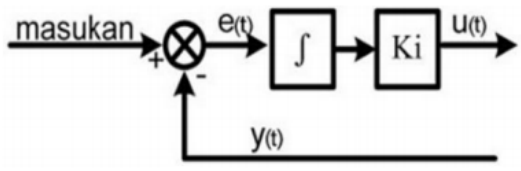
$$u(t) = K_p e(t) \quad \text{... (2.2)}$$

Ciri-ciri kendali proporsional dapat dijelaskan sebagai berikut.

1. Jika nilai K*P* kecil maka kendali proporsional hanya mampu melakukan koreksi kesalahan yang kecil, sehingga akan menghasilkan respons sistem yang lambat.
2. Jika nilai K*P* dinaikkan, respons sistem menunjukkan semakin cepat mencapai keadaan mantapnya.
3. Jika nilai K*P* diperbesar mencapai harga yang berlebihan, akan mengakibatkan sistem bekerja tidak stabil, atau sistem akan berosilasi.

2.7 Kendali Integral

Kendali integral akan menghilangkan kesalahan keadaan tunak (offset) yang dihasilkan oleh kendali proporsional. Kendali integral akan menghasilkan nilai yang berbanding lurus dengan besar dan lamanya *error*. Kendali integral mempercepat perpindahan proses menuju nilai *setpoint* (nilai yang diinginkan) dan menghilangkan *steady-state error* yang muncul pada kendali proporsional. Namun karena kendali integral merespons terhadap nilai *error* yang terakumulasi sebelumnya, maka dapat menyebabkan *overshoot*. Kendali integral memiliki karakteristik seperti halnya sebuah integral. Keluaran pengendali sangat dipengaruhi oleh perubahan yang sebanding dengan nilai sinyal kesalahan. Keluaran pengendali ini merupakan jumlahan yang terus menerus dari perubahan masukannya. Kalau sinyal kesalahan tidak mengalami perubahan, keluaran akan menjaga keadaan seperti sebelum terjadinya perubahan masukan [10].



Gambar 2.14 Diagram blok Kendali Integral

Diagram blok di atas dapat diwujudkan dalam bentuk persamaan yang menunjukkan hubungan antara nilai keluaran kendali integral u(t) dengan sinyal error e(t) sebagai berikut:

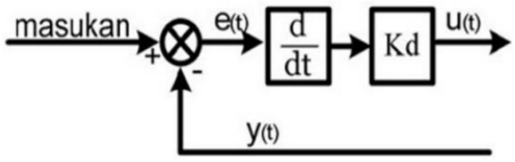
$$u(t) = K_i \int_0^t e(t)\, dt \qquad \text{... (2.3)}$$

Ciri-ciri kendali Integral dijelaskan sebagai berikut.

1. Keluaran pengendali membutuhkan selang waktu tertentu sehingga kendali integral cenderung memperlambat respons.
2. Ketika sinyal kesalahan berharga nol, keluaran pengendali akan bertahan pada nilai sebelumnya.
3. Jika sinyal kesalahan tidak berharga nol, keluaran akan menunjukkan sinyal kesalahan dan nilai K_i.
4. Kendali integral K_i yang berharga besar akan mempercepat hilangnya *offset*. Tetapi semakin besar nilai konstanta K_i akan mengakibatkan peningkatan osilasi dari sinyal keluaran pengendali.

2.8 Kendali Derivatif

Keluaran kendali derivatif memiliki sifat seperti halnya suatu operasi derivatif. Perubahan yang mendadak pada masukan pengendali akan mengakibatkan perubahan yang sangat besar dan cepat juga pada keluaran [10].



Gambar 2.15 Diagram blok Kendali Derivatif

Diagram blok di atas diwujudkan dalam bentuk persamaan yang menunjukkan hubungan antara nilai keluaran kendali integral u(t) dengan sinyal *error* e(t) sebagai berikut:

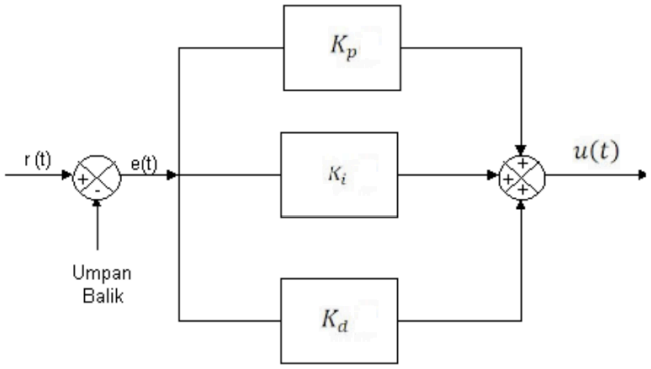
$$u(t) = K_d \frac{de(t)}{dt} \quad \text{... (2.4)}$$

