KECERDASAN BUATAN

Penulis:
Victor Amrizal
Qurrotul Aini

Halaman Moeka

KECERDASAN BUATAN

Penulis:
Victor Amrizal
Qurrotul Aini

Editor: Qurrotul Aini
Desainer Isi: Tim HalamanMoeka.com
Desainer Sampul: Tim HalamanMoeka.com

Jakarta, September 2013

ISBN: 978-602-269-024-5

Diterbitkan:
Halaman Moeka Publishing
Jl. Manggis IV No. 2 Rt 07/04 Tanjungduren Selatan
Grogol Petamburan, Jakarta Barat Telp. 021 5644157
halamanmoeka.blogspot.com | halamanmoeka@gmail.com

MOTO

يَٰمَعۡشَرَ ٱلۡجِنِّ وَٱلۡإِنسِ إِنِ ٱسۡتَطَعۡتُمۡ أَن تَنفُذُواْ مِنۡ أَقۡطَارِ ٱلسَّمَٰوَٰتِ وَٱلۡأَرۡضِ فَٱنفُذُواْۚ لَا تَنفُذُونَ إِلَّا بِسُلۡطَٰنٖ ٣٣

"Hai jama'ah jin dan manusia, jika kamu sanggup menembus (melintasi) penjuru langit dan bumi, maka lintasilah, kamu tidak dapat menembusnya kecuali dengan kekuatan" (QS. Ar-Rahman [55]:33)

يَٰقَوۡمِ لَكُمُ ٱلۡمُلۡكُ ٱلۡيَوۡمَ ظَٰهِرِينَ فِي ٱلۡأَرۡضِ فَمَن يَنصُرُنَا مِنۢ بَأۡسِ ٱللَّهِ إِن جَآءَنَاۚ قَالَ فِرۡعَوۡنُ مَآ أُرِيكُمۡ إِلَّا مَآ أَرَىٰ وَمَآ أَهۡدِيكُمۡ إِلَّا سَبِيلَ ٱلرَّشَادِ ٢٩

"(Musa berkata): "Hai kaumku, untukmulah kerajaan pada hari ini dengan berkuasa di muka bumi. siapakah yang akan menolong kita dari azab Allah jika azab itu menimpa kita!" Fir'aun berkata: "Aku tidak mengemukakan kepadamu, melainkan apa yang aku pandang baik; dan aku tiada menunjukkan kepadamu selain jalan yang benar" (QS. Al-Mu'min [40]:29)

كِتَٰبٌ أَنزَلۡنَٰهُ إِلَيۡكَ مُبَٰرَكٞ لِّيَدَّبَّرُوٓاْ ءَايَٰتِهِۦ وَلِيَتَذَكَّرَ أُوْلُواْ ٱلۡأَلۡبَٰبِ ٢٩

"Ini adalah sebuah kitab yang Kami turunkan kepadamu penuh dengan berkah supaya mereka memperhatikan ayat-ayatNya dan supaya mendapat pelajaran orang-orang yang mempunyai fikiran" (QS. Ash-Shaad [38]:29)

هُدًى وَذِكۡرَىٰ لِأُوْلِي ٱلۡأَلۡبَٰبِ ٥٤

"untuk menjadi petunjuk dan peringatan bagi orang-orang yang berfikir" (QS. Al-Mu'min [40]:54)

Kata Pengantar

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

Segala puji bagi Allah سبحانه وتعالى, Pemilik segala pujian yang selalu memberikan ramat dan nikmat kepada hamba-Nya serta salam dan shalawat selalu tercurah pada Nabi Muhammad ﷺ. Semoga kita tetap menjadi umatnya hingga akhir zaman.

Penulis dapat menyelesaikan buku "Kecerdasan Buatan" dalam rentang waktu yang cukup lama. Buku ini terdiri atas 13 bab yang dimulai dengan pendahuluan yang berisi sejarah awal munculnya kecerdasan buatan hingga *Decision Support System*. Adapun buku ini mencakup sistem pakar, *Natural Language Processing, Speech Recognition, Computer Vision, Fuzzy Logic*, algoritma genetika, *immune* dan koloni semut, *Tabu Search* serta robotika dan sistem sensor.

Penulis menyadari masih banyak kekurangan dalam penulisan dan penyusunan buku Kecerdasan Buatan ini. Saran dan kritik dari pembaca sangat diperlukan untuk perbaikan di masa datang. Selamat membaca, semoga bermanfaat.

حَسْبُنَا ٱللَّهُ وَنِعْمَ ٱلْوَكِيلُ

Ciputat, Agustus 2013

Penulis

DAFTAR ISI

MOTO | ii

KATA PENGANTAR | iv

DAFTAR ISI | v

Bab 1 Pendahuluan

1.1 Sejarah Kecerdasan Buatan | 1
1.2 Revolusi Pengolahan Data oleh Komputer | 2
1.3 Apakah Komputer Dapat Berpikir? | 4
1.4 Faktor Pendorong Perkembangan Kecerdasan Buatan | 7
1.5 Kedudukan Ilmu Kecerdasan Buatan | 10
1.6 Permasalahan yang Diselesaikan oleh Kecerdasan Buatan | 13
1.7 Konsep Komputasi Kecerdasan Buatan dan Komputasi Konvensional | 17

Bab 2 Sistem Pakar

2.1 Sejarah Sistem Pakar | 23
2.2 Definisi Sistem Pakar | 25
2.3 Kelebihan dan Kekurangan Sistem Pakar | 28
2.4 Struktur Sistem Pakar | 30
2.5 Arsitektur Sistem Pakar | 30
2.6 Komponen Utama Sistem Pakar | 31
2.7 Fase-Fase Pengembangan Sistem Pakar | 45
2.8 Pemeran dalam Suatu Proyek Sistem Pakar | 47
2.9 Kategori Masalah Sistem Pakar Secara Umum | 48
2.10 Bentuk/Tipe Sistem Pakar | 50
2.11 Pengetahuan (*Knowledge*) | 51

2.12 Teknik Representasi Pengetahuan | 52
2.13 Representasi Logika | 56
2.14 Bingkai (*Frame*) | 59
2.15 Naskah (*Script*) | 60
2.16 Sistem Produksi | 63
2.17 Perbedaan antara Sistem Konvensional dengan Sistem Pakar | 65

Bab 3 Natural Language Processing

3.1 *Natural Language Processing* | 69
3.2 Pemahaman Kalimat | 74
3.3 CFG (Context Free Grammar) Recursive -Descent | 78
3.4 *Parsing* | 80
3.5 Semantik | 81
3.6 *Semantic Grammar* | 82
3.7 Sistem Pemrosesan Bahasa Alami | 85
3.7.1 Sistem *Speech Recognition* | 85
3.7.2 Sistem *Text to Speech* | 87
3.7.3 Sistem Natural Language Processing | 89
3.8 Aplikasi Pengolahan Bahasa Alami | 90

Bab 4 Speech Recognition

4.1 Pengantar | 96
4.2 Definisi *Speech Recognition* | 97
4.3 Sejarah *Speech Recognition* | 99
4.4 Pemodelan *Speech Recognition* | 103
4.5 Cara Kerja *Speech Recognition* | 106
4.6 Implementasi *Speech Recognition* | 113

Bab 5 Computer Vision Sebagai Pengindraan

5.1 Pengantar | 116
5.2 Definisi Visi Komputer (*Computer Vision*) | 118
5.3 Elemen-Elemen Visi Komputer | 120
5.4 Tahapan-Tahapan Proses dalam Visi Komputer | 122

5.5 Proses Visi Komputer | 131
5.5.1 Akuisisi Citra | 134
5.5.2 Pengolahan Citra | 142
5.5.3 Pemahaman Citra | 147
5.6 Aplikasi Visi Komputer | 148
5.7 Implementasi Visi Komputer pada Mesin OCR | 149

Bab 6 Fuzzy Logic

6.1 Pengantar | 156
6.2 Sejarah *Fuzzy Logic* | 156
6.3 Definisi Logika *Fuzzy* | 157
6.4 Himpunan *Fuzzy* | 161
6.5 Dasar Logika *Fuzzy* | 173
6.6 Proses Sistem Kontrol Logika Fuzzy | 178
6.7 *Fuzzy Inference System* (FIS) | 182
6.8 Aplikasi | 187

BAB 7 Robotika dan Sistem Sensor

7.1 Pengantar | 189
7.2 Sensor | 190
7.2.1 Persyaratan Umum Sensor | 194
7.2.2 Jenis Sensor | 197
7.2.3 Klasifikasi Sensor | 198
7.3 Robot | 199
7.3.1 Definisi Robot dan Robotik | 200
7.3.2 Istilah Robot | 202
7.3.3 Komponen Dasar Sebuah Robot | 213
7.4 Tingkatan Teknologi Robot | 213
7.5 Geometri Robot dan Istilah-Istilahnya | 217
7.6 Konfigurasi Robot | 219
7.7 Spesifikasi Teknis yang Lain | 222
7.8 Sistem Kontrol | 225
7.8.1 Jenis Robot Kontrol | 227
7.8.2 Bagian-Bagian Kontrol Robot | 228
7.9 Aplikasi Robot di Industri/Manufaktur | 230

7.10 Tipe Robot | 231

Bab 8 Tabu Search

8.1 Sejarah Tabu Search | 239
8.2 Definisi dan Algoritma Tabu Search | 241
8.3 Struktur Memori dan Komponen Tabu Search | 24(
8.4 Penerapan Tabu Search | 248
8.5 Kelebihan dan Kekurangan Tabu Search | 265

Bab 9 Algoritma Koloni Semut

9.1 Pengantar | 268
9.2 Sejarah | 269
9.3 Perilaku Semut | 269
9.4 Karakteristik Optimasi Koloni Semut | 272
9.5 Algoritma *Ants Colony Optimization* (ACO) | 273

Bab 10 Algoritma Immune

10.1 Pengantar | 277
10.2 Sistem Kekebalan Tubuh Manusia | 277
10.3 *Artificial Immune System* (AIS) | 282

Bab 11 Algoritma Genetika

11.1 Pengantar | 287
11.2 Definisi Algoritma Genetika | 289
11.3 Definisi Penting AG | 294
11.4 Perbedaan Algoritma Genetika dengan Algoritma Konvensional | 297
11.5 Definisi Nilai *Fitness* | 298
11.6 Siklus Algoritma Genetika | 298
11.7 Mutasi | 305
11.8 Operasional AG | 311
11.9 Kendali Parameter Algoritma Genetika | 318
11.10 Kritik terhadap Algoritma Genetika | 319
11.11 *Word Matching* dengan Algoritma Genetika | 321

Bab 12 Jaringan Saraf Tiruan

12.1 Definisi Jaringan Saraf Tiruan | 326
12.2 Keuntungan Penggunaan *Neural Network* | 328
12.3 Jaringan Saraf Manusia | 329
12.4 Jaringan Saraf Tiruan (*Artificial Neural Networks*) | 332
12.5 Algoritma Pembelajaran | 348
12.6 Keuntungan dan Kerugian *Neural Networks* | 377

Bab 13 Decision Support System (DSS)

13.1 Pengantar | 381
13.2 Definisi DSS | 382
13.3 Mengapa Menggunakan DSS? | 384
13.4 Konsep DSS | 385
13.5 Tingkat Pengambilan Keputusan | 394
13.6 Komponen DSS | 397
13.7 Ciri, Keuntungan dan Keterbatasan DSS | 398
13.8 Sistem Penunjang Keputusan Kelompok (GDSS) | 400

DAFTAR PUSTAKA | 402

BIOGRAFI PENULIS | 409

Bab 1

Pendahuluan

1.1 Sejarah Kecerdasan Buatan

Kecerdasan buatan sebenarnya sudah dimulai sejak musim panas tahun 1956. pada waktu itu sekelompok pakar komputer, pakar dan peneliti dari disiplin ilmu lain dari berbagai akademi, industri serta berbagai kalangan berkumpul di Dartmouth College untuk membahas potensi komputer dalam rangka menirukan atau mensimulasi kepandaian manusia. Beberapa ilmuwan yang terlibat adalah **Allen Newel**, **Herbert Simon**, **Marvin Miskey**, **Oliver Selfridge**, dan **John McCarthy**. Sejak saat itu, para ahli mulai bekerja keras untuk membuat, mendiskusikan, merubah dan mengembangkan sampai mencapai titik kemajuan yang penuh. Mulai dari laboratorium sampai pada pelaksanaan kerja nyata.

Pada mulanya kecerdasan buatan hanya ada di universitas dan laboratorium penelitian, dan hanya sedikit sekali – jika ada produk praktis yang sudah dikembangkan. Menjelang akhir tahun 1970-an dan awal tahun 1980-an, mulai dikembangkan secara penuh dan hasilnya secara berangsur-angsur mulai dipasarkan. Saat ini, sudah banyak hasil penelitian yang sedang dan sudah

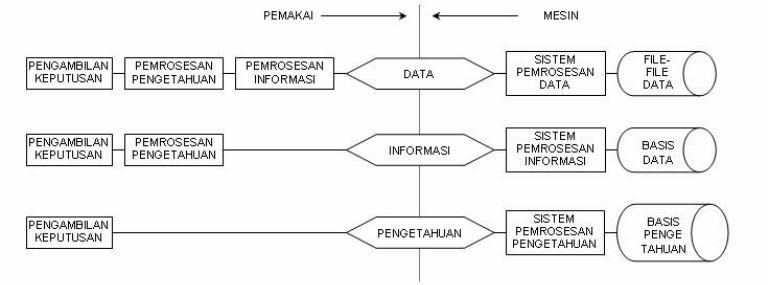
dikonversikan menjadi produk nyata yang membawa keuntungan bagi pemakainya.

Kecerdasan buatan atau *artificial intelligence* (AI), definisinya menurut beberapa pakar:

1. Schalkoff (1990): AI adalah bidang studi yang berusaha menerangkan dan meniru perilaku cerdas dalam bentuk proses komputasi.
2. Rich dan Knight (1991): AI adalah studi tentang cara membuat komputer melakukan sesuatu yang, sampai saat ini, orang dapat melakukannya lebih baik.
3. Luger dan Stubblefield (1993): AI adalah cabang ilmu komputer yang berhubungan dengan otomasi perilaku yang cerdas.
4. Haag dan Keen (1996): AI adalah bidang studi yang berhubungan dengan penangkapan, pemodelan, dan penyimpanan kecerdasan manusia dalam sebuah sistem teknologi informasi sehingga sistem tersebut dapat memfasilitasi proses pengambilan keputusan yang biasanya dilakukan oleh manusia.

1.2 Revolusi Pengolahan Data oleh Komputer

Awalnya komputer hanya mengolah data, kemudian menghasilkan informasi untuk pengambilan keputusan. Seiring dengan perkembangan, saat ini komputer dapt mengolah pengetahuan sehingga proses pengambilan keputusan menjadi lebih cepat dan akurat.



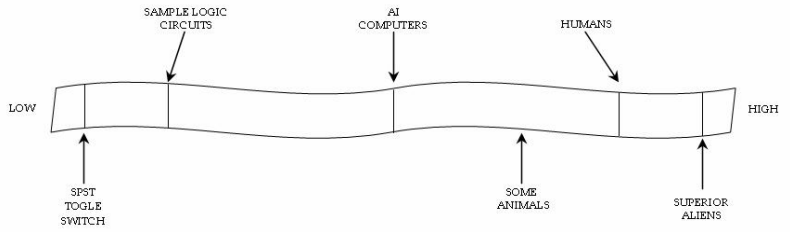
Gambar 1.1 Pengolahan Data, Informasi dan Pengetahuan

Apabila komputer mengerjakan pengolahan data, maka manusia harus mengkonversi data tersebut menjadi informasi yang dapat digunakan dalam mengolah pengetahuan untuk menghasilkan keputusan. Apabila komputer mengerjakan pengolahan informasi – yang berarti juga termasuk pengolahan data – maka manusia cukup mengerjakan pengolahan pengetahuan untuk menghasilkan keputusan. Akan tetapi apabila komputer dapat melakukan pengolahan pengetahuan – juga terkandung di dalamnya pengolahan data dan informasi – maka akan sangat sedikitlah bagian pekerjaan yang harus dilakukan manusia, termasuk dalam penerapan hasil untuk keperluan tertentu.

Teknik yang membuat komputer mampu mengolah pengetahuan ini dinamakan **teknik kecerdasan buatan** (*artificial intelligence technique*). Dengan pendekatan ini manusia mencoba membuat komputer dapat berpikir seperti cara yang dipakai manusia dalam memecahkan masalah.

1.3 Apakah Komputer Dapat Berpikir?

Terdapat beberapa tingkat kecerdasan seperti yang diilustrasikan oleh gambar berikut.



Gambar 1.2 Spektrum Kecerdasan atau "kemampuan arah"

Pikiran manusia menduduki tingkat tertinggi dalam spektrum kecerdasan, rangkaian logika sederhana ada pada tingkat spektrum terendah, sedangkan kecerdasan buatan terletak pada kedua hal tersebut. Beberapa pertanyaan seputar Kecerdasan Buatan yang diterapkan pada komputer:

- Apakah kemudian komputer menjadi lebih pintar?
- Apakah benar, komputer sekarang mempunyai kecerdasan seperti yang dimiliki manusia dalam melaksanakan tugasnya?
- Apabila kecepatan dan kemudahan serta peningkatan produktifitas kerja dan kemampuannya melakukan pekerjaan dengan baik sudah tercapai, apakah komputer itu sudah bisa dikatakan mempunyai tingkat kecerdasan tertentu?

- Bagaimana kita bisa menyatakan bahwa komputer itu memiliki kecerdasan?
- Kriteria apakah yang bisa digunakan untuk sampai pada kesimpulan itu?

Kenyataannya, hampir semua program Kecerdasan Buatan yang dilaksanakan dengan teknik pelacakan dan pencocokan pola mengarah kepada suatu kesimpulan bahwa komputer tidak benar-benar pintar. Kita boleh memberi informasi sebanyak-banyaknya kepada komputer dan beberapa pengarahan tentang cara penggunaannya. Dengan menggunakan informasi dan kriteria yang berlaku, komputer bisa menemukan suatu solusi. Semua yang dilakukannya itu merupakan percobaan atas berbagai alternatif dan usaha untuk menemukan beberapa kombinasi yang cocok dengan kriteria terancang. Bila semua itu sudah dikerjakan, maka secara tipikal solusi tersebut bisa dicapai. Dengan demikian, walaupun komputer itu tidak betul-betul pintar, tetapi kelihatannya seperti "berfikir" dan sering memberikan solusi yang menakjubkan.

Seorang ahli matematika dan Kecerdasan Buatan bernama **Allan Turing** berkebangsaan Inggris yang memiliki sumbangan dalam teori kemampuan penghitungan (*computability*), bergumul dengan pertanyaan *apakah sebuah mesin dapat berpikir atau tidak?* dengan melakukan sebuah percobaan yang disebut dengan imitasi permainan (*pseudo game*). Dalam uji ini, dilibatkan seorang penanya (manusia) dan 2 objek yang ditanyai (manusia

dan mesin). Penanya dan 2 objek yang ditanyai itu terletak pada jarak yang jauh sehingga tidak dapat melihat atau berbicara dengan kedua objek yang ditanyai itu. Penanya hanya dapat berkomunikasi dengan keduanya lewat suatu alat terminal komputer. Penanya mengira bahwa ia sedang berhubungan dengan dua operator lainnya. Dalam mengajukan pertanyaan, penanya bebas memilih pertanyaan. Misalnya, untuk mengetahui mana objek yang benar-benar manusia, penanya dapat mengajukan pertanyaan yang didasarkan pada sifat emosi kedua objek yang ditanyai, yakni pertanyaan tentang suatu puisi atau karya seni lain. Walaupun secara dasar percobaan itu masih menimbulkan kesangsian manusiawi, tetapi Turing berpendapat bahwa jika penanya tidak dapat membedakan mana yang manusia dan mana yang mesin, maka mesin tersebut dapat diasumsikan sebagai cerdas. Beberapa hal penting dari uji ini adalah:

1. Memberikan tanda-tanda yang objektif dari kecerdasan, yaitu respon tingkah laku dari kecerdasan yang telah dikenal terhadap sejumlah pertanyaan tertentu. Cara ini memberikan standar dalam menentukan kecerdasan dan menghindarkan beda pendapat tentang apa itu sifat kecerdasan yang sebenarnya.
2. Dapat membuat kita mempunyai pedoman dalam menerima jawaban yang membingungkan dan ketidakmampuan objek menjawab pertanyaan kita, terlepas apakah mesin tersebut menggunakan proses internal yang memadai atau tidak

peduli apakah komputer itu sadar atau tidak dengan responnya sendiri.

3. Menghapus setiap bias yang menguntungkan organisme hidup (termasuk manusia) dibandingkan mesin cerdas dengan memaksa si penanya agar hanya memfokuskan perhatiannya pada jawaban-jawaban dari pertanyaan yang diajukannya.

Uji Turing ini menjadi dasar bagi banyak strategi yang digunakan dalam menilai program-program Kecerdasan Buatan modern. Setiap percobaan untuk inteligensia mempunyai nilai dan kedayagunaan yang terbatas. Dia mungkin bisa mengerjakan suatu kasus, tapi belum tentu bisa mengerjakan kasus yang lain.

1.4 Faktor Pendorong Perkembangan Kecerdasan Buatan

Faktor pendorong bagi terlaksananya aplikasi Kecerdasan Buatan adalah:

1. **Pesatnya perkembangan teknologi perangkat keras.**
   Hampir semua aplikasi Kecerdasan Buatan memerlukan perangkat keras yang memiliki kecepatan daya tampung yang lebih tinggi, walaupun hanya menjalankan perangkat lunak Kecerdasan Buatan yang paling sederhana sekalipun. Di samping itu, harga perangkat keras yang dengan kemampuan lebih memiliki harga yang relatif semakin murah.

2. **Pengembangan perangkat lunak Kecerdasan Buatan**
   Dewasa ini bahasa dan alat pemrograman baru yang lebih canggih sudah banyak dikembangkan dan dipasarkan secara luas, termasuk bahasa khusus untuk Kecerdasan Buatan.
3. **Perkembangan khusus komputer pribadi (personal computer/PC).**
   Sekarang sudah sangat banyak orang menggunakan komputer mikro (*microcomputer*) khususnya komputer pribadi baik di sekolah, perusahaan atau bahkan di rumah yang menyebabkan permintaan mereka akan perangkat lunak yang lebih unggul untuk pekerjaan mereka.
4. **Turut andilnya para investor dalam mendanai penelitian dan pengembangan teknologi Kecerdasan Buatan.**
   Hal ini mengakibatkan terjadinya semacam tekanan di kalangan masyarakat Kecerdasan Buatan untuk berlomba-lomba dalam mempercepat gerak dan langkah penelitiannya dan segera memproduksi Kecerdasan Buatan dalam waktu yang singkat.

Masalah utama Kecerdasan Buatan adalah sulitnya merumuskan dan memvisualisasi inteligensia itu sendiri, karena mempunyai arti yang banyak. Walaupun AI telah banyak membuat komputer menjadi lebih pintar dan lebih canggih, tapi tampaknya impian manusia agar bisa membuat komputer yang betul-betul bisa membuat duplikasi otak manusia, atau bisa menjadi pengganti otak manusia yang sebenarnya masih jauh dari kenyataan.

Mungkin belum bisa terlaksana pada zaman atau masa kini. Kenyataannya, masih banyak persoalan-persoalan yang timbul tentang apakah kita akan selamanya bisa membuat sebuah komputer yang secara akurat dapat melakukan hal-hal yang terbaik seperti yang dilakukan oleh pikiran manusia.

Hubert Dreyfus – ahli filsafat dari **Universitas California** di **Berkeley** – berpendapat bahwa masyarakat sekarang ini sedang dikacaukan oleh oleh pengertian Kecerdasan Buatan yang mengira seolah-olah kegunaannya sangat berlebihan dan tidak mungkin bisa mencapai tujuan. **Dreyfus** berkata: "*Kita tidak akan pernah bisa membuat suatu kaidah untuk semua cara kita berfikir, karena hal itu sangat kompleks*".

Tentu saja, para peneliti yang ahli akan lebih bisa mendekati pada komputer pintar, tapi masih banyak masalah yang harus dijawab. Misalnya, bagaimana kita bisa mencerminkan keterampilan dalam menangani masalah manusia, kemampuan belajar, selera, imajinasi, emosi, kreativitas dan 'rasa berani'. Untuk menjawab masalah-masalah tersebut, para ahli kembali pada bidang yang berkaitan, seperti: filosofi, psikologi, linguistik dan sains saraf (***neuro science***) dan tentu saja sains komputernya itu sendiri. Dengan demikian akan lahir bidang sains kognitif antar disiplin ilmu tersebut.

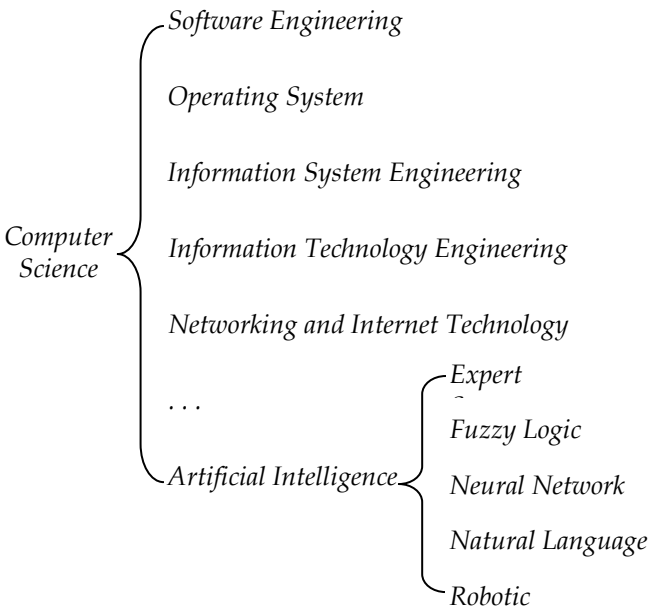
Walaupun masih banyak kritik, metode Kecerdasan Buatan tetap memperlihatkan nilai lebih dibanding dengan yang lain.

Teknik Kecerdasan Buatan menunjukkan bagaimana kita berfikir dan bagaimana kita menerapkan inteligensia dengan lebih baik. Teknik Kecerdasan Buatan akan membuat komputer lebih mudah digunakan dan pengetahuan akan semakin tersebar luas di kalangan masyarakat. Kita membuat komputer pintar bukan untuk menggantikan manusia tapi hanya sekedar untuk menjadi alat bantu manusia.

1.5 Kedudukan Ilmu Kecerdasan Buatan

Kecerdasan berasal dari kata dasar *cerdas*. Cerdas dapat memiliki konotasi makna lebih baik, cepat, *capable*, *adapted* dengan kondisi umumnya/normal. Cerdas juga dapat berarti kemampuan untuk mengerti/memahami. Kecerdasan (*intelligence*) dimiliki seseorang yang pandai melaksanakan pengetahuan yang dimilikinya. Walaupun seseorang memiliki banyak pengetahuan, tetapi bila ia tidak bisa melaksanakannya dalam praktek, maka ia tidak bisa digolongkan ke dalam kecerdasan. Dengan perkataan lain, kecerdasan adalah kemampuan manusia untuk memperoleh pengetahuan dan pandai melaksanakannya dalam praktek.

Kecerdasan buatan merupakan upa-bidang ilmu komputer (*computer science*) yang khusus ditujukan untuk membuat perangkat lunak dan perangkat keras yang sepenuhnya bisa menirukan beberapa fungsi otak manusia atau cabang ilmu komputer yang mempelajari otomatisasi tingkah laku cerdas (*intelligent*).



Gambar 1.3 Bagan Kedudukan Ilmu Kecerdasan Buatan

Kecerdasan harus didasarkan pada prinsip-prinsip teoritikal dan terapan yang menyangkut:

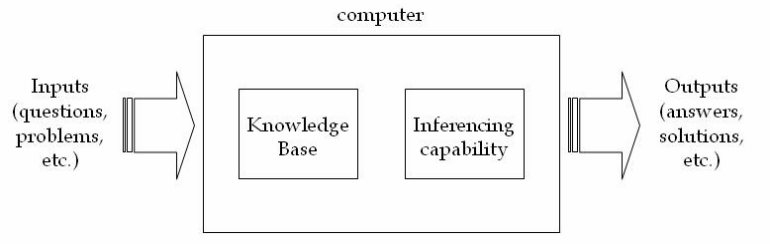
- struktur data yang digunakan dalam representasi pengetahuan (*knowledge representation*),
- algoritma yang diperlukan dalam penerapan pengetahuan itu,
- teknik-teknik bahasa dan pemrograman yang dipakai dalam implementasinya.

Kecerdasan buatan menawarkan baik media maupun uji teori kecerdasan. Teori-teori semacam ini dapat dinyatakan dalam bahasa program komputer dan dibuktikan melalui eksekusinya pada komputer.

Bagian utama aplikasi kecerdasan buatan adalah pengetahuan (*knowledge*), yaitu suatu pengertian tentang beberapa wilayah subyek yang diperoleh melalui pendidikan dan pengalaman. Bagian utama yg dibutuhkan untuk aplikasi kecerdasan buatan:

- Basis pengetahuan (*knowledge base*)
- Motor inferensi (*inference engine*)

Pengetahuan merupakan informasi terorganisir dan teranalisis agar bisa lebih mudah dimengerti dan bisa diterapkan pada pemecahan masalah dan pengambilaan keputusan. Pengetahuan terdiri atas fakta, pemikiran, teori, prosedur, dan hubungannya satu sama lain.



Gambar 1.4 Penerapan Konsep Kecerdasan Buatan dalam Komputer

Komputer tidak mungkin mendapatkan pengetahuannya sendiri dengan belajar, berpengalaman atau melakukan

penelitian, akan tetapi ia memperolehnya melalui upaya yang diberikan oleh seorang pakar manusia. Hampir semua pangkalan pengetahuan (*knowledge base*) sangat terbatas, dalam arti terfokuskan kepada suatu masalah khusus. Pada saat pangkalan pengetahuan itu sudah terbentuk, teknik Kecerdasan Buatan bisa digunakan untuk memberi kemampuan baru kepada komputer agar bisa berfikir, menalar, dan membuat inferensi (mengambil keputusan berdasarkan pengalaman) dan membuat pertimbangan-pertimbangan yang didasarkan kepada fakta dan hubungan-hubungannya yang terkandung dalam pangkalan pengetahuan itu. Dengan pangkalan pengetahuan dan kemampuan untuk menarik kesimpulan melalui pengalaman (inferensi), komputer dapat disejajarkan sebgai alat bantu yang bisa digunakan secara praktis dalam memecahkan masalah dan pengambilan keputusan serta bisa mencapai satu atau lebih solusi alternatif pada maslah yang diberikan.

1.6 Permasalahan yang Diselesaikan oleh Kecerdasan Buatan

Mula-mula hal yang digeluti oleh Kecerdasan Buatan adalah pembuktian teorema dan permainan (*game*). Misalnya **Newell**, ahli teori logika, berusaha untuk membuktikan teorema-teorema matematika dan Samuel yang membuat program permainan catur. Kemudian para periset Kecerdasan Buatan terus mengembangkan berbagai teknik baru untuk menangani sejumlah besar persoalan, termasuk persepsi, pemahaman bahasa

alamiah, dan problema spesifik seperti diagnosa medis. Permasalahan yang ditangani oleh Kecerdasan Buatan adalah:

1. Pembuktian teorema (*theorem proving*), misalnya: MACSYMA untuk tugas-tugas matematika.
2. Permainan (*game*), seperti: *chess*, *tic* (= saraf tak sadar) *tac toe* (= jari kaki), *othelo*, dan sebagainya.
3. Pemecahan masalah umum (*general problem solving*), misalnya pengambilan keputusan otomatis (*automated decision making*) dan pemodelan kinerja manusia.
4. Persepsi/*perception* (visi/*vision* dan percakapan/ *conversation*)
5. Pemahaman bahasa alamiah (*natural language*), misalnya ELIZA yang dapat memberikan saran psikologis.
6. Pengenalan pola (*pattern recognition*), seperti pengolahan citra digital untuk kebutuhan ramalan cuaca, foto (kopi), monitor tv, dan sebagainya.
7. Pemecahan masalah pakar (*expert*), mencakup bidang matematika simbolik, diagnosa medis, rekayasa rancang bangun, analisis kimia.
8. Pembuatan perangkat lunak otomatis (*automated software generation*).

Bidang-bidang teknik kecerdasan buatan di antaranya adalah:

- Sistem pakar (*expert system*)
- Pengolahan bahasa alami (*natural language processing*)
- Pengenalan ucapan (*speech recognition*)

- Pengolahan citra (*image processing*)
- Robotik (*robotics*), dan sensor
- Logika Samar (*fuzzy logic*),
- Algoritma Genetika (*genetic algorithm*)
- Jaringan saraf tiruan (*neural networks*)
- *Intelligent computer-aided instruction*
- *Game playing*
- *Evolutionary computing* (Optimasi)
- *Probabilistic reasoning* (mengakomodasi ketidakpastian)

Beberapa sifat penting Kecerdasan Buatan yang muncul dalam pelbagai bidang penerapan:

1. Digunakannya komputer untuk melakukan pertimbangan dengan proses yang memakai simbol.
2. Pemfokusan ditujukan pada persoalan yang tidak memberikan respon terhadap solusi algoritmik. Hal inilah yang mendasari pencarian (*search*) heuristik sebagai teknik pemecahan masalah Kecerdasan Buatan.
3. Usaha yang dilakukan lebih ditujukan untuk menangkap dan memanipulasi sifat-sifat kualitatif penting dari suatu situasi daripada metode numerik.
4. Usaha yang dilakukan adalah untuk menangani arti-arti semantik dan bentuk sintaksis.
5. Jawaban yang diberikan tidaklah eksak atau optimal, namun lebih bersifat cukup (*sufficient*). Ini merupakan hasil penting dalam metode pemecahan masalah heuristik pada situasi di

mana hasil yang eksak atau optimal akan berharga terlalu mahal atau tidak mungkin dilakukan.

6. Penggunaan sejumlah besar pengetahuan khusus dalam memecahkan persoalan. Ini merupakan dasar bagi sistem pakar.
7. Penggunaan pengetahuan tingkat meta (*meta-level*) untuk mempengaruhi pengendalian lebih canggih dari strategi pemecahan masalah. Meskipuan hal ini merupakan persoalan yang sangat sulit dan hanya ditujukan pada sejumlah kecil sistem, namun muncul sebagai objek riset yang penting.

Perangkat lunak Kecerdasan Buatan didasarkan kepada representasi dan manipulasi simbol (*symbol*). Sebuah simbol bisa merupakan huruf, kata atau bilangan yang digunakan untuk menggambarkan objek, proses dan saling hubungannya. Ia bisa merupakan cetakan atau elektronik. Objek bisa berupa orang, benda, ide, pikiran, peristiwa atau pernyataan suatu fakta. Dengan menggunakan simbol memungkinkan komputer bisa menciptakan suatu basis data yang menyatakan fakta, pikiran dan hubungannya satu sama lain. Berbagai proses digunakan untuk memanipulasi simbol agar mampu memecahkan masalah. Pengolahannya bersifat kuantitatif dan bukan kualitatif seperti halnya komputasi yang didasarkan kepada algoritma. Apabila basis pengetahuan, fakta dan hubungannya sudah dibuat, maka penggunaannya untuk memecahkan masalah harus sudah dimulai. Teknik dasar untuk melakukan penalaran dan menarik kesimpulan dari pengalaman melalui basis pengetahuan adalah

pelacakan (*searching*) dan pencocokan pola (*pattern matching*). Secara harfiah komputer terus memburu dan mencari pengetahuan yang ada sampai ia menemukan jawaban yang terbaik atau jawaban yang paling cocok.

Walaupun secara langsung pemecahan Kecerdasan Buatan tidak didasarkan kepada algoritma, tetapi sebenarnya dalam implementasi proses pelacakan, algoritma tetap digunakan. Program yang didasarkan kepada algoritma melaksanakan manipulasi simbolik yang menyebabkan suatu masalah dapat terpecahkan dengan cara yang sangat mendekati dengan cara kerja berfikir manusia.

1.7 Konsep Komputasi Kecerdasan Buatan dan Komputasi Konvensional

Berikut dijelaskan beberapa persamaan dan perbedaan komputasi kecerdasan buatan dan konvensional.

Persamaan:

1. Sama-sama mengolah simbol-simbol yang dapat berupa huruf, kata, atau bilangan yang digunakan untuk menggambarkan objek, proses, dan saling hubungannya. Objek dapat berupa orang, benda, ide, pikiran, peristiwa atau pernyataan suatu fakta.
2. Menggunakan komputer digital untuk melaksanakan operasi.

Perbedaan:

KOMPUTASI KONVENSIONAL	KECERDASAN BUATAN
Menggunakan fungsi otak manusia.	Meniru beberapa fungsi otak manusia.
Komputer diperintahkan untuk menyelesaikan suatu masalah.	Komputer diberitahu tentang suatu masalah.
Hanya dapat ditulis dalam bahasa pemrograman biasa seperti Assembler, C/C++, Fortran, Basic dan Pascal.	Programnya dapat ditulis dalam semua bahasa pemrograman termasuk bahasa pemrograman khusus untuk aplikasi Kecerdasan Buatan seperti Prolog dan LISP.
Dapat dijalankan pada semua jenis komputer tetapi tidak dibuatkan *hardware* khusus.	Dapat dibuatkan *hardware* khusus dan dapat pula dijalankan pada semua jenis komputer.
Komputer diberikan data dan program yang berisi spesifikasi langkah demi langkah bagaimana cara data itu digunakan dan diolah untuk menghasilkan solusi.	Komputer diberi pengetahuan tentang suatu wilayah subyek masalah tertentu dengan ditambah kemampuan inferensi.
Didasarkan pada suatu algoritma yang dapat berupa rumus matematika atau prosedur berurutan yang tersusun jelas.	Didasarkan pada representasi dan manipulasi simbol.
Pengolahan objek bersifat kualitatif.	Pengolahan objek bersifat kuantitatif.