Ciri-ciri kendali derivatif sebagai berikut

1. Pengendali ini tidak dapat menghasilkan keluaran bila tidak ada perubahan pada masukannya (berupa sinyal kesalahan).
2. Jika sinyal kesalahan berubah terhadap waktu, maka keluaran yang dihasilkan pengendali tergantung pada nilai Kd dan laju perubahan sinyal kesalahan.
3. Kendali derivatif mempunyai karakteristik untuk mendahului, sehingga pengendali ini dapat menghasilkan koreksi yang signifikan sebelum pembangkit kesalahan menjadi sangat besar. Jadi kendali derivatif dapat mengantisipasi pembangkit kesalahan, memberikan aksi yang bersifat korektif, dan cenderung meningkatkan stabilitas sistem.

2.9 Kendali PID

Kendali PID merupakan gabungan dari kendali Proporsional, kendali Integral, dan kendali Derivatif. Elemen-elemen kendali P, I dan D masing-masing secara keseluruhan bertujuan untuk mempercepat reaksi sebuah sistem, menghilangkan *offset* dan menghasilkan perubahan awal besar.



Gambar 2.16 Diagram blok Kendali PID

Dari diagram blok di atas dapat diperoleh persamaan

$$u(t) = K_p e(t) + K_i \int_0^t e(t)\, dt + K_d \frac{de(t)}{dt} \quad \text{... (2.5)}$$

2.10 Android

Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis Linux yang mencakup sistem operasi, *middleware* dan aplikasi. Android menyediakan platform yang terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Awalnya, Google Inc. membeli Android Inc. yang merupakan pendatang baru yang membuat peranti lunak untuk ponsel/*smartphone*. Kemudian untuk mengembangkan Android, dibentuklah Open Handset Aliance yang merupakan 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* [11].

Sekitar September 2007 sebuah studi melaporkan bahwa Google mengajukan hak paten aplikasi telepon seluler (akhirnya Google mengenalkan Nexus One, salah satu jenis telepon pintar yang menggunakan Android pada 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 pembaruan berupa perbaikan *bug* dan penambahan fitur baru.

Telepon pertama yang memakai sistem operasi Android adalah HTC Dream, yang dirilis pada 22 Oktober 2008. Pada penghujung tahun 2009 diperkirakan di dunia ini paling sedikit terdapat 18 jenis telepon seluler yang menggunakan Android.

2.10.1 Fundamental Aplikasi

Aplikasi Android ditulis dalam bahasa pemrograman Java. Kode Java dikompilasi bersama dengan data file resource yang dibutuhkan oleh aplikasi, dimana prosesnya di-*package* oleh *tools* yang dinamakan "apt tools" ke dalam paket Android sehingga menghasilkan file dengan ekstensi apk. Empat komponen dasar pada aplikasi Android yaitu :

1. *Activity*

   Suatu Activity akan menyajikan *user interface* (UI) kepada pengguna, sehingga pengguna dapat melakukan interaksi
2. *Service*

   Service tidak memiliki *Graphic User Interface* (GUI), tetapi *service* berjalan secara *background*. *Service* umumnya digunakan untuk menjalankan tugas yang membutuhkan waktu yang lama.
3. *Content Providers*

   Content Providers membuat applikasi dapat mengakses dan mengatur data yang disimpan oleh aplikasi itu sendiri maupun data yang disimpan oleh aplikasi lainnya juga memberikan cara untuk membagikan data yang disimpan oleh aplikasi itu sendiri agar dapat

diakses oleh aplikasi lainnya. sehingga bisa digunakan oleh aplikasi lain

4. *Broadcast Receiver*

   Broadcast Receiver berfungsi menerima dan bereaksi untuk menyampaikan notifikasi atau *broadcast event*.

2.10.2 Versi Android

Beberapa versi android yang telah dikembangkan adalah sebagai berikut [12, 13]:

1. Android Versi 1.5 (*Cupcake*)
2. Android Versi 1.6 (*Donut*)
3. Android Versi 2.1 (*Eclair*)
4. Android Versi 2.2 (*Froyo*)
5. Android Versi 2.3 (*Gingerbread*)
6. Android Versi 3.2 (*Honeycomb*)
7. Android Versi 4.0 (*Ice Cream Sandwich*)
8. Android Versi 4.1 (*Jelly Bean*)
9. Android Versi 4.4 (*KitKat*)
10. Android Versi 5.0 (*Lollipop*)
11. Android Versi 6.0 (*Marshmallow*)
12. Android Versi 7.0 (*Nougat*)

2.10.3 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 di dikeluarkan 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 memberi kesempatan untuk membuat aplikasi yang kita butuhkan yang bukan merupakan aplikasi bawaan *smartphone*.

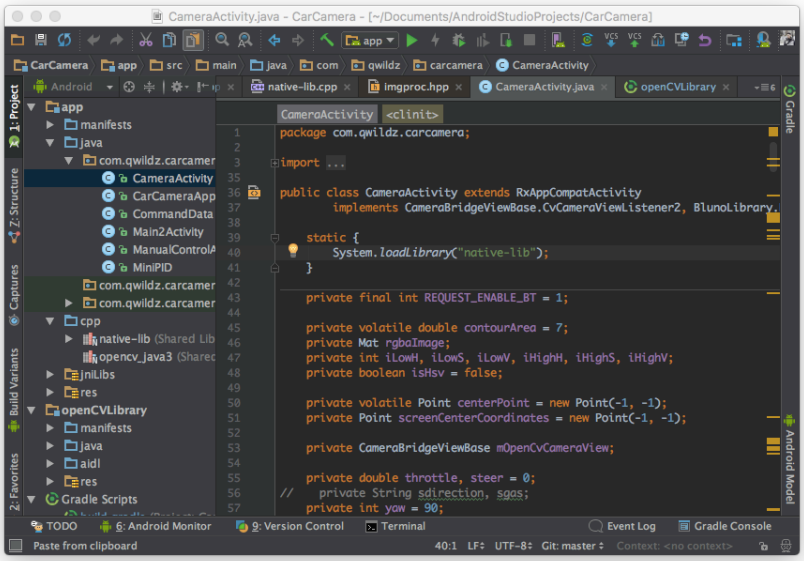
2.10.4 Android NDK

Secara singkat, Android NDK (Native Development Kit) adalah toolset yang dapat digunakan untuk mengimplementasikan bahasa pemrograman *native* seperti C dan C++ kedalam aplikasi Android. Sehingga kode ataupun pustaka yang telah dikembangkan melalui bahasa pemrograman C/C++ dapat diintegrasikan kedalam aplikasi. Andoid NDK menggunakan *Java Native Interface* (JNI) untuk komunikasi antara kode Java dengan C/C++. Dalam kasus performa penggunaan bahasa C/C++ pada aplikasi android dapat meningkatkan performa komputasi dan pemrosesan data yang kompleks.

2.10.5 Android Studio

Android studio adalah lingkungan pengembangan terpadu (IDE) resmi untuk pengembangan platform Android. Hal itu disampaikan pada tanggal 16 Mei 2013 Google I/O,android studio sudah tersedia secara bebas dibawah lisensi Apache 2.0.

Android studio pada awalnya tahap *preview* versi 0.1 yang dipakai pada tanggal 1 mei 2013 dan memasuki tahap beta pada bulan Juni 2014 dan mulai stabil dirilis pada Desember 2014 dengan versi 1.0 ,berdasarkan jetBrains IDEA Intellij Software, Android Studio dirancang khusus untuk pengembangan Android yang tersedia untuk Windows, Mac OS X dan Linux sebagai pengganti Eclipse.



Gambar 2.17 Tampilan jendela Android Studio

2.11 Java

Bahasa Java dikembangkan oleh Sun Microsystem tahun 1991 sebagai bagian dari suatu proyek penelitian untuk mengembangkan software bagi konsumer barang-barang elektronik seperti televisi, VCR, *toaster* dan mesin-mesin lainnya yang dapat dibeli di swalayan. Tujuan penciptaan Java pada waktu itu adalah menjadi suatu program yang berukuran kecil, efisien, dan *portable* di segala jenis *hardware* [14]. Tujuan yang sama ini membuat Java menjadi satu bahasa yang ideal untuk mendistribusikan program-program yang dapat dijalankan melalui www dan juga suatu bahasa pemrograman untuk segala tujuan untuk mengembangkan program-program yang dapat digunakan dengan mudah dan *portable* di berbagai platform yang berbeda.