Adapun keuntungan dan kerugian/kelemahan Kecerdasan Buatan adalah sebagai berikut:

1. Komputer masa depan akan memberikan kenikmatan, kenyamanan, dan kesenangan yang lebih bagi penggunanya, tetapi sebaliknya akan mendorong harga komputer menjadi semakin mahal. Hal-hal yang ditawarkan bagi para pengguna komputer Kecerdasan Buatan:
   - akan bisa berkomunikasi dengan komputer dengan bahasa alami/bahasa manusia sehari-hari
   - akan terbebas dari keharusan belajar bahasa pemrograman dan sistem operasi.
   - para pengguna komputer yang tidak terlatih sekalipun akan menghasilkan karya yang sangat berguna bagi kepentingannya dengan menggunakan komputer
   - menggunakan komputer akan tidak lebih sulit daripada menggunakan pesawat telepon
2. Komputer akan menjadi semakin lebih berguna.
   Hal ini karena bidang-bidang masalah yang tadinya tidak dapat dipecahkan oleh komputer kini akan dapat dapat dipecahkan dengan teknik Kecerdasan Buatan.
3. Biaya pengembangan dan penelitian Kecerdasan Buatan sangat mahal.
4. Pengembangan aplikasi Kecerdasan Buatan merupakan hal yang sangat sulit dan diperlukan waktu yang sangat lama.

5. Masih sedikitnya perangkat lunak khusus untuk Kecerdasan Buatan.

   Padahal dengan perangkat lunak khusus ini, pekerjaan pembuatan dan pengembangan perangkat lunak Kecerdasan Buatan menjadi lebih mudah dan cepat.

6. Belum terciptanya antarmuka (*interface*) bahasa alami khusus untuk Kecerdasan Buatan.

   Perangkat lunak Sistem Manajemen Basis Data (*DataBase Management System/DBMS*) merupakan salah satu tipe perangkat lunak konvensional pertama yang sudah bisa mengambil keuntungan dari terciptanya bahasa alami ini. Perangkat lunak DBMS ini bisa lebih cepat dan lebih mudah mengambil data yang disimpan dalam basis data tanpa harus menggunakan teknik pemrograman yang rumit.

Beberapa contoh penerapan AI:

- Deep Blue adalah program catur yang pada tahun 1997 dapat mengalahkan pecatur dunia Garyy Kasparov dengan kedudukan 3,5 – 2,5.
- Logic Theorist adalah program yang mampu membuktikan beberapa teorema yang terdapat pada bab pertama buku *Principia Matematica* karya Whitehead dan Russel.
- Systran adalah perangkat lunak yang dapat dipakai untuk melakukan penerjemahan dari dan ke bahasa-bahasa berikut: Jerman, Perancis, Italia, Jepang, Korea, Rusia, Portugis, Spanyol.

- Delco Electronics menciptakan sebuah mobil yang dapat mengemudikan sendiri. Mobil ini menggunakan pendeteksi tepi untuk untuk tetap bertahan di jalan.
- Volkswagen AG (Jerman) menciptakan sistem pengemudi kendaraan otomatis. Bahasa pemrograman Kecerdasan Buatan (AI) Bahasa pemrograman AI disebut juga dengan bahasa pemrograman generasi kelima yang merupakan kelompok bahasa-bahasa pemrograman yang ditujukan untuk menangani kecerdasan buatan. Bahasa tersebut adalah:
- LISP (List Processing) diciptakan oleh John McCarthy di MIT sekitar tahun 1960 dan merupakan bahasa kecerdasan buatan yang pertama. Kehadiran bahasa ini merupakan kebangkitan dari aplikasi kecerdasan buatan. Sejak saat itu, masih dalam dekade 1960-an bermunculan program-program yang meniru kecerdasan buatan manusia, seperti: ELIZA yang diciptakan oleh Joseph Wizenbaum (program yang memungkinkan komputer bermain catur dan dapat bertindak sebagai psikoanalis). Ciri khas program ini banyak menggunakan tanda kurung.
- PROLOG (aslinya memiliki kepanjangan PROgrammation en LOGique atau kemudian menjadi PROgramming in LOGic) adalah bahasa untuk aplikasi kecerdasan buatan yang dibuat pada tahun 1972 di Universitas Marseille-Aix, Perancis. Penciptanya adalah Alsin Colmerauer. Bahasa ini menjadi sangat populer semenjak Jepang mengumumkan

pada tahun 1981 bahwa Jepang akan menggunakannya sebagai basis komputer “generasi kelima”.

Bab 2

Sistem Pakar

2.1 Sejarah Sistem Pakar

Sistem pakar (*Expert Sistem* (ES)) dikembangkan pada pertengahan tahun 1960-an oleh *Artificial Intelligence Corporation*. Periode penelitian *artificial intelligence* ini didominasi oleh suatu keyakinan bahwa nalar yang digabungkan dengan komputer canggih akan menghasilkan prestasi pakar atau manusia super. Suatu usaha ke arah ini adalah *General Purpose Problem-Solver* (GPS). GPS yang berupa sebuah prosedur yang dikembangkan Allen Newell, John Cliff Show dan Herbert Alexander Simon dari *Logic Theorist*, yang merupakan sebuah percobaan untuk menciptakan mesin yang cerdas. GPS sendiri merupakan sebuah *Predecessor* menuju *Expert Sistem* (ES). GPS berusaha untuk menyusun langkah-langkah yang dibutuhkan untuk mengubah situasi awal menjadi *state* tujuan yang telah ditentukan sebelumnya.

Pada pertengahan tahun 1960-an, terjadi pergantian dari program serba bisa (*general-purpose*) ke program yang spesialis (*special-purpose*) dengan dikembangkannya *DENDRAL* oleh E. Feigenbaum dari Universitas Stanford dan kemudian diikuti oleh

MYCIN. Masalah yang kompleks membutuhkan pengetahuan yang banyak sekali tentang area problem. Pada pertengahan tahun 1970-an, beberapa ES mulai muncul, sebuah pengetahuan kunci yang dipelajari saat itu adalah kekuatan dari ES berasal dari pengetahuan spesifik yang dimilikinya bukan dari formalisme khusus dan pola penarikan kesimpulan yang digunakan.

Awal 1980-an teknologi ES yang mula-mula dibatasi oleh suasana akademis mulai muncul sebagai aplikasi komersial, khususnya XCON**,** XSEL (dikembangkan dari R-1 pada *Digital Equipment Corp*) dan CATS-1 (dikembangkan oleh *General Electric*). Sistem Pakar untuk melakukan diagnosis pertama dibuat oleh *Bruce Buchanan* dan *Edward Shortliffe* di *Stanford University*. Sistem ini diberi nama MYCIN. MYCIN merupakan program interaktif yang melakukan diagnosis penyakit meningitis dan infeksi *bacremia* serta memberikan rekomendasi terapi antimikrobia. MYCIN mampu memberikan penjelasan atas penalarannya secara detail. Dalam uji coba, ia mampu menunjukkan kemampuan seperti seorang spesialis. Meskipun MYCIN tidak pernah digunakan secara rutin oleh dokter, MYCIN merupakan referensi yang bagus dalam penelitian kecerdasan buatan yang lain.

2.2 Definisi Sistem Pakar

- Menurut **Durkin**: suatu program komputer yang dirancang untuk memodelkan kemampuan penyelesaian masalah yang dilakukan oleh seorang pakar.
- Menurut **Ignizio**: suatu model dan prosedur yang berkaitan, dalam suatu domain tertentu, yang mana tingkat keahliannya dapat dibandingkan dengan keahlian seorang pakar.
- Menurut **Giarratano** dan **Riley**: suatu sistem komputer yang bisa menyamai atau meniru kemampuan seorang pakar.

Sistem pakar adalah sebuah program komputer yang dirancang untuk memodelkan kemampuan menyelesaikan masalah seperti layaknya seorang pakar (*human expert*). Sistem pakar yang baik dirancang agar dapat menyelesaikan suatu permasalahan tertentu dengan meniru kerja dari para ahli di bidangnya masing-masing. Sistem pakar terkadang lebih baik unjuk kerjanya daripada seorang pakar manusia, sehingga sistem pakar dapat disimpulkan yaitu sebagai sebuah kepakaran yang ditransfer dari sebuah pakar (atau sumber kepakaran yang lain) ke komputer, pengetahuan yang ada kemudian disimpan dalam memori komputer dan pengguna dapat berkonsultasi dengan komputer untuk suatu keperluan tertentu, lalu komputer dapat menyimpulkan seperti layaknya seorang pakar, kemudian mnjelaskannya kepada pengguna tersebut, dengan menyertakan alasan-alasannya.

Sistem Pakar dibangun bukan berdasarkan algoritma tertentu tetapi bedasarkan basis pengetahuan (*Knowledge-Base*) dan aturan (*Rule*). Basis pengetahuan berisi pengetahuan penting untuk pengertian, formulasi dan pemecahan masalah. Dalam penyusunannya, sistem pakar mengkombinasikan kaidah-kaidah penarikan kesimpulan (*inference rules*) dengan basis pengetahuan tertentu yang diberikan oleh satu atau lebih pakar dalam bidang tertentu. Kombinasi dari kedua hal tersebut disimpan dalam komputer, yang selanjutnya digunakan dalam proses pengambilan keputusan untuk penyelesaian masalah tertentu. Sistem pakar setidak-tidaknya mempunyai dua unsur manusai atau lebih yang terlibat di dalam pembangunan dan pengembangan serta penggunaannya. Minimal, ada seseorang yang membangun dan ada penggunanya. Sering juga ada pakar dan perekayasa pengetahuan (*knowledge engineer*). Menurut Turban (2001), ada 4 unsur manusia dalam sistem pakar yaitu:

1. **Pakar (*The Expert*)**

   Pakar merupakan orang yang menguasai bidang ilmu pengetahuan tertentu, berpengalaman, pengambil keputusan dan menguasai metode-metode tertentu, serta mampu memanfaatkan talentanya dalam memberikan nasehat/saran terhadap penyelesaian suatu permasalahan. Juga merupakan tugas dari seorang pakar untuk memberikan atau menyediakan pengetahuan bagaimana seseorang membentuk suatu sistem berbasis pengetahuan yang hendak dibuatnya. Selain itu, pakar juga mengetahui mana fakta

yang penting dan tidak penting di antara fakta-fakta yang ada.

2. Perekayasa Pengetahuan (*Knowledge Engineer*)

Knowledge engineer adalah orang yng membantu pakar dalam menyusun area permasalahan dengan menginterprestasikan dan mengintegrasikan jawaban-jawaban pakar atas pertanyaan yang diajukan, menggambarkan analogi, mengajukan *counter example* dan menerangkan kesulitan-kesulitan konseptual.

3. Pemakai (*User*)

Sistem pakar memiliki beberapa kelas pemakai, yaitu:

a. Pemakai bukan pakar. Dalam hal ini, sistem pakar berperan sebagai seorang konsultan atau pemberi nasihat.
b. Siswa yang ingin belajar, di sini sistem pakar berperan sebagai instruktur.
c. Pembangunan sistem pakar yang ingin meningkatkan dan menambah basis pengetahuan, dalam hal ini sistem pakar berperan sebagai rekan kerja (partner).
d. Pakar, dalam hal ini sistem pakar berperan sebagai kolega atau asisten dan unsur lainnya

4. Beberapa unsur lainnya yang mungkin termasuk ke dalam unsur manusia untuk sistem pakar adalah ***system builder*** (pembangun sistem), ***system analyst*** yang membantu mengintegrasikan sebuah sistem pakar dengan sistem terkomputerisasi lainnya. Suatu *tool builder* dapat menyediakan atau membangun *tool-tool* yang khusus.

2.3 Kelebihan dan Kekurangan Sistem Pakar

Kelebihan dari sistem pakar yaitu:

1. Memungkinkan orang awam bisa mengerjakan pekerjaan para ahli.
2. Bisa melakukan proses secara berulang secara otomatis.
3. Menyimpan pengetahuan dan keahlian para pakar.
4. Mampu mengambil dan melestarikan keahlian para pakar (terutama yang termasuk keahlian langka).
5. Mampu beroperasi dalam lingkungan yang berbahaya.
6. Memiliki kemampuan untuk bekerja dengan informasi yang tidak lengkap dan mengandung ketidakpastian. Pengguna bisa merespon dengan jawaban ’tidak tahu’ atau ’tidak yakin’ pada satu atau lebih pertanyaan selama konsultasi dan sistem pakar tetap akan memberikan jawaban.
7. Tidak memerlukan biaya saat tidak digunakan, sedangkan pada pakar manusia memerlukan biaya sehari-hari.
8. Dapat digandakan (diperbanyak) sesuai kebutuhan dengan waktu yang minimal dan sedikit biaya.
9. Dapat memecahkan masalah lebih cepat daripada kemampuan manusia dengan catatan menggunakan data yang sama.
10. Menghemat waktu dalam pengambilan keputusan.
11. Meningkatkan kualitas dan produktivitas karena dapat memberi nasehat yang konsisten dan mengurangi kesalahan.

12. Meningkatkan kapabilitas sistem terkomputerisasi yang lain. Integrasi Sistem Pakar dengan sistem komputer lain membuat lebih efektif dan bisa mencakup lebih banyak aplikasi.
13. Mampu menyediakan pelatihan. Pengguna pemula yang bekerja dengan sistem pakar akan menjadi lebih berpengalaman. Fasilitas penjelas dapat berfungsi sebagai guru.

Adapun kelemahan dari sistem pakar yaitu:

1. Biaya yang diperlukan untuk membuat, memelihara, dan mengembangkannya sangat mahal.
2. Sulit dikembangkan, hal ini erat kaitannya dengan ketersediaan pakar di bidangnya dan kepakaran sangat sulit diekstrak dari manusia karena sangat sulit bagi seorang pakar untuk menjelaskan langkah mereka dalam menangani masalah.
3. Sistem pakar tidak 100% benar karena seseorang yang terlibat dalam pembuatan sistem pakar tidak selalu benar. Oleh karena itu perlu diuji ulang secara teliti sebelum digunakan.
4. Pendekatan oleh setiap pakar untuk suatu situasi atau masalah bisa berbeda-beda, meskipun sama-sama benar.
5. Transfer pengetahuan dapat bersifat subjektif dan bias.
6. Kurangnya rasa percaya pengguna dapat menghalangi pemakaian sistem pakar.

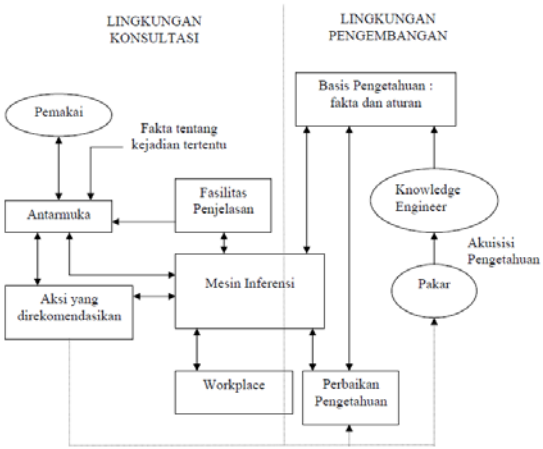
2.4 Struktur Sistem Pakar

Terdapat 2 bagian utama sistem pakar:

1. Lingkungan Pengembangan (*Development Environment*)
   Digunakan untuk memasukkan pengetahuan pakar ke dalam lingkungan sistem pakar.
2. Lingkungan Konsultasi (*Consultation Development*)
   Digunakan oleh pengguna yang bukan pakar untuk memperoleh pengetahuan pakar.

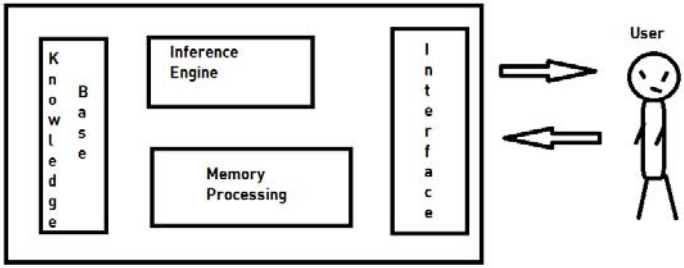
2.5 Arsitektur Sistem Pakar

Struktur dari Sistem Pakar dapat digambarkan sebagai berikut.



Gambar 2.1 Struktur Sistem Pakar

Struktur dari Sistem Pakar dapat juga secara sederhana ditunjukkan Gambar 2.2.



Gambar 2.2 Struktur Sistem Pakar Sederhana

2.6 Komponen Utama Sistem Pakar

Komponen utama pada sistem pakar, yaitu:

a. Basis Pengetahuan (*Knowledge Base*)

Basis pengetahuan merupakan inti dari suatu sistem pakar, yaitu berupa representasi pengetahuan dari pakar. Basis pengetahuan tersusun atas fakta dan kaidah. Fakta adalah informasi tentang objek, peristiwa atau situasi. Kaidah adalah cara untuk membangkitkan suatu fakta baru dari fakta yang sudah diketahui. Basis pengetahuan merupakan representasi dari seorang pakar, yang kemudian dapat dimasukkan ke dalam bahasa pemrograman khusus untuk kecerdasan buatan (misalnya PROLOG atau LISP) atau shell sistem pakar (misalnya EXSYS, PC-PLUS, CRYSTAL, dsb.)

b. Mesin Inferensi (*Inference Engine*)

Mesin inferensi berperan sebagai otak dari sistem pakar.

Mesin inferensi berfungsi untuk memandu proses penalaran terhadap suatu kondisi, berdasarkan pada basis pengetahuan yang tersedia. Di dalam mesin inferensi terjadi proses untuk memanipulasi dan mengarahkan kaidah, model, dan fakta yang disimpan dalam basis pengetahuan dalam rangka mencapai solusi atau kesimpulan. Dalam prosesnya, mesin inferensi menggunakan strategi penalaran dan strategi pengendalian. Strategi penalaran terdiri atas:

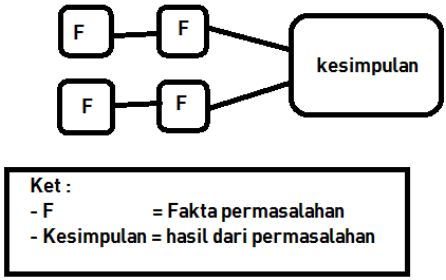
- strategi penalaran pasti (*Exact Reasoning*)
  Exact reasoning akan dilakukan jika semua data yang dibutuhkan untuk menarik suatu kesimpulan tersedia
- strategi penalaran tak pasti (*Inexact Reasoning*).
  Dilakukan jika semua data yang dibutuhkan untuk menarik suatu kesimpulan tidak tersedia.

Strategi pengendalian berfungsi sebagai panduan arah dalam melakukan proses penalaran. Terdapat tiga teknik pengendalian yang sering digunakan, yaitu *forward chaining*, *backward chaining*, dan gabungan dari kedua teknik pengendalian tersebut. Secara umum ada dua teknik utama yang digunakan mesin inferensi untuk pengujian aturan, yaitu: penalaran maju (*forward reasoning*/*forward chaining*) dan penalaran mundur (*backward chaining*).

a. *Forward Chaining*

   Pencocokan fakta atau pernyataan dimulai dari bagian

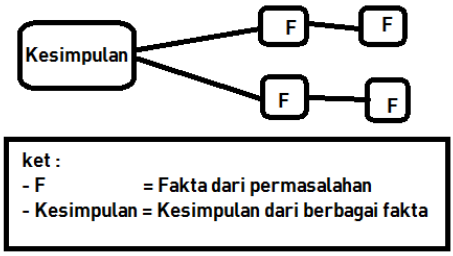
sebelah kiri dulu (IF dulu). Dengan kata lain penalaran dimulai dari fakta terlebih dahulu untuk menguji kebenaran hipotesis.



Gambar 2.3 *Forward Chaining*

b. *Backward Chaining*

Pencocokan fakta atau pernyataan dimulai dari bagian sebelah kanan (THEN dulu). Dengan kata lain penalaran dimulai dari hipotesis terlebih dahulu dan untuk menguji kebenaran hipotesis tersebut harus dicari fakta-fakta yang ada dalam basis pengetahuan.

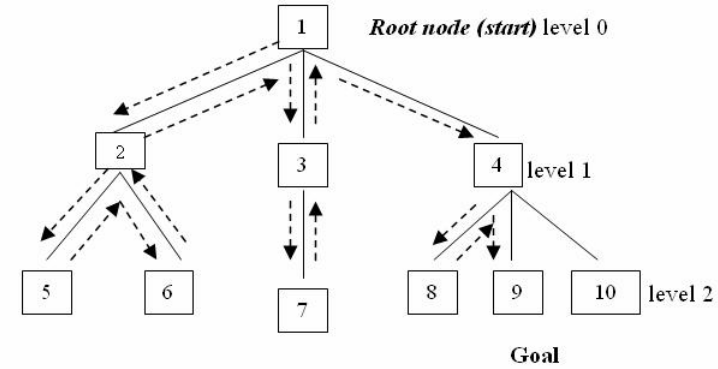


Gambar 2.4 *Backward Chaining*

Kedua metode inferensi tersebut dipengaruhi oleh tiga macam teknik penulusuran:

- **Dept-first search**

  Depth-first search Merupakan metode penelusuran yang dimulai dari akar (level 0) dan dilanjutkan dengan penelusuran *node* paling kiri yang berada pada level di bawahnya sampai dasar dari level. Bila tidak ditemukan *goal* maka pencarian diteruskan pada level 1 dan seterusnya (Gambar 2.5).

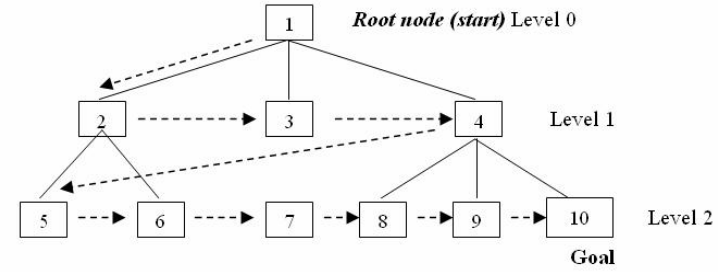


Gambar 2.5 *Depth-first Search*

- **Breadth-first search**

  Breadth-first search merupakan metode penelusuran yang dimulai dari akar (level 0) dan dilanjutkan ke level selanjutnya. Pelacakan ini dilakukan dengan menelusuri pada semua *node* yang mempunyai level yang sama sampai menemukan *goal* pada level

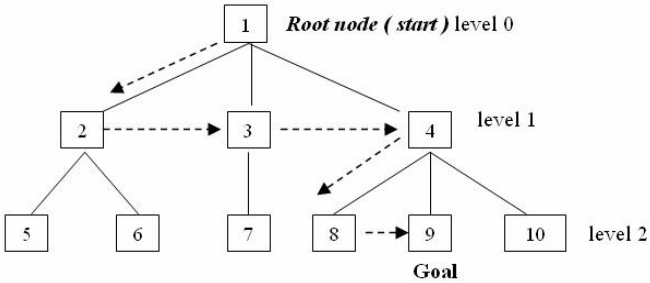
tersebut. Bila tidak ditemukan maka akan pindah ke level selanjutnya (Gambar 2.6).



Gambar 2.6 *Breadth-first Search*

- **Best-first search**

  Best-first search merupakan gabungan dari kedua metode sebelumnya, di mana dalam mencari *goal* penelusuran menggunakan kedua metode tersebut. Pencarian jenis ini dikenal juga sebagai *heuristic*. Pendekatan yang dilakukan adalah mencari solusi yang terbaik berdasarkan pengetahuan yang dimiliki sehingga penelusuran dapat ditentukan harus dimulai dari mana. Keuntungan jenis penelusuran ini adalah mengurangi beban komputasi karena hanya solusi yang memberi harapan saja yang akan diuji dan akan berhenti apabila solusi sudah mendekati yang terbaik (Gambar 2.7).

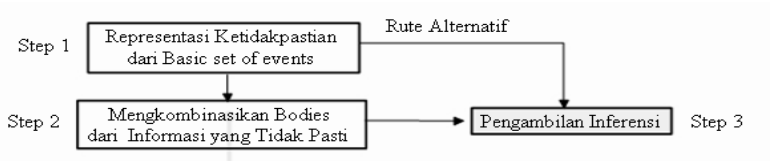


Gambar 2.7 *Best-first Search*

Untuk sebuah sistem pakar yang besar, dengan jumlah "rule" yang relatif banyak, metode pelacakan ke depan akan dirasakan sangat lamban dalam pengambilan kesimpulan, sehingga untuk sistem-sistem yang besar biasannya digunakan metode pelacakan ke belakang.

Inferensi dengan Ketidakpastian

Ketidakpastian dalam AI digambarkan dalam proses tiga tahap seperti Gambar 2.8.



Gambar 2.8 Ketidakpastian AI

Step 1 : Pakar memperoleh pengetahuan yang tidak pasti, berupa numerik, grafik atau simbolik ("hampir pasti bahwa ....")

Step 2 : Pengetahuan yang tidak pasti dapat digunakan untuk menarik kesimpulan dalam kasus sederhana (step 3)

Step 3 : Maksud dari sistem berbasis pengetahuan adalah untuk penarikan kesimpulan

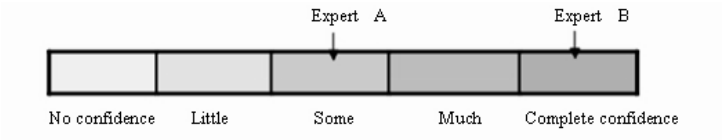
Representasi Ketidakpastian

- **Numerik**

  Metode yang paling umum untuk merepresentasikan ketidakpastian adalah numerik, dengan menggunakan skala dari 2 angka ekstrim. Biasanya 0 menggambarkan sangat ketidakpastian, sedangkan 1 atau 100 menggambarkan sangat kepastian. Permasalahannya, pakar memberikan angka tertentu sesuai dengan kognisi dan pengalamannya sendiri, selain itu orang cenderung tidak konsisten dalam menilai.

- **Grafik**

  Orang sering sulit mengerti angka-angka. Dengan menggunakan horizontal *bar*, dapat membantu pakar dalam menggambarkan kepercayaannya dalam *event* tertentu. Masalah dengan grafik adalah tidak seakurat numerik. Beberapa pakar tidak biasa memberikan angka dalam skala, mereka lebih suka memberi *ranking*.



Gambar 2.9 Representasi Ketidakpastian Grafik

- **Simbolik**

  Contohnya adalah Likert *scale*, *ranking*, dan *analytical hierarchy process* (AHP).

Probabilitas dan Pendekatan Lainnya

- **Rasio Probabilitas**

  Derajat keyakinan dari kepercayaan dalam suatu premise atau konklusi dapat dinyatakan dengan probabilitas:

$$\mathbf{P}(x) = \frac{\textbf{Jumlah outcome dari occurence } x}{\textbf{Jumlah seluruh events}} \qquad (2.1)$$

  Jika $P_1 = 0.9$, $P_2 = 0.7$, dan $P_3 = 0.65$, maka P dengan *Bayesian approach*

$$\mathbf{P}(\mathbf{A}_1|\mathbf{B}) = \frac{\mathbf{P}(\mathbf{A}_1) * \mathbf{P}(\mathbf{B}|\mathbf{A}_1)}{\mathbf{P}(\mathbf{B}|\mathbf{A}_1) * \mathbf{P}(\mathbf{A}_1) + \ldots + \mathbf{P}(\mathbf{B}|\mathbf{A}_\mathbf{n}) * \mathbf{P}(\mathbf{A}_\mathbf{n})} \qquad (2.2)$$

  di mana $P(A_1) + P(A_2) + ... + P(A_n) = 1$

- **Dempster - Shafer Approach**

  Dempster-Shafer *approach* membedakan antara ketidakpastian dengan *ignore* dengan membuat *belief funcion*.

- **Certainty Factors and Beliefs**

  Certainty factors (CF) menyatakan kepastian (*belief*) dalam suatu *event* (atau fakta, atau hipotesis) didasarkan kepada *evidence* (atau *expert's assessment*)

  CF[P,E] = MB[P,E] - MD[P,E] (2.3)

  Di mana:

  CF = *certainty factor*

  MB = *measure of belief*

  MD = *measure of disbelief*

  P = *probability*

  E = *evidence* atau *event*

  Mengkombinasikan beberapa *certainty factors* dalam satu *rule*, dengan operator AND:

  IF *inflation is high*, CF = 50 percent, (A), dan

  IF *unemployment rate is above* 7 *percent*, CF = 70 *percent*, (B), dan

  IF *bond prices decline*, CF = 100 *percent*, (C) THEN *stock prices decline*

  CF(A, B, dan C) = minimum[CF(A),CF(B),CF(C)] (2.4)

  Jika 2 CF dianggap benar, maka konklusi mempunyai maksimum CF dari 2 CF yang ada:

  CF(A atau B) = maksimum[CF(A), CF(B)] (2.5)

- **Kombinasi dari 2 Atau Lebih Rules**

  Contoh:

  R1 : IF *the inflation rate is less than* 5 *percent*, THEN *stock market prices go up* (CF = 0.7).

  R2 : IF *unemployment level is less than* 7 *percent*, THEN *stock market prices go up* (CF = 0.6).

$$\mathbf{CF(R_1, R_2) = CF(R_1) + CF(R_2)[1 - CF(R_1)]; \text{ atau}}$$
$$\mathbf{CF(R_1, R_2) = CF(R_1) + CF(R_2) - CF(R_1) \times CF(R_2)} \quad (2.6)$$

Efek kombinasi dihitung seperti Persamaan 2.6.

Dalam probabilitas, jika kita menggabungkan dua probabilitas yang dependen, akan didapat:

$$\mathbf{CF(R_1,R_2) = CF(R_1) \times CF(R_2)} \quad (2.7)$$

Misalkan ada *rule* baru:

R3: IF *bond price increases*, THEN *stock prices go up* (CF = 0.85) maka digunakan formula:

$$\mathbf{CF(R_1,R_2,R_3) = CF(R_1,R_2) + CF(R_3)\,[1 - CF(R_1,R_2)]} \quad (2.8)$$

c. Basis Data (*Database*)

Basis data terdiri atas semua fakta yang diperlukan, di mana fakta-fakta tersebut digunakan untuk memenuhi kondisi dari kaidah-kaidah dalam sistem. Basis data menyimpan semua fakta, baik fakta awal pada saat sistem mulai beroperasi, maupun fakta-fakta yang diperoleh pada

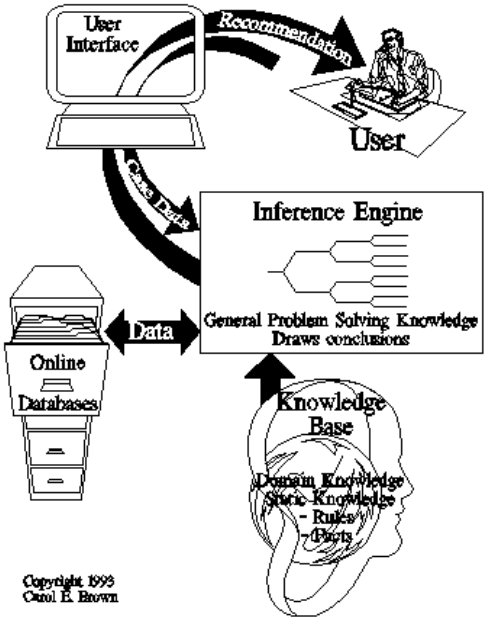
saat proses penarikan kesimpulan sedang dilaksanakan. Basis data digunakan untuk menyimpan data hasil observasi dan data lain yang dibutuhkan selama pemrosesan.

d. Antarmuka Pemakai (*User Interface*)

Fasilitas ini digunakan sebagai perantara komunikasi antara pemakai dengan sistem. *User interface* dirancang untuk mempermudah dialog dua arah antara sistem dan pemakai dengan menampilkan teknik tanya jawab dan pengisian formulir kemudian muncul bahasa perintah & menu *electronic spreadesheet* dan sistem manajemen basis data (DBMS). Hubungan antar komponen penyusun struktur sistem pakar dapat dilihat pada Gambar 2.10.

e. Akuisisi Pengetahuan (Knowledge Acquisition)

Akuisisi pengetahuan adalah akumulasi, transfer, dan transformasi keahlian dalam menyelesaikan masalah dari sumber pengetahuan ke dalam program komputer. Dalam tahap ini *knowledge engineer* berusaha menyerap pengetahuan untuk selanjutnya ditransfer ke dalam basis pengetahuan.



Gambar 2.10 Komponen Struktur Sistem Pakar

Pengetahuan diperoleh dari pakar, dilengkapi dengan buku, basis data, laporan penelitian dan pengalaman pemakai. Metode akuisisi pengetahuan:

- **Wawancara**

  Metode yang paling banyak digunakan, yang melibatkan pembicaraan dengan pakar secara langsung dalam suatu wawancara.

- **Analisis protokol**

  Dalam metode ini pakar diminta untuk melakukan suatu pekerjaan dan mengungkapkan proses

pemikirannya dengan menggunakan kata-kata. Pekerjaan tersebut direkam, dituliskan, dan dianalisis.

- **Observasi pada pekerjaan pakar**

  Pekerjaan dalam bidang tertentu yang dilakukan pakar direkam dan diobservasi.

- **Induksi aturan dari contoh**

  Induksi adalah suatu proses penalaran dari khusus ke umum. Suatu sistem induksi aturan diberi contoh-contoh dari suatu masalah yang hasilnya telah diketahui. Setelah diberikan beberapa contoh, sistem induksi aturan tersebut dapat membuat aturan yang benar untuk kasus-kasus contoh. Selanjutnya aturan dapat digunakan untuk menilai kasus lain yang hasilnya tidak diketahui.

f. Workplace/Blackboard/Memory Processing

Memory Processing atau *Working Memory* merupakan bagian dari sistem pakar yang berisi fakta-fakta masalah yang ditemukan dalam suatu sesi berisi fakta-fakta tentang suatu masalah yang ditemukan dalam proses konsultasi *Workplace* merupakan area dari sekumpulan memori kerja (*working memory*), digunakan untuk merekam kejadian yang sedang berlangsung termasuk keputusan sementara. Ada 3 keputusan yang dapat direkam:

- Rencana : bagaimana menghadapi masalah

- Agenda : aksi-aksi yang potensial yang sedang menunggu untuk dieksekusi
- Solusi : calon aksi yang akan dibangkitkan

g. Fasilitas Penjelasan

Merupakan komponen tambahan yang akan meningkatkan kemampuan sistem pakar. Digunakan untuk melacak respon dan memberikan penjelasan tentang kelakuan sistem pakar secara interaktif melalui pertanyaan:

- Mengapa suatu pertanyaan ditanyakan oleh sistem pakar?
- Bagaimana konklusi dicapai?
- Mengapa ada alternatif yang dibatalkan?
- Rencana apa yang digunakan untuk mendapatkan solusi?

h. Perbaikan Pengetahuan

Pakar memiliki kemampuan untuk menganalisis dan meningkatkan kinerjanya serta kemampuan untuk belajar dari kinerjanya. Kemampuan tersebut adalah penting dalam pembelajaran terkomputerisasi, sehingga program akan mampu menganalisis penyebab kesuksesan dan kegagalan yang dialaminya dan juga mengevaluasi apakah pengetahuan-pengetahuan yang ada masih cocok untuk digunakan di masa mendatang.

2.7 Fase-Fase Pengembangan Sistem Pakar

1. Identifikasi

   Tahap ini merupakan tahap penentuan hal-hal penting sebagai dasar permasalahan yang akan dianalisis. Tahap ini merupakan tahap untuk mengkaji dan membatasi masalah yang akan diimplementasikan dalam sistem. Setiap masalah yang diidentifikasikan harus dicari solusi, fasilitas yang akan dikembangkan, penentuan jenis bahasa pemrograman dan tujuan yang ingin dicapai dari peroses pengembangan tersebut. Pada tahap ini seorang pengembang (*developer*) mengumpulkan data-data dari pakar mengenai suatu permasalahan yang akan dibuat menjadi sistem pakar.

2. Konseptualisasi

   Hasil identifikasi masalah dikonseptualisasi dalam bentuk hubungan antar data, hubungan antar pengetahuan dan konsep-konsep penting dan ideal yang akan diterapkan dalam sistem. Konseptualisasi juga menganalisis data-data penting yang perlu dibahas secara detail dengan pakar. Pengembang merancang *knowledge base* dan *working memory*, pengembang juga melakukan representasi pengetahuan berdasarkan data-data yang diperoleh.

3. Formalisasi

   Pada tahap formalisasi konsep-konsep tersebut diimplemetasikan secara formal. Misalnya pertimbangkan

beberapa faktor pengambilan keputusan. Pengembang melakukan perancangan pengambilan keputusan yang akan dilakukan oleh mesin inferensi, misalnya menggunakan *backward chaining*, *forward chaining*, dan sebagainya.

4. Implementasi

   Pengembang mengimplementasikan hasil rancangan yang telah dilakukan pada tahap konseptualisasi dan formalisasi ke dalam suatu bahasa pemrograman tertentu. Tetapi pengembang terlebih dahulu merancang *interface* yang akan digunakan.

5. Evaluasi

   Setelah proses implementasi selesai, sebagaimana lazimnya dalam proses pembuatan perangkat lunak, perlu dilakukan evaluasi/*testing* pada perangkat lunak tersebut dengan kaidah-kaidah yang telah ditentukan dalam *software engineering*.

6. Pengembangan sistem lebih lanjut

   Pengembangan sistem diperlukan karena terkadang suatu permasalahan tidaklah statis tetapi terkadang juga dinamis, misalnya ditemukan aturan-aturan baru dalam permasalahan itu.

2.8 Pemeran dalam Suatu Proyek Sistem Pakar

1. *Domain Expert*

Definisi: orang yang memiliki keterampilan (*skill*) dan pengetahuan (*knowledge*) untuk menyelesaikan masalah khusus dengan cara-cara yang superior dibanding orang kebanyakan.