Saat ini distribusi Java dan kelas pendukungnya dibagi dalam tiga bagian yang masing-masing memiliki konsentrasi tersendiri yaitu:

1. Java 2 Standard Edition (J2SE), untuk aplikasi desktop.
2. Java 2 Enterprise Edition (J2EE), untuk aplikasi server.
3. Java 2 Micro Edition (J2ME), untuk peranti dengan kemampuan terbatas.

2.12 Object Oriented Programming (OOP)

Object Oriented Programming atau pemrograman berorientasi objek atau yang biasa disingkat OOP adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Metode berorientasi objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas. Metode berorientasi objek meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek.

Pada saat ini, metode berorientasi objek banyak dipilih karena metodologi banyak menimbulkan masalah seperti adanya kesulitan pada saat mentransformasi hasil dari satu tahap pengembangan ke tahap berikutnya, misalnya pada metode pendekatan terstruktur, jenis aplikasi yang dikembangkan saat ini berbeda dengan masa lalu. Aplikasi yang dikembangkan saat ini sangat beragam (aplikasi bisnis, *real-time*, *utility*, dan sebagainya) dengan platform yang berbeda-beda, sehingga menimbulkan tuntutan kebutuhan metodologi pengembangan yang dapat mengakomodasi ke semua jenis aplikasi tersebut.

Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut[5]:

a. Meningkatkan produktivitas

   Karena kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk masalah lainnya yang melibatkan objek tersebut (*reusable*).

b. Kecepatan pengembangan

   Karena sistem yang dibangun dengan baik dan benar pada saat analisis dan perancangan akan menyebabkan berkurangnya kesalahan pada saat pengkodean.

c. Kemudahan pemeliharaan

Karena dengan model objek, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering diubah-ubah.

d. Adanya konsistensi

Karena sifat pewarisan dan penggunaan notasi yang sama pada saat analisis, perancangan maupun pengkodean.

e. Meningkatkan kualitas perangkat lunak

Karena adanya pendekatan pengembangan lebih dekat dengan dunia nyata dan adanya konsistensi pada saat pengembangannya, perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

2.13 OpenCV

OpenCV adalah *library* yang bersifat *open-source*, *multi platform* yang memberikan kemudahan dalam melakukan eksperimen ataupun pembuatan aplikasi yang menggunakan computer vision. Library ini memberikan kemudahan karena menyediakan high-level interface untuk mengambil, memproses, serta menampilkan gambar [15]. OpenCV dapat berjalan pada *desktop* (Windows, Linux, MacOS) maupun *mobile* (Android, iOS) dan dikembangkan dengan bahasa pemrogramman C++ dengan tambahan *wrappers* agar dapat digunakan pada bahasa pemrograman Python atau Java.

2.14 Canny Edge Detector

Canny Edge Detector dikembangkan oleh John F. Canny pada tahun 1986 untuk mendeteksi berbagai tepi pada citra. *Canny Edge Detector* menggunakan algoritma multi-tahap untuk mendeteksi berbagai tepi pada gambar.

Algoritma Canny berjalan dalam 5 langkah yang terpisah yaitu:

1. *Smoothing*: Menggabungkan gambar untuk menghilangkan *noise*.

2. *Finding gradient*: Tepian harus ditandai pada gambar memiliki gradien yang besar.
3. *Non-maximum suppression*: Hanya maxima lokal yang harus ditandai sebagai *egde*.
4. *Double thresholding*: Tepian yang berpotensi ditentukan oleh *Thresholding*.
5. *Edge Tracking by hysteresis*: Tepian final ditentukan dengan menekan semua sisi yang tidak terhubung dengan tepian yang sangat kuat.

Tidak dapat dipungkiri bahwa semua gambar yang diambil akan berisi sejumlah *noise*. Untuk mencegah *noise* salah dideteksi sebagai tepian, maka *noise* harus dikurangi. Oleh karena itu pada langkah pertama gambar harus diperhalus dengan menggunakan *Gaussian filter*. Inti dari *Gaussian filter* adalah standar deviasi dengan σ = 1,4 ditentukan pada persamaan:

$$\mathbf{B} = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \quad \text{... (2.6)}$$

Pada persamaan di atas merupakan matriks *Gaussian filter* yang digunakan untuk menghilangkan untuk menghilangkan *noise*.

Algoritma Canny pada dasarnya menemukan titik tepi pada gambar *grayscale* dengan perubahan nilai intensitas yang paling besar, daerah ini ditemukan dengan menentukan gradien gambar. Gradien pada setiap *pixel* citra yang telah diperhalus ditentukan dengan menerapkan operator Sobel. Langkah Kedua adalah memperkirakan gradien pada arah x dan y. Hal tersebut ditunjukkan dalam persamaan:

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \; M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad \text{... (2.7)}$$

Magnitudo gradien (juga dikenal sebagai kekuatan tepi) dapat ditentukan sebagai jarak *Euclidean* yang diukur mengukur dengan menerapkan hukum Pythagoras seperti yang ditunjukkan dalam persamaan (2.8). Yang terkadang disederhanakan dengan menerapkan ukuran jarak Manhattan seperti yang ditunjukkan dalam persamaan (2.9) untuk mengurangi kompleksitas komputasi.

$$|G| = \sqrt{G_x^2} + \sqrt{G_y^2} \quad \text{... (2.8)}$$

$$|G| = |G_x| + |G_y| \quad \text{... (2.9)}$$

Gx dan Gy adalah gradien pada masing-masing arah x dan y. Hal ini tampak jelas dari Gambar 3, bahwa gambar dengan gradien yang besar sering menunjukkan tepian yang cukup jelas. Namun, tepian biasanya luas dan dengan demikian tidak dapat menunjukkan persis di mana tepian yang sebenarnya. Untuk menentukan tepian yang sebenarnya ini, arah tepian harus ditentukan dan disimpan seperti ditunjukkan dalam persamaan (2.10).

$$\theta = arctan\left(\frac{|G_x|}{|G_y|}\right) \quad \text{... (2.10)}$$

Pada langkah ketiga bertujuan untuk mengonversikan tepian yang masih kabur pada gambar hasil *magnitude* gradien hingga menghasilkan tepian yang tajam. Pada dasarnya hal ini dilakukan dengan mempertahankan semua *maxima* lokal dalam gambar gradien dan menghapus segala sesuatu yang lain. Algoritma adalah untuk setiap *pixel* pada gambar gradien adalah sebagai berikut:

1. Putar arah gradien θ ke arah 45˚terdekat, kemudian hubungkan dengan 8 titik tetangga yang terhubung dengannya.
2. Bandingkan nilai *pixel* tepian saat ini dengan nilai *pixel* tepian dalam arah positif dan negatif gradien. Jika arah gradien adalah utara (θ =90 ◦), bandingkan dengan *pixel* ke utara dan selatan.
3. Jika nilai *pixel* tepian saat ini adalah yang terbesar, maka simpan nilai tepian tersebut, namun jika bukan, hapus nilai tersebut.

2.15 Transformasi Hough

Hough transform pertama kali diperkenalkan oleh Paul Hough pada tahun 1962 untuk mendeteksi garis lurus. Hough transform adalah teknik transformasi citra yang dapat digunakan untuk mengisolasi atau dengan kata lain memperoleh fitur dari sebuah citra. Karena tujuan dari sebuah transformasi adalah mendapatkan suatu fitur yang lebih spesifik, *classical hough transform* merupakan teknik yang paling umum digunakan untuk mendeteksi objek yang berbentuk kurva seperti garis, lingkaran, elips dan parabola. Keuntungan utama dari transformasi *Hough* adalah dapat mendeteksi sebuah tepian dengan celah pada batas fitur dan secara relatif tidak dipengaruhi oleh derau atau *noise*.