- Memiliki pengetahuan kepakaran
- Memiliki keterampilan *problem-solving* yang efisien
- Dapat mengkomunikasikan pengetahuan
- Dapat menyediakan waktu
- Dapat bekerja sama

2. *Knowledge Engineer*

Definisi: orang yang melakukan proses desain, mengembangkan dan menguji suatu sistem pakar.

- Memiliki ketrampilan rekayasa pengetahuan (*knowledge engineering*)
- Memiliki keterampilan komunikasi yang baik
- Dapat menyesuaikan masalah kepada *software*
- Memiliki ketrampilan pemrograman sistem pakar

3. *End-User*

- Dapat membantu mendefinisikan spesifikasi *interface*
- Dapat membantu proses akuisisi pengetahuan
- Dapat membantu proses pengembangan sistem

Ciri-ciri Sistem Pakar

- Terbatas pada domain keahlian tertentu
- Dapat memberikan penalaran untuk data yang tidak pasti**.**
- Dapat mengemukakan rangkaian alasan-alasan yang diberikannya dengan cara yang dapat dipahami.
- Berdasarkan pada kaidah/ketentuan/*rule* tertentu**.**
- Dirancang untuk dapat dikembangkan secara bertahap**.**
- *Output*-nya bersifat anjuran**.**

2.9 Kategori Masalah Sistem Pakar Secara Umum

1. **Interpretasi**

   Pengambilan keputusan dari hasil observasi, di antaranya: pengawasan, pengenalan ucapan, analisis citra, interpretasi sinyal, dan beberapa analisis kecerdasan.

2. **Prediksi**

   Memprediksi akibat-akibat yang dimungkinkan dari situasi-situasi tertentu, di antaranya: peramalan, prediksi demografis, peramalan ekonomi, prediksi lalulintas, estimasi hasil, militer, pemasaran atau peramalan keuangan.

3. **Diagnosis**

   Menentukan sebab malfungsi dalam situasi kompleks yang didasarkan pada gejala-gejala yang teramati, di antaranya: medis, elektronis, mekanis, dan diagnosis perangkat lunak.

4. **Desain**

   Menentukan konfigurasi komponen-komponen sistem yang cocok dengan tujuan-tujuan kinerja tertentu dan kendala-kendala tertentu, di antaranya: *layout* sirkuit, perancangan bangunan.

5. **Perencanaan**

   Merencanakan serangkaian tindakan yang akan dapat mencapai sejumlah tujuan dengan kondisi awal tertentu, di antaranya: perencanaan keuangan, komunikasi, militer, pengembangan politik, *routing* dan manajemen proyek.

6. **Monitoring**

   Membandingkan tingkah laku suatu sistem yang teramati dengan tingkah laku yang diharapkan darinya, di antaranya: *Computer Aided Monitoring System*.

7. **Debugging dan repair**

   Menentukan dan mengimplementasikan cara-cara untuk mengatasi malfungsi, di antaranya memberikan resep obat terhadap suatu kegagalan.

8. **Instruksi**

   Melakukan instruksi untuk diagnosis, *debugging* dan perbaikan kinerja.

9. **Kontrol**

   Mengatur tingkah laku suatu *environment* yang kompleks seperti kontrol terhadap interpretasi, prediksi, perbaikan, dan *monitoring* kelakuan sistem.

10. **Seleksi**

    Mengidentifikasi pilihan terbaik dari sekumpulan (*list*) kemungkinan.

11. **Simulasi**

    Pemodelan interaksi antara komponen-komponen sistem.

2.10 Bentuk/Tipe Sistem Pakar

Adapun bentuk atau tipe sistem pakar sebagai berikut:

1. **Mandiri**

   Sistem pakar yang murni berdiri sendiri, tidak digabung dengan *software* lain, bisa dijalankan pada komputer pribadi, *mainframe*.

2. **Terkait/Tergabung**

   Dalam bentuk ini sistem pakar hanya merupakan bagian dari program yang lebih besar. Program tersebut biasanya menggunakan teknik algoritma konvensional tapi bisa mengakses sistem pakar yang ditempatkan sebagai subrutin, yang bisa dimanfaatkan setiap kali dibutuhkan.

3. **Terhubung**

   Merupakan sistem pakar yang berhubungan dengan *software* lain misal: *spreadsheet*, DBMS, program grafik. Pada saat proses inferensi, sistem pakar bisa mengakses data dalam *spreadsheet* atau DBMS atau program grafik bisa dipanggil untuk menayangkan *output* visual.

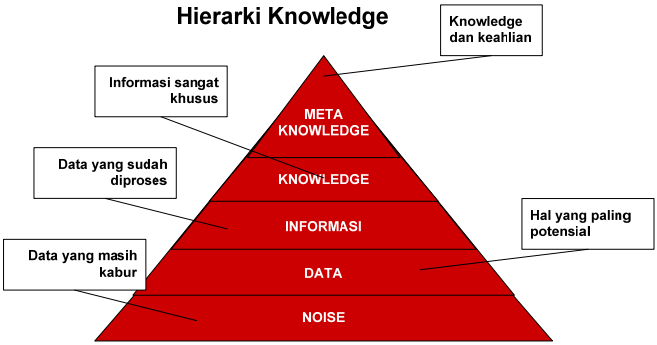
4. **Sistem Mengabdi**

   Merupakan bagian dari komputer khusus yang diabdikan kepada fungsi tunggal. Sistem tersebut bisa membantu analisis data radar dalam pesawat tempur atau membuat keputusan intelijen tentang bagaimana memodifikasi pembangunan kimiawi.

2.11 Pengetahuan (*Knowledge*)

Pengetahuan adalah fakta atau kondisi sesuatu atau keadaan yang timbul karena suatu pengalaman. Cabang ilmu yang mempelajari tentang pengetahuan disebut epistomologi. Epistomologi dibagi menjadi lima, yaitu:

- Pengetahuan Priori → Disebut secara universal benar dan tidak dapat ditentukan tanpa kontradiksi
- Pengetahuan Posteriori → Diperoleh dari arti. Kebenaran dari pengetahuannya menggunakan pengalaman
- Pengetahuan Prosedural → Bagaimana melakukan sesuatu
- Pengetahuan Deklaratif → Mengacu pada benar/salah
- Pengetahuan *Tacit*/*Unconscious* → Tidak dapat diekspresikan dengan bahasa.



Gambar 2.11 Hierarki Pengetahuan

2.12 Teknik Representasi Pengetahuan

Representasi pengetahuan adalah suatu teknik untuk merepresentasikan basis pengetahuan yang diperoleh ke dalam suatu skema/diagram tertentu sehingga dapat diketahui relasi/keterhubungan antara suatu data dengan data yang lain. Teknik ini membantu *knowledge engineer* dalam memahami struktur pengetahuan yang akan dibuat sistem pakarnya.

Representasi pengetahuan sangat penting karena:

1. *Shell* dalam sistem pakar didesain untuk tipe representasi pengetahuan tertentu seperti baris dan logika.
2. Akan memberikan efek/akibat pengembangan, efisiensi, kecepatan dan perawatan sistem.

Representasi Pengetahuan (*knowledge representation*) dimaksudkan untuk menangkap sifat-sifat penting masalah dan membuat infomasi dapat diakses oleh prosedur pemecahan

masalah. Bahasa representasi harus dapat membuat seorang programmer mampu mengekspresikan pengetahuan untuk mendapatkan solusi suatu masalah.

Dalam representasi pengetahuan ada dua hal yang harus diperhatikan, yaitu:

- Fakta

  Objek yang akan direpresentasikan. Ini menerangkan keadaan tentang benda yang ada dalam domain masalah. Fakta dapat berupa keterangan/kalimat dalam bahasa lamai, logika atau benda.
- Formula

  Bentuk representasi yang dapat dimanipulasi dalam proses pemecahan soal. Bentuk ini harus dapat melukiskan hubungan antara komponen-komponen domain masalah.

Dalam representasi sebuah fakta yang kita gunakan dalam sebuah program, kita juga harus konsisten dengan representasi yang menggunakan bahasa natural (bahasa Inggris). Representasi yang baik, harus:

- Mengemukakan hal secara eksplisit
- Membuat masalah menjadi transparan
- Komplit dan efisien
- Menampilkan batasan-batasan alami yang ada
- Menekan/menghilangkan detail-detail yang diperlukan
- Dapat dilakukan komputasi (ada batasan/*constraint*)

Dengan representasi, banyak hal yang akan kita dapatkan dalam menyelesaikan suatu permasalahan. Berikut ini adalah beberapa keuntungan yang akan kita dapatkan ketika kita membuat representasi pengetahuan, yaitu:

- Dengan representasi yang baik, membuat objek dan relasi yang penting menjadi jelas.
- Representasi menyingkap *constraint* (batasan) dalam suatu permasalahan. Kita dapat mengungkapkan pengaruh sebuah objek atau relasi terhadap objek atau relasi yang lain.
- Dengan representasi kita akan dapatkan objek dan relasi secara bersama-sama. Kita akan dapat melihat semua yang kita inginkan dalam satu waktu.
- Kita dapat menghilangkan semua komponen yang tidak berhubungan dengan permasalahan yang sedang kita selesaikan. Atau kita dapat menyembunyikan beberapa informasi yang tidak kita butuhkan untuk sementara dan pada saat kita membutuhkannya kita dapat menampilkan kembali.
- Dengan representasi akan membuat permasalahan yang sedang kita selesaikan menjadi transparan. Kita akan memahami permasalahan yang kita selesaikan.
- Dengan representasi kita akan dapat menyingkap suatu permasalahan secara lengkap, sehingga permasalahan dapat diselesaikan.

- Dengan representasi akan membua permasalahan menjadi ringkas. Kita akan berpikir ringkas (merepresentasikan apa yang ingin kita representasikan secara *efficient*).
- Dengan representasi, maka akan menjadikan pekerjaan kita menjadi cepat.
- Dengan representasi, menjadikan permasalahan yang kita selesaikan dapat terkomputerisasi. Dengan representasi ini kita akan dapat melakukan prosedur-prosedur dalam menyelesaikan suatu permasalahan.

Di samping keuntungan-keuntungan tersebut satu hal yang menjadi prinsip dalam representasi pengetahuan adalah Jika suatu permasalahan dideskripsikan dengan menggunakan representasi yang tepat, maka dapat dipastikan bahwa permasalahan tersebut dapat diselesaikan.

Menurut **Mylopoulos dan Levesque**, susunan atau pola klasifikasi representasi pengetahuan dibagi menjadi empat kategori, yaitu:

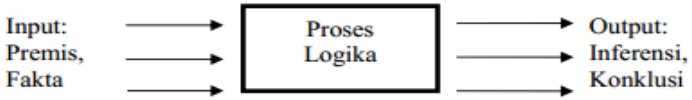
1. Representasi Logika/*Rule-Based Knowledge*
   Representasi ini menggunakan ekspresi-ekspresi dalam logika formal untuk merepresentasikan basis pengetahuan.
2. Representasi Prosedural/*Case-Base Reasoning*
   Menggambarkan pengetahuan sebagai sekumpulan instruksi untuk memecahkan suatu masalah. Dalam

sistem yang berbasis aturan, aturan if-then dapat ditafsirkan sebagai sebuah prosedur untuk mencapai tujuan pemecahan masalah.

3. Representasi Network/*Object-Based Knowledge*
   Menyatakan pengetahuan sebagai sebuah graf di mana simpul-simpulnya menggambarkan objek atau konsep dalam masalah yang dihadapi, sedangkan lengkungannya menggambarkan hubungan antar mereka. Contohnya adalah jaringan semantik.
4. Representasi Terstruktur/*Frame-Based Knowledge*
   Memperluas *network* dengan cara membuat setiap simpulnya menjadi sebuah struktur data kompleks yang berisi tempat-tempat bernama *slot* dengan nilai-nilai tertentu. Nilai-nilai ini dapat merupakan data numerik atau simbolik sederhana, *pointer* ke bingkai (*frame*) lain, atau bahkan merupakan prosedur untuk mengerjakan tugas tertentu. Contoh: skrip (*script*), bingkai (*frame*) dan objek (*object*).

2.13 Representasi Logika

Pada dasarnya proses logika adalah proses membentuk kesimpulan atau menarik inferensi berdasarkan fakta yang ada. Input dari proses logika berupa premis atau fakta yang diakui kebenarannya sehingga dengan melakukan penalaran pada proses logika dapat dibentuk suatu inferensi atau kesimpulan yang benar pula.



Gambar 2.12 Proses Logika

Ada dua penalaran untuk mendapatkan konklusi:

1. **Penalaran Deduksi.**

   Penalaran dimulai dari prinsip umum untuk mendapatkan konklusi yang lebih khusus.

2. **Penalaran Induktif**

   Penalaran dimulai dari fakta-fakta khusus untuk mendapatkan kesimpulan umum.

Pada representasi logika ini ada dua metode yaitu:

1. **Logika Proposisi**
2. **Kalkulus Predikat**

1. Logika preposisi

Proposisi adalah pernyataan yang dapat bernilai benar (B) atau salah (S). Simbol-simbol seperti P dan Q menunjukkan proposisi. Dua atau lebih proposisi dapat digabungkan dengan menggunakan operator logika yaitu:

- Konjungsi
- Disjungsi
- Negasi

- Implikasi
- Ekuivalensi

Untuk nilai kebenaran dari semua operator logika dapat dilihat di table kebenaran (*truth table*) berikut.

Tabel 2.1 Tabel Kebenaran

P	Q	P AND Q	P OR Q	P → Q	P ⇔ Q
B	B	B	B	B	B
B	S	S	B	S	S
S	B	S	B	B	S
S	S	S	S	B	B

2. Kalkulus Predikat

Kalkulus Predikat adalah satu formula yang terdiri atas predikat, variabel, konstanta atau fungsi. Kalkulus predikat digunakan untuk merepresentasikan hal-hal yang tidak dapat direpresentasikan dengan menggunakan logika proposisi. Pada logika predikat (kalkulus predikat) kita dapat merepresentasikan fakta-fakta sebagai pernyataan yang disebut dengan *wff (well formed formula)*.

Contoh :

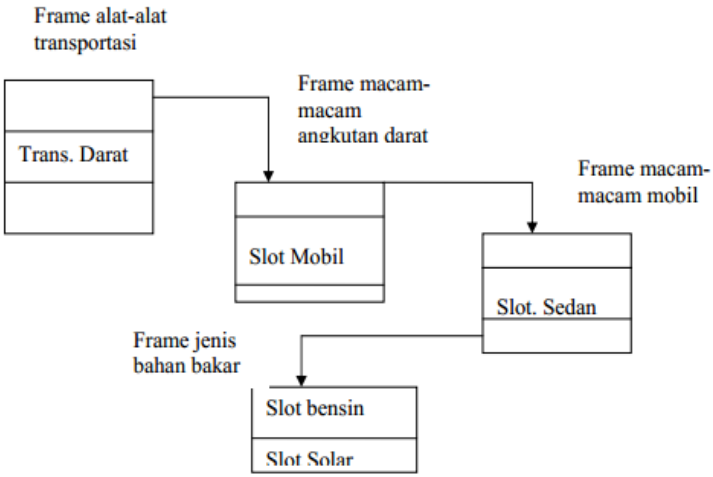
- **WARNA (RUMAH, MERAH):**

Predikat ini menggambarkan warna rumah merah, di mana WARNA adalah predikat, RUMAH dan MERAH adalah sautu konstanta.

- **WARNA (x,MERAH):**
  x adalah variabel yang menyatakan sembarang benda yang berwarna merah.
- **WARNA (x,y):**
  Predikat ini menyatakan suatu sifat warna antara variabel x dan y.

2.14 Bingkai (*Frame*)

Frame merupakan kumpulan pengetahuan tentang suatu objek tertentu, peristiwa, lokasi, situasi dan lain-lain. *Frame* memiliki *slot* yang menggambarkan rincian (atribut) dan karakteristik objek. *Frame* biasanya digunakan untuk merepresentasikan pengetahuan yang berdasarkan karakteristik yang sudah dikenal yang merupakan pengalaman-pengalaman. Dengan menggunakan *frame*, sangat mudah untuk membuat inferensi tentang objek karena menyediakan basis pengetahuan yang berasal dari pengalaman. Contoh *frame*:



Gambar 2.13 Proses Logika

Gambar 2.13 menunjukkan *frame* alat-alat trasnportasi. *Frame* memiliki 3 *slot*, yaitu alat-alat transportasi di darat, udara, dan laut. Ada beberapa *slot* yang bernilai tetap dan tidak tetap (prosedural). *Slot* yang bernilai tetap misal jumlah roda pada sedan (4). Jenis *slot* lain yang bersifat prosedural artinya *slot* yang memungkinkan penambahan informasi baru yang bisa ditambahkan pada aturan IF. Misal informasi tentang kecepatan perjalanan, pengisian tangki bahan bakar atau pemakaian bahan bakar tiap km.

2.15 Naskah (*Script*)

Script adalah skema representasi pengetahuan yang hampir sama dengan *frame*. Perbedaannya *frame*

menggambarkan objek sedangkan *script* menggambarkan urutan peristiwa. Dalam menggambarkan urutan peristiwa, skrip menggunakan *slot* yang berisi informasi tentang orang, objek, dan tindakan yang terjadi dalam suatu peristiwa.

Elemen-elemen *script* meliputi:

1. Kondisi *input*, yaitu kondisi yang harus dipenuhi sebelum terjadi suatu peristiwa.
2. *Track*, yaitu variasi yang mungkin terjadi dalam suatu *script*.
3. *Prop*, berisi objek-objek pendukung yang digunkana selama peristiwa terjadi.
4. *Role*, yaitu peran yang dimainkan oleh seseorang dalam peristiwa.
5. *Scene*, yaitu adegan yang dimainkan.
6. Hasil, yaitu kondisi yang ada setelah urutan peristiwa yang terjadi.

Contoh:

Jalur (*track*)	:	Ujian tertulis mata kuliah Kecerdasan Buatan
Role (peran)	:	Mahasiswa, Pengawas.
Prop (pendukung):		lembar soal, lembar jawab, presensi, pena dan lain-lain.
Kondisi *input*	:	Mahasiswa terdaftar untuk mengikuti ujian

Adegan (*scene*) – 1 : Persiapan pengawas

- Pengawas menyiapkan lembar soal.
- Pengawas menyiapkan lembar jawab
- Pengawas menyiapkan lembar kehadiran siswa

Adegan - 2 : Mahasiswa masuk ruangan

- Pengawas mempersilahkan mahasiswa untuk masuk ruangan
- Pengawas membagikan lembar soal
- Pengawas membagikan lembar jawab
- Pngawas memimpin doa

Adegan – 3 : Mahasiswa mengerjakan soal ujian

- Mahasiswa menulis identitas di lembar jawab
- Mahasiswa menandatangani lembar jawab
- Mahasiswa mengerjakan soal
- Mahasiswa mengecek jawaban

Adegan – 4 : Mahasiswa telah selesai ujian

- Pengawas mempersilahkan mahsiswa keluar ruangan
- Mahasiswa mengumpulkan kembali lembar jawab
- Mahasiswa keluar ruangan

Adegan - 5 : Pengawas mengemasi lembar jawab

- Pengawas mengurutkan lembar jawab
- Pengawas mengecek lembar jawab dan mengurutkan
- Pengawas meninggalkan ruangan

Hasil:

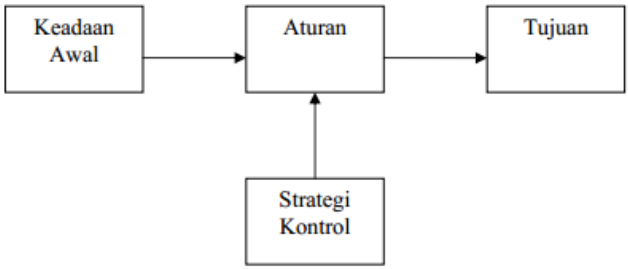
- Mahasiswa merasa lega dan senang sudah ujian

- Mahasiswa merasa kecewa
- Mahasiswa pusing
- Mahasiswa sangat bersyukur

2.16 Sistem Produksi

Sistem Produksi secara umum terdiri atas komponen-komponen:

1. Ruang keadaan, yang berisi keadaan awal (*initial state*), tujuan (*goal state*) dan kumpulan aturan (*production rule*) yang digunakan untuk mencapai tujuan.
2. Strategi Kontrol, yang berguna untuk mengarahkan bagaimana proses pencarian akan berlangsung dan mengendalikan arah eksplorasi.



Gambar 2.14 Sistem Produksi

Sistem produksi ini sangat populer dan banyak digunakan. Representasi pengetahuan dengan sistem produksi pada dasarnya berupa aplikasi aturan yang berupa:

1. *Antecedent*, yaitu bagian yang mengekspresikan situasi atau premis (pernyataan berawalan IF).

2. Konsekuen, yaitu bagian yang menyatakan suatu tindakan tertentu atau konklusi yang diterapkan jika situasi atau premis bernilai benar (berawalan THEN).

Ada beberapa faktor yang mempengaruhi pemilihan *forward reasoning* atau *backward reasoning* dalam memilih metode penalaran:

- Banyaknya keadaan awal dan tujuan. Jika jumlah keadaan awal lebih kecil dari pada tujuan, maka digunakan *forward reasoning* dan sebaliknya.
- Rata-rata jumlah *node* yang dapat diraih secara langsung dari suaru *node*. Lebih baik dipilih yang jumlah *node* tiap cabangnya lebih sedikit.
- Apakah program butuh menanyai *user* untuk melakukan justifikasi terhadap proses penalaran? Jika iya, maka dipilih yang memudahkan *user*.
- Bentuk kejadian yang akan memicu penyelesaian masalah. Jika kejadian itu berupa fakta baru, maka dipilih penalaran *forward reasoning*, tapi jika kejadian berupa *query*, maka lebih baik digunakan *backward reasoning*.

Adapun perbandingan antara seorang pakar dan sistem pakar adalah seperti Tabel 2.2.

Tabel 2.2 Perbandingan Pakar dan Sistem Pakar

Faktor	Human Expert	Expert System
Time Availability	Hari kerja	Setiap saat
Geografis	Lokal/tertentu	Di mana saja
Keamanan	Tidak tergantikan	Dapat diganti
Perishable/Dapat habis	Ya	Tidak
Performansi	Variabel	Konsisten
Kecepatan	Variabel	Konsisten & lebih cepat
Biaya	Tinggi	Terjangkau

2.17 Perbedaan antara Sistem Konvensional dengan Sistem Pakar

Sistem konvensional yang dimaksud dalam hal ini adalah suatu sistem yang tidak mengimplementasikan AI ke dalam sistem tersebut atau dengan kata lain sistem yang tidak memiliki representasi pengetahuan yang tertanam dalam sistem. Perbedaan sistem konvensional dan sistem pakar , adalah sebagai berikut:

Tabel 2.3 Perbandingan Sistem Konvensional dan Sistem Pakar

Sistem Konvensional	Sistem Pakar
Informasi dan pemrosesannya biasanya jadi satu dengan program	Basis pengetahuan merupakan bagian terpisah dari mekanisme inferensi
Program tidak pernah salah (kecuali pemrogramnya yang salah)	Program bisa saja melakukan kesalahan
Biasanya tidak bisa menjelaskan mengapa suatu input data itu dibutuhkan atau bagaimana output itu diperoleh	Penjelasan adalah bagian terpenting dari sistem pakar
Pengubahan program cukup sulit dan merepotkan	Pengubahan pada aturan/kaidah dapat dilakukan dengan mudah
Sistem hanya akan bekerja jika sistem tersebut sudah lengkap	Sistem dapat bekerja hanya dengan beberapa aturan
Eksekusi dilakukan langkah demi langkah secara algoritmik	Eksekusi dilakukan pada keseluruhan basis pengetahuan secara heuristik dan logis
Menggunakan data	Menggunakan pengetahuan
Tujuan utamanya adalah efisiensi	Tujuan utamanya adalah efektivitas

Beberapa Contoh Sistem Pakar

1. Dendral
   - Merupakan produk peneliti di Universitas Stanford.
   - Dengan menggunakan pengetahuan struktur molekular dan kimia

- Berusaha mengidentifikasi struktur molekul campuran yang tak dikenal.

2. Prospector
   - Membantu ahli geologi dalam mencari & menemukan hasil tambang di bumi.
   - Berisi bentuk-bentuk kandungan mineral/batu-batuan yang diperoleh dari para ahli geologi.
3. XCON & XSEL

   XCON
   - Membantu konfigurasi sistem komputer besar.
   - Membantu melayani order langganan sistem komputer DEC VAX II/780 le dalam sistem spesifikasi final yang lengkap.

   XSEL
   - Membantu karyawan bagian penjualan dalam memilih komponen sistem VAX (dengan pengetahuannya sistem komputer VAX II/780).
   - Pengetahuan dalam XSEL membantu untuk memilih konfigurasi yang dikehendaki, kemudian XSEL memilih CPU, memori, periferal, dalam menyarankan paket *software* tertentu yang paling tepat dengan konfigurasinya.
4. ACE (AT & T), digunakan untuk memberikan laporan *trouble-shooting* dan analisis untuk perawatan kabel telepon.

5. AS/ASQ (Arthur Young), digunakan untuk membantu dalam prosedur *auditing*.
6. AUDITOR, memilihkan prosedur audit untuk memverifikasi rekening pendapatan sebuah perusahaan.
7. AUTHORIZER'S ASSISTANT (American Express), membantu dalam meninjau penipuan kartu kredit.
8. BUSINESS PLAN (Sterling Wentworth Corp.), membantu pegawai professional dan pemilik bisnis tentang semua aspek yang menyangkut perencanaan keuangan.
9. CASH VALUE (Heuros Ltd.), mendukung perencanaan proyek modal.
10. COMPASS (GTE Corp.), troubleshoots tidak berfungsinya sirkuit telepon.
11. CONCEPT (Tyashare), memproduksi model-model dari pasar yang disenangi konsumen.
12. DELTA (GE), membantu mendiagnosa dan memperbaiki kereta api listrik diesel.
13. EXPERT TAX (Coopers & Lybrand), memberikan bimbingan menghitung pajak.
14. FIN PLAN (Wright Patterson Air Force Base), mendukung perencanaan keuangan pribadi.
15. FINANCIAL ADVISOR (Palladian), memberikan bimbingan keuangan pada proyek, produk, dan penggabungan serta akuisisi.

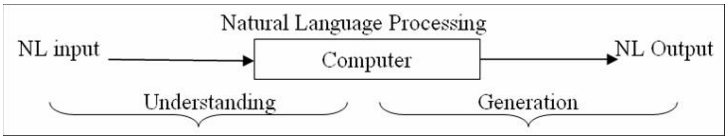
Bab 3

Natural Language Processing

3.1 *Natural Language Processing*

Natual Language Processing (NLP) atau Pengolahan Bahasa Alami (PBA) merupakan salah satu bidang ilmu kecerdasan buatan (*artificial intelligence*) yang mempelajari komunikasi antara manusia dengan komputer. *Natural Languange* adalah sebuah bahasa yang diucapkan, ditulis oleh manusia untuk berkomunikasi. Bahasa adalah sebuah sistem yang terdiri atas kumpulan simbol dan aturan (*grammar*). NLP mencakup semua yang diperlukan komputer untuk mengerti Bahasa Alami baik yang ditulis maupun diucapkan dan menghasilkan sebuah bahasa alami. NLP terdiri atas NLU dan NLG.

- **Natural Language Understanding (NLU)**
  Tugas NLU adalah memahami *input* (bahasa alami)
- **Natural Language Generation (NLG)**
  NLG adalah salah satu bagian dari NLP, NLG sering juga disebut *text generation*.



Gambar 3.1 *Natural Language Processing*

Berikut ini dijelaskan bidang-bidang pengetahuan yang berhubungan dengan NLP:

- Fonetik dan fonologi: berhubungan dengan suara yang menghasilkan kata yang dapat dikenali. Bidang ini menjadi penting dalam proses aplikasi yang memakai metoda *speech based system*.
- Morfologi: yaitu pengetahuan tentang kata dan bentuknya dimanfaatkan untuk membedakan satu kata dengan lainnya. Pada tingkat ini juga dapat dipisahkan antara kata dan elemen lain seperti tanda baca.
- Sintaksis: yaitu pemahaman tentang urutan kata dalam pembentukan kalimat dan hubungan antar kata tersebut dalam proses perubahan bentuk dari kalimat menjadi bentuk yang sistematis.
- Semantik: yaitu pemetaan bentuk struktur sintaksis dengan memanfaatkan tiap kata ke dalam bentuk yang lebih mendasar dan tidak tergantung struktur kalimat. Semantik mempelajari arti suatu kata dan bagaimana dari arti kata tersebut membentuk suatu arti dari kalimat yang utuh. Dalam tingkatan ini belum tercakup konteks dari kalimat tersebut.

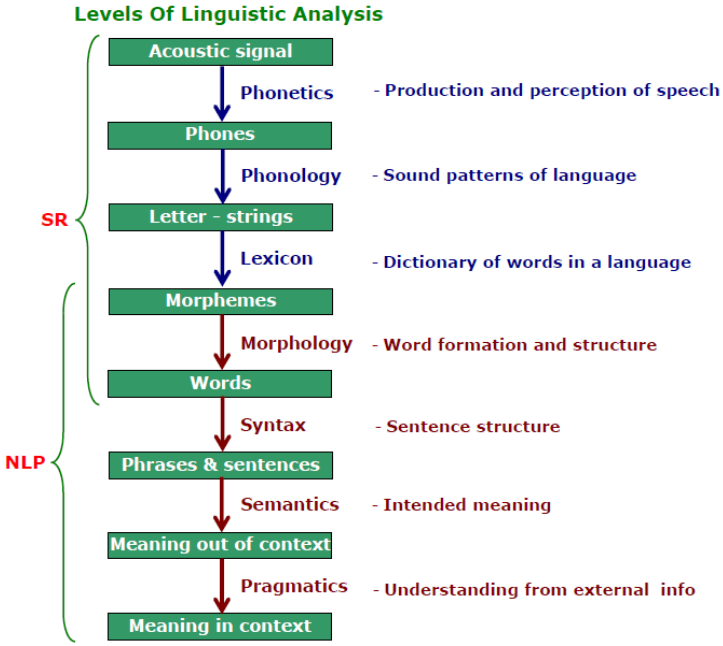
- Pragmatik: pengetahuan pada tingkatan ini berkaitan dengan masing-masing konteks yang berbeda tergantung pada situasi dan tujuan pembuatan sistem.
- *Discourse Knowledge*: melakukan pengenalan apakah suatu kalimat yang sudah dibaca dan dikenali sebelumnya akan mempengaruhi arti dari kalimat selanjutnya. Informasi ini penting diketahui untuk melakukan pengolahan arti terhadap kata ganti orang dan untuk mengartikan aspek sementara dari informasi.
- *World Knowledge*: mencakup arti sebuah kata secara umum dan apakah ada arti khusus bagi suatu kata dalam suatu percakapan dengan konteks tertentu.

Level- level pada analisis linguistik adalah sebagai berikut, level yang lebih tinggi merupakan *speech recognition*, level yang lebih rendah merupakan NLP.

Phonetic dan *phonology* adalah sesuatu yang berhubungan dengan suara yang menghasilkan kata yang dapat dikenali. Bidang ini menjadi penting dalam proses aplikasi yang memakai metoda *speech system.* Pada tahap ini sinyal bunyi akan dikenali dan menghasilkan kata-kata yang sesuai dengan bunyi tersebut. Dengan NLP komputer dapat perkerjaan sebagai berikut:

- NLP adalah mencoba untuk membuat komputer dapat mengerti perintah-perintah yang ditulis dalam standar bahasa manusia.

- NLP tidak memperdulikan bagaimana sebuah kalimat dimasukkan ke komputer tetapi menduplikasi informasi dari kalimat tersebut.



Gambar 3.2 Level Analisis Linguistik

NLP adalah pengolahan *input* data untuk membuat komputer dapat mengerti perintah-perintah yang ditulis dalam standar bahasa manusia. NLP tidak mempedulikan bagaimana sebuah kalimat dimasukkan ke komputer tetapi menduplikasi informasi dari kalimat tersebut. Inti dari NLP adalah *parser*, di mana

parser tersebut membaca setiap kalimat, kata demi kata, untuk menentukan apa yang dimaksud.

Parser terdiri atas 3 jenis:

1. *Parser State machine*

   Parser state machine menggunakan keadaan yang sesungguhnya dari kalimat untuk memprediksi tipe apa dari kata yg berlaku. *State machine*: *directed gap* yang menunjukkan transisi yang valid dari *state* ke yang lainnya.

2. *Parser context-free Recursive Descent*

   Sebuah kalimat adalah gabungan dari berbagai item dan item ini adalah gabungan dari item lain dan seterusnya sampai dipotong ke elemen-elemen seperti *noun*, *adjective* dan sebaginya. CFG merupakan suatu cara untuk menyatakan struktur dari suatu tata bahasa. CFG ini mempunyai aturan sebagai berikut: "<simbol><simbol1><simbol2><simboln>, di mana n>=1 <simbol> harus merupakan simbol bukan terminal, sedangkan <simbol1>, <simbol2>, ...<simboln> dapat merupakan simbol terminal atau bukan terminal. Bentuk tersebut mempunyai arti <simbol> dapat diganti dengan <simbol1>, <simbol2>, ...<simboln>", contoh untuk tata bahasa Indonesia yang sederhana.

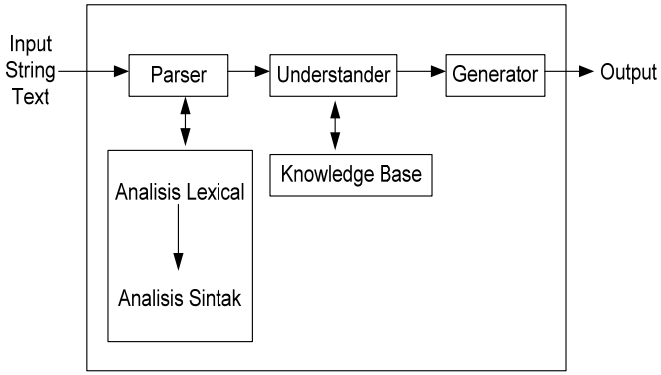
3. *Parser Noise Disposal*

   Tipe *parser* ini sesungguhnya sangat umum dalam aplikasi tipe *database* seperti *command processor*.

Hal-hal yang bertentangan dengan pendekatan NLP sebagai berikut:

- Bahwa sesungguhnya NLP menggunakan semua informasi dalam sebuah kalimat, hanya manusia yang dapat melakukan hal tersebut.
- Mencoba memperbolehkan komputer menerima perintah bahasa alami, tetapi hanya menduplikasi inti informasi pada perintah (*command*).

Konsep NLP



Gambar 3.3 Diagram Alir NLP/Tipe Analisis Semantik

3.2 Pemahaman Kalimat

Bagian dari proses pemahaman adalah menggabungkan kata-kata untuk membentuk suatu struktur yang memiliki arti. Karena begitu banyak hal yang terkait dengan proses pemahaman suatu kalimat, yaitu:

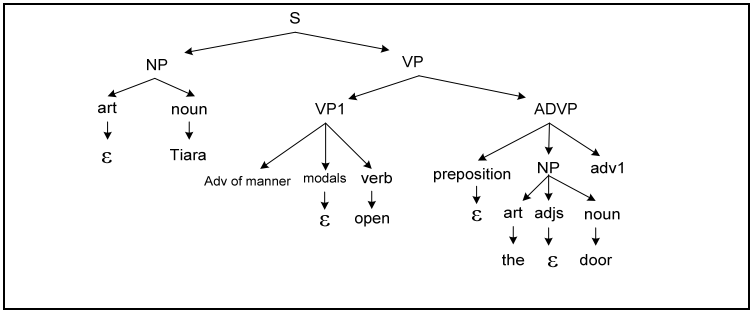
i. Analisis Sintak

Menentukan detail struktur dari suatu kalimat adalah penting. Ini dilakukan melalui proses yang disebut *parsing*. Untuk mem-*parsing* suatu kalimat adalah penting untuk menggunakan *grammar* (aturan tata bahasa) yang menggambarkan struktur dari *string-string* dalam sebuah bahasa. Dengan adanya *grammar*, suatu *parser* (program yang melakukan proses *parsing*) dapat menentukan struktur ini disebut *parse tree*. Gambar 3.4 menunjukkan suatu *grammar* atau aturan dalam bahasa Inggris. Gambar 3.5 menunjukkan *parse tree* yang akan dihasilkan untuk kalimat "Tiara open the door" menggunakan *grammar* di atas. Perhatikan bahwa *parse tree* sesuai aturan dengan mentransformasikan simbol awal S sampai akhir kalimat.

S	→	NP VP
NP	→	art + noun
NP	→	art + adjs + noun
VP	→	V + ADVP
VP1	→	adv of manner + modals + verb
ADVP	→	preposition + NP + adv1

Gambar 3.4 Suatu Aturan atau *Grammar*

Urutan dari kata-kata yang ditransformasikan ke dalam struktur menunjukkan hubungan antara kata yang satu dengan yang lain. Beberapa urutan kata bisa ditolak jika melanggar aturan bahasa.



Gambar 3.5 *Parse Tree*

Sebagai contoh, penganalisis sintak akan menolak kalimat "Wirda the eat an apple".

Proses *parsing* melakukan dua hal:

- *Parsing* melakukan kalimat mana yang dapat diterima secara sintaktis benar (sesuai dengan *grammar*) dan mana yang tidak.
- Untuk kalimat yang secara sintaktis benar, *parsing* dapat menentukan struktur.

j. Analisis Leksikal

Analisis leksikal membaca *input* dan mengubahnya ke dalam rangkaian dari *token* yang akan dianalisis oleh *parser*, karena kalimat-kalimat dari suatu bahasa terdiri atas rangkaian *token*. Barisan dari karakter *input* yang membentuk suatu *token* tertentu disebut *lexeme*. Penganalisis leksikal dapat memisahkan suatu pengurai dari representasi *lexeme* untuk *token*. Mula-mula akan

diberikan oleh suatu penganalisis leksikal untuk melakukan fungsinya.

k. Analisis Semantik

Menghasilkan sintak dari suatu kalimat hanya merupakan langkah maju pertama dalam pemahaman kalimat. Suatu cara untuk interpretasi semantik suatu kalimat adalah dengan menghasilkan sintak yang lengkap dan memberikan struktur tersebut kepada *semantic interpreter* secara terpisah. Kesulitan besar dengan menggunakan pendekatan ini adalah biasanya tidak mungkin untuk menghasilkan interpretasi sintak yang benar tanpa mempertimbangkan beberapa informasi semantik. Struktrur kalimat di mana tidak ada pemetaan tentang bagaimana hubungan antar objek kemungkinan akan ditolak. Contohnya yaitu kalimat "Gorgeous funny girl walk fastly" akan ditolak, karena penggunaan dua *adjective* sebelum *noun* (kata benda).

4. Analisis Pragmatik

Struktur kalimat yang diinterpretasikan untuk mengetahui apa maksud sebenarnya. Sebagai contoh: "What time is it?'', harus diartikan sebagai permintaan untuk diberitahu tentang waktu.

5. Semantik *Grammar*

Suatu aturan bebas konteks (*context free grammar*) di mana pilihan dari non-terminal dan aturan produksi diatur oleh semantik sebagaimana fungsi sintaksis. Salah satu

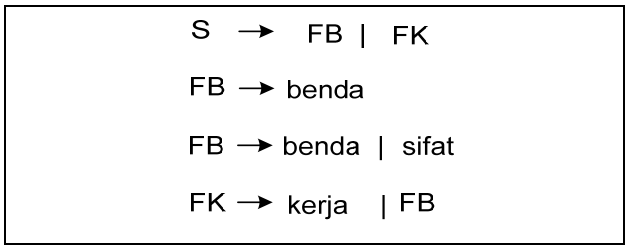
kegunaan dari suatu *semantic grammar* adalah menyediakan *interface* bagi sebuah *intelligent computer-aided instruction system* SOPHIE, yang mengajarkan bagaimana men-*debug* sirkuit elektronik.

Pemrosesan sintak adalah pemrosesan untuk mendapatkan struktur dari kalimat. Dalam pemrosesan sintak ada dua hal harus diperhatikan, yaitu: tata bahasa (*grammar*) dan proses penguraian (*parsing*).

Tata bahasa menunjukkan spesifikasi formal dari struktur bahasa yang diperbolehkan, sedangkan penguraian adalah suatu metode untuk menganalisis suatu kalimat sesuai dengan tata bahasa yang ada. Untuk menyatakan suatu tata bahasa dapat digunakan CFG (Context Free Grammar).

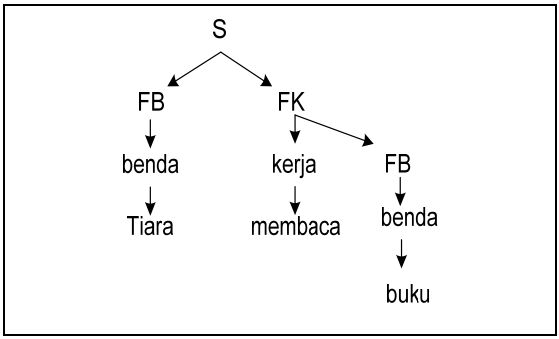
3.3 CFG (Context Free Grammar) Recursive -Descent

CFG merupakan suatu cara untuk menyatakan struktur dari suatu tata bahasa. CFG ini mempunyai aturan sebagai berikut: "<simbol><simbol1><simbol2><simboln>, di mana $n \geq 1$ <simbol> harus merupakan simbol bukan terminal, sedangkan <simbol1>, <simbol2>, ...<simboln> dapat merupakan simbol terminal atau bukan terminal. Bentuk di atas mempunyai arti <simbol> dapat diganti dengan <simbol1>, <simbol2>, ...<simboln>" , contoh untuk tata bahasa Indonesia yang sederhana, dapat diberikan atruan-aturan seperti berikut.



Gambar 3.6 *Grammar* Sederhana

S adalah simbol awal, FB (Frasa Benda) dan FK (Frasa Kerja) disebut simbol bukan terminal, sedangkan sifat, benda dan kerja merupakan simbol terminal. Simbol terminal ini, umumnya menunjukkan kategori dari suatu kata, contoh untuk kalimat "Tiara membaca buku", kalimat ini dapat diuraikan sesuai dengan tata bahasa tersebut.



Gambar 3.7 *Parse Tree* dari Suatu Kalimat

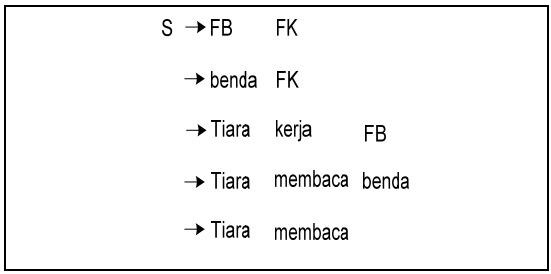
Parser Recursive descent menggunakan kumpulan rutin *recursive* di mana menurunkannya melalui *production rule* sampai kalimat selesai ditelusuri seluruhnya. *Parsing* turun-

berulang (*recursive-descent parsing*) merupakan suatu metoda analisis sintaks puncak-ke-bawah yang dilakukan dengan menjalankan suatu himpunan dari prosedur secara berulang (*recursive*) untuk memproses *input*. Untuk membentuk *parser context-free recursive-descent* dibutuhkan beberapa *vocabulary database*.

3.4 *Parsing*

Ada dua cara yang umum dilakukan untuk melakukan penguraian suatu kalimat yaitu *Top Down Parsing* dan *Bottom Up Parsing*. Dalam pembuatan *Translator English* – Indonesia ini menggunakan *top down parsing*. *Top down parsing* dimulai dari simbol awal (S), kemudian diuraikan misalnya, menjadi bagian kanannya yaitu: FB dan FK. Simbol ini kemudian diuraikan kembali menjadi bagian kanannya sampai ditemukan simbol terminal yang kemudian akan diperiksa dengan kategori kata yang ada.

Top down parsing ini juga dikenal dengan dengan penguraian dari kiri ke kanan (*left to right parsing*), yaitu dimulai dari bagian kiri, kemudian diuraikan terus simbol bukan terminal sampai diperoleh simbol terminal. Contoh *top down parsing* untuk “ “.



Gambar 3.8 *Top Down Parsing*

Aspek yang paling sulit dalam pembentukan sistem NLP adalah pengakomodasian kekompleksan dan kefleksibelan bahasa manusia dalam sistem. Bentuk standar:

Subjek-verb-Objek

Diasumsikan:

- *Adjectiver* mengawali *noun*
- *Adverb* mengawali *Verb*
- Semua kalimat diakhiri dengan titik

3.5 Semantik

Semantic analyzer mempunyai himpunan *rule* dalam basis pengetahuan untuk menginterprestasikan sebuah kalimat.

Rule tersebut adalah:

1. IF *determiner* adalah bagian pertama dalam kalimat dan diikuti oleh *noun* THEN *noun* tersebut dianggap sebagai subjek.
2. IF *verb* diikuti subjek THEN *verb* menjelaskan tentang apa yang dikerjakan oleh subjek.

3. IF *noun* diikuti subjek dan *verb* THEN *noun* tersebut dianggap sebagai objek.
4. IF kalimat mempunyai bentuk subjek, *verb*, objek THEN subjek mengerjakan (*verb*) yang ada hubungannya dengan objek.

Natural Language Processing dapat dipakai sebagai *front* (bagian depan) pada sistem AI, di mana data dilewatkan secara verbal. Pendekatan yang dipakai adalah pendekatan *Semantic Grammar* yang dipadukan dengan *Dictionary* tambahan dan *Template Grammar*. *Semantic Grammar* sebagai *grammar* utama dipilih dengan alasan dalam *grammar* ini sudah terkandung unsur semantik yang dapat membantu pembentukan semantik dari kalimat. Selain itu karena format dari kalimat sudah dibatasi pada bentuk tertentu (representasi data tabel) maka *grammar* ini dapat diandalkan terutama untuk bentuk-bentuk tanya dan perintah. Mendefinisikan semantik dan arti sebenarnya adalah proses yang sulit karena hal ini tergantung pada maksud dalam kalimat dan juga adanya kemungkinan arti lain dalam kalimat.

3.6 *Semantic Grammar*

Pada *semantic grammar*, dipakai sekumpulan *rule* yang bukan hanya bersifat sintaksis tapi juga bersifat semantis dan pragmatis. Hasil dari proses *parsing* dengan menggunakan *grammar* tersebut adalah langsung berupa representasi semantis

dari kalimat yang diolah. Dapat dilihat pada contoh dengan domain sistem jadwal penerbangan pesawat udara:

the flight to Chicago
the 8 o'clock flight
flight 457 to Chicago

Grammar untuk sistem ini pada umumnya dikenali sebagai:

NP → DET CNP (the flight)
CNP → N (flight)
CNP → CNP PP (flight to Chicago)
CNP → PRE-MOD CNP (8 o'clock flight)
NP → N NUMB (flight 457)

Tetapi perlu diingat bahwa *grammar* tersebut masih bersifat umum, masih memungkinkan terjadi kesalahan karena luasnya jangkauan *grammar* seperti:

the city to Chicago
the 8 o'clock city

Untuk itu maka dilakukan pembatasan dengan jalan memberikan kategori leksikal baru pada suatu kata yang berdasar pada keperluan semantis. Dapat kita gambarkan bahwa pada:

NP → DET CNP (the flight) diubah menjadi
FLIGHT-NP → DET FLIGHT-CNP
CNP → N (flight) diubah menjadi
FLIGHT-CNP → FLIGHT-N

Dengan demikian kita sudah melakukan pembatasan bahwa yang mungkin dibentuk untuk kata benda dari sistem adalah ‘the flight’ dan ‘flight’ tidak untuk yang lain. Perlu diingat perubahan ini juga akan merubah kata yang lain misal ‘Chicago’ dari NP menjadi misalnya CITY-NAME. Secara lengkap maka *grammar* tersebut tadi berubah menjadi:

FLIGHT-NP → DET FLIGHT-CNP
FLIGHT-CNP → FLIGHT-N
FLIGHT-CNP → FLIGHT-CNP FLIGHT-DEST
FLIGHT-CNP → FLIGHT-CNP FLIGHT-SOURCE
FLIGHT-CNP → FLIGHT-N FLIGHT-PART
FLIGHT-CNP → FLIGHT-PRE-MOD FLIGHT-CNP
FLIGHT-NP → FLIGHT-N NUMB
CITY-NP → CITY-NAME
CITY-NP → DET CITY-CNP
CITY-CNP → CITY-N
CITY-CNP → CITY-MOD CITY-CNP CITY-MOD-ARG

Dari *grammar* dasar tersebut dapat dibentuk *grammar* tambahan misalnya untuk pertanyaan:
TIME-QUERY → When does FLIGHT-CNP (When does flight to Chicago)

Dengan pendekatan ini, interpretasi dari *rule* yang bersangkutan menjadi lebih mudah karena sebagian besar dari informasi semantik yang diperlukan dapat dilihat dari *rule* yang

digunakan. Kekurangan dari pendekatan ini adalah domain sistem yang tidak begitu besar, di mana domain yang baru akan memerlukan aturan yang baru yang sesuai. Selain itu terjadi pembengkakan jumlah *rule* yang diperlukan, hal ini karena dengan langsung mengacu ke semantik maka banyak generalisasi linguistik yang harus diperinci lebih jauh. Seperti pada contoh sebelumnya, kita harus memisahkan antara NP untuk 'flight' dan 'Chicago' menjadi FLIGHTNP dan CITY-NAME.

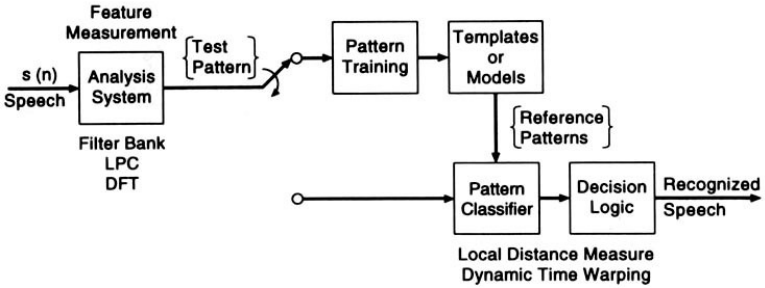
3.7 Sistem Pemrosesan Bahasa Alami

Suatu sistem pemrosesan bahasa alami secara lisan dapat dibentuk dari tiga sub-sistem, yaitu sebagai berikut:

3.7.1 Sistem *Speech Recognition*

Sistem *Speech Recognition* atau sistem pengenalan ucapan adalah sistem yang berfungsi untuk mengubah bahasa lisan menjadi bahasa tulisan. Masukan sistem adalah ucapan manusia, selanjutnya sistem akan mengidentifikasikan kata atau kalimat yang diucapkan dan menghasilkan teks yang sesuai dengan apa yang diucapkan.

Pada Gambar 3.9, Sinyal ucapan (s(n)) pertama kali akan dilewatkan pada bagian Penganalisis Ucapan untuk mendapatlan besaran-besaran atau ciri-ciri yang mudah diolah pada tahap berikutnya. Untuk setiap ucapan yang berbeda akan dihasilkan pola ciri yang berbeda.



Gambar 3.9 Diagram Blok Sistem Pengenal Ucapan

Sistem *speech recognition* yang dapat mengenali seluruh kata dalam suatu bahasa melakukan pengenalan untuk setiap unit bunyi pembentuk ucapan (fonem), selanjutnya mencoba mencari kemungkinan kombinasi hasil ucapan yang paling dapat diterima. Sistem yang lebih sederhana adalah sistem yang hanya dapat mengenal sejumlah kata yang jumlahnya terbatas. Sistem ini biasanya lebih akurat dan lebih mudah dilatih, tetapi tidak dapat mengenal kata yang berada di luar kosa kata yang pernah diajarkan.

Sistem *speech recognition* biasanya dapat dioperasikan pada dua mode yang berbeda. Pertama adalah mode belajar. Pada mode ini, sistem akan dilatih menggunakan sejumlah kata atau kalimat yang memenuhi suatu kriteria tertentu. Setiap contoh kata atau kalimat ajar tersebut akan menghasilkan pola tertentu yang akan dipelajari oleh sistem dan disimpan sebagai *template* atau referensi.

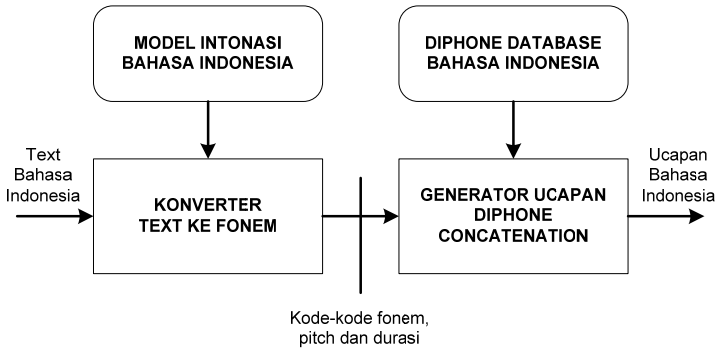
Kedua adalah mode produksi atau pengenalan ucapan. Pada mode ini, setiap kalimat yang ingin dikenali akan dianalisis

polanya. Berdasarkan hasil perbandingan dengan *template* atau referensi, modul klasifikasi pola serta pengambil keputusan akan mengidentifikasikan kata atau kalimat yang diucapkan tersebut. Pada prinsipnya, teknik-teknik atau algoritma yang digunakan pada sistem Pengenal Ucapan tidak bersifat sensitif terhadap bahasa. Artinya, sistem yang sama dapat digunakan untuk bahasa apapun. Namun demikian, kemampuan sistem untuk mengenali ucapan pada bahasa tertentu sangat tergantung dari *template* atau referensi yang diperoleh melalui proses belajar di dalam sistemnya itu sendiri.

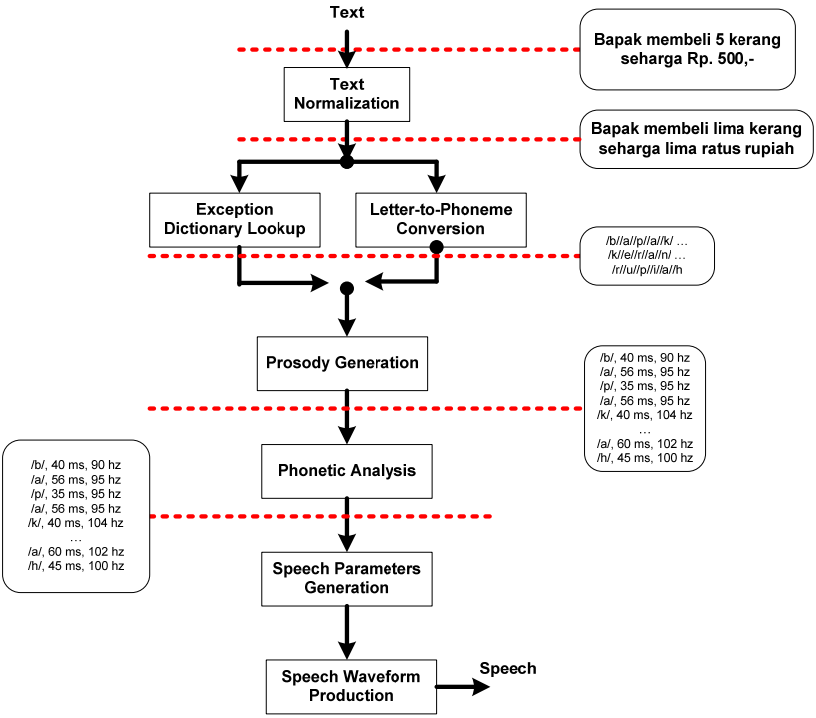
3.7.2 Sistem *Text to Speech*

TTS adalah suatu sistem yang dapat mengubah *text* menjadi ucapan. Suatu sistem pensintesa ucapan atau *Text to Speech* pada prinsipnya terdiri atas dua sub sistem, yaitu:

1) bagian Konverter Teks ke Fonem (*Text to Phoneme*), serta
2) bagian Konverter Fonem ke Ucapan (*Phoneme to Speech*).



Gambar 3.10 Diagram Blok Sistem *Text to Speech* (Bahasa Indonesia)



Gambar 3.11 Urutan Proses Konversi Teks Menjadi Ucapan (*Text to Speech*)

Bagian Konverter Teks ke Fonem berfungsi untuk mengolah kalimat masukan dalam suatu bahasa tertentu yang berbentuk teks menjadi urutan kode-kode bunyi yang direpresentasikan dengan kode fonem, durasi serta *pitch*-nya. Kode-kode fonem adalah kode yang merepresentasikan unit bunyi yang ingin diucapkan. Pengucapan kata atau kalimat pada prinsipnya adalah urutan bunyi atau secara simbolik adalah urutan kode fonem. Setiap fonem harus dilengkapi dengan

informasi durasi dan *pitch*. Informasi durasi diperlukan untuk menentukan berapa lama suatu fonem diucapkan, sedangkan informasi *pitch* diperlukan untuk menentukan tinggi rendahnya nada pengucapan suatu fonem. Durasi dan *pitch* bersama-sama akan membentuk intonasi suatu ucapan. Kedua informasi ini dalam suatu sistem TTS biasanya dibangkitkan oleh modul pembangkit/model intonasi. Bagian Konverter Fonem ke Ucapan akan menerima masukan kode-kode fonem serta *pitch* dan durasi yang telah dihasilkan oleh bagian sebelumnya. Berdasarkan kode-kode tersebut, bagian ini akan menghasilkan bunyi atau sinyal ucapan yang sesuai dengan kalimat yang ingin diucapkan.

3.7.3 Sistem Natural Language Processing

Teknologi Natural Language Processing (NLP) atau pemrosesan bahasa alami adalah teknologi yang memungkinkan untuk melakukan berbagai macam pemrosesan terhadap bahasa alami yang biasa digunakan oleh manusia. Beberapa di antara berbagai kategori aplikasi NLP adalah sebagai berikut:

1. *Natural Language Translator*, yaitu *translator* dari satu bahasa alami ke bahasa alami lainnya, misalnya *translator* bahasa Inggris ke bahasa Indonesia, Bahasa Indonesia ke Bahasa Jawa dan sebagainya. *Translator* bahasa alami bukan hanya kamus yang menerjemahkan kata per kata, tetapi harus juga mentranslasikan sintaks dari bahasa asal ke bahasa tujuannya.

2. *Translator* bahasa alami ke bahasa buatan, yaitu translator yang mengubah perintah-perintah dalam bahasa alami menjadi bahasa buatan yang dapat dieksekusi oleh mesin atau komputer. Sebagai contoh, *translator* yang memungkinkan kita memberikan perintah bahasa alami kepada komputer. Dengan sistem seperti ini, pengguna sistem dapat memberikan perintah dengan bahasa sehari-hari, misalnya, untuk menghapus semua *file*, pengguna cukup memberikan perintah ”komputer, tolong hapus semua *file*!” *Translator* akan mentranslasikan perintah bahasa alami tersebut menjadi perintah bahasa formal yang dipahami oleh komputer, yaitu “dir *.*”.
3. *Text Summarization*, yaitu suatu sistem yang dapat ”membuat ringkasan” hal-hal yang penting dari suatu wacana yang diberikan.

3.8 Aplikasi Pengolahan Bahasa Alami

Jenis Aplikasi yang dapat dibuat pada bidang Natural Processing Language adalah:

a. *Text-based application*
b. *Dialogue-based application* (*speech-based application*).

Text-based application mencakup segala macam aplikasi yang melakukan proses terhadap teks. Contoh penggunaan *text-based application* adalah:

- Mencari topik tertentu dari buku di perpustakaan
- Mencari isi dari suatu berita atau artikel

- Mencari isi dari *email*
- Menterjemahkan dokumen dari suatu bahasa ke bahasa lain

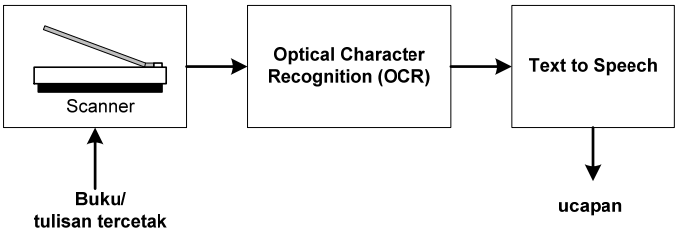
Dialogue-based application adalah aplikasi yang melibatkan bahasa lisan atau pengenalan suara, tetapi bidang ini juga memasukkan interaksi dengan cara memasukkan teks pertanyaan melalui *keyboard*. Contoh aplikasinya adalah:

- Sistem tanya jawab, di mana *Natural Language* digunakan dalam mendapatkan informasi dalam suatu *database*.
- Sistem otomatis pelayanan dalam telepon.
- Control suara pada peralatan elektronik
- Sistem *problem-solving* (pemecah masalah) yang membantu untuk melakukan penyelesaian masalah yang umum dihadapi dalam suatu pekerjaan.
- Aplikasi peningkatan kemampuan berbahasa.

Banyak manfaat yang dapat dicapai dari ketersediaan Aplikasi Teknologi Bahasa, khususnya untuk Bahasa Indonesia. Contohnya sebagai berikut:

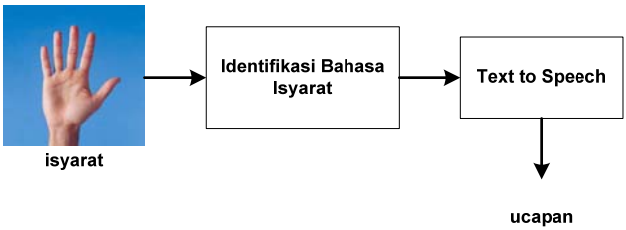
- **Alat bantu membaca untuk tunanetra.** Alat bantu membaca bagi tunanetra mempunyai masukan berupa teks tercetak (misalnya buku) dan mempunyai keluaran berupa ucapan dari teks tercetak yang diberikan. Pada prinsipnya ada dua komponen utamanya, yaitu bagian "pengenal karakter" yang menggunakan teknologi OCR

(*Optical Character Recognition*), serta bagian TTS. Dengan alat bantu ini, orang tunanetra dapat membaca suatu buku atau dokumen. Bahkan, jika teks yang ingin dibacakan sudah tersedia di dalam komputer, dengan teknologi *text to speech* dapat langsung diucapkan.

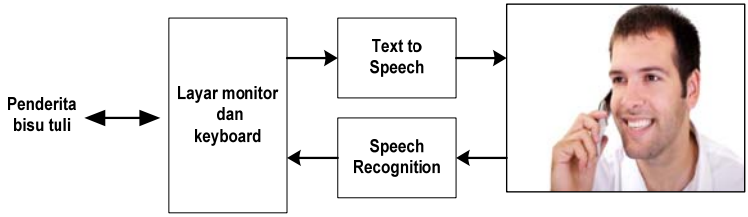


Gambar 3.12 Aplikasi Alat Bantu Baca

- **Alat bantu bicara untuk tunawicara.** Alat bantu membaca bagi tunawicara mempunyai masukan posisi tangan yang dideteksi oleh suatu sensor dan unit identifikasi. Rangkaian huruf yang diidentifikasikan akan disusun membentuk suatu kata yang pada akhirnya akan diumpankan pada bagian TTS.

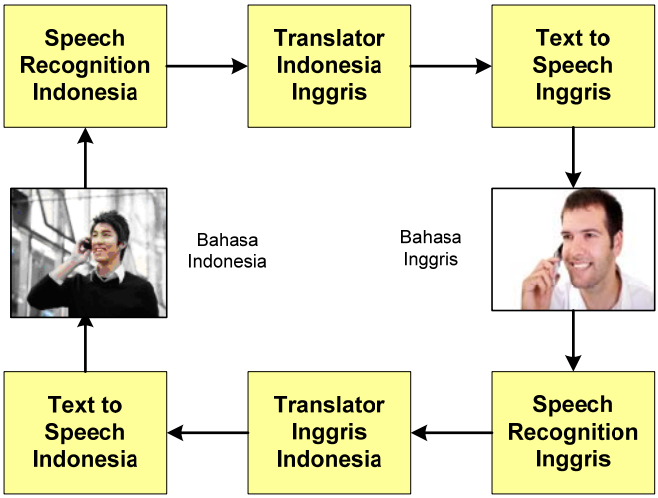


Gambar 3.13 Aplikasi Alat untuk Tuna Wicara



Gambar 3.14 Telepon untuk Penderita Bisu-Tuli

- ***Online translator.*** yang dimaksud adalah *translator* yang secara otomatis dapat menerjemahkan kalimat lisan dari suatu bahasa alami (misalnya Bahasa Inggris) menjadi ucapan hasil terjemahannya dalam bahasa alami lainnya (misalnya Bahasa Indonesia). *Online translator* terdiri atas 3 bagian yaitu:
  - *Speech recognition*, berfungsi untuk mengenali rangkaian kata dari bahasa sumber menjadi teks dalam bahasa sumber.
  - *Translator* teks ke teks. Hasil bagian kedua ini adalah kalimat bahasa tujuan yang masih berupa teks.
  - Berupa sistem TTS dalam bahasa tujuan.



Gambar 3.15 Aplikasi *Translator Online*

- ***Talking email* atau aplikasi lainnya.** TTS juga memungkinkan diintegrasikan dengan berbagai program aplikasi, seperti *email*, *web browser*, aplikasi-aplikasi multimedia atau aplikasi lainnya.
- **Aplikasi *Telephony*.** TTS dapat digunakan pada aplikasi telephony, seperti sistem informasi *billing* atau sistem informasi lainnya yang diucapkan secara lisan. TTS juga dapat digunakan untuk konversi dari SMS (*Short Message System*) ke ucapan sehingga pesan SMS dapat didengar.

Contoh aplikasi NLP lainnya:

- Eliza yang dibekali pengetahuan psikologi, sehingga beberapa orang terdorong untuk mampu merubah sikap dan perilakunya.
- Jupiter yang mampu memberikan informasi cuaca melalui telepon.
- Alvin yang mampu menjawab pertanyaan mengenai DOS
- SEXPERT yang dirancang untuk perbincangan mengenai pendidikan seksual.
- *Email translator*: alat yang akan menjawab masalah perbedaan bahasa, karena *email translator* mampu menterjemahkan bahasa, seperti yang kita inginkan. *Email translator* akan menterjemahkan kaimat-kalimat di dalam *mailbox*, jika *email* yang kita terima tidak sesuai dengan bahasa kita sehari-hari.
- *Web translator*: suatu mesin aplikasi berbasis *World Wide Web* yang dapat menterjemahkan bahasa dalam suatu *website*. *Web translator* akan menterjemahkan bahasa di dalam semua *link*, *page* per *page* menjadi bahasa seperti yang kita inginkan.
- *World translator*: suatu pengembangan dari *word translator* yang sudah ada, diharapkan dengan teknik ini hasil terjemahan bahasa akan menjadi lebih sempurna, mengikuti kaidah tata bahasa. Terjemahan akan lebih cepat, akurat, bukan lagi dengan sistem menterjemahkan per kata, tapi per kalimat dengan melihat Subjek-Predikat-Objek.

Bab 4

Speech Recognition

4.1 Pengantar

Kecerdasan buatan atau *artificial intelligence* didefinisikan sebagai kecerdasan yang ditunjukkan oleh suatu entitas buatan. Sistem seperti ini umumnya dianggap komputer. Kecerdasan diciptakan dan dimasukkan ke dalam suatu mesin (komputer) agar dapat melakukan pekerjaan seperti yang dapat dilakukan manusia. Namun seiring dengan perkembangan zaman, maka peran komputer semakin mendominasi kehidupan umat manusia. Komputer tidak hanya digunakan sebagai alat hitung, lebih dari itu, komputer diharapkan untuk dapat diberdayakan untuk mengerjakan segala sesuatu yang bisa dikerjakan oleh manusia.

Semakin pesatnya perkembangan teknologi menyebabkan adanya perkembangan dan perluasan lingkup yang membutuhkan kehadiran kecerdasan buatan. Karakteristik 'cerdas' sudah mulai dibutuhkan di berbagai disiplin ilmu dan teknologi. Kecerdasan buatan tidak hanya dominan di bidang ilmu komputer (informatika), namun juga sudah merambah di berbagai disiplin ilmu yang lain. meliputi sistem pakar (*Expert System*), pengolahan bahasa alami (*Natural Language Processing*), Pengenalan Ucapan (*Speech Recognition*), Robotika & Sistem

Sensor (*Robotics & Sensory Systems*), *Computer Vision*, *Intelligent Computer-Aided Instruction*, *Game Playing*.

4.2 Definisi *Speech Recognition*

Speech Recognition adalah proses identifikasi suara berdasarkan kata yang diucapkan dengan melakukan konversi sebuah sinyal akustik, yang ditangkap oleh *audio device* (perangkat *input* suara). *Speech Recognition* juga merupakan sistem yang digunakan untuk mengenali perintah kata dari suara manusia dan kemudian diterjemahkan menjadi suatu data yang dimengerti oleh komputer. Pada saat ini, sistem ini digunakan untuk menggantikan peranan *input* dari *keyboard* dan *mouse*.

Pengenalan ucapan atau pengenalan wicara—dalam istilah bahasa Inggrisnya, *automatic speech recognition* (ASR) adalah suatu pengembangan teknik dan sistem yang memungkinkan komputer untuk menerima masukan berupa kata yang diucapkan. Teknologi ini memungkinkan suatu perangkat untuk mengenali dan memahami kata-kata yang diucapkan dengan cara digitalisasi kata dan mencocokkan sinyal digital tersebut dengan suatu pola tertentu yang tersimpan dalam suatu perangkat. Kata-kata yang diucapkan diubah bentuknya menjadi sinyal digital dengan cara mengubah gelombang suara menjadi sekumpulan angka yang kemudian disesuaikan dengan kode-kode tertentu untuk mengidentifikasikan kata-kata tersebut. Hasil dari identifikasi kata yang diucapkan dapat ditampilkan dalam bentuk tulisan atau

dapat dibaca oleh perangkat teknologi sebagai sebuah komando untuk melakukan suatu pekerjaan, misalnya penekanan tombol pada telepon genggam yang dilakukan secara otomatis dengan komando suara.

Alat pengenal ucapan yang sering disebut dengan *speech recognizer*, membutuhkan sampel kata sebenarnya yang diucapkan dari pengguna. Sampel kata akan didigitalisasi, disimpan dalam komputer dan kemudian digunakan sebagai basis data dalam mencocokkan kata yang diucapkan selanjutnya. Sebagian besar alat pengenal ucapan sifatnya masih tergantung kepada pengeras suara. Alat ini hanya dapat mengenal kata yang diucapkan dari satu atau dua orang saja dan hanya bisa mengenal kata-kata terpisah yaitu kata-kata yang dalam penyampaiannya terdapat jeda antar kata. Hanya sebagian kecil dari peralatan yang menggunakan teknologi ini yang sifatnya tidak tergantung pada pengeras suara. Alat ini sudah dapat mengenal kata yang diucapkan oleh banyak orang dan juga dapat mengenal kata-kata kontinu atau kata-kata yang dalam penyampaiannya tidak terdapat jeda antar kata. Parameter yang dibandingkan ialah tingkat penekanan suara yang kemudian akan dicocokkan dengan *template database* yang tersedia. Sedangkan sistem pengenalan suara berdasarkan orang yang berbicara dinamakan *speaker recognition*.

Pengenalan ucapan dalam perkembangan teknologinya merupakan bagian dari pengenalan suara (proses identifikasi seseorang berdasarkan suaranya). Pengenalan suara sendiri

terbagi menjadi dua yaitu pengenalan pengguna (identifikasi suara berdasarkan orang yang berbicara) dan pengenalan ucapan (identifikasi suara berdasarkan kata yang diucapkan).

Perkembangan alat pengenal ucapan (*speech recognizer*)

- Tahun 1940

  Perusahaan *American Telephone and Telegraph Company* (AT&T) mulai mengembangkan suatu perangkat teknologi yang dapat mengidentifikasi kata yang diucapkan manusia.
- Tahun 1960-an

  Para peneliti perusahaan tersebut berhasil membuat suatu perangkat yang dapat mengidentifikasi kata-kata terpisah.
- Tahun 1970-an:

  Berhasil membuat perangkat yang dapat mengidentifikasi kata-kata kontinu.
- Tahun 1980-an:

  Speech recognizer menjadi sangat fungsional dan masih dikembangkan dan terus ditingkatkan keefektifannya hingga sekarang.

4.3 Sejarah *Speech Recognition*

Speech recognition pertama muncul pada tahun 1952 dan terdiri atas sebuah perangkat untuk pengakuan lisan digit tunggal awal perangkat lainnya adalah IBM Kotak Sepatu yang dipamerkan di Fair tahun 1964 New York. Akhir-akhir ini telah ada banyak perbaikan seperti *capabilitiy massa* kecepatan transkripsi tinggi pada satu sistem seperti Sonic Extractor salah

satu domain yang paling menonjol untuk aplikasi komersial pengenalan suara di Amerika Serikat pada perawatan kesehatan dan khususnya karya *transcriptionist* medis (MT).Menurut para ahli industri, pada awal berdirinya, pengenalan pembicaraan (SR) yang dijual sebagai cara untuk sepenuhnya menghilangkan transkripsi daripada membuat proses transkripsi lebih efisien, maka tidak diterima. Ini juga merupakan kasus yang SR pada waktu itu sering secara teknis kurang. Selain itu, untuk digunakan secara efektif, dibutuhkan perubahan cara dokter bekerja dan didokumentasikan pertemuan klinis, yang banyak jika tidak semua enggan untuk melakukannya. Keterbatasan terbesar pidato pengakuan mengotomatisasi transkripsi, bagaimanapun, dipandang sebagai perangkat lunak. Sifat naratif dikte sangat interpretatif dan seringkali memerlukan penilaian yang dapat diberikan oleh manusia sesungguhnya tetapi belum oleh sistem otomatis. Keterbatasan lainnya telah jumlah ekstensif waktu yang diperlukan oleh pengguna dan atau penyedia sistem untuk melatih perangkat lunak. Perbedaan dalam ASR sering dibuat antara "sistem sintaks buatan" yang biasanya domain-spesifik dan "pemrosesan bahasa alami" yang biasanya bahasa-spesifik. Masing-masing jenis aplikasi menyajikan tujuannya sendiri tertentu dan tantangan.

Automatic Speech Recognition (ASR) sekarang ini telah banyak dikembangkan dalam berbagai penelitian. Terdapat bermacam-macam metode yang dapat digunakan untuk mengenali ucapan manusia. Penelitian ini akan membahas

penggunaan metode Hidden Markov Model (HMM) untuk pengenalan ucapan berbahasa Indonesia. Dalam penelitian ini, digunakan HMM diskrit untuk proses pelatihan dan pengujiannya. Berdasarkan hasil pengujian dengan menggunakan metode tersebut, kemudian dianalisis faktor keberhasilannya (tingkat ketelitiannya dalam %) berdasarkan parameter-parameter Linear Predictive Coding (LPC), parameter *pitch* (Fo) dan parameter energi (Eo) dalam proses mengenali suatu ucapan dalam bahasa Indonesia.