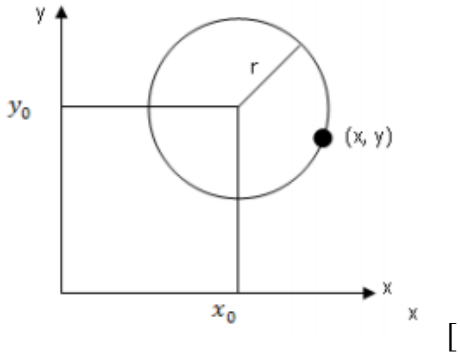
Prinsip kerja dari metode transformasi *Hough* sendiri adalah dengan mencari bentuk geometri yang paling sesuai dengan kumpulan titik pada suatu citra. Jika objek yang dicari adalah berupa lingkaran, maka digunakan transformasi lingkaran *Hough*. Prosedur yang digunakan dalam mendeteksi lingkaran adalah sama dengan transformasi *Hough* pada objek garis, tapi dikerjakan pada ruang dimensi yang lebih kompleks, yaitu dalam parameter ruang 3D dimana x_0 dan y_0 merupakan koordinat pusat lingkaran dan r adalah jari-jari lingkaran seperti persamaan berikut.

$$(x - x_0)^2 + (y - y_0)^2 = r^2 \quad \text{... (2.10)}$$

Sedangkan untuk vektor parameternya adalah:

$$\vec{p} = \begin{bmatrix} x_0 \\ y_0 \\ r \end{bmatrix} \quad \text{... (2.11)}$$

Dan jika diilustrasikan dalam koordinat adalah sebagai berikut :



[

Gambar 2.18 Peta Kartesian Koordinat Titik Tepi

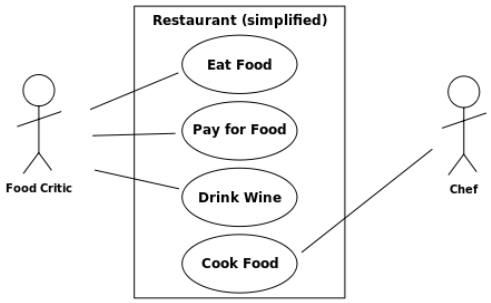
2.16 Unified Modelling Language (UML)

Unified Modelling Language atau yang disingkat *UML* adalah suatu bahasa pemodelan yang digunakan untuk mendokumentasikan perangkat lunak. Dengan menggunakan UML, dapat membuat model untuk semua jenis aplikasi peranti lunak, dimana aplikasi tersebut dapat berjalan pada peranti keras, sistem operasi dan jaringan apa pun, serta ditulis dalam bahasa pemrograman apa pun. Tetapi, karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka UML lebih cocok untuk penulisan peranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C.

2.16.1 Diagram Use Case

Diagram *Use Case* menggambarkan kejadian-kejadian apa saja yang dapat dilakukan oleh aktor dan fungsionalitas apa saja yang terdapat pada sistem yang akan dibangun tanpa melihat bagaimana sistem melakukan fungsionalitas tersebut. Pada diagram *Use Case* juga dijabarkan batasan fungsionalitas yang bisa diakses oleh aktor.

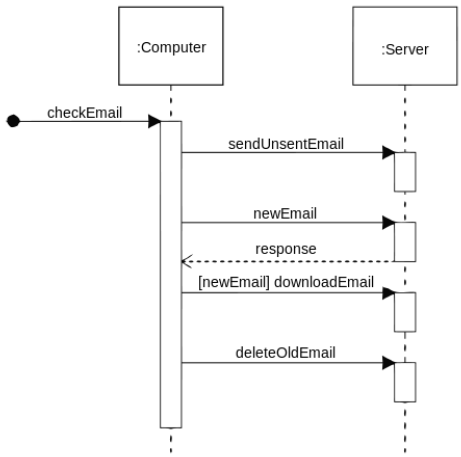
Ada dua hal utama pada *diagram Use Case* yaitu pendefinisian apa yang disebut aktor dan *Use Case*. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem yang akan dibuat. Sedangkan *Use Case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.



Gambar 2.19 Diagram *Use Case*

2.16.2 Diagram Sequence

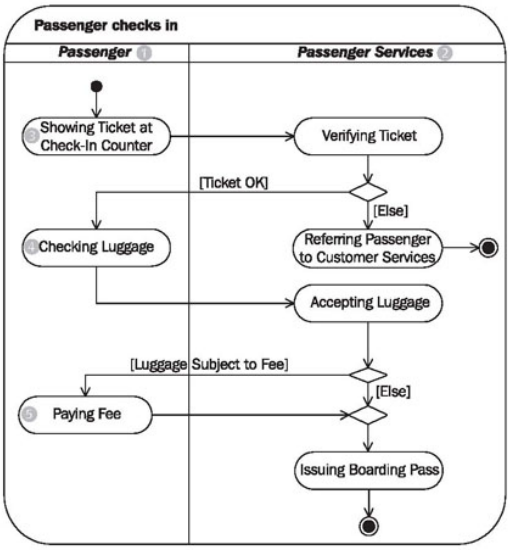
Diagram *Sequence* menggambarkan interaksi antar objek di dalam dan di sekitar sistem yang menekankan pada pengiriman pesan dalam suatu waktu tertentu. Diagram *Sequence* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan dari sebuah *event* untuk menghasilkan keluaran tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan keluaran apa yang dihasilkan