Prinsip kerja sistem pengenalan ucapan adalah dengan membandingkan informasi ucapan yang ada pada referensi dengan informasi ucapan yang menjadi masukan sistem pengenal ucapan tersebut. Blok pengenalan ucapan dengan HMM dapat dibagi menjadi tiga tahap yaitu bagian depan, tahap *feature extraction* dan tahap sistem pengenalan HMM. Pada tahap yang pertama dilakukan pemfilteran sinyal suara dan mengubah sinyal suara analog ke digital. Tahap *feature extraction* adalah untuk mendapatkan parameter-parameter yang dapat merepresentasikan sinyal suara tersebut dan dilakukan analisis serta kuantisasi vektor. Tahap yang ketiga, dapat dibagi menjadi dua tugas yaitu tugas pemodelan dan tugas pengenalan. Untuk tugas pemodelan dibuatkan suatu model HMM dari data-data yang berupa sampel ucapan dari sebuah kata. HMM yang dipakai adalah densitas diskrit.

Metode Hidden Markov Model mulai diperkenalkan dan dipelajari pada akhir tahun 1960, metode yang berupa model

statistik dari rantai Markov ini semakin banyak dipakai pada tahun-tahun terakhir terutama dalam bidang *speech recognition*, seperti dijelaskan oleh Lawrence R. Rabiner dalam laporannya yang berjudul "*A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*".

Proses dalam dunia nyata secara umum menghasilkan *observable output* yang dapat dikarakterisasikan sebagai sinyal. Sinyal bisa bersifat diskrit (karakter dalam alfabet) maupun kontinu (pengukuran temperatur, alunan musik). Sinyal bisa bersifat stabil (nilai statistiknya tidak berubah terhadap waktu) maupun nonstabil (nilai signal berubah-ubah terhadap waktu). Dengan melakukan pemodelan terhadap sinyal secara benar, dapat dilakukan simulasi terhadap sumber dan pelatihan sebanyak mungkin melalui proses simulasi tersebut. Sehingga model dapat diterapkan dalam sistem prediksi, sistem pengenalan, maupun sistem identifikasi. Secara garis besar model signal dapat dikategorikan menjadi 2 golongan yaitu: model deterministik dan model statistik. Model deterministik menggunakan nilai-nilai properti dari sebuah sinyal seperti: amplitudo, frekuensi, fase dari gelombang sinus. Sedangkan model statistik menggunakan nilai-nilai statistik dari sebuah sinyal seperti: proses Gaussian, proses Poisson, proses Markov, dan proses Hidden Markov. Suatu model HMM secara umum memiliki unsur-unsur sebagai berikut:

- N, yaitu jumlah *state* dalam model. Secara umum *state* saling terhubung satu dengan yang lain, dan suatu *state*

bisa mencapai semua *state* yang lain dan sebaliknya (disebut *model ergodic*). Namun hal tersebut tidak mutlak, terdapat kondisi lain di mana suatu *state* hanya bisa berputar ke diri sendiri dan berpindah ke satu *state* berikutnya, hal ini bergantung pada implementasi dari model.

- M, yaitu jumlah *observation* simbol secara unik pada tiap *state*-nya, misalnya: karakter dalam alfabet, di mana *state* adalah huruf dalam kata.
- *State Transition Probability*
- *Observation Symbol Probability* pada *state*
- *Initial State Distribution*

Dengan memberikan nilai pada N, M, A, B, dan p, HMM dapat digunakan sebagai *generator* untuk menghasilkan urutan observasi, di mana tiap observasi adalah salah satu simbol dari V, dan T adalah jumlah observasi dalam suatu *sequence*.

4.4 Pemodelan *Speech Recognition*

Ada dua pemodelan dasar untuk *speech recognition*, yaitu:

1. Hidden Markov Model (HMM)-*based recognition*. Model ini digunakan pada sistem modern *general-purpose speech recognition*. Model ini merupakan model yang statistikal di mana *output* adalah *sequence* dari simbol atau kuantitas. Model ini digunakan karena sebuah sinyal dari pengucapan bisa dilihat seperti *piecewise stationary signal* atau *short-*

time stationary signal, selain itu model ini sederhana dan secara komputasional bisa digunakan.

2. Dynamic time warping (DTW)-*based speech recognition*, adalah pendekatan yang pernah sejarahnya digunakan untuk *speech recognition* yang sekarang sudah digantikan oleh model Hidden Markov. DTW pertama kali dikenalkan pada tahun 60-an dan dieksplorasi sampai tahun 70-an yang menghasilkan alat *speech recognizer*.

Berdasarkan kemampuan dalam mengenal kata yang diucapkan, terdapat 5 jenis pengenalan kata yaitu:

1. Kata-kata yang terisolasi

   Proses pengidentifikasian kata yang hanya dapat mengenal kata yang diucapkan jika kata tersebut memiliki jeda waktu pengucapan antar kata
2. Kata-kata yang berhubungan

   Proses pengidentifikasian kata yang mirip dengan kata-kata terisolasi, namun membutuhkan jeda waktu pengucapan antar kata yang lebih singkat
3. Kata-kata yang berkelanjutan

   Proses pengidentifikasian kata yang sudah lebih maju karena dapat mengenal kata-kata yang diucapkan secara berkesinambungan dengan jeda waktu yang sangat sedikit atau tanpa jeda waktu. Proses pengenalan suara ini sangat rumit karena membutuhkan metode khusus untuk membedakan

kata-kata yang diucapkan tanpa jeda waktu. Pengguna perangkat ini dapat mengucapkan kata-kata secara natural.

4. Kata-kata spontan

   Proses pengidentifikasian kata yang dapat mengenal kata-kata yang diucapkan secara spontan tanpa jeda waktu antar kata.

5. Verifikasi atau identifikasi suara

   Proses pengidentifikasian kata yang tidak hanya mampu mengenal kata, namun juga mengidentifikasi siapa yang berbicara.

Kelebihan dari peralatan yang menggunakan teknologi ini adalah:

1. Cepat

   Teknologi ini mempercepat transmisi informasi dan umpan balik dari transmisi tersebut. Contohnya pada komando suara. Hanya dalam selang waktu sekitar satu atau dua detik setelah kita mengkomandokan perintah melalui suara, komputer sudah memberi umpan balik atas komando kita.

2. Mudah digunakan

   Kemudahan teknologi ini juga dapat dilihat dalam aplikasi komando suara. Komando yang biasanya kita masukkan ke dalam komputer dengan menggunakan *mouse* atau *keyboard* kini dapat dengan mudahnya kita lakukan tanpa perangkat keras, yakni dengan komando suara.

Adapun kekurangan dari peralatan yang menggunakan teknologi ini adalah:

1. Rawan terhadap gangguan

   Hal ini disebabkan oleh proses sinyal suara yang masih berbasis frekuensi. Ketika sebuah informasi dalam sinyal suara mempunyai komponen frekuensi yang sama banyaknya dengan komponen frekuensi gangguannya, akan sulit untuk memisahkan gangguan dari sinyal suara
2. Jumlah kata yang dapat dikenal terbatas

   Hal ini disebabkan pengenal ucapan bekerja dengan cara mencari kemiripan dengan basis data yang dimiliki.

4.5 Cara Kerja *Speech Recognition*

Teknologi *speech recognition* memiliki dua pilihan, yakni menangkap percakapan terputus (kata per kata) atau percakapan tersambung (per kalimat). Komputer sebenarnya lebih mudah memahami suara dalam bentuk kata per kata, yang di antara masing-masing kata terdapat jeda, namun kebanyakan orang lebih menyukai jika teknologi ini mampu menangkap sebuah percakapan normal.

Untuk mengubah percakapan menjadi teks *on-screen* atau perintah tertentu, komputer melakukan beberapa langkah yang kompleks. Ketika berbicara, kita mengeluarkan getaran di udara. Kemudian, analog-to-digital converter (ADC) yang ada di

soundcard menerjemahkan gelombang analog ini menjadi data digital yang dapat dimengerti oleh komputer.

Untuk melakukan hal tersebut, sistem *speech recognition* melakukan *sampling* atau *voice digitizing* dengan cara mengambil ukuran yang paling pas dari gelombang. Sistem menyaring suara yang telah didigitalkan dan membuang gangguan (*noise*), serta memisahkannya ke dalam pita frekuensi yang berbeda. Frekuensi adalah panjang gelombang suara, yang terdengar oleh telinga manusia sebagai tinggi nada (*pitch*) yang berbeda.

Sistem ini juga menormalkan suara atau mengaturnya ke dalam tingkat volume yang tetap, terkadang juga mendatarkan suara. Manusia tidak berbicara dalam kecepatan yang sama sehingga suara harus diatur dengan kecepatan yang sama dengan sampel-sampel *template* suara yang tersimpan dalam komputer.

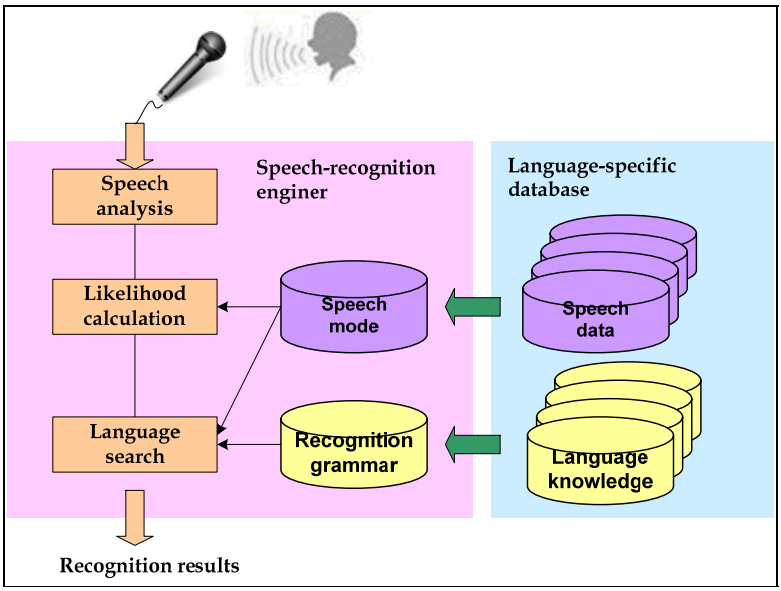
Langkah selanjutnya adalah memecah sinyal menjadi bagian-bagian kecil, dengan durasi seperseratus detik, atau bahkan seperseribu pada kasus bunyi-bunyi konsonan atau mati. Konsonan memberhentikan produksi suara dengan menghalangi aliran gelombang pada bidang vokal, seperti “p” atau “t”. Program di komputer kemudian mencocokkan bagian-bagian kecil ini dengan fonem yang dikenal dalam bahasa tertentu. Fonem adalah elemen terkecil dalam sebuah bahasa, merepresentasikan suara yang kita hasilkan, dan merangkainya ke dalam bentuk ujaran yang memiliki makna. Tahap berikutnya kelihatan sederhana, tapi pada dasarnya merupakan proses yang

paling susah diselesaikan, sekaligus merupakan inti dari sebagian besar penelitian di bidang *speech recognition*. Komputer memeriksa fonem-fonem dalam konteks (hubungan) dengan fonem-fonem lain yang menyertainya. Komputer menjalankan alur (*plot*) melalui sebuah model statistika yang kompleks dan membandingkannya dengan koleksi kata, frase, dan kalimat yang telah dikenal. Program *speech recognition* selanjutnya menentukan apa yang mungkin dikatakan oleh pengguna dan juga mengetikkannya sebagai teks atau mengeluarkannya sebagai perintah pada komputer.

Alat pengenal ucapan memiliki empat tahapan dalam prosesnya, yaitu:

1. Tahap penerimaan masukan

   Masukan berupa kata-kata yang diucapkan lewat pengeras suara. Secara umum, *speech recognizer* memproses sinyal suara yang masuk dan menyimpannya dalam bentuk digital. Hasil proses digitalisasi tersebut kemudian dikonversi dalam bentuk spektrum suara yang akan dianalisis dengan membandingkannya dengan *template* suara pada *database* sistem.



Gambar 4.1 Skema *Speech Recognition*

2. Tahap ekstraksi

   Tahap ini adalah tahap penyimpanaan masukan yang berupa suara sekaligus pembuatan basis data sebagai pola. Proses ekstraksi dilakukan berdasarkan metode Model Markov Tersembunyi atau Hidden Markov Model (HMM), yang merupakan model statistik dari sebuah sistem yang diasumsikan oleh Markov sebagai suatu proses dengan parameter yang tidak diketahui. Tantangan dalam model statistik ini adalah menentukan parameter-parameter tersembunyi dari parameter yang dapat diamati. Parameter-parameter yang telah kita tentukan kemudian digunakan

untuk analisis yang lebih jauh pada proses pengenalan kata yang diucapkan. Berdasarkan HMM, proses pengenalan ucapan secara umum menghasilkan keluaran yang dapat dikarakterisasikan sebagai sinyal. Sinyal dapat bersifat diskrit (karakter dalam abjad) maupun kontinu (pengukuran temperatur, alunan musik). Sinyal dapat pula bersifat stabil (nilai statistiknya tidak berubah terhadap waktu) maupun nonstabil (nilai sinyal berubah-ubah terhadap waktu). Dengan melakukan pemodelan terhadap sinyal secara benar, dapat dilakukan simulasi terhadap masukan dan pelatihan sebanyak mungkin melalui proses simulasi tersebut sehingga model dapat diterapkan dalam sistem prediksi, sistem pengenalan, maupun sistem identifikasi. Secara garis besar model sinyal dapat dikategorikan menjadi dua golongan, yaitu:

a. Model deterministik, menggunakan nilai-nilai properti dari sebuah sinyal seperti: amplitudo, frekuensi, dan fase dari gelombang sinus.
b. Model statistikal menggunakan nilai-nilai statistik dari sebuah sinyal seperti: proses Gaussian, proses Poisson, proses Markov, dan proses Markov Tersembunyi.

Suatu model HMM secara umum memiliki unsur-unsur sebagai berikut:

a. N, yaitu jumlah bagian dalam model. Secara umum bagian tersebut saling terhubung satu dengan yang lain, dan suatu bagian bisa mencapai semua bagian

yang lain, serta sebaliknya (disebut dengan model ergodik). Namun hal tersebut tidak mutlak karena terdapat kondisi lain di mana suatu bagian hanya bisa berputar ke diri sendiri dan berpindah ke satu bagian berikutnya. Hal ini bergantung pada implementasi dari model.

b. M, yaitu jumlah simbol observasi secara unik pada tiap bagiannya, misalnya: karakter dalam abjad, di mana bagian diartikan sebagai huruf dalam kata.
c. Probabilitas Perpindahan
d. Probabilitas Simbol Observasi
e. Inisial Distribusi Bagian i p p

   Setelah memberikan nilai N, M, A, B, dan p, maka proses ekstraksi dapat diurutkan. Berikut adalah tahapan ekstraksi pengenalan ucapan berdasarkan HMM:

f. Tahap ekstraksi tampilan. Penyaringan sinyal suara dan pengubahan sinyal suara analog ke digital
g. Tahap tugas pemodelan. Pembuatan suatu model HMM dari data-data yang berupa sampel ucapan sebuah kata yang sudah berupa data digital
h. Tahap sistem pengenalan HMM. Penemuan parameter-parameter yang dapat merepresentasikan sinyal suara untuk analisis lebih lanjut.

3. Tahap pembandingan

   Tahap ini merupakan tahap pencocokan data baru dengan data suara (pencocokan tata bahasa) pada pola. Tahap ini dimulai dengan proses konversi sinyal suara digital hasil dari proses ekstraksi ke dalam bentuk spektrum suara yang akan dianalisis dengan membandingkannya dengan pola suara pada basis data. Sebelumnya, data suara masukan dipilah-pilah dan diproses satu per satu berdasarkan urutannya. Pemilihan ini dilakukan agar proses analisis dapat dilakukan secara paralel. Proses yang pertama kali dilakukan ialah memproses gelombang kontinu spektrum suara ke dalam bentuk diskrit. Langkah berikutnya ialah proses kalkulasi yang dibagi menjadi dua bagian:

   a. Transformasi gelombang diskrit menjadi data yang terurut.

      Gelombang diskrit berbentuk masukan berukuran n yang menjadi objek yang akan dibagi pada proses konversi dengan cara pembagian rincian waktu.

   b. Menghitung frekuensi pada tiap elemen data yang terurut.

      Selanjutnya tiap elemen dari data yang terurut tersebut dikonversi ke dalam bentuk bilangan biner. Data biner tersebut nantinya akan dibandingkan dengan pola data suara dan kemudian diterjemahkan sebagai keluaran yang dapat berbentuk tulisan ataupun perintah pada perangkat.

4. Tahap validasi identitas pengguna

   Alat pengenal ucapan yang sudah memiliki sistem verifikasi/identifikasi suara akan melakukan identifikasi orang yang berbicara berdasarkan kata yang diucapkan setelah menerjemahkan suara tersebut menjadi tulisan atau komando.

4.6 Implementasi *Speech Recognition*

Hardware yang dibutuhkan dalam implementasi *speech recognition*:

a. *Sound card*: merupakan perangkat yang ditambahkan dalam suatu Komputer yang fungsinya sebagai perangkat *input* dan *output* suara untuk mengubah sinyal elektrik, menjadi analog maupun menjadi digital.
b. *Microphone*: perangkat *input* suara yang berfungsi untuk mengubah suara yang melewati udara, air dari benda orang menjadi sinyal elektrik.
c. Komputer atau Komputer *Server*: dalam proses suara digital menterjemahkan gelombang suara menjadi suatu simbol biasanya menjadi suatu nomor biner yang dapat diproses lagi kemudian diidentifikasikan dan dicocokan dengan *database* yang berisi berkas suara agar dapat dikenali.

Contoh aplikasi *speech recognition*:

1. Bidang Komunikasi
   a. Komando Suara: suatu program pada komputer yang melakukan perintah berdasarkan komando suara dari pengguna. Contohnya pada aplikasi Microsoft Voice yang berbasis bahasa Inggris.
   b. Pendiktean: sebuah proses mendikte yang sekarang ini banyak dimanfaatkan dalam pembuatan laporan atau penelitian. Contohnya pada aplikasi Microsoft Dictation, yang merupakan aplikasi yang dapat menuliskan apa yang diucapkan oleh pengguna secara otomatis.
   c. Telepon, pada telepon teknologi pengenal ucapan digunakan pada proses penekanan tombol otomatis yang dapat menelpon nomor tujuan dengan komando suara
2. Bidang Kesehatan

   Alat pengenal ucapan banyak digunakan dalam bidang kesehatan untuk membantu para penyandang cacat dalam beraktivitas. Contohnya pada aplikasi Antarmuka Suara Pengguna atau Voice User Interface (VUI) yang menggunakan teknologi pengenal ucapan di mana pengendalian saklar lampu misalnya, tidak perlu dilakukan secara manual dengan menggerakkan saklar tetapi cukup dengan mengeluarkan perintah dalam bentuk ucapan sebagai saklarnya. Metode ini membantu manusia yang

secara fisik tidak dapat menggerakkan saklar karena cacat pada tangan misalnya. Penerapan VUI ini tidak hanya untuk lampu saja tapi bisa juga untuk aplikasi-aplikasi kontrol yang lain.

3. Bidang Militer
   a. Pelatihan penerbangan. Aplikasinya adalah pada pengatur lalu lintas udara atau yang dikenal dengan Air Traffic Controllers (ATC) yang dipakai oleh pilot untuk mendapatkan keterangan mengenai keadaan lalu lintas udara seperti radar, cuaca dan navigasi.
   b. Helikopter. Aplikasinya digunakan untuk berkomunikasi lewat radio dan menyesuaikan sistem navigasi.

Bab 5

Computer Vision Sebagai Pengindraan

5.1 Pengantar

Kecerdasan buatan merupakan teknologi yang mensimulasikan kecerdasan manusia. Komputer dapat menyelesaikan secara cepat suatu masalah dengan meniru bagaimana manusia menyelesaikan permasalahan tersebut. Salah satu cabang dari AI yang difokuskan pada pengembangan algoritma untuk menganalisis isi dari suatu gambar. *Computer vision* mencoba meniru cara kerja sistem visual manusia. Computer vision bertujuan agar komputer dapat mengenali suatu image dan mengambil suatu keputusan.

Perkembangan internet yang pesat menyebabkan perkembangan volume informasi tekstural yang luar biasa. Pada dasarnya masyarakat dunia sangat bergantung pada adanya informasi dari berbagai bidang baik itu informasi berupa teks, gambar, video ataupun informasi yang berasal dari halaman suatu *web*. Selain itu perkembangan teknologi yang pesat mempermudah akses terhadap informasi tekstural yang sangat besar jumlahnya, baik yang terdapat pada Internet maupun pada koleksi dokumen yang spesifik. Akan tetapi, kemampuan

manusia untuk membaca informasi tersebut dan memahami isinya tidak bertambah cepat dari sebelumnya.

Saat ini teknologi *information retrieval* saja tidak akan mampu mencukupi kebutuhan informasi yang lebih spesifik karena teknologi ini hanya menyediakan informasi pada level koleksi dokumen. Oleh karena itu, dibutuhkan suatu mekanisme untuk menjadikan informasi tersebut ke dalam format yang terstruktur, sebagai contohnya adalah ke dalam basis data relasional agar dapat lebih mudah untuk memahami suatu informasi yang berasal dari format tak struktur seperti gambar, video, dokumen teks atau halaman *web*.

Visi komputer dalam perkembangannya mampu untuk mengekstraksi informasi penting dari sebuah gambar atau video. Konsep visi komputer didasari oleh penglihatan manusia (*human vision)* yang sangat kompleks, ide dasarnya adalah komputer mampu mengenali setiap objek yang diamatinya dan memberikan hasil analisis sesuai dengan tugas yang diberikan ke komputer tersebut, dengan visi komputer semua permasalahan mengenai ekstraksi informasi dari gambar dan video dapat dilakukan dengan proses yang berkaitan dengan segmentasi citra.

Computer vision diciptakan untuk membangun sebuah mesin pandai yang dapat melihat, seperti robot AIBO yang dibuat oleh perusahaan SONY, di mana robot yang berbentuk anjing tersebut dapat melihat dan mengenali pemiliknya.

5.2 Definisi Visi Komputer (*Computer Vision*)

Visi komputer adalah ilmu dan teknologi mesin yang memiliki fitur untuk melihat, di mana mesin mampu mengekstrak informasi dari gambar yang diperlukan untuk menyelesaikan tugas tertentu. Sebagai suatu disiplin ilmu, visi komputer berkaitan dengan teori dalam bagian sistem buatan mengenai ekstrak informasi dari gambar. Data gambar dapat mengambil banyak bentuk, seperti urutan video, pandangan dari beberapa kamera atau data multi-dimensi dari *scanner* medis. Sedangkan sebagai disiplin teknologi, *computer vision* berusaha untuk menerapkan teori dan model untuk pembangunan sistem *computer vision*.

Visi komputer didefinisikan sebagai salah satu cabang ilmu pengetahuan yang mempelajari bagaimana komputer dapat mengenali objek yang diamati. Cabang ilmu ini bersama *artificial intelligence* akan mampu menghasilkan *visual intelligence system*. Pendapat lain mengatakan *computer vision* adalah proses otomatis yang mengintegrasikan sejumlah besar proses untuk persepsi visual, seperti akusisi citra, pengolahan citra, pengenalan dan pembuatan keputusan. *Computer vision* mencoba meniru cara kerja sistem visual manusia *(human vision*) yang sesungguhnya sangat kompleks, bagaimana manusia melihat objek dengan indra penglihatan (mata), lalu citra objek diteruskan ke otak untuk diinterpretasi sehingga manusia mengerti objek apa yang tampak dalam pandangan mata.

Selanjutnya hasil interpretasi ini digunakan untuk pengambilan keputusan.

Computer vision adalah salah satu cabang dari AI yang difokuskan pada pengembangan algoritma untuk menganalisis isi dari suatu gambar. *Computer vision* merupakan proses otomatis yang mengintegrasikan sejumlah besar proses untuk persepsi visual, seperti mengakuisisi citra, pengolahan citra, pengenalan dan membuat keputusan. *Computer vision* mencoba meniru cara kerja sistem visual manusia (*human vision*) yang sesungguhnya sangat kompleks. Manusia melihat dengan objek dengan indra penglihatan (mata), lalu citra objek diteruskan ke otak untuk diinterpretasi sehingga manusia mengerti objek apa yang tampak dalam pandangan mata. Hasil interpretasi ini digunakan untuk pengambilan keputusan.

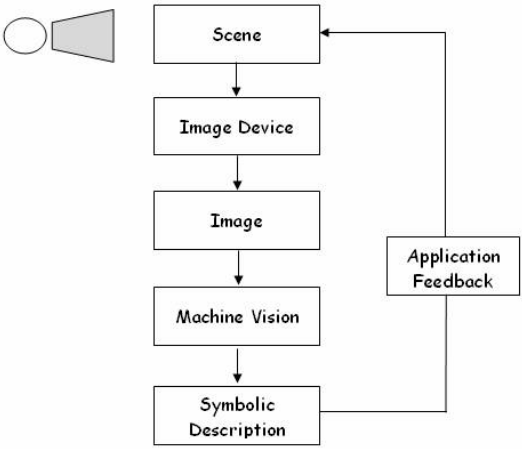
Computer vision mempunyai tujuan utama untuk menganalisis objek fisik yang nyata berdasarkan *image* yang ditangkap dari sensor. *Computer vision* diciptakan untuk membangun sebuah mesin pandai yang dapat melihat, seperti robot AIBO yang dibuat oleh perusahaan SONY, di mana robot yang berbentuk anjing tersebut dapat melihat dan mengenali pemiliknya. Contoh penerapan *computer vision*:

- Pengendalian proses (misalnya, sebuah robot industri atau kendaraan otonom).
- Mendeteksi peristiwa (misalnya, untuk pengawasan visual atau orang menghitung).

- Mengorganisir informasi (misalnya untuk pengindeksan *database* foto dan gambar urutan).
- Modeling benda atau lingkungan (misalnya industri inspeksi, analisis gambar medis/topografis).
- Interaksi (misalnya sebagai *input* ke perangkat untuk interaksi manusia komputer).

5.3 Elemen-Elemen Visi Komputer

Dalam visi komputer terdapat 7 struktur yang mendasari elemen-elemen suatu komputer vision, yaitu *light sources, scene, image device, machine vision, symbolic description,* dan *possible application feetback.* Gambar berikut merupakan struktur dari suatu mesin visi.

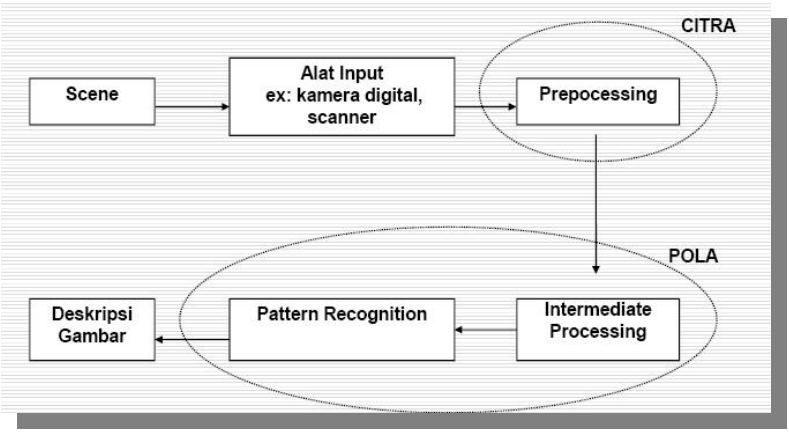


Gambar 5.1 Struktur Visi Komputer

Adapun komponen visi komputer terdiri atas:

- *Light sources*, merupakan sumber cahaya yang digunakan sebagai sumber untuk aplikasi seperti layaknya laser, sistem robotika dan sebagainya.
- *Scene*, merupakan kumpulan objek.
- *Image Device*, merupakan alat yang digunakan untuk mengubah representasi dari keadaan sesungguhnya.
- *Machine Vision*, merupakan sistem yang menginterpresentasikan gambar yang berkenaan dengan ciri-ciri, pola maupun objek yang dapat ditelusuri oleh sistem.
- *Symbolic Description*, merupakan sistem yang dapat digunakan menganalogikan kinerja sistem ke simbol-simbol tertentu yang dimengerti oleh sistem.
- *Possible Application Feedback*, merupakan suatu keadaan yang dapat memberikan respon untuk menerima gambar dari suatu sistem penglihatan.

Penerapan visi komputer meliputi pengendalian proses (misalnya sebuah robot industri atau kendaraan kemudi otomatis); mengorganisir informasi (misalnya untuk *face recoqnition*); *modeling* benda atau lingkungan (misalnya, industri inspeksi, analisis gambar medis/topografis); interaksi (misalnya, sebagai *input* ke perangkat untuk interaksi manusia komputer).



Gambar 5.2 Proses dalam Visi Komputer

5.4 Tahapan-Tahapan Proses dalam Visi Komputer

- *Image Acquisition*

  Pada manusia, *image acquisition* dimulai dengan mata, kemudian informasi visual diterjemahkan ke dalam suatu format yang kemudian dapat dimanipulasi oleh otak. Pada komputer umumnya untuk menangkap sebuah sinyal visual digunakanlah kamera video. Kamera menerjemahkan sebuah *scene* atau *image*. Kemudian sinyal listrik ini diubah menjadi bilangan biner yang akan digunakan oleh komputer untuk pemrosesan. Keluaran dari kamera adalah berupa sinyal analog, di mana frekuensi dan amplitudonya (frekuensi berhubungan dengan jumlah sinyal dalam satu detik, sedangkan amplitudo berkaitan dengan tingginya sinyal listrik yang dihasilkan) merepresentasikan detail ketajaman (*brightness*) pada *scene*. Kamera mengamati

sebuah kejadian pada satu jalur dalam satu waktu, memindainya dan membaginya menjadi ratusan garis horizontal yang sama. Tiap-tiap garis membuat sebuah sinyal analog yang amplitudonya menjelaskan perubahan *brightness* sepanjang garis sinyal tersebut. Karena komputer tidak bekerja dengan sinyal analog, maka sebuah ADC dibutuhkan untuk memproses semua sinyal tersebut oleh komputer. ADC ini akan mengubah sinyal analog yang direpresentasikan dalam bentuk informasi sinyal tunggal ke dalam sebuah aliran (*stream*) sejumlah bilangan biner. Bilangan biner ini kemudian disimpan di dalam memori dan akan menjadi data *raw* yang akan diproses.

- *Image processing*

  Image processing membantu peningkatan dan perbaikan kualitas *image*, sehingga dapat dianalisis dan diolah lebih jauh secara lebih efisien. Tahapan-tahapan dalam *image processing*:

1. *Pre processing*: mengatur cahaya, warna, dan kontras. Melakukan pengaturan jumlah cahaya, warna dan kontras berbagai objek di dalam *scene*. Perhatikan kontrol terhadap intensitas iluminasi, posisi sumber cahaya atau objek, sehingga diperoleh tingkat kelihatan yang maksimum dan informasi yang lengkap.
2. *Noise reduction*: mengeliminasi *noise* dan distorsi. *Noise reduction* disebut juga dengan *image averaging*, yang digunakan untuk mengeliminasi *noise* dan distorsi. Pada

proses ini sistem visi meng-*capture* sejumlah *view* yang berurutan dari sebuah *scene*, kemudian meratakannya. Misalnya sebagai akibat dari perubahan cahaya, posisi oebjek berubah, maka akan berakibat bagi perubahan image yang ditangkap oleh kamera, maka reduksi *noise* dilakukan secara *random*.

3. *Gray scale modification*: mengatur tingkat terang dan gelap dari sebuah *image*. Teknik pemrosesan berikutnya adalah dengan mengatur tingkat terang (*lighten*) dan gelap (*darken*) dari sebuah *scene*. Hal ini disebut dengan *grayscale modification*. Sejalan dengan itu, maka untuk pemrosesan 8 bit biner *array*, angka 0 (00000000) merepresentasikan hitam dan angka 255 (11111111) merepresentasikan warna putih. Nilai-nilai yang berada di antara nilai tersebut disebut dengan *grayscale*.
4. Histogram *Flattering*: meratakan sinyal *input* dari hasil modifikasi *grayscale*, sehingga diperoleh detail sinyal biner yang lebih jelas dan tajam. Histogram *flattering* adalah salah satu metoda lain yang digunakan untuk meningkatkan kualitas *images*. Histogram adalah sebuah grafik *bar* digital yang digunakan untuk mengatur informasi secara statistik. Histogram bisa dibangun dari *image* biner dengan menghitung jumlah beda level cahaya *gray*. Proses *flattering* ini dilakukan dengan mengatur level *threshold*, dengan membagi *pixel brightness range* ke dalam blok *brightness values*. Nilai *pixel* yang ada dalam

range yang bervariasi kemudian diidentifikasi dan *pixel* yang berada dalam *range* itu ditandai sebagai nilai intensitas yang sama (*common intensity value*).

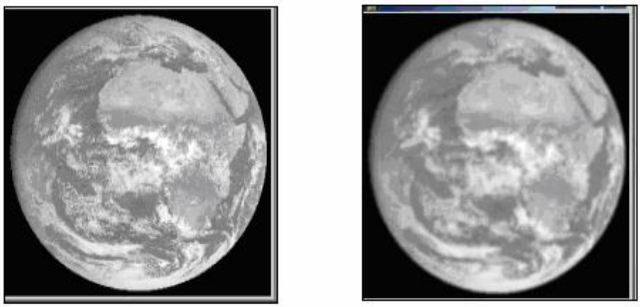
◎ *Image analyst*

Image analyst akan mengeksplorasi *image* ke dalam bentuk karateristik utama dari objek melalui suatu proses/teknik. *Image analysis* akan mengeksplorasi *scene* ke dalam bentuk karakteristik utama dari objek melalui suatu proses investigasi. Sebuah program komputer akan mulai melihat melalui bilangan biner yang merepresentasikan informasi visual untuk mengidentifikasi fitur-fitur spesifik dan karakteristiknya. Lebih khusus lagi program *image analysis* digunakan untuk mencari tepi dan batas-batasan objek dalam *image*. Sebuah tepian (*edge*) terbentuk antara objek dan latar belakangnya atau antara dua objek yang spesifik. Tepi ini akan terdeteksi sebagai akibat dari perbedaan level *brightness* pada sisi yang berbeda dengan salah satu batasnya.

Tahapan teknik *image analysis*:

- *Surface Smoothing*: menghilangkan efek *shading* pada *image* dengan intensitas cahaya yang rendah. Permukaan suatu objek pada dasarnya sangat mudah untuk diidentifikasi, karena pada umumnya permukaan tersebut relatif seragam dalam intensitas cahaya. Persoalan muncul pada bagian permukaan yang memperoleh intensitas cahaya rendah, sehingga permukaan tersebut sedikit

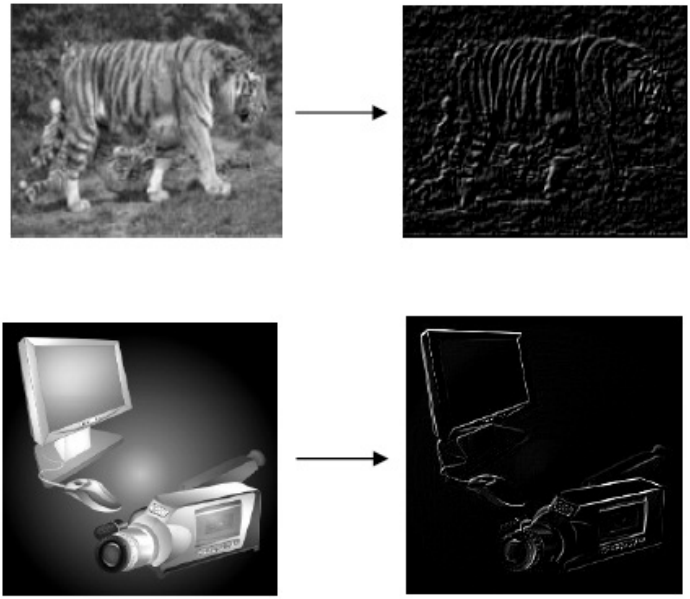
membayang (*shading*). Kondisi *shading* ini menjadi persoalan ketika proses deteksi sudut (*edge detection*) dilakukan. Untuk mengatasinya, maka pada permukaan yang mengalami *shading*, dilakukan proses *surface smoothing*, sehingga sisi shading menjadi tampak lebih jelas pada saat *edge detection* dilakukan.



Gambar 5.3 Citra Asli dan Citra Hasil *Mask* 3x3

- *Edge Detection*: mendeteksi sudut-sudut objek pada suatu *image* sehingga dapat dibedakan batas dari keduanya. Setelah permukaan objek dihaluskan, maka selanjutnya komputer melakukan pendeteksian terhadap sudut-sudut objek, sehingga bisa dibedakan batas antara dua objek. Pada *edge detection* ini, program komputer melakukan pembandingan terhadap intensitas *pixel*. Jika dari hasil pembandingan diperoleh perbedaan yang kecil, maka dianggap objek mempunyai area permukaan yang luas. Namun jika dari hasil perbandingan diperoleh peredaan

yang cukup besar, maka itu mengindikasikan adanya *edge* pada objek.



Gambar 5.4 Deteksi Tepi

- *Image understanding*

  Komputer bisa memiliki pengetahuan (*knowledge*) tentang suatu objek yang ada dalam *image*. Pada tiga tahapan sebelumnya, komputer telah memperoleh sebuah *image* yang telah diperbaiki kualitasnya dan dianalisis. Namun sejauh ini komputer tetap tidak mengetahui apa arti dari *image*/*scene* tersebut, apa yang ketahui dari sebuah objek atau bagaimana hubungan antar objek. Tahapan akhir dari *computer vision* kemudian adalah bagaimana komputer

bisa memiliki pengetahuan (*knowledge*) tentang suatu objek yang ada dalam *scene*.

Tahapan metode *image understanding*:

1. *Templates matching*: membandingkan suatu model objek yang telah tersimpan dalam objek biner dalam komputer dengan objek biner yang dilihat oleh sistem *computer vision* secara *pixel by pixel*. *Template matching* melakukan pembandingan suatu model objek yang telah tersimpan dalam objek biner dalam komputer dengan objek biner yang dilihat oleh sistem *computer vision*. Proses pembandingan dilakukan dalam bentuk basis *pixel by pixel*. Perbandingan *pixel* dalam dua *image* akan menghasilkan suatu nilai yang berbeda. Perbedaan inilah yang kemudian dihitung. Jika perbedaannya sama dengan nol, maka kedua pola adalah sesuai, jika perbedaannya kecil, maka *image*-nya saling mendekati dan jika besar maka *image* tidaklah sama.
2. *Featured matching*: membandingkan fitur-fitur yang dimiliki oleh objek dengan fitur tentang suatu objek yang telah disimpan dalam komputer. Metoda lain selain *pattern matching* adalah *feature matching*. *Feature matching* tidak membandingkan *actual image* dengan *stored image* seperti pada *template matching*, tetapi pencocokan dilakukan pada fitur-fitur yang dimiliki oleh objek dengan fitur tentang suatu objek yang telah disimpan dalam komputer. Fitur-fitur tersebut disimpan dalam suatu blok yang disebut

dengan *block world* yang berisi informasi tentang objek yang mempunyai fitur-fitur yang sama.

Dari semua penjelasan mengenai struktur tersebut, ada 3 elemen yang sangat mendasari suatu sistem visi, yaitu *image processing, pattern classfication,* dan *scene analysis.*

1. ***Image Processing***

   Bagian ini berfungsi mengubah atau mengkonversi gambar-gambar eksternal menjadi suatu representasi yang dibutuhkan.Berikut bagian dari *image processing*.



Gambar 5.5 Fase *Image Processing*

2. **Klasifikasi Pola**

   Ide klasifikasi pola (*pattern classification)* ini adalah bagaimana suatu mesin pintar (dalam hal ini adalah komputer) dapat mengetahui berbagai macam dan bentuk pola, seperti garis, kurva, bayangan dan berbagai pola lainnya. Artinya, jika mesin tersebut diberi suatu *input* berupa pola tertentu maka mesin dapat mengerti pola yang diberikan itu. Berikut adalah bagian dari suatu proses *image processing*.



Gambar 5.6 Fase Klasifikasi Pola

3. *Scene Analysis*

Bagian ini berfungsi untuk menganalisis suatu informasi yang didapat dari suatu paparan (baik itu berupa gambar atau pola-pola) sehingga didapat suatu hasil analisis yang diperlukan untuk tahap selanjutnya. Pada bagian ini visi komputer mengidentifikasi sebuah gambar untuk menentukan ciri-ciri dan karakteristiknya.

Ada beberapa hal yang perlu diperhatikan dalam teori yang dipaparkan David Marr, yaitu:

1. *Problem with Earlier Approaches*
2. *Template Matching*
3. *Feature Extraction*

Metode ini adalah metode yang menggunakan teknik pemisahan objek menjadi unsur-unsur fundamentalnya.

a. Fourier Analysis

Metode ini biasanya digunakan dalam beberapa teknik peningkatan kualitas citra digital, seperti penajaman citra, penghalusan citra, kontras citra dan lain-lain.

b. Structural Description

Metode ini adalah metode yang menjabarkan suatu objek terhadap komponen-komponennya serta dengan relasi yang berlaku pada sistem tersebut dengan menggunakan kalkulus predikat.

5.5 Proses Visi Komputer

Sebuah komputer menirukan penglihatan manusia dalam 4 langkah dasar, yaitu akuisisi citra, pengolahan citra, analisis citra dan pemahaman citra. Berikut ini adalah penjelasan singkat dari langkah-langkah tersebut.

1. Akuisisi Citra

Akuisisi citra menerjemahkan transformasi visual ke dalam sebuah format yang bisa dimanipulasi lebih lanjut oleh otak. Untuk awalnya, sebuah sistem penglihatan komputer atau visi komputer memerlukan sebuah mata. Di dalam hampir semua visi komputer, matanya adalah sebuah kamera TV (video). Kamera menerjemahkan sebuah adegan atau gambar ke dalam sinyal elektris. Sinyal-sinyal bisa diterjemahkan ke dalam bilangan biner seperti yang bisa dilakukan oleh komputer. *Output* kamera televisi adalah sebuah sinyal analog yang frekuensi dan amplitudonya mewakili *brightness* detail sebuah adegan. Kamera mengobservasi sebuah adegan pada satu saat, lalu *scanning* membaginya ke dalam ratusan garis halus. Setiap garis menciptakan sebuah sinyal analog yang amplitudonya

menggambarkan atau mewakili perubahan *brightness* sepanjang garis. Pada komputer digital hal ini tidak dapat dilakukan karena komputer digital tidak dapat menerima sinyal analog, oleh karena itu dibutuhkan sebuah *converter* yang dapat merubah sinyal analog menjadi sinyal digital. ADC merupakan alat yang dapat mengkonversi sinyal analog yang mewakili satu garis informasi ke dalam sejumlah arus bilangan biner. Kemudian, bilangan biner disimpan dalam memori dan menjadi data mentah yang akan diproses oleh komputer.

2. Pengolahan Citra

Tahap berikutnya penglihatan komputer (visi komputer) melibatkan beberapa awal manipulasi data biner. Pengolahan citra membantu memperbaiki kualitas citra agar komputer dapat menganalisis dan memahami lebih efisien. Pengolahan citra memperbaiki *signal to noise ration.* Sinyal tentu merupakan informasi yang mencerminkan objek dalam gambar. *Noise* adalah setiap gangguan, gelombang atau penyimpangan yang mengganggu objek. Dengan berbagai alat komputasi, komputer dapat memperbaiki *signal to noise ratio*. Misalnya kontras dalam suatu adegan bisa diperbaiki. Gangguan, seperti pantulan yang tidak diharapkan, bisa diperbaiki pula. Setelah citra dibersihkan dan diperbaiki, baru kemudian dianalisis.

3. **Analisis Citra**

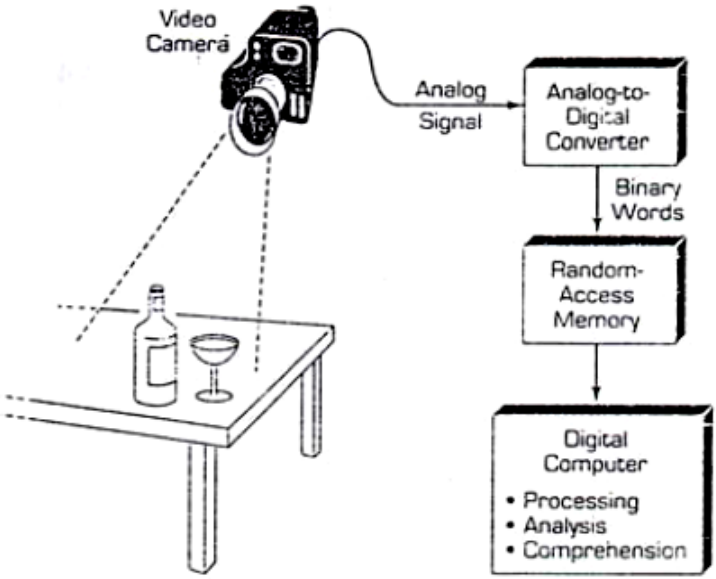
   Analisis citra mengeksplorasi adegan untuk menentukan adanya sesuatu. Program komputer mulai melihat bilangan yang menggambarkan informasi visual agar bisa mengidentifikasikan ciri-ciri khas dan karakteristiknya. Secara lebih spesifik, program analisis citra mencari sisi-sisi dan batasnya. Batas-batas atau sisi-sisi bisa diidentifikasi karena tingkat *brightness* yang berbeda pada setiap sisi dari batas. Analisis gambar mencari tekstur dan bayangan antara garis-garis. Keduanya sangat berguna untuk pemroses di kemudian hari saat mengidentifikasikan adegan.

4. **Pemahaman Citra**

   Langkah terakhir dalam proses visi komputer adalah pemahaman dengan mengidentifikasikan objek-objek spesifik dan hubungannya. Pemahaman adegan memerlukan pencocokan *template*. Komputer sudah diprogram sebelumnya dengan citra biner yang sudah tersimpan atau *template* yang menggambarkan objek spesifik. Dengan menggunakan program pencarian dan teknik pencocokan pola (*pattern matching)*, komputer mencari melalui informasi biner yang merepsentasikan adegan dan membandingkan pada berbagai *template*. Bila sudah cocok, maka suatu objek telah teridentifikasi. Proses pencocokan *template* berlangsung sampai semua infromasi dalam adegan dianalisis dan semua objek teridentifikasi.

5.5.1 Akuisisi Citra

Kamera TV menerjemahkan adegan untuk dianalisis ke dalam sinyal analog yang kemudian diterjemahkan ke dalam bentuk biner oleh *analog to digital converter*. Bilangan biner disimpan dalam RAM, lalu diproses oleh komputer. Berikut ini merupakan gambaran yang menjelaskan konsep akuisisi citra dengan kamera TV.



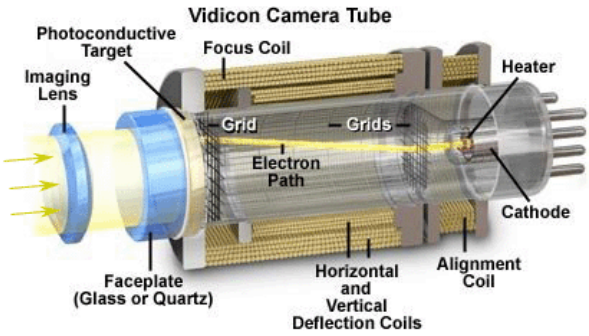
Gambar 5.7 Konsep Akuisisi Citra

a. Kamera Video

Dua alat yang umum digunakan dalam kamera visi komputer untuk mengkonversi cahaya ke dalam sinyal elektrik adalah tabung *vidicon* dan *array* CCD. Selama bertahun-tahun, tabung *vidicon* sudah menjadi alat utama yang digunakan dalam kamera televisi komersial. Walaupun demikian, untuk sistem komputer *vision*, Charged Coupled Devices (CCD's) sudah lama digunakan secara luas. Alat semikonduktor ini berukuran kecil, sensitivitas cahayanya lebih besar, dan kebutuhan *power*-nya lebih kecil dibandingkan *vidicon*. Baik vidicon maupun kamera CCD menghasilkan *output* sinyal video analog hitam dan putih.

1. Tabung Vidicon

Gambar 5.8 menunjukkan gambaran sederhana tabung vidicon. Komponen kunci dalam tabung adalah target *photoconductive* yang ada di sebelah kiri. Lensa memfokuskan adegan yang akan dilihat pada target dan target menyimpan adegan dalam bentuk muatan listrik. Konduktifitas target bervariasi sesuai dengan jumlah cahaya yang diserap. Di mana ada cahaya yang minimum, material memiliki resistensi elektris tinggi. Dengan demikian, titik pada target dengan sedikit cahaya mempunyai resistansi lebih rendah.



Gambar 5.8 Contoh Tabung Vidicon

Tabung selebihnya terdiri atas *electron gun* dengan *magnetic coil* yang ada di sekitarnya agar menghasilkan arus elektron yang difokuskan menjadi berkas sempit untuk menabrak bagian dalam target. *Magnetic coil* lainnya yang berada di sekitar tabung vidicon bertujuan menghasilkan defleksi vertikal dan horizontal dari berkas elektron. *Coil-coil* ini menyebabkan elektron men-*scan* dari sebelah kiri ke kanan dan dari atas ke bawah melintas bagian dalam target. Akibatnya adalah arus listrik mengalir di antara target dan transparan sinyal elektroda. Arus listrik ini proporsional dengan amplitudo cahaya. Arus ini merupakan sinyal *output* video.

2. Charge Coupled Devices (CCD)

CCD adalah sirkuit terpadu sensitif cahaya yang dirancang untuk mengkonversi citra visual ke dalam

sinyal elektris video. Sistem lensa memfokuskan adegan pada sebuah substrat foto konduktif seperti silikon. Alat menyerap cahaya dan menyimpannya sebagai muatan dalam ribuan kapasitor kecil persegi dan persegi panjang. Silikon membentuk plat tunggal untuk semua kapasitor. Plat-plat (lempengan) lainnya merupakan elektroda metal kecil individu yang dipisahkan dari *photoconductor silicon* oleh *insulator* tipis *layer silicon* dioksida. Cahaya jatuh pada substrat menyebabkan kapasitor diis dan dikosongkan pada tingkat tertentu tergantung pada intensitas cahaya yang datang pada substrat langsung di bawah plat kapasitor kecil. Sinyal *output* CCD bervariasi dari sekitar 0 sampai 1 volt dengan kedua ujung mewakili hitam dan putih. *Interface scanning* tidak digunakan. Sebagai gantinya, baris matriks CCD merupakan hal yang sudah ter-*scan*. Kamera video CCD sangat dibutuhkan untuk visi komputer. Karena ukurannya kecil, sensitivitasnya tinggi, dan sangat menguntungkan. Kamera pun lebih kasap dibandingkan dengan tabung vidicon yang lunak dan tidak membutuhkan voltase *filament* atau voltase yang tinggi seperti vidicon. Bobotnya ringan dan kebutuhan *power* rendah serta mudah dibawa.

b. Konversi Analog To Digital

Sinyal *output* video dari kamera dimasukkan ke dalam alat konversi *analog to digital* (Analog to Digital, ADC). Kemudian, secara periodik ADC men-*sampling* sinyal analog mengkonversi amplitudo ke dalam bilangan biner paralel. Banyak metode digunakan agar bisa menghasilkan konversi analog to digital, yaitu metode *counter feedback,* metode*sequential approximation.* Meskipun demikian, tak ada satu pun di antaranya digunakan dalam aplikasi ADC video. Sebaliknya, paralel kecepatan tinggi khas atau *flashconverter* banyak digunakan. Hanya ADC tipe ini yang cukup cepat untuk sinyal video frekuensi tinggi. Intensitas cahaya sebuah garis *scan* berubah cepat dan nilainya bervariasi dalam rentang yang lebar dari gelap ke terang. Untuk beberapa adegan, frekuensi sinyal *output* audio mungkin sangat rendah, tetapi adegan-adegan dengan detail yang tinggi dan rentang intensitas cahaya sangat luas maka frekuensi *output* akan mencapai beberapa MHz. Dalam hampir semua sistem visi komputer, satu garis *scan* disampel 200 sampai 500 kali per garis. Garis scan memerlukan waktu kira-kira 60 mikrodetik. Artinya, rata-rata *sampling* ADC harus dilakukan satu kali dalam setiap 120 sampai 300 nanodetik. Untuk setiap sampel, *output* ADC adalah bilangan 8 bit biner yang nilainya proporsional pada intensitas cahaya.

1. *Pixel*

Pixel (*picture element*) adalah nilai intensitas cahaya titik tertentu pada garis *scan*. Dengan demikian, *pixel* adalah elemen kecil di mana setiap garis *scan* dipecahkan. Setiap garis *scan* mengandung kira-kira 200 sampai 500 *pixel*. *Pixel* diwujudkan dalam bayangan keabuan yang dibentuk oleh pola suatu bilangan biner. Dalam beberapa sistem yang rendah, 4 bit *output* sudah memuaskan namun untuk sistem dengan resolusi lebih tinggi biasanya menggunakan 8 bit dan 6 bit. Sekarang, 8 bit lebih populer karena menyediakan 256 tingkat keabuan individu yang menyediakan detail sangat halus. lebih lanjut, 8 bit merupakan ukuran kata standar yang digunakan secara luas dalam komputer digital. Kata yang terdiri atas 8 bit mengacu pada byte dan hampir semua komputer digital mampu memproses pada tingkat byte.

2. RAM

Setiap *pixel* direpresentasikan dengan bilangan binder 8 bit yang disinpan dalam RAM yang besar. *Chip* semikonduktor RAM digunakan dalam memori. Waktu aksesnya harus sangat cepat agar bisa menerima output kecepatan tinggi dari CDC. Memorinya juga harus besar untuk menyimpan banyak byte *pixel* yang memuat adegan-adegan gambar. Pada saat membutuhkan penyimpanan gambar, komputer menyiapkan tugas pada RAM khusus. Pada hampir semua visi komputer, RAM

ini dipisahkan dari RAM yang digunakan dalam komputer. Kita menyebutnya *buffer* RAM atau *frame buffer,* di mana *frame* mengacu pada digitalisasi adegan gambar. Pada tahap ini, komputer telah menyimpan di dalam memori representasi digital adegan gambar yang harus dianalisis dan dimengerti. Sekali citra adegan gambar ada di dalam memori, komputer bisa mengambil alih dan melaksanakan banyak operasi untuk memperbaiki adegan gambar, menganalisisnya, menterjemahkannya ke dalam berbagai bentuk yang berbeda dan akhirnya memahami keadaan sebenarnya yang ada di sana.

3. Monitor

Hampir semua sistem visi komputer memasukkan sebuah monitor video, suatu *set* TV tanpa mekanisme *channel tunning*, untuk melihat adegan yang dilihat kamera. Monitor sangat berguna bagi operator sistem visi komputer untuk mengetahui secara pasti apa yang dilihat kamera dan apa yang telah disimpan komputer di dalam memorinya. Salah satu cara mudah mengerjakannya adalah dengan mengisi sinyal video dari *output* kamera secara langsung ke dalam monitor video, sehungga memudahkan operator mengarahkan kamera, memfokuskannya atau cara lain menyesuaikannya agar bisa terlihat apa yang diinginkannya. Beberapa sistem visi komputer dapat menerjemahkan citra digital yang

disimpan dalam RAM kembali ke dalam video untuk ditayangkan dalam monitor. Caranya adalah dengan memasukkan bilangan biner secara berurutan ke *Digital to Analog Converter* (DAC). DAC adalah kebalikan dari ADC yaitu mengambil bilangan biner dan mengkonversinya ke dalam suatu *output* analog. Setiap nilai *output* tertentu yang *fix*. Bila nilai-nilai dimasukkan dengan cepat ke dalam DAC, maka *output* mengubah *incrementasi discrete* atau pencipitaan ulang suatu perkiraan sinyal *output* analog yang sebenarnya sudah didigitasi. Semua byte informasi *pixel* satu garis diisikan pada ADC ke DAC untuk menciptakan kembali sinyal analog yang mencerminkan satu garis *scan*. Pada ujung garis, pulsa *sync* dibuat agar *beam* elektron di dalam CRT membersihkan kembali dan memulai dengan garis horizontal yang baru. Kemudian, urutan byte *pixel* yang mencerminkan garis *scan* berikut dikeluarkan ke DAC, lalu ke monitor. Dengan cara ini, proses *scanning* dari kamera mulai kembali dan adegan yang sudah didigitasi ditayangkan.

4. 3D ke 2D

Kamera tidak melihat dalam 3D. Kita akan mendapatkan representasi 2 dimensi dari apa yang dilihat kamera apa pun. Kita melihat lebar dan tinggi subjek dengan akurat, tetapi kehilangan dimensi dalam perspektif kedalamannya. Tanpa informasi kedalaman, kita sulit

menentukan jarak antara kamera dan objek yang sedang dibidik. Selanjutnya, kita sulit menentukan jarak antara objek yang berbeda di dalam suatu adegan. Teknik pengindraan kedalaman di dalam sistem visi komputer adalah dengan menggunakan 2 kamera yang akan mengasilkan *binocular* atau *stereo vision* yang memungkinkan kita bisa menentukan kedalaman. Di dalam sistem 3D, adegan yang sama dibidik oleh dua kamera. Kamera ditempatkan di jarak yang pasti dan terpisah. Kemudian, adegan dari dua kamera didigitasi dan disimpan dalam memori. Satu objek dalam adegan telah diidentifikasi, sehingga komputer dapat melakukan berbagai operasi matematika untuk membantu menghitung jarak dengan objek dan antara objek-objek.

5.5.2 Pengolahan Citra

Pemrosesan/pengolahan citra dikenal sebagai penghalusan citra yang merupakan proses perbaikan kualitas citra. Banyak permasalahan yang membuat kualitas gambar menjadi buruk, misalnya lensa kamera bisa menimbulkan distorsi atau gangguan, ketidaktepatan atau inkonsistensi dalam area target vidicon atau CCD akan menimbulkan intensitas cahaya yang tidak seimbang, Non-linearitas dalam proses *scanning* bisa juga menimbulkan gangguan.

Masalah lain yang patut diperhatikan adalah *noise* yang bisa mengganggu sinyal yang diharapkan. *Noise* bisa

menyebabkan pantulan yang tidak biasa karena tekstur yang melekat pada permukaan objek yang sedang dibidik. Ada beberapa cara untuk menangani permasalahan tersebut seperti mengurangi *noise*, modifikasi skala keabuan, dan histogram *flattening*.

1. Prapemrosesan

Sebelum melakukan perbaikan citra, kita perlu mengambil beberapa langkah yang disebut prapemrosesan. Langkah pertama yaitu penggunaan filter pada kamera untuk mengembalikan julmah cahaya yang masuk serta warna dan kontras pada berbagai objek di dalam adegan. Kedua, banyak sistem visi komputer yang bekerja dalam suatu lingkungan terkendali yang bukan hanya mengendalikan tingkat iluminasi, tetapi juga posisi sumber-sumber cahaya atau objek yang dibidik agar mencapai visibilitas dan komprehensi maksimal.

2. Mengurangi Gangguan *Noise*

Image averaging atau pemerataan citra membantu mengurangi *noise* dan gangguan lainnya. Dalam proses, sistem visi komputer menangkap pandangan beberapa objek secara berurutan dan merata-ratakannya. Variasi minor mungkin terjadi dalam suatu adegan. Misalnya, bila posisi cahaya berubah atau objek pindah, maka refleksi berbagai elemen yang ada di dalam adegan akan berubah pula. Pemindahan sumber cahaya yang disengaja tentu kadang-kadang bisa dilakukan dengan maksud memindahkan

penyediaan pandangan berbeda dari objek yang sama untuk memberikan kesempatan kepada komputer dan pemerataanya. Ketika sumber cahaya berpindah, bayangan dan pantulan permukaan berubah pula. Karena *noise* ini bersifat acak maka proses pemerataan berkurang sampai tingkat terendah.

3. Modifikasi Skala Keabuan

Perbaikan kontras atau *contrast improvement* adalah salah satu jenis modifikasi skala abu-abu. Kontras adalah rentang antara titik tergelap dan titik terterang dalam suatu gambar. Rincian halus muncul menjadi terbaik bila ada kontras tinggi di antara *pixel* yang berdekatan warna objek yang berdekatan, intensitas, posisi cahaya dan distorsi sistemnya bisa membunyikan sisi dan batas karena ada kontras yang tidak sempurna. Dalam kasus semacam itu, kontras dapat diperbaiki dengan cara pemrosesan. Kita dapat melakukannya dengan mengidentifikasikan rentang nilai intensitas *pixel* ditekan atau dinaikkan. Dengan kata lain, nilai *threshold* atas dan bawah sudah di-*set* dan setiap *pixel* yang jatuh pada rentang di atas akan dimodifikasi. Modifikasi dilakukan dengan menambah, mengurangi, mengalikan, dan membagi nilai-nilai *pixel* dengan *constant* tergantung kepada penghalusan yang dibutuhkan.

4. Histogram *Flattening*

Histogram *flattening* adalah suatu *bar chart vertical* yang digunakan untuk mengeksploitasi informasi statistik.

Histogram bisa dibangun dari citra biner dengan menghitung jumlah waktu setiap tingkat perbedaan keabuan. Dengan histogram yang dibangun dalam memori, sebuah program bisa menentukan bila ada nilai cahaya yang berlebih atau rendah. Misalnya, suatu analisis mungkin menemukan suatu jumlah cahaya yang tidak biasa dari tingkat sangat cerah atau tingkat sangat gelap. Bila nilai intensitas terntentu terdapat dalam seluruh adegan gambar, maka histogram *flattening* bisa menguranginya. Histogram *flattening* mempertajam citra dengan menonjolkan intensitas langka yang terjadi pada batas atau sisi. Nilai intensitas yang umum terjadi pada daerah luas atau intensitas yang sama tidak ditekankan. Dalam setiap kasus, hal ini akan menjadi teknik yang berguna bagi usaha memperbaiki rincian citra sebelum menganalisisnya.

5. Aplikasi Lain Pemrosesan Citra

Pemrosesan citra biner bisa dilakukan dalam berbagai bentuk. Karena komputer bisa melakukan operasi matematika virtual, maka *pixel* di dalam suatu adegan bisa dimanipulasi dengan berbagai cara. Di samping penambahan, pengurangan, perkalian, dan pembagian, teknik pemrosesan *pixel* termasuk pula fungsi kalkulus seperti diferensiasi. Semua rangkaian skema sudah dikembangkan untuk memperbaiki kualitas sinyal. Berikut adalah contoh-contoh kasus yang memakai pemrosesan citra: *scan* CAT pada ilmu

medik, radar dan sonar pada kapal laut, dan pengambilan gambar-gambar planet.

6. Analisis Citra

Setelah gambar (*image*) yang ditangkap oleh kamera diperbaiki kualitasnya pada tahap pengolahan citra, gambar tersebut selanjutnya dianalisis untuk menentukan objek-objek yang terdapat pada gambar tersebut dengan mendeteksi *outline* dari suatu objek (inti: memisahkan setiap objek dari *background* dan dari objek lainnya). Analisis citra terdiri atas dua tahap, yaitu:

- **Pelembutan Permukaan**

  Pada tahap ini, mula-mula dilakukan pendeteksian terhadap permukaan-permukaan yang ada di dalam gambar. Pendeteksian permukaan ini relatif lebih mudah dilakukan karena *pixel-pixel* yang menjadi bagian dari permukaan tersebut memiliki intensitas cahaya dan nilai warna yang relatif berdekatan. Setelah itu, permukaan-permukaan yang ada pada gambar diperhalus dengan cara penyederhanaan warna pada *pixel-pixel* dari permukaan tersebut agar lebih mudah diidentifikasi lebih lanjut oleh komputer (warna gradiasi halus diubah menjadi *solid color*).

- **Pendeteksian sisi**

  Pada tahap ini, gambar yang permukaannya telah diperhalus kemudian dideteksi bagian sisi-sisinya untuk menentukan objek-objek yang terdapat pada gambar

tersebut. Pendeteksian ini dilakukan dengan membandingkan intensitas dari *pixel* yang saling berdekatan. Jika intensitas antar *pixel* memiliki perbedaan yang signifikan, maka sisi dari suatu objek telah terdeteksi. Setelah seluruh sisi yang terdapat pada gambar telah terdeteksi, maka komputer akan membuat *outline* gambar berdasarkan sisi-sisi tersebut lalu mengubah gambar menjadi *binary image* (gambar hitam-putih, dengan *outline* gambar berwarna hitam dan *background* berwarna putih).

5.5.3 Pemahaman Citra

Sampai sini, komputer sudah mampu mendeteksi objek-objek yang telah tertangkap oleh kamera. Namun, komputer belum menyadari isi/maksud dari objek-objek yang dilihat dan hubungan antar objek. Oleh karena itu, tujuan dari langkah akhir visi komputer ini adalah memberikan pengetahuan kepada komputer mengenai adegan yang dilihatnya. Komputer akan dibekali dengan *template-template* yang menggambarkan objek spesifik. Dengan menggunakan program pencarian dan teknik pencocokan pola (*pattern matching*), komputer mencari melalui informasi biner yang merepresentasikan adegan dan membandingkannya dengan *template-template* yang ada. Jika komputer menemukan kecocokan antara adegan yang dilihat dengan *template* yang ada, maka dapat dikatakan bahwa komputer memahami maksud dari apa yang dilihatnya. Jika tidak

ada *template* yang cocok, berarti komputer tidak mengerti maksud dari apa yang dilihat.

5.6 Aplikasi Visi Komputer

Beberapa contoh dari penerapan atau aplikasi visi komputer adalah sebagai berikut:

1. **Mesin Visi**

   Aplikasi terbesar dari visi komputer adalah mesin visi. Mesin visi mengacu pada penggunaan peralatan visi komputer dan teknik dengan proses manufaktur yang biasa dilakukan oleh beberapa jenis mesin. Tujuan mesin visi dalam aplikasi manufaktur adalah untuk menggantikan manusia dalam beberapa tugas dan membantu mempercepat atau mudahnya proses manufaktur dan lainnya. Biasanya, mesin visi menggantikan manusia dalam mengerjakan tugas berulang-ulang yang dapat menimbulkan kebosanan.

2. **Robot Vision**

   Salah satu aplikasi umum visi komputer adalah robot. Seperti yang kita tahu, robot adalah sebuah mesin yang diprogram untuk mengerjakan manipulasi fisik. Sebenarnya, robot tidak pintar karena tidak bisa berfikir untuk dirinya sendiri. Namun, bila kita berikan program *artificial intelligence*, maka robot bisa melakukan sesuatu dengan lebih pintar secara karakteristik. Agar menjadi benar-benar pintar, robot membutuhkan penglihatan yang

menyediakan *feedback* agar robot bisa menyesuaikan diri dalam pekerjaannya terhadap berbagai kondisi. Dalam hal ini, visi komputer digunakan agar robot memiliki kemampuan penglihatan.

5.7 Implementasi Visi Komputer pada Mesin OCR

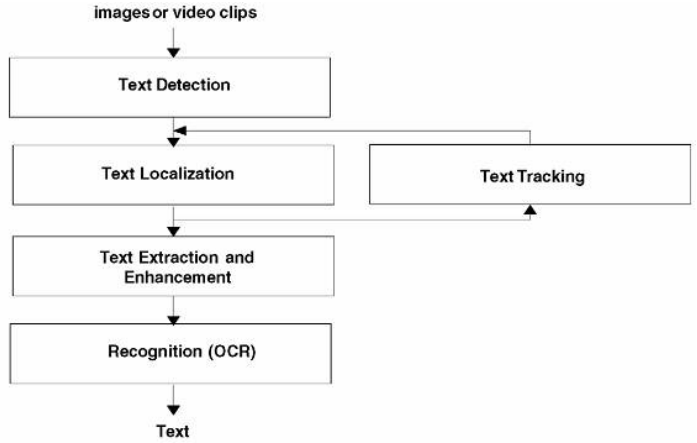
1. Mesin OCR

Mesin OCR *(optical character recognization)* adalah suatu aplikasi yang dapat mengidentifikasi karakter yang digambarkan oleh *user* atau *image* pada bidang *input.* Misalnya *user* menggambar suatu karakter alfabet pada bidang yang ditentukan kemudian OCR akan mendeteksi apakah yang digambarkan tersebut mendekati suatu karakter alfabet tertentu. Mesin OCR pada dasarnya menggunakan sebuah metode *text information extraction* (TIE) pada gambar yang berguna untuk mengenali, mengalisa dan menghasilkan *output* informasi sesuai dengan analisis mesin OCR. Ada 5 proses pada metode TIE yaitu (1) *detection*; (2) *localization*; (3) *tracking*; (4) *extraction and enhancement*; dan (5) *recognition*.

2. Text Information Extraction (TIE) pada Gambar

Masalah TIE harus didefinisikan secara akurat sebelum dilanjutkan ketahap yang lebih jauh. Sistem TIE menerima masukan dalam bentuk satu atau serangkaian gambar. Gambar-gambar tersebut dapat berupa *grayscale* atau berwarna, dikompresi atau tidak dikompresi, dan teks di

dalamnya mungkin bergerak atau tidak bergerak. Masalah TIE dapat dibagi ke dalam sub-masalah berikut: (1) *detection*; (2) *localization*; (3) *tracking*; (4) *extraction and enhancement*; dan (5) *recognition*.



Gambar 5.9 Arsitektur Sistem TIE

Text detection mengacu pada penentuan keberadaan teks didalam *frame* (biasanya pada video atau gambar yang berurutan). *Text localization* adalah proses menentukan lokasi teks pada gambar dan menciptakan semacam kotak pembatas di sekitar teks. *Text tracking* dilakukan untuk mengurangi waktu pemrosesan untuk *text localization* dan untuk memelihara integritas di antara *frame* yang berdekatan. *Text extraction* adalah tahap di mana komponen-komponen teks disegmentasi dari *background*. Pada tahap ini, *enhancement* diperlukan

karena daerah teks biasanya memiliki resolusi yang rendah serta rentan terhadap *noise*. *Text recognition* adalah tahap di mana teks dalam format gambar yang merupakan hasil dari *extraction* diubah menjadi teks murni.

a. Text Detection

Tahap *text detection* mendeteksi keberadaan teks di dalam gambar. Pada pengaplikasian TIE yang berhubungan gambar statis seperti sampul buku, poster, kartu pos, dan sebagainya, mungkin tahap ini tidak diperlukan, karena memang sudah seharusnya gambar-gambar tersebut memiliki teks di dalamnya. Namun dalam pengaplikasian TIE yang berhubungan dengan video, tahap ini sangat diperlukan bagi sistem secara keseluruhan untuk mengurangi kompleksitas waktu, karena tidak seluruh *frame* memiliki teks di dalamnya.

b. Text Localization

Berdasarkan fitur yang dimiliki, metode *text localization* dapat dikategorikan ke dalam dua tipe: *region based method* dan *texture based method*.

1) Region Based Method

Region based method menggunakan sifat dari warna atau *grayscale* pada *text region* atau perbedaannya berdasarkan sifat yang sesuai dengan *background*. Metode ini dapat dibagi lagi menjadi dua sub-

pendekatan: berbasis *connected component* (CC) dan berbasis *edge* (pinggiran).

- *CC Based Method*

  Metode berbasis CC menggunakan pendekatan *button-up* dengan mengelompokkan komponen-komponen kecil ke dalam komponen-komponen yang lebih besar secara berurutan sampai semua *region* di dalam gambar teridentifikasi.

- *Edge Based Method*

  Di antara sifat tekstual beberapa gambar, *edge-based method* fokus pada "kontras yang tinggi antara text dengan *background*". Tepi dari batas teks diidentifikasi dan digabungkan, kemudian beberapa heuristik digunakan untuk menyaring daerah non-teks. Biasanya, *edge* filter digunakan untuk *edge detection* dan sebuah operasi yang mulus atau operator morfologi digunakan untuk tahap penggabungan.

2) Texture-Based Method

Texture-based method menggunakan pengamatan bahwa teks dalam gambar mempunyai perbedaan sifat-sifat *texture* yang membedakan mereka dari *background*. Teknis-teknis yang berdasarkan Gabor filters, Wavelet, FFT, *spatial variance*, dan lain-lain dapat digunakan untuk mengetahui sifat-sifat tekstur dari sebuah *text region* pada gambar.

c. Text Extraction in Compressed Domain

Berdasarkan gagasan bahwa banyak gambar digital biasanya disimpan, diproses, dan ditransmisikan dalam bentuk terkompresi, maka metode TIE (*text information extraction*) *in compressed domain* dapat beroperasi secara langsung pada gambar terkompresi yang berformat MPEG atau JPEG. Metode ini hanya menggunakan sedikit *decoding*, sehingga menghasilkan algoritma yang lebih cepat.

d. Pendekatan Lain

Teknik *Binarization* menggunakan, global, lokal, atau *adaptive thresholding* adalah metode yang paling sederhana untuk *text localization*. Metode ini banyak digunakan untuk segmentasi dokumen gambar, gambar-gambar ini biasanya termasuk karakter hitam dalam *background* putih, sehingga menghasilkan segmentasi *thresholding* yang sukses.

e. Text Tracking

Text tracking digunakan untuk penentuan teks dalam sebuah gambar. *Text tracking* biasanya saling berkaitan dengan *text localization,* dalam *text tracking* banyak digunakan beberapa pendekatan seperti algoritma blok *matching* (sebuah gambar setelah diubah menjadi berwarna *gray* dalam proses *localization* barulah ditentukan blok atau *pixel* dalam gambar yang memiliki kemiripan dengan blok-blok yang membentuk sebuah

teks), *motion vector, text contour,* dan *multi resolution matching*.

f. Text Extraction and Enhancement

Setelah teks dilokalisasi (*localization)*, tahap berlanjut ke tahap *text segmentation,* tahap ini merupakan pemisahan *pixel* suatu teks dari *pixel* suatu *background*. *Output* dari tahap ini adalah *binary image* dengan karakter teks berwarna hitam dan *background* berwarna putih. Pada tahap ini terdapat pengekstraksian dari *text region* sebenarnya dengan membagi *pixel* yang memiliki sifat serupa menjadi kontur atau segmen dan menghilangkan bagian redudansi dari *frame*. Pada *text region*, seringkali terdapat gangguan dari *background*. Intensitas *background* serupa dengan intensitas teks sehingga menyebabkan *binary image* dari *text region* tidak sesuai untuk *recognition* secara langsung. Hal ini dapat diatasi dengan proses *enhancement* yaitu proses untuk memperhalus batasan-batasan antara hasil segmentasi untuk karakter teks yang berwarna hitam dan *background* yang berwarna putih, sehingga didapat sebuah *text extraction* berupa segmentasi *binary image* yang sesuai antara karakter teks dan *background*.

g. Recognition

Pada tahap ini terdapat pengenalan sebenarnya dari karakter yang diekstrak dengan mengkombinasikan berbagai fitur yang diekstraksi pada tahap sebelumnya untuk memberikan teks sebenarnya dengan bantuan *neural network*. Pada tahap ini, *output* dari tahap *segmentation* dipertimbangkan dan karakter yang berada di dalam gambar dibandingkan dengan *neural network training set* standar dan tergantung pada nilai karakter yang muncul pada gambar, karakter yang merepresentasikan nilai *training set* terdekat ditampilkan sebagai karakter yang dikenali. Hasil dari *recognition* adalah rasio antara jumlah karakter yang diekstrak secara tepat dengan total karakter hasil ekstraksi dan mengevaluasi persentase karakter yang diekstraksi secara tepat dari *background*-nya. Untuk setiap hasil ekstraksi dari karakter, apabila garis utama pembentuk karakter tidak terlewat/hilang, maka akan dianggap sebagai karakter yang benar. Hasil ekstraksi langsung dikirim ke mesin OCR.