Gambar 2.20 Diagram *Sequence*

2.16.3 Diagram Activity

Diagram *Activity* menggambarkan berbagai alur aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* atau keputusan yang mungkin terjadi, dan bagaimana mereka berakhir. Diagram *Activity* merupakan *state* Diagram khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu Diagram *Activity* tidak menggambarkan *behaviour* internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum

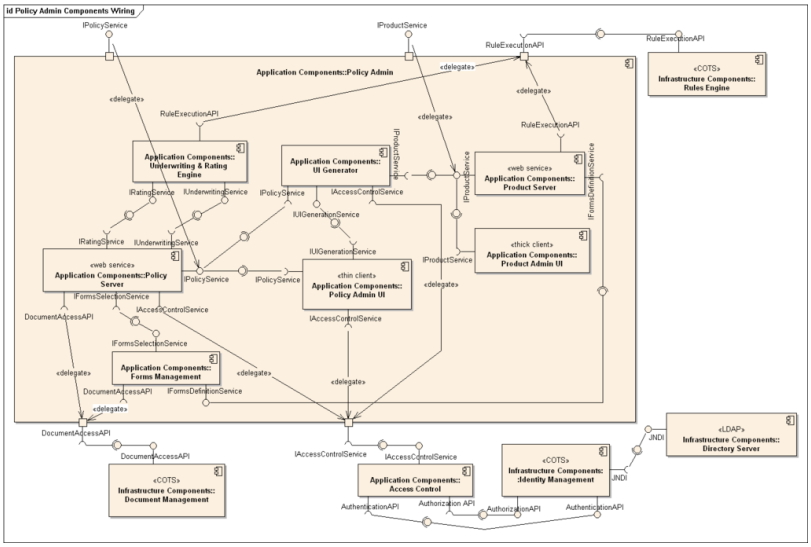


Gambar 2.21 Diagram *Activity*

2.16.4 Diagram Class

Diagram *Class* digunakan untuk menggambarkan keadaan suatu sistem dengan menjelaskan keterhubungan antara suatu *class* dengan *class* yang lain yang terdapat pada sistem. Sebuah *class* terdiri dari nama, atribut dan *method*. Atribut dan *method* dari sebuah *class* atau kelas mempunyai *visibility*. Ada tiga jenis visibility yang digunakan yaitu *private*, *public*, dan *protected*. Setiap kelas pada Diagram *Class* mempunyai hubungan dengan kelas lainnya. Ada beberapa jenis hubungan kelas, yaitu *dependency*, asosiasi, agregasi dan generalisasi.

1. *Dependency* merupakan hubungan terlemah antar kelas. *Dependency* bermakna satu kelas menggunakan atau memiliki pengetahuan terhadap kelas lain, namun hubungannya hanya sementara dan tidak ada batas waktu yang jelas.
2. Asosiasi mempunyai hubungan yang lebih kuat dari hubungan *dependency*. Dimana suatu class tetap berhubungan dengan kelas lain seterusnya, hubungan asosiasi dibagi menjadi dua jenis yaitu *directional* dan *bidirectional*.
3. Agregasi merupakan bentuk hubungan yang mengimplikasikan kepemilikan suatu kelas. Agregasi juga merupakan bentuk yang lebih kuat dari asosiasi. Hubungan agregasi dinyatakan dengan simbol *diamond* pada pemilik kelas dan garis utuh berpanah ke *class* yang dimiliki.
4. Generalisasi merupakan bentuk hubungan antar *class*, dari *class* yang umum dengan *class* yang lebih khusus. Contoh *class* hewan memiliki hubungan generalisasi dengan *class* kucing, karena kucing merupakan *class* khusus dari kelas hewan yang lebih umum.



Gambar 2.22 Diagram *Class*