Bab 6

Fuzzy Logic

6.1 Pengantar

Fuzzy Logic adalah metodologi pemecahan masalah dengan beribu-ribu aplikasi dalam pengendali yang tersimpan dan pemrosesan informasi. *Fuzzy logic* menyediakan cara sederhana untuk menggambarkan kesimpulan pasti dari informasi yang ambigu, samar-samar atau tidak tepat. Sedikit banyak, *fuzzy logic* menyerupai pembuatan keputusan pada manusia dengan kemampuannya untuk bekerja dari data yang ditafsirkan dan mencari solusi yang tepat.

6.2 Sejarah *Fuzzy Logic*

Konsep *fuzzy logic* diperkenalkan oleh Prof. Lotfi Zadeh dari Universitas California di Berkeley pada 1965 dan dipresentasikan bukan sebagai suatu metodologi kontrol, tetapi sebagai suatu cara pemrosesan data dengan memperkenankan penggunaan *partial set membership* dibanding *crisp set membership* atau *non-membership*. Pendekatan pada *set* teori ini tidak diaplikasikan pada sistem kontrol sampai tahun 70-an karena kemampuan komputer yang tidak cukup pada saat itu.

Profesor Zadeh berpikir bahwa orang tidak membutuhkan kepastian, masukan informasi numerik dan belum mampu terhadap kontrol adaptif yang tinggi.

Konsep *fuzzy logic* kemudian berhasil diaplikasikan dalam bidang kontrol oleh E.H. Mamdani. Sejak saat itu aplikasi *fuzzy* berkembang kian pesat. Di tahun 1980-an negara Jepang dan negara-negara di Eropa secara agresif membangun produk nyata sehubungan dengan konsep logika *fuzzy* yang diintegrasikan dalam produk-produk kebutuhan rumah tangga seperti *vacum cleaner*, *microwave oven* dan kamera video. Sementara pengusaha di Amerika Serikat tidak secepat itu mencakup teknologi ini. *Fuzzy logic* berkembang pesat selama beberapa tahun terakhir. Terdapat lebih dari dua ribu produk di pasaran yang menggunakan konsep *fuzzy logic*, mulai dari mesin cuci hingga kereta berkecepatan tinggi. Setiap aplikasi tentunya menyadari beberapa keuntungan dari *fuzzy logic* seperti performa, kesederhaan, biaya rendah dan produktifitasnya.

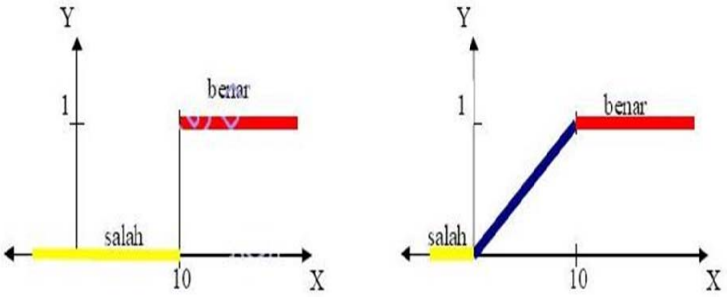
6.3 Definisi Logika *Fuzzy*

Logika *fuzzy* merupakan sebuah logika yang memiliki nilai kekaburan atau kesamaran (*fuzzyness*) antara benar dan salah secara bersamaan namun berapa besar kebenaran dan kesalahan suatu nilai tergantung kepada bobot keanggotaan yang dimilikinya. Adapun perbedaan logika fuzzy dengan logika tegas adala:

- logika tegas memiliki nilai

tidak = 0.0, dan ya = 1.0

- *fuzzy* memiliki nilai antara 0,0 hingga 1,0



Gambar 6.1 Nilai Tegas dan *Fuzzy*

Fuzzy logic menawarkan beberapa karakteristik unik yang menjadikannya suatu pilihan yang baik untuk banyak masalah kontrol. Karakteristik tersebut antara lain:

1. Sudah menjadi sifatnya yang kuat selama tidak membutuhkan ketepatan, *input* yang bebas derau dan dapat diprogram untuk gagal dengan aman jika sensor arus balik dimatikan atau rusak. *Control output* adalah fungsi kontrol halus meskipun jarak variasi *input* yang cukup besar.
2. Selama *fuzzy logic controller* memproses aturan-aturan yang dibuat *user* yang memerintah target sistem kontrol, maka dapat dimodifikasi dengan mudah untuk meningkatkan atau mengubah secara drastis performa sistem. Sensor yang baru dapat dengan mudah digabungkan ke dalam sistem secara sederhana dengan menghasilkan aturan memerintah yang sesuai.

3. *Fuzzy logic* tidak terbatas pada sedikit masukan umpan-balik dan satu atau dua *output control*, tidak juga penting untuk menilai atau menghitung parameter rata-rata perubahan dengan tujuan agar diimplementasikan. Sensor data yang menyediakan beberapa indikasi untuk aksi dan reaksi sistem sudah cukup. Hal ini memungkinkan sensor menjadi murah dan tidak tepat sehingga menghemat biaya sistem keseluruhan dan kompleksitas rendah.
4. Karena operasi-operasi yang berbasiskan aturan, jumlah *input* yang masuk akal dapat diproses (1 sampai 8 atau lebih) dan banyak *output* (1 sampai 4 atau lebih) dihasilkan, walaupun pendefinisian *rulebase* secara cepat menjadi rumit jika terlalu banyak *input* dan *output* dipilih untuk implementasi tunggal selama pendefinisian *rules* (aturan), hubungan timbal baliknya juga harus didefinisikan. Akan lebih baik jika memecah sistem ke dalam potongan-potongan yang lebih kecil dan menggunakan *fuzzy logic controller* yang lebih kecil untuk didistribusikan pada sistem, masing-masing dengan tanggung jawab yang lebih terbatas.
5. *Fuzzy logic* dapat mengontrol sistem nonlinear yang akan sulit atau tidak mungkin untuk dimodelkan secara matematis. Hal ini membuka pintu bagi sistem kontrol yang secara normal dianggap tidak mungkin untuk otomatisasi.

Sedangkan karakteristik utama dari *fuzzy logic* yang ditemukan oleh Prof. Lotfi A. Zadeh adalah sebagai berikut:

- Dalam *fuzzy logic*, penalaran tepat dipandang sebagai suatu kasus terbatas dari penalaran kira-kira.
- Dalam *fuzzy logic* segala sesuatunya adalah masalah derajat.
- Sistem logis manapun dapat difuzzifikasi.
- Dalam *fuzzy logic*, pengetahuan diinterpretasikan sebagai koleksi dari *fuzzy* yang dipaksakan pada sekumpulan variabel.
- Kesimpulan dipandang sebagai sebuah proses dari perkembangan pembatas elastis.

Adapun langkah-langkah penggunaan *fuzzy logic* adalah sebagai berikut:

- Definisikan objektif dan kriteria kontrol:

1. Apa yang kita coba kontrol ?
2. Apa yang harus kita lakukan untuk mengontrol sistem ?
3. Respon seperti apa yang kita butuhkan ?
4. Apa mode kegagalan sistem yang mungkin ?

- Tentukan hubungan antara *input* dan *output* serta memilih jumlah minimum variabel *input* pada mesin *fuzzy logic* (secara khusus *error* dan rata-rata perubahan *error*).
- Dengan menggunakan struktur berbasis aturan dari *fuzzy logic*, jabarkan permasalahan kontrol ke dalam aturan IF X AND Y THEN Z yang mendefinisikan respon *output system* yang diinginkan untuk kondisi *input system* yang diberikan. Jumlah dan kompleksitas dari *rules* bergantung

pada jumlah parameter *input* yang diproses dan jumlah variabel *fuzzy* yang bekerjasama dengan tiap-tiap parameter. Jika mungkin, gunakan setidaknya satu variabel dan turunan waktunya. Walaupun mungkin untuk menggunakan sebuah parameter tunggal yang *error* saat itu juga tanpa mengetahui rata-rata perubahannya, hal ini melumpuhkan kemampuan sistem untuk meminimalisasi keterlampauan untuk sebuah tingkat *input*.

- Buat fungsi keanggotaan yang menjelaskan nilai *input* atau *output* yang digunakan di dalam *rules*.
- Buat rutinitas proses awal dan akhir yang penting jika diimplementasikan dalam *software*, sebaliknya program rules ke dalam mesin *hardware fuzzy logic*.
- *Test system*, evaluasi hasil, atur *rules* dan fungsi keanggotaan, dan *retest* sampai hasil yang memuaskan didapat.

6.4 Himpunan *Fuzzy*

Sangat penting sekali bagi kita untuk terlebih dahulu mengetahui apa itu *crisp set* atau yang dikenal juga dengan *conventional set*, sebelum kita mengarah pada bagaimana himpunan *fuzzy* dibuat untuk kekurangan pada *crisp set*.

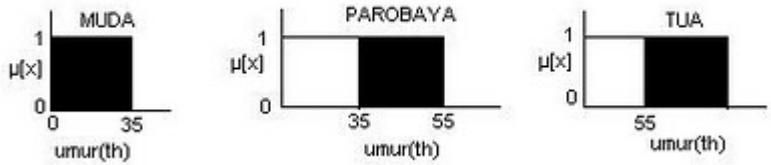
a. Crisp Set

Dalam kebanyakan jenis pemikiran setiap harinya dan refleksi bahasa darinya, orang-orang menggunakan *crisp set* untuk

mengelompokkan sesuatu. Menjadi anggota dari *crisp set* adalah seluruhnya berhubungan atau tidak sama sekali. Seorang wanita dikatakan hamil ataupun tidak, ia tidak pernah "hamil sebagian" atau "sedikit hamil". Berpikir dengan *crisp set* menjadikan segala sesuatunya lebih sederhana, karena sesuatu bisa merupakan anggota dari suatu *crisp set* atau tidak. *Crisp set* dapat digunakan untuk merepresentasikan gambaran pengertian hitam dan putih. Seringkali juga, saat sesuatu itu merupakan anggota dari sebuah *crisp set* maka ia kemudian (pada waktu yang sama) bukan merupakan anggota dari *crisp set* manapun. Kembali hal ini menyederhanakan penggunaan logika dengan proses pemikiran semacam ini. Konstruksi linguistik yang menggambarkan jenis pemikiran ini dapat benar-benar berguna, terutama saat kategori *crisp* digunakan. Pada himpunan tegas (*crisp*), nilai keanggotaan suatu item *x* dalam suatu himpunan A, yang sering ditulis dengan μA[*x*], memiliki 2 kemungkinan, yaitu:

- Satu (1), yang berarti bahwa suatu item menjadi anggota dalam suatu himpunan atau
- Nol (0), yang berarti bahwa suatu item tidak menjadi anggota dalam suatu himpunan.

Untuk lebih jelasnya, bisa dilihat dari contoh berikut:



Gambar 6.2 Nilai Keanggotaan

Dari Gambar 6.2 dapat dijelaskan bahwa:

- Apabila seseorang berusia 34 tahun, maka dikatakan MUDA (μMUDA[34] = 1);
- Apabila seseorang berusia 35 tahun, maka dikatakan TIDAK MUDA (μMUDA[35] = 0);
- Apabila seseorang berusia 35 tahun kurang 1 hari, maka dikatakan TIDAK MUDA (μMUDA[35 – 1hr] = 0);
- Apabila seseorang berusia 35 tahun, maka dikatakan PAROBAYA (μPAROBAYA[35] = 1);
- Apabila seseorang berusia 34 tahun, maka dikatakan TIDAK PAROBAYA (μPAROBAYA[34] = 0);
- Apabila seseorang berusia 55 tahun, maka ia dikatakan PAROBAYA (μPAROBAYA[55] = 1);
- Apabila seseorang berusia 35 tahun kurang 1 hari, maka dikatakan TIDAK PAROBAYA (μPAROBAYA[35 – 1hr] = 0);

Dari sini bisa katakan bahwa pemakaian himpunan *crisp* untuk menyatakan umur sangat tidak adil, adanya perubahan

kecil saja pada suatu nilai mengakibatkan perbedaan kategori yang cukup signifikan. Oleh karena itu digunakanlah himpunan *fuzzy* untuk mengantisipasi hal tersebut.

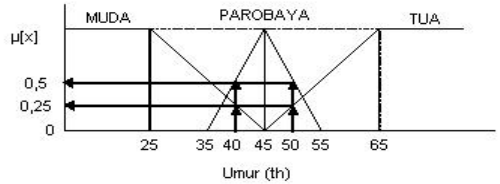
b. Fuzzy Set

Logika *fuzzy* lahir berdasarkan fenomena-fenomena alam yang serba tidak tepat dan samar ditinjau dari cara berpikir manusia, di mana pada kenyataannya tidak ada suatu kondisi atau pernyataan yang tepat 100% benar atau 100% salah. Prof. Lotfi A. Zadeh mengemukakan bahwa *true* atau *false* dalam logika Boolean tidak dapat merepresentasikan pernyataan yang tidak pasti yang berada di antara pernyataan *true* atau *false* tadi, seperti yang sering terjadi dalam dunia nyata. Untuk merepresentasikan nilai ketidakpastian antara *true* atau *false* tersebut, Prof. Lotfi A. Zadeh mengembangkan suatu teori berdasarkan *conventional set* yang disebutnya *fuzzy set* (himpunan *fuzzy*). Sebagai ganti dari pernyataan dengan nilai seluruhnya *true* atau semuanya *false*, logika *fuzzy* memberikan nilai yang spesifik pada setiap nilai di antara pernyataan *true* atau *false* dengan menentukan fungsi keanggotaan (*membership function*) bagi tiap nilai *input* dari proses *fuzzy* (*crisp input*) dan derajat keanggotaan (*degree of membership*) yaitu menyatakan derajat dari *crisp input* sesuai *membership function* antara 0 sampai 1, sehingga memungkinkan bagi suatu persamaan memiliki nilai *true* dan *false* secara bersamaan.

Menurut Prof. Lotfi A Zadeh, *fuzzy set* adalah sebuah kelas dari objek dengan serangkaian kesatuan dari *grades of membership* (nilai keanggotaan). Sebuah *set* dikarakterisasikan oleh sebuah fungsi keanggotaan (karakteristik) yang memberikan tiap objek sebuah nilai keanggotaan yang rentang nilainya antara 0 dan 1. Gagasan pencantuman (*inclusion*), penyatuan (*union*), persimpangan (*intersection*), pelengkap (*complement*), hubungan (*relation*), kecembungan (*convexity*), dan sebagainya diberikan pada *set* tersebut dan berbagai macam sifat dari pemikiran ini dalam konteks dari *fuzzy set* dibangun. Secara khusus, dalil untuk *fuzzy set* cembung dibuktikan tanpa perlu *fuzzy set* terputus. Aturan umum untuk teori *fuzzy set* dituliskan sebagai berikut:

$$f:[0,1]^n \rightarrow [0,1] \qquad (6.1)$$

di mana n merupakan jumlah kemungkinan. Persamaan (6.1) menyatakan bahwa kita dapat mengambil n jumlah *event* yang mungkin dan menggunakan f untuk menghasikan hasil tunggal yang mungkin. Untuk lebih jelasnya mengenai himpunan *fuzzy* dapat dilihat pada contoh persoalan berikut:



Gambar 6.3 Himpunan *Fuzzy*

Dengan adanya himpunan *fuzzy* memungkinkan seseorang untuk dapat masuk ke dalam 2 himpunan yang berbeda, MUDA dan PAROBAYA, PAROBAYA dan TUA, dan sebagainya. Seberapa besar eksistensinya dalam himpunan tersebut dapat dilihat pada nilai keanggotaannya. Dari Gambar 6.3, dapat dilihat bahwa:

- Seseorang yang berumur 40 tahun, termasuk dalam himpunan MUDA dengan μMUDA[40] = 0,25; namun dia juga termasuk dalam himpunan PAROBAYA dengan μPAROBAYA[40] = 0,5.
- Seseorang yang berumur 50 tahun, termasuk dalam himpunan MUDA dengan μMUDA[50] = 0,25; namun dia juga termasuk dalam himpunan PAROBAYA dengan μPAROBAYA[50] = 0,5.

Kalau pada himpunan *crisp*, nilai keanggotaan hanya ada 2 kemungkinan, yaitu 0 atau 1, pada himpunan *fuzzy* nilai keanggotaan terletak pada rentang 0 sampai 1. Apabila x memiliki nilai keanggotaan *fuzzy* $\mu A[x] = 0$ berarti x tidak menjadi anggota himpunan A, demikian pula apabila x memiliki nilai keanggotaan *fuzzy* $\mu A[x] = 1$ berarti x menjadi anggota penuh pada himpunan A.

Terkadang kemiripan antara keanggotaan *fuzzy* dengan probabilitas menimbulkan kerancuan. Keduanya memiliki nilai pada interval [0,1], namun interpretasi nilainya sangat berbeda antara kedua kasus tersebut. Keanggotaan *fuzzy* memberikan

suatu ukuran terhadap pendapat atau keputusan, sedangkan probabilitas mengindikasikan proporsi terhadap keseringan suatu hasil bernilai benar dalam jangka panjang. Misalnya, jika nilai keanggotaan suatu himpunan *fuzzy* MUDA adalah 0,9 maka tidak perlu dipermasalahkan berapa seringnya nilai itu diulang secara individual untuk mengharapkan suatu hasil yang hampir pasti muda. Di lain pihak, nilai probabilitas 0,9 muda berarti 10% dari himpunan tersebut diharapkan tidak muda. Himpunan *fuzzy* memiliki 2 atribut, yaitu:

- Linguistik, penamaan suatu grup yang mewakili suatu keadaan atau kondisi tertentu dengan menggunakan bahasa alami, seperti: MUDA, PAROBAYA, TUA.
- Numerik, suatu nilai (angka) yang menunjukkan ukuran dari suatu variabel seperti: 40, 25, 50, dsb.

c. Fuzzy Set Operation

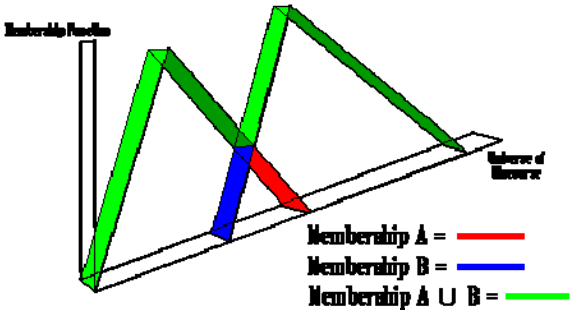
Fuzzy set operation adalah operasi yang dilakukan pada *fuzzy set*. Operasi-operasi ini merupakan generalisasi dari operasi *crisp set*. Terdapat lebih dari satu generalisasi yang mungkin. Operasi-operasi yang paling banyak digunakan secara luas disebut *standard fuzzy set operations*. Terdapat tiga operasi yaitu: *fuzzy unions*, *fuzzy intersections*, dan *fuzzy complements*.

d. Fuzzy Unions

Fungsi keanggotaan dari *union* dari dua *fuzzy set* A dan B dengan fungsi keanggotaan μA dan μB berturut-turut ditetapkan sebagai

maksimum dari dua fungsi keanggotaan tersendiri. Ini disebut standar maksimum.

$$\mu_{A\cup B} = \max(\mu_A, \mu_B) \tag{6.2}$$



Gambar 6.4 Keanggotaan $A \cup B$

Operasi *union* dalam teori *fuzzy set* ekuivalen dengan operasi OR pada aljabar Boolean. Sifat (*property*) dari *fuzzy union* mencakup:

- *Boundary Condition*

  u(a,0) = a
- *Monotonicity*

  b ≤ d secara tidak langsung menyatakan u(a,b) ≤ u(a,d)
- *Commutativity*

  u(a,b) = u(b,a)
- *Associativity*

  u(a,u(b,d)) = u(u(a,b),d)

- *Continuity*

  u adalah fungsi yang berkelanjutan (*continuous*)
- *Superidempotency*

  $u(a,a) > a$
- *Strict monotonicity*

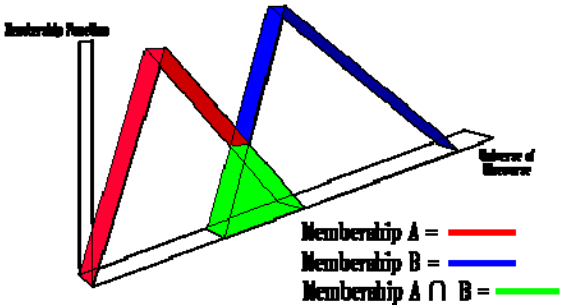
  $a_1 < a_2$ dan $b_1 < b_2$ secara tidak langsung menyatakan bahwa $u(a_1,b_1) < u(a_2,b_2)$

Keterangan: u menyatakan *union* a, b, d, a_1, a_2, b_1, b_2 menyatakan *fuzzy set*.

e. Fuzzy Intersections

Fungsi keanggotaan dari *intersection* dari dua *fuzzy set* A dan B dengan fungsi keanggotaan μA dan μB berturut-turut ditetapkan sebagai minimum dari dua fungsi keanggotaan tersendiri. Ini disebut standar minimum.

$$\mu_{A\cap B} = \min(\mu_A, \mu_B) \tag{6.3}$$



Gambar 6.5 Keanggotaan $A \cap B$

Operasi *intersection* dalam teori *fuzzy set* ekuivalen dengan operasi AND pada aljabar Boolean. Sifat (*property*) dari *fuzzy intersection* mencakup:

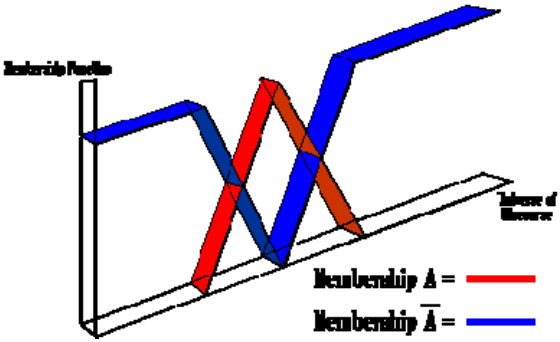
- *Boundary Condition*
  i(a,1) = a
- *Monotonicity*
  b ≤ d secara tidak langsung menyatakan i(a,b) ≤ i(a,d)
- *Commutativity*
  i(a,b) = i(b,a)
- *Associativity*
  i(a,i(b,d)) = i(i(a,b),d)
- *Continuity*
  i adalah fungsi yang berkelanjutan (*continuous*)
- *Subidempotency*
  i (a,a) ≤ a

Keterangan: i menyatakan *intersection*; a, b, d, a_1, a_2, b_1, b_2 menyatakan *fuzzy set*.

f. Fuzzy Complement

Fungsi keanggotaan dari *intersection* dari sebuah *fuzzy set* A dengan fungsi keanggotaan μA ditetapkan sebagai negasi dari fungsi keanggotaan yang ditentukan. Ini disebut standar negasi.

$$\mu_{\bar{A}} = 1 - \mu_A \tag{6.4}$$



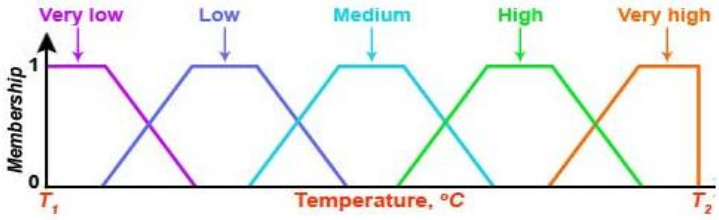
Gambar 6.6 Keanggotaan $\overline{A}$

Operasi *complement* dalam teori *fuzzy set* ekuivalen dengan operasi NOT pada aljabar Boolean. Sifat (*property*) dari *fuzzy complement* mencakup:

- *Boundary Condition*
  c(0) = 1 dan c(1) = 0
- *Monotonicity*
  Untuk semua a,b[0,1], jika a ≤ b maka c(a) ≥ c(b)
- *Continuity*
  c adalah fungsi yang berkelanjutan (*continuous*)
- *Involutions*
  c adalah suatu *involution*, yang berarti bahwa c(c(a)) = a untuk setiap a[0,1]

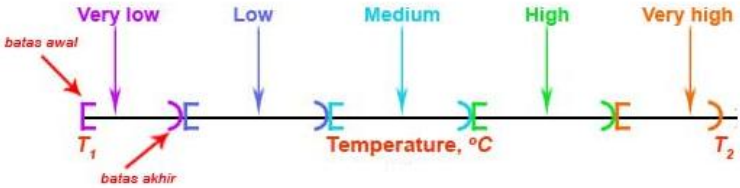
g. Himpunan Klasik vs Himpunan Fuzzy

Perbedaan himpunan *fuzzy* dengan himpunan klasik dapat diilustrasikan pda Gambar 6.7. Dari gambar tersebut dapat terlihat himpunan *fuzzy* memiliki batas yang tidak jelas, sedangkan himpunan klasik memiliki batas yang jelas. Pada gambar tanda ‘)’ mengartikan batas akhir dari sebuah *scope* dan tanda ‘[‘ mengartikan batas awal sebuah *scope* dari himpunan klasik. Rentang suhu yang dinyatakan dalam himpunan *fuzzy*:



Gambar 6.7 Himpunan *Fuzzy* Suhu

Rentang suhu yang dinyatakan dalam Himpunan Klasik:



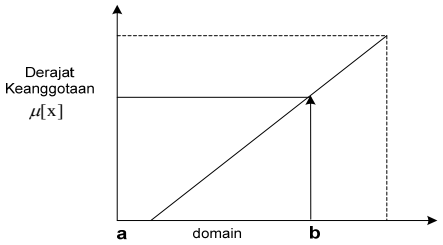
Gambar 6.8 Himpunan Klasik Suhu

6.5 Dasar Logika Fuzzy

- Fungsi keanggotaan (*membership function*) adalah suatu kurva yang menunjukkan pemetaan titik-titik *input* data ke dalam nilai keanggotaannya (sering juga disebut dengan derajat keanggotaan) yang memiliki interval antara 0 sampai 1 atau dapat dinyatakan dengan notasi $0 \leq \mu \leq 1$. Fungsi keanggotaan dapat dinyatakan dengan dua cara yaitu secara numerik dan fungsi.
- Domain adalah keseluruhan nilai yang diijinkan dalam semesta pembicaraan dan boleh dioperasikan dalam himpunan *fuzzy*.
- Ada beberapa fungsi yang dapat digunakan dalam merepresentasikan fungsi keanggotaan, yaitu:

1. Representasi Linear, ada 2 kemungkinan himpunan *fuzzy linear* yaitu:

a. Representasi Linear Naik, kenaikan himpunan dimulai pada nilai domain yang memiliki derajat keanggotaan nol (0) bergerak ke kanan menuju nilai domain yang memiliki derajat keanggotaan lebih tinggi.

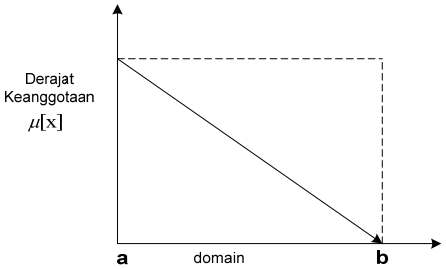


Gambar 6.9 Fungsi Keanggotaan Linear Naik

Fungsi Keanggotaan:

$$\mu(x)=\begin{cases}0; & x \le a \\ (x-a)/(b-a); & a \le x \le b \\ 1; & x \ge b\end{cases} \quad (6.5)$$

b. Representasi Linear Turun, Garis lurus dimulai dari nilai domain dengan derajat keanggotaan tertinggi pada sisi kiri, kemudian bergerak menurun ke nilai domain yang memiliki derajat keanggotaan lebih rendah.

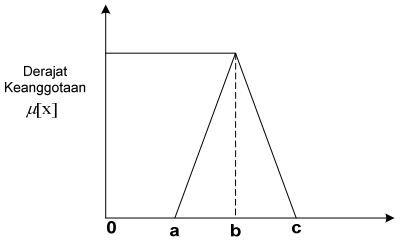


Gambar 6.10 Fungsi Keanggotaan Linear Turun

Fungsi Keanggotaan:

$$\mu(x)=\begin{cases}(b-x)/(b-a) & ; a \le x \le b \\ 0 & ; x \ge b\end{cases} \quad (6.6)$$

2. Representasi Kurva Segitiga

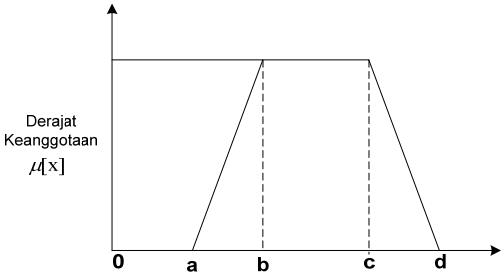


Gambar 6.11 Fungsi Keanggotaan Kurva Segitiga

Fungsi Keanggotaan:

$$\mu(x)=\begin{cases}0 & ; x \le a \text{ atau } x \ge c \\ (x-a)/(b-a) & ; a \le x \le b \\ (c-x)/(c-b) & ; b \le x \le c\end{cases} \quad (6.7)$$

3. Representasi Kurva Trapesium

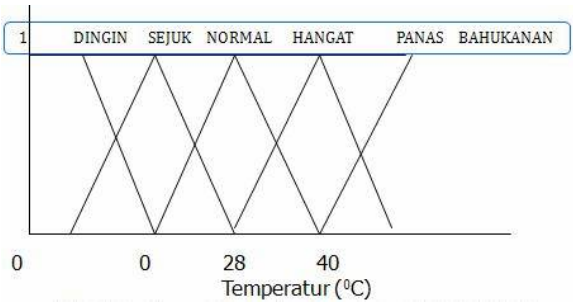


Gambar 6.12 Fungsi Keanggotaan Kurva Trapesium

Fungsi Keanggotaan:

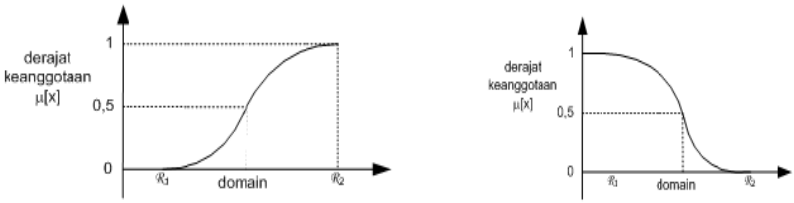
$$\mu(x)=\begin{cases}0 & ; x \le a \text{ atau } x \ge d \\ (x-a)/(b-a) & ; a \le x \le b \\ 1 & ; b \le x \le c \\ (d-a)/(d-c) & ; c \le x \le d\end{cases} \quad (6.8)$$

4. Representasi Kurva Bentuk Bahu



Gambar 6.13 Fungsi Keanggotaan Kurva Bentuk Bahu

5. Representasi Kurva – S



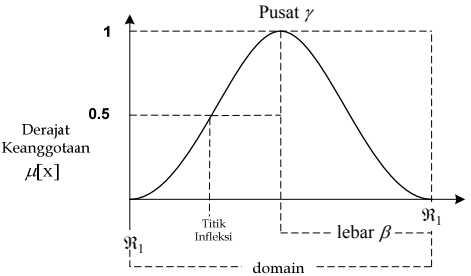
Gambar 6.14 Fungsi Keanggotaan Kurva S

Fungsi Keanggotaan:

$$S(x;\alpha,\beta,\gamma)=\begin{cases}0; & x\le\alpha\\ 2((x-\alpha)/(\gamma-\alpha))^2; & \alpha\le x\le\beta\\ 1-2((\gamma-x)/(\gamma-\alpha))^2; & \beta\le x\le\gamma\\ 1; & x\ge\gamma\end{cases} \qquad (6.9)$$

$$S(x;\alpha,\beta,\gamma)=\begin{cases}1; & x\le\alpha\\ 1-2((x-\alpha)/(\gamma-\alpha))^2 & \alpha\le x\le\beta\\ 2((\gamma-x)/(\gamma-\alpha))^2 & \beta\le x\le\gamma\\ 0; & x\ge\gamma\end{cases} \qquad (6.10)$$

6. Representasi Kurva Bentuk Lonceng



Gambar 6.15 Fungsi Keanggotaan Kurva Bentuk Lonceng

Fungsi Keanggotaan:

$$\mu(x) = \begin{cases} S\left(x; \gamma - \beta, \gamma - \frac{\beta}{2}, \gamma\right); & x \le \gamma \\ 1 - S\left(x; \gamma, \gamma + \frac{\beta}{2}, \gamma + \beta\right); & x > \gamma \end{cases} \quad (6.11)$$

- **Aritmatika Logika Fuzzy**
  - Gabungan (*union*), dalam sistem logika *fuzzy* dikenal dengan istilah *Max*, operasi kesamaan dinyatakan dengan persamaan:

    $\mu A \cup B = \max(\mu A[x], \mu B[y])$ (6.12)
  - Irisan (*intersection*), dalam sistem logika *fuzzy* dikenal dengan istilah *Min*, operasi kesamaan dinyatakan dengan persamaan:

    $\mu A \cap B = \min(\mu A[x], \mu B[y])$ (6.13)
  - Kesamaan (*equality*), operasi kesamaan dinyatakan dengan persamaan:

    $\mu A(x) = \mu B(x); x \in U$ (6.14)
  - Produk (Product), operasi produk dinyatakan dengan persamaan:

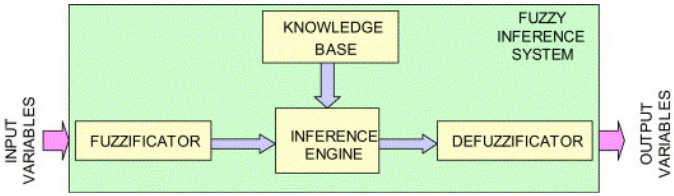
    $\mu A'(x) = 1 - \mu A(x); x \in U$ (6.15)
  - Komplemen (*complement*), operasi komplemen dinyatakan dengan persamaan:

    $\mu(A.B)(x) = \mu A(x).\mu B(x); x \in U$ (6.16)

6.6 Proses Sistem Kontrol Logika Fuzzy

Dalam sistem kontrol logika *fuzzy* (Gambar 6.16) terdapat beberapa tahapan operasional yang meliputi:

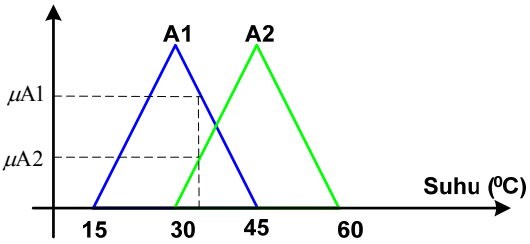
1. Fuzzifikasi
2. Aturan Dasar (*rule base*)
3. Penalaran (*inference machine*)
4. Defuzzifikasi



Gambar 6.16 Diagram Kontrol Logika *Fuzzy*

a. Fuzzifikasi

Fuzzifikasi adalah suatu proses pengubahan dari *input* (nilai tegas) yang ada menjadi nilai fungsi keanggotaan. Misal: merujuk pada Gambar 6.17 berikut, fuzzifikasi dari suhu 35^0C.



Gambar 6.17 Diagram Kontrol Logika *Fuzzy*

Pada Persamaan 6.17 contoh perhitungan fuzzifikasi dapat ditunjukkan sebagai berikut:

$$\mu A_1 = \frac{c-x}{c-b} = \frac{45-35}{45-30} = \frac{2}{3}$$
$$\mu A_2 = \frac{x-a}{b-a} = \frac{35-30}{45-30} = \frac{1}{3} \quad (6.17)$$

b. Konstruksi Dasar Aturan Fuzzy (*Fuzzy Rule Base*)

Dasar aturan *fuzzy* terdiri atas satu *set* aturan *if-then fuzzy*, di mana dasar aturan ini merupakan inti dari sistem *fuzzy* yang merupakan sensor terhadap semua komponen lain yang digunakan untuk menerapkan suatu aturan yang memiliki alasan mendasar dan efisien. Hal penting yang harus dipertimbangkan dalam membangun sebuah dasar aturan *fuzzy* untuk sistem *fuzzy* maupun kontrol *fuzzy*, yaitu:

1. Variabel *input* dan *output*, pemilihan variabel *input* dan *output* yang tepat sangat menentukan kinerja dari model sistem yang akan dibuat.
2. Jangkauan nilai linguistik, nilai linguistik berhubungan dengan fungsi keanggotaan dari data. Untuk menghasilkan nilai batas lingustik yang tepat, maka dibutuhkan proses pengaturan sehingga dapat dihasilkan kinerja model sistem yang baik.
3. Penurunan aturan *fuzzy,* beberapa faktor yang sangat membantu ketika menurunkan suatu dasar aturan *fuzzy*, yaitu:

- Pengalaman atau pengetahuan
- Model *fuzzy*
- Model matematika

Interpretasi Aturan Implikasi *Fuzzy*:

- Fungsi implikasi adalah fungsi yang menyatakan hubungan akibat dari penerapan aturan *if-then* (jika-maka).
- Fungsi implikasi dalam matematika biasanya dinotasikan dengan (jika *p* maka *q*).

Metode untuk menentukan fungsi implikasi dalam sistem dan kontrol *fuzzy*:

1. Implikasi Dienes-Rescher
2. Implikasi Lukasiewicz
3. Implikasi Zadeh
4. Implikasi Gödel
5. Implikasi Mamdani

c. Penalaran (Inference Machine/Engine)

Fuzzy inference engine adalah prinsip logika *fuzzy* yang digunakan untuk mengkombinasikan aturan *if-then fuzzy* dalam suatu dasar aturan *fuzzy* pada suatu pemetaan *fuzzy set* pada himpunan *input* dengan *fuzzy* set pada himpunan *output*. Secara garis besar berdasarkan kombinasi yang digunakan, maka ada dua metode *inference engine*, yaitu:

1. *Inference engine* yang menggunakan kombinasi gabungan, berdasarkan dasar aturan individu *inference*

dengan menggunakan kombinasi gabungan, maka terdapat dua metode *inference engine* yaitu:

- *Product Inference Engine*
- *Minimum Inference Engine*

2. *Inference engine* yang menggunakan kombinasi irisan. Berdasarkan dasar aturan individu dengan menggunakan kombinasi irisan maka terdapat tiga metode *inference engine*, yaitu:
   - Lukasiewicz *Inference Engine*
   - Zadeh *Inference Engine*
   - Dienes-Rescher *Inference Engine*

d. Defuzzifikasi

Defuzzy adalah suatu mekanisme untuk mengkonversi atau mengubah hasil *output fuzzy* menjadi suatu *output nonfuzzy* yang tegas. *Output* dalam bentuk *fuzzy* belum dapat hasil yang dapat langsung digunakan. Dalam teori logika *fuzzy*, terdapat beberapa metode *defuzzy* yang digunakan, yaitu:

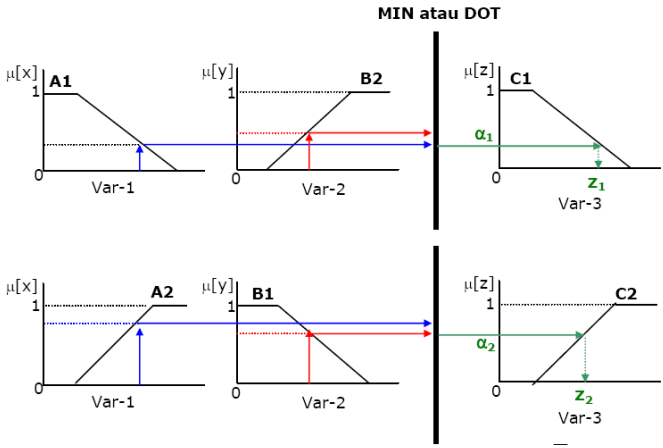
1. *Center of Gravity*
2. *Center of Average*
3. *Bisector*
4. MOM (*Mean of Maximum*)
5. LOM (*Largest of Maximum*)
6. SOM (*Smallest of Maximum*)

6.7 *Fuzzy Inference System* (FIS)

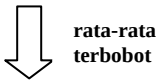
FIS merupakan salah satu metode logika *fuzzy* yang dapat digunakan untuk pengambilan suatu keputusan. Metode ini banyak digunakan sebagai sistem kontrol karena kemampuan metode ini untuk menggantikan seorang operator dalam mengambil keputusan.

a. Metode Tsukamoto

Pada metode Tsukamoto, setiap konsekuen pada aturan yang berbentuk IF-Then harus direpresentasikan dengan suatu himpunan *fuzzy* dengan fungsi keanggotaan yang monoton. Sebagai hasilnya, *output* hasil inferensi dari tiap-tiap aturan diberikan secara tegas (*crisp*) berdasarkan α-predikat (*fire strength*). Hasil akhirnya diperoleh dengan menggunakan rata-rata terbobot.



Gambar 6.18 Inferensi Menggunakan Metode Tsukamoto



$$z = \frac{\alpha_1 z_1 + \alpha_2 z_2}{\alpha_1 + \alpha_2} \qquad \textbf{(}6.18\text{)}$$

b. Metode Mamdani

Metode Mamdani sering juga dikenal dengan nama Metode Max-Min. Metode ini diperkenalkan oleh Ebrahim Mamdani pada tahun 1975. Untuk mendapatkan *output*, diperlukan 4 tahapan:

1. Pembentukan himpunan *fuzzy*
2. Aplikasi fungsi implikasi (aturan)
3. Komposisi aturan
4. Penegasan (*deffuzy*)

1. Pembentukan himpunan *fuzzy*

   Pada metode Mamdani, baik variabel *input* maupun variabel *output* dibagi menjadi satu atau lebih himpunan *fuzzy*.
2. Aplikasi fungsi implikasi

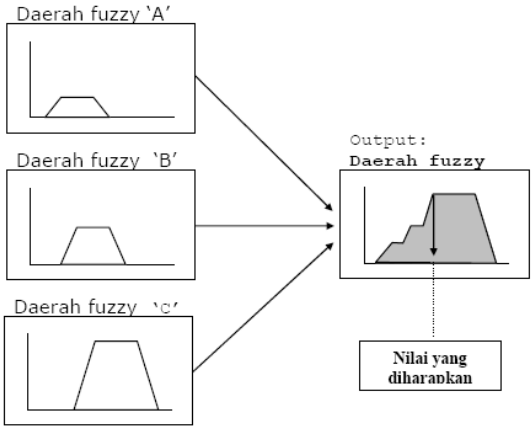
   Pada metode Mamdani, fungsi implikasi yang digunakan adalah Min.
3. Komposisi Aturan

   Tidak seperti penalaran monoton, apabila sistem terdiri atas beberapa aturan, maka inferensi diperoleh dari kumpulan dan korelasi antar aturan. Ada 3 metode yang

digunakan dalam melakukan inferensi sistem *fuzzy*, yaitu: *max*, *additive* dan probabilistik OR (probor).

4. Penegasan (*defuzzy*)

   Input dari proses defuzzifikasi adalah suatu himpunan *fuzzy* yang diperoleh dari komposisi aturan-aturan *fuzzy*, sedangkan *output* yang dihasilkan merupakan suatu bilangan pada domain himpunan *fuzzy* tersebut, sehingga jika diberikan suatu himpunan *fuzzy* dalam *range* tertentu, maka harus dapat diambil suatu nilai *crisp* tertentu sebagai *output* seperti terlihat pada Gambar 6.19.



Gambar 6.19 Proses Defuzzifikasi

Ada beberapa metode defuzzifikasi pada komposisi aturan Mamdani, antara lain:

a. Metode *Centroid* (*Composite Moment*)

Pada metode ini, solusi *crisp* diperoleh dengan cara mengambil titik pusat (z*) daerah *fuzzy*. Secara umum dirumuskan:

$$Z^* = \frac{\int_Z Z\mu(Z)dz}{\int_z \mu(z)dz}$$

$$Z^* = \frac{\sum_{j=1}^{n} z_j \mu(z_j)}{\sum_{j=1}^{n} \mu(z_j)} \qquad (6.19)$$

b. Metode Bisektor

Pada metode ini, solusi *crisp* diperoleh dengan cara mengambil nilai pada domain *fuzzy* yang memiliki nilai keanggotaan separo dari jumlah total nilai keanggotaan pada daerah *fuzzy*. Secara umum dituliskan:

$$z_p \text{ sedemikian hingga } \int_{R1}^{p} \mu(z)dz = \int_{p}^{Rn} \mu(z)dz \qquad (6.20)$$

c. Metode Mean of Maximum (MOM)

Pada metode ini, solusi *crisp* diperoleh dengan cara mengambil nilai rata-rata domain yang memiliki nilai keanggotaan maksimum.

d. Metode Largest of Maximum (LOM)

   Pada metode ini, solusi *crisp* diperoleh dengan cara mengambil nilai terbesar dari domain yang memiliki nilai keanggotaan maksimum.

e. Metode Smallest of Maximum (SOM)

   Pada metode ini, solusi *crisp* diperoleh dengan cara mengambil nilai terkecil dari domain yang memiliki nilai keanggotaan maksimum.

b. Metode Sugeno

Penalaran dengan metode Sugeno hampir sama dengan penalaran Mamdani, hanya saja *output* (konsekuen) sistem tidak berupa himpunan *fuzzy*, melainkan berupa konstanta atau persamaan linear. Metode ini diperkenalkan oleh Takagi-Sugeno Kang pada tahun 1985.

- Model Fuzzy Sugeno Orde-Nol

  Secara umum bentuk model *fuzzy* Sugeno Orde-Nol adalah:

$$\text{IF } (x_1 \text{ is } A_1) \bullet (x_2 \text{ is } A_2) \bullet (x_3 \text{ is } A_3) \bullet \ldots \bullet (x_n \text{ is } A_n) \text{ THEN z} = \text{k}$$

  dengan A_i adalah himpunan *fuzzy* ke-*i* sebagai anteseden dan k adalah suatu konstanta (tegas) sebagai konsekuen.

- Model Fuzzy Sugeno Orde-Satu

  Secara umum bentuk model *fuzzy* Sugeno Orde-Satu adalah:

$$\text{IF } (x_1 \text{ is } A_1) \bullet \ldots \bullet (x_N \text{ is } A_N) \text{ THEN z} = p_1 * x_1 + \ldots + p_N * x_N + q$$

dengan A_i adalah himpunan *fuzzy* ke-*i* sebagai anteseden, dan pi adalah suatu konstanta (tegas) ke-*i* dan q juga merupakan konstanta dalam konsekuen. Apabila komposisi aturan menggunakan metode Sugeno, maka deffuzifikasi dilakukan dengan cara mencari nilai rata-ratanya.

c. Basis Data Fuzzy

- ❖ Model Tahani
- ❖ Model Umano

6.8 Aplikasi

Beberapa aplikasi logika fuzzy, antara lain:

1. Pada tahun 1990 pertama kali dibuat mesin cuci dengan logika *fuzzy* di Jepang (Matsushita Electric Industrial Company). Sistem *fuzzy* digunakan untuk menentukan putaran yang tepat secara otomatis berdasarkan jenis dan banyaknya kotoran serta jumlah yang akan dicuci. Input yang digunakan adalah: seberapa kotor, jenis kotoran, dan banyaknya yang dicuci. Mesin ini menggunakan sensor optik, mengeluarkan cahaya ke air dan mengukur bagaimana cahaya tersebut sampai ke ujung lainnya. Makin kotor, maka sinar yang sampai makin redup. Di samping itu, sistem juga dapat menentukan jenis kotoran (daki atau minyak).

2. Transmisi otomatis pada mobil. Mobil Nissan telah menggunakan sistem *fuzzy* pada transmisi otomatis, dan mampu menghemat bensin 12 – 17%.
3. Kereta bawah tanah Sendai mengontrol pemberhentian otomatis pada area tertentu.
4. Ilmu kedokteran dan biologi, seperti sistem diagnosis yang didasarkan pada logika *fuzzy*, penelitian kanker, manipulasi peralatan prostetik yang didasarkan pada logika *fuzzy*.
5. Manajemen dan pengambilan keputusan, seperti manajemen basisdata yang didasarkan pada logika *fuzzy*, tata letak pabrik yang didasarkan pada logika *fuzzy*, sistem pembuat keputusan di militer yang didasarkan pada logika *fuzzy*, pembuatan *games* yang didasarkan pada logika *fuzzy*.
6. Ekonomi, seperti pemodelan *fuzzy* pada sistem pemasaran yang kompleks.
7. Klasifikasi dan pencocokan pola.
8. Psikologi, seperti logika *fuzzy* untuk menganalisis kelakuan masyarakat, pencegahan dan investigasi kriminal.
9. Ilmu-ilmu sosial, terutama untuk pemodelan informasi yang tidak pasti.
10. Ilmu lingkungan, seperti kendali kualitas air, prediksi cuaca.

Bab 7

Robotika dan Sistem Sensor

7.1 Pengantar

Kemajuan ilmu pengetahuan dan teknologi dari masa ke masa berkembang cepat terutama di bidang otomasi industri. Perkembangan ini tampak jelas di industri pemabrikan, di mana sebelumnya banyak pekerjaan menggunakan tangan manusia, kemudian beralih menggunakan mesin, berikutnya dengan *electro-mechanic* (semi otomatis) dan sekarang sudah menggunakan robotik.

Hampir semua orang tahu apa itu robot. Dalam sepuluh tahun terakhir ini dunia robotika berkembang pesat dan cukup populer. Hal-hal yang dilakukan manusia kini banyak yang diganti mesin. Robot merupakan suatu sistem yang mampu bekerja otomatis. Hasil penelitian menunjukan secanggih apapun sistem kendali yang dipakai akan sangat tergantung kepada sensor yang digunakan.

Sensor merupakan peralatan atau komponen yang mempunyai peranan penting dalam sebuah sistem pengaturan otomatis. Ketepatan dan kesesuaian dalam memilih sebuah

sensor akan sangat menentukan kinerja dari sistem pengaturan secara otomatis.

7.2 Sensor

Sensor adalah alat untuk mendeteksi atau mengukur sesuatu yang digunakan untuk mengubah variasi mekanis, magnetis, panas, sinar dan kimia menjadi tegangan dan arus listrik. Sensor itu sendiri terdiri atas *transducer* dengan atau tanpa penguat/pengolah sinyal yang terbentuk dalam satu sistem pengindera. Dalam lingkungan sistem pengendali dan robotika, sensor memberikan kesamaan yang menyerupai mata, pendengaran, hidung, lidah yang kemudian akan diolah oleh kontroler sebagai otaknya.

Adapun macam-macam sensor dijelaskan sebagai berikut:

a) Sensor Kedekatan (*proximity*), yaitu sensor atau saklar yang dapat mendeteksi adanya target (jenis logam) dengan tanpa adanya kontak fisik. Sensor jenis ini biasanya tediri atas alat elektronis *solid-state* yang terbungkus rapat untuk melindunginya dari pengaruh getaran, cairan, kimiawi, dan korosif yang berlebihan. Sensor ini dapat diaplikasikan pada kondisi penginderaan pada objek yang dianggap terlalu kecil/lunak untuk menggerakkan suatu mekanis saklar. Prinsip kerjanya adalah dengan memperhatikan

perubahan amplitudo suatu lingkungan medan frekuensi tinggi.

b) Sensor Magnet, juga disebut relai buluh, adalah alat yang akan terpengaruh medan magnet dan akan memberikan perubahan kondisi pada keluaran. Seperti layaknya saklar dua kondisi (*on*/*off*) yang digerakkan oleh adanya medan magnet di sekitarnya. Biasanya sensor ini dikemas dalam bentuk kemasan yang hampa dan bebas dari debu, kelembapan, asap ataupun uap.

c) Sensor Sinar, terdiri atas 3 kategori. *Photovoltaic* atau sel solar adalah alat sensor sinar yang mengubah energi sinar langsung menjadi energi listrik, dengan adanya penyinaran cahaya akan menyebabkan pergerakan elektron dan menghasilkan tegangan. Demikian pula dengan fotokonduktif (fotoresistif) yang akan memberikan perubahan tahanan (resistansi) pada sel-selnya, semakin tinggin intensitas cahaya yang terima, maka akan semakin kecil pula nilai tahanannya. Sedangkan Fotolistrik adalah sensor yang berprinsip kerja berdasarkan pantulan karena perubahan posisi/jarak suatu sumber sinar (inframerah atau laser) ataupun target pemantulnya, yang terdiri atas pasangan sumber cahaya dan penerima.

d) Sensor Efek-Hall, dirancang untuk merasakan adanya objek magnetis dengan perubahan posisinya. Perubahan

medan magnet yang terus menerus menyebabkan timbulnya pulsa yang kemudian dapat ditentukan frekuensinya, sensor jenis ini biasa digunakan sebagai pengukur kecepatan.

e) **Sensor Ultrasonik**, bekerja berdasarkan prinsip pantulan gelombang suara, di mana sensor ini menghasilkan gelombang suara yang kemudian menangkapnya kembali dengan perbedaan waktu sebagai dasar penginderaannya. Perbedaan waktu antara gelombang suara dipancarkan dengan ditangkapnya kembali gelombang suara tersebut adalah berbanding lurus dengan jarak atau tinggi objek yang memantulkannya. Jenis objek yang dapat diindera di antaranya adalah: objek padat, cair, butiran maupun tekstil.

f) **Sensor Tekanan**, sensor ini memiliki *transducer* yang mengukur ketegangan kawat, di mana mengubah tegangan mekanis menjadi sinyal listrik. Dasar penginderaannya pada perubahan tahanan pengantar (*transducer*) yang berubah akibat perubahan panjang dan luas penampangnya.

g) **Sensor Suhu**, ada 4 jenis utama sensor suhu yang biasa digunakan; thermocouple (T/C), resistance temperature detector (RTD), termistor dan IC sensor. *Thermocouple* pada pokoknya terdiri atas sepasang *transducer* panas dan dingin yang disambungkan/dilebur bersama, perbedaan

yang timbul antara sambungan tersebut dengan sambungan referensi yang berfungsi sebagai pembanding. Resistance Temperature Detector (RTD) didasari pada tahanan listrik dari logam yang bervariasi sebanding dengan suhu. Kesebandingan variasi ini adalah presisi dengan tingkat konsisten/kestabilan yang tinggi pada pendeteksian tahanan. Platina adalah bahan yang sering digunakan karena memiliki tahanan suhu, kelinearan, stabilitas dan reproduksibilitas. Termistor adalah resistor yang peka terhadap panas yang biasanya mempunyai koefisien suhu negatif, karena saat suhu meningkat maka tahanan menurun atau sebaliknya.

h) IC Sensor, adalah sensor suhu dengan rangkaian terpadu yang menggunakan *chip* silikon untuk kelemahan penginderanya. Mempunyai konfigurasi *output* tegangan dan arus yang sangat linear.

i) Sensor Kecepatan/RPM, proses pengindraan merupakan proses kebalikan dari suatu motor, di mana suatu poros/objek yang berputar pada suatui generator akan menghasilkan suatu tegangan yang sebanding dengan kecepatan putaran objek. Kecepatan putar sering pula diukur dengan menggunakan sensor yang mengindra pulsa magnetis (induksi) yang timbul saat medan magnetis terjadi.

j) **Sensor Penyandi** (**Encoder**), digunakan untuk mengubah gerakan linear atau putaran menjadi sinyal digital, di mana sensor putaran memonitor gerakan putar dari suatu alat. Sensor ini biasanya terdiri atas 2 lapis jenis penyandi, yaitu: (1) penyandi rotari tambahan (yang mentransmisikan jumlah tertentu dari pulsa untuk masing-masing putaran) yang akan membangkitkan gelombang kotak pada objek yang diputar. (2) Penyandi absolut (yang memperlengkapi kode biner tertentu untuk masing-masing posisi sudut) mempunyai cara kerja sang sama dengan perkecualian, lebih banyak atau lebih rapat pulsa gelombang kotak yang dihasilkan sehingga membentuk suatu pengkodean dalam susunan tertentu.

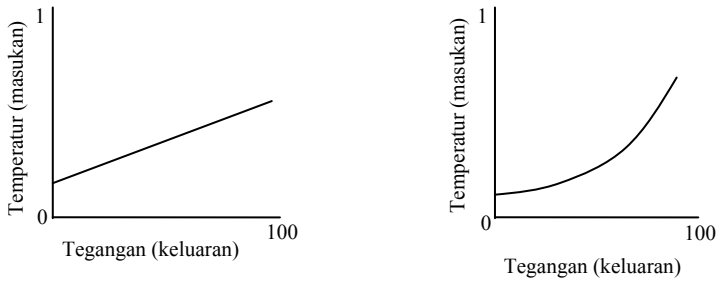
7.2.1 Persyaratan Umum Sensor

Dalam memilih peralatan sensor yang tepat dan sesuai dengan sistem yang akan disensor maka perlu diperhatikan persyaratan umum sensor berikut ini:

a) Linearitas

Ada banyak sensor yang menghasilkan sinyal keluaran yang berubah secara kontinu sebagai tanggapan terhadap masukan yang berubah secara kontinu. Sebagai contoh, sebuah sensor panas dapat menghasilkan tegangan sesuai dengan panas yang dirasakannya. Dalam kasus seperti ini,

biasanya dapat diketahui secara tepat bagaimana perubahan keluaran dibandingkan dengan masukannya berupa sebuah grafik. Gambar 7.1 memperlihatkan hubungan dari dua buah sensor panas yang berbeda. Garis lurus pada Gambar 7.1(a). memperlihatkan tanggapan linear, sedangkan pada Gambar 7.1(b). adalah tanggapan non-linear.



(a) Tanggapan linear

(b) Tanggapan non linear

Gambar 7.1 Keluaran dari *Transducer* Panas

b) Sensitivitas

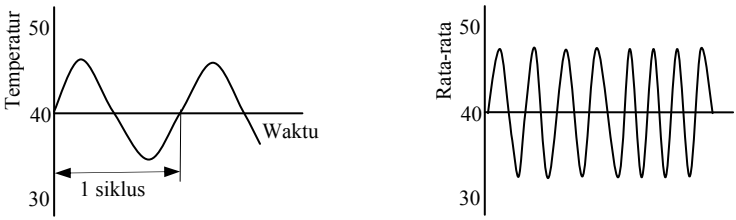
Sensitivitas akan menunjukan seberapa jauh kepekaan sensor terhadap kuantitas yang diukur. Sensitivitas sering juga dinyatakan dengan bilangan yang menunjukan "perubahan keluaran dibandingkan unit perubahan masukan". Beberapa sensor panas dapat memiliki kepekaan yang dinyatakan dengan "satu volt per derajat", yang berarti *perubahan* satu derajat pada masukan akan

menghasilkan *perubahan* satu volt pada keluarannya. Sensor panas lainnya dapat saja memiliki kepekaan "dua volt per derajat", yang berarti memiliki kepakaan dua kali dari sensor yang pertama. Linearitas sensor juga mempengaruhi sensitivitas dari sensor. Apabila tanggapannya linear, maka sensitivitasnya juga akan sama untuk jangkauan pengukuran keseluruhan. Sensor dengan tanggapan paga Gambar 7.1(b) akan lebih peka pada temperatur yang tinggi dari pada temperatur yang rendah.

c) Tanggapan Waktu

Tanggapan waktu pada sensor menunjukkan seberapa cepat tanggapannya terhadap perubahan masukan. Sebagai contoh, instrumen dengan tanggapan frekuensi yang jelek adalah sebuah termometer merkuri. Masukannya adalah temperatur dan keluarannya adalah posisi merkuri. Misalkan perubahan temperatur terjadi sedikit demi sedikit dan kontinyu terhadap waktu, seperti tampak pada Gambar 7.2(a). Frekuensi adalah jumlah siklus dalam satu detik dan diberikan dalam satuan hertz (Hz) [1 hertz berarti 1 siklus per detik, 1 kilohertz berarti 1000 siklus per detik]. Pada frekuensi rendah, yaitu pada saat temperatur berubah secara lambat, termometer akan mengikuti perubahan tersebut dengan "setia". Tetapi apabila perubahan temperatur sangat cepat lihat Gambar 7.2(b) maka tidak

diharapkan akan melihat perubahan besar pada termometer merkuri, karena bersifat lamban dan hanya akan menunjukan temperatur rata-rata.



(a) Perubahan lambat (b) Perubahan cepat

Gambar 7.2 Temperatur Berubah secara Kontinu

Ada bermacam cara untuk menyatakan tanggapan frekuensi sebuah sensor. Misalnya "satu milivolt pada 500 hertz". Tanggapan frekuensi dapat pula dinyatakan dengan "*decibel* (db)", yaitu untuk membandingkan daya keluaran pada frekuensi tertentu dengan daya keluaran pada frekuensi referensi.

7.2.2 Jenis Sensor

Perkembangan sensor sangat cepat sesuai kemajuan teknologi otomasi, semakin komplek suatu sistem otomasi dibangun maka semakin banyak jenis sensor yang digunakan. Robotik adalah sebagai contoh penerapan sistem otomasi yang

kompleks, di sini sensor yang digunakan dapat dikatagorikan menjadi dua jenis sensor yaitu:

a. *Internal sensor*, yaitu sensor yang dipasang di dalam bodi robot.

   Sensor internal diperlukan untuk mengamati posisi, kecepatan, dan akselerasi berbagai sambungan mekanik pada robot, dan merupakan bagian dari mekanisme servo.

b. *External sensor*, yaitu sensor yang dipasang di luar bodi robot.

   Sensor eksternal diperlukan karena dua macam alasan yaitu untuk keamanan dan untuk penuntun.

Sesuai dengan fungsi sensor sebagai pendeteksi sinyal dan m eng-informasikan sinyal tersebut ke sistem berikutnya, maka peranan dan fungsi sensor akan dilanjutkan oleh *transducer*. Karena keterkaitan antara sensor dan *transducer* begitu erat maka pemilihan *transducer* yang tepat dan sesuai juga perlu diperhatikan.

7.2.3 Klasifikasi Sensor

Secara umum berdasarkan fungsi dan penggunaannya sensor dapat dikelompokkan menjadi 3 bagian yaitu:

a. Sensor termal (panas), adalah sensor yang digunakan untuk mendeteksi gejala perubahan panas/temperatur/suhu pada

suatu dimensi benda atau dimensi ruang tertentu. Contohnya: bimetal, termistor, termokopel, RTD, *photo transistor, photo dioda, photo multiplier, photovoltaik, infrared pyrometer, hygrometer*.

b. Sensor mekanis, adalah sensor yang mendeteksi perubahan gerak mekanis, seperti perpindahan atau pergeseran atau posisi, gerak lurus dan melingkar, tekanan, aliran, level dan sebagainya. Contoh: *strain gage*, *linear variable deferential transformer* (LVDT), *proximity*, *potensiometer, load cell, bourdon tube*.
    - Sensor Posisi
    - Sensor Kecepatan (*motion sensor*)
    - Sensor Tekanan (*pressure sensor*)
    - Sensor Aliran Fluida (*flow sensor*)
    - Sensor Level

c. Sensor optik (cahaya), adalah sensor yang mendeteksi perubahan cahaya dari sumber cahaya, pantulan cahaya ataupun bias cahaya yang mengernai benda atau ruangan. Contoh: *photo cell, photo transistor, photo diode, photo voltaic, photo multiplier, pyrometer optic*.

7.3 Robot

Pertama kali kata “ROBOT” digunakan di New York pada Oktober 1922 pada sebuah pentas teater yang berjudul “RVR”,

dinaskahi oleh Karel Caper. Kata *Robot* itu sendiri berasal dari sebuah kata *robota* yang berarti kerja. Tahun 1956, UNIMATION memulai bisnis robot dan baru pada tahun 1972 mendapatkan laba dari usahanya tersebut. Istilah *robot* makin populer setelah ada film *Starwars* dan Robot R2D2 yaitu sekitar tahun 70-an.

Robot adalah motor primitif kapasitas *intelligence* di komputer sains dan *control theory* (*sensory perception*, *decision making*, kemampuan *intellegence*). Robot bahasa cheko "worker"

- *Asimov's low of robotic*:
    1. Tidak boleh melukai manusia.
    2. Harus patuh terhadap perintah manusia, kecuali yang bertentangan dengan no.1.
    3. Harus mempertahankan diri, kecuali bertentangan dengan no.1 dan no.2.

7.3.1 Definisi Robot & Robotik

Kata "robot" diambil dari bahasa Ceko (Chech), yang memiliki arti "pekerja" (*worker*). Robot merupakan suatu perangkat mekanik yang mampu menjalankan tugas-tugas fisik, baik di bawah kendali dan pengawasan manusia ataupun yang dijalankan dengan serangkaian program yang telah didefinisikan terlebih dahulu atau kecerdasan buatan (*artificial intelligence*).

Jika sebelumnya robot hanya dioperasikan di laboratorium ataupun dimanfaatkan untuk kepentingan industri, saat ini robot telah digunakan sebagai alat untuk membantu pekerjaan manusia. Seiring dengan berkembangnya teknologi, khususnya teknologi elektronik, peran robot menjadi semakin penting tidak saja dibidang sains, tapi juga di berbagai bidang lainnya, seperti di bidang kedokteran, pertanian, bahkan militer. Secara sadar atau tidak, saat ini robot telah "masuk" dalam kehidupan manusia sehari-hari dalam berbagai bentuk dan jenis. Ada jenis robot sederhana yang dirancang untuk melakukan kegiatan yang sederhana, mudah dan berulang-ulang ataupun robot yang diciptakan khusus untuk melakukan sesuatu yang rumit, sehingga dapat berperilaku sangat kompleks dan secara otomatis dapat mengontrol dirinya sendiri sampai batas tertentu.

Banyak terdapat tanggapan mengenai konsep robot, di mana robot diandalkan sebagai tiruan manusia. Karena itu dicoba dibuat sebuah definisi untuk menghindari hal-hal yang tidak diinginkan. Definisi yang paling dapat diterima adalah dari "Robot Institute Of America".

> *"Sebuah robot adalah sesuatu yang dapat diprogram dan diprogram ulang, dengan memiliki manipulator mekanik/pengerak yang didesain untuk memindahkan barang-barang, komponen-komponen atau alat-alat khusus dengan berbagai program*

yang fleksibel/mudah disesuaikan untuk melaksanakan berbagai macam tugas"

Dari definisi tersebut dapat dikatakan robot sebagai automasi yang dapat diprogram (*Programmable Automation*). Sedangkan istilah robotik berdasarkan Webster adalah:

"*Teknologi yang behubungan dengan mendesain, membuat, dan mengoperasikan robot*."

Robotik ruang lingkupnya mencakup *artificial intelligence*, ilmu komputer, rekayasa mekanik, psikologi, anatomi, dan bidang ilmu lainnya. Kata Robotik sendiri pertama kali digunakan oleh Issac Asimov pada tahun 1942.

7.3.2 Istilah Robot

Sebagai pemahaman awal maka perlu diketahui beberapa istilah dasar dari robot, sehingga untuk mempelajari bagian selanjutnya kita tidak akan menemukan kesulitan untuk mendefinisikan.

Robot : Peralatan yang dapat diprogram ulang, memiliki banyak fungsi yang didesain untuk memindahkan material, *part*, peralatan atau peralatan khusus

Controller : suatu peralatan yang bertugas sebagai pengendali dari gerakan robot. *Controller*

membentuk sistem kontrol yang akan menentukan *input* dan *output* suatu robot.

Manipulator : lengan yang memberikan gerakan robot untuk memutar, melipat, menjangkau objek. Gerakan ini disebut dengan derajat kebebasan robot atau jumlah sumbu yang ada pada robot. manipulator terdiri atas beberapa segmen dan sambungan (*joint*).

Joint : *joint* atau sambungan merupakan hubungan antara lengan (*arm*) dengan lengan yang lain sehinga dipisahkan oleh sumbu (*axis*).

Open loop : lup terbuka adalah suatu sistem kontrol yang tidak memiliki *feedback* atau umpan balik, sehingga suatu peralatan tidak dapat mengenali kesalahan sebagai pembanding kerja selanjutnya. *Feedback* digunakan pada *close loop* (lup tertutup).

Aplikasi robot terdapat pada:

- Bidang industri: keandalan keefektivitasan biaya.
- Bidang ilmiah: *sensor protection*, *motor control*, *intelligent behavor*.

Robot mempunyai dua katagori

- Robot manual. Dalam robot manual semua kendali masih dikendalikan oleh manusia.
- Robot Cerdas. Dalam robot cerdas semua kendalinya dikendalikan oleh mikrokontroler, yang di mana diberi kercerdasan sebelumnya.

Tipe robot

- Tipe robot pertama membutuhkan tempat yang tetap, robot-robot perangkat industri seperti yang digunakan untuk merakit mobil. Jenis robot ini dioperasikan dalam lingkungan terkendali tinggi yang telah dirancang untuk hal tersebut.
- Tipe kedua terdiri atas berbagai jenis robot otomatis. Robot-robot ini dirancang untuk dioperasikan dalam dunia nyata, seperti robot asimo, robot aibo (robot anjing).

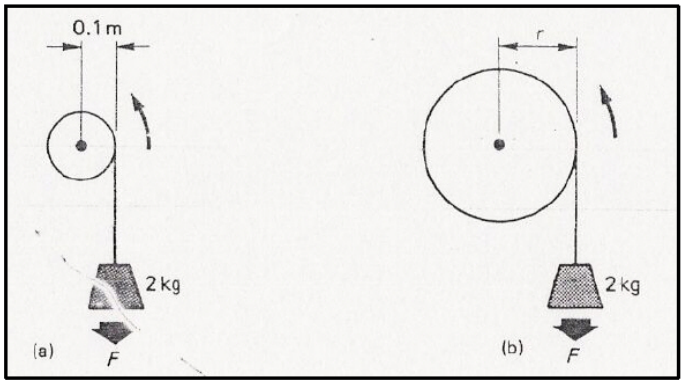
Sistem Penggerak robot

a. Motor Listrik

- Daya: terdapat 2 daya dalam satuan yang sama yaitu watt, daya listrik (=satuan listrik yang digunakan oleh motor) dan daya mekanik (=satuan daya yang dihasilkan oleh motor). Efisiensi motor idealnya 100%, artinya 1 watt daya listrik menghasilkan 1 watt daya mekanik, namun pada prakteknya efisiensi motor kurang dari 50%,

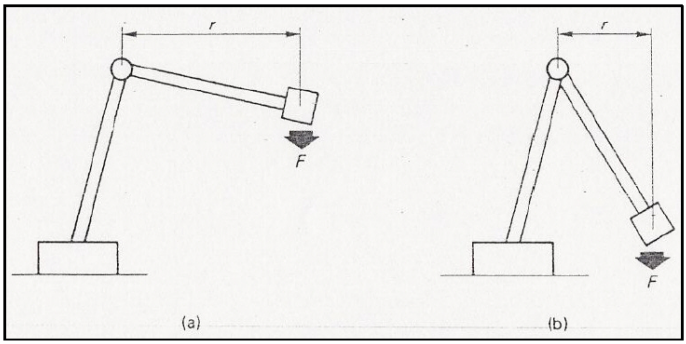
berarti 1 daya listrik hanya menghasilkan ½ atau lebih kecil lagi daya mekanik, sisanya berupa panas dan derau.

- Torsi: kekuatan atau gaya puntir yang dihasilkan satuannya Nm (Newton meter), defInisinya adalah gaya dikalikan dengan jarak terpendek yang diukur dari sumbu rotasi ke garis di sepanjang gaya tersebut bekerja.



Gambar 7.3 Torsi untuk Mengangkat Beban

Pada Gambar 7.3, torsi sebesar 2 Nm diperlukan untuk mengangkat beban dengan menggunakan katrol. Diperlukan torsi yang lebih besar untuk mengangkat beban ini karena jarak yang ditempuh untuk setiap putaran/revolusi katrol juga lebih besar. Jika Gaya F adalah 2kg (20 Newton) dan r adalah 0,1 meter, maka torsi = 20 x 0,1 = 2 Nm.



Gambar 7.4 Torsi dengan Beban Bergerak

Pada Gambar 7.4, torsi semakin berkurang apabila beban bergerak ke arah poros motor.

- Kecepatan: dinyatakan dalam rpm (*revolution per minute*) adalah penghubung antara daya dan torsi. Kita dapat menghitung berapa torsi T (dalam Newton meter) bila mengetahui daya mekanik P (dalam watt) dan jumlah revolusi per menit R dengan formula:

$$T = 10\,P / R \text{ newton meter} \qquad (7.1)$$

b. Jenis Motor Listrik

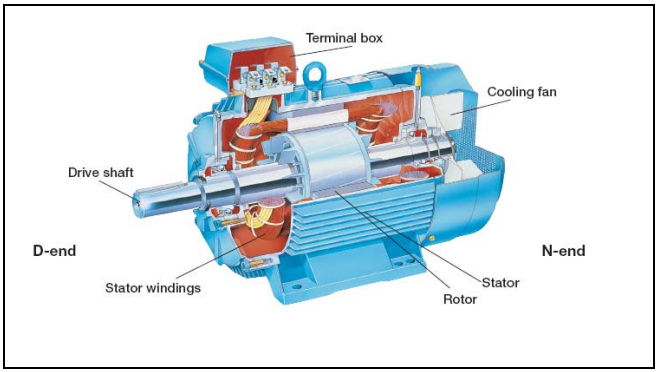
- DC motor: istilah lain yang dikenal dengan dinamo, yang banyak digunakan pada mobil robot. Tipe ini motor bergerak bebas.
- Servo motor: motor DC dengan tambahan kontrol *power*, di mana biasa digunakan pada penggerak mainan seperti mobil, perahu dan pesawat.

- *Stepper Motor*: gerakan motor secara bertahap pada motor ini, sesuai dengan derajatnya.
- Motor AC, dioperasikan oleh arus listrik bolak-balik.

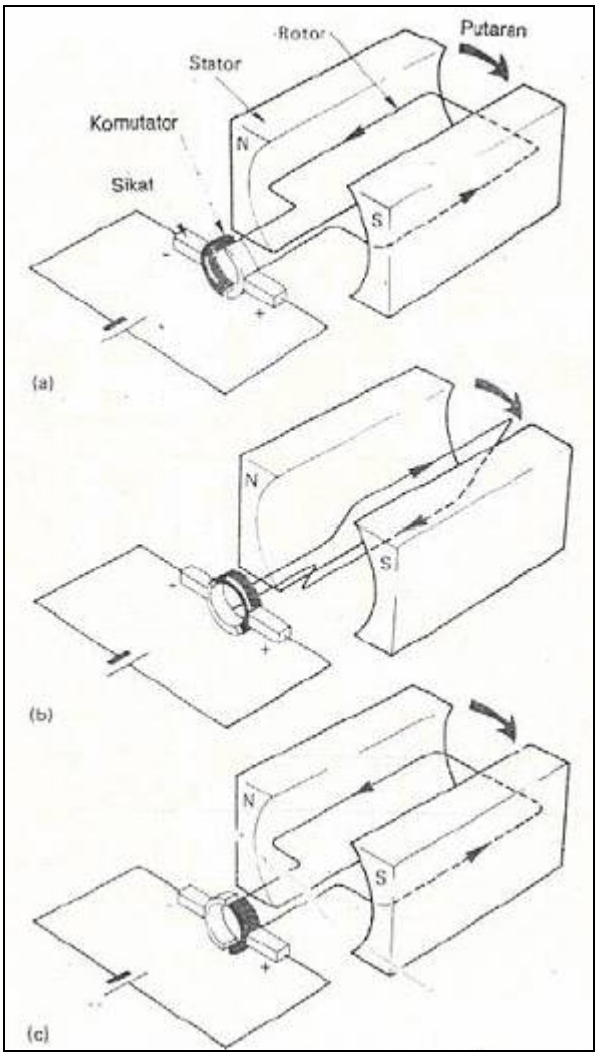
c. Konstruksi Motor

- **Motor DC:**

  Terdapat 2 bagian utama, yaitu bagian yang tetap dan menghasilkan medan magnet dari koilnya yang disebut **stator** dan bagian yang berputar disebut **rotor** atau *armature* berupa koil di mana arus listrik mengalir.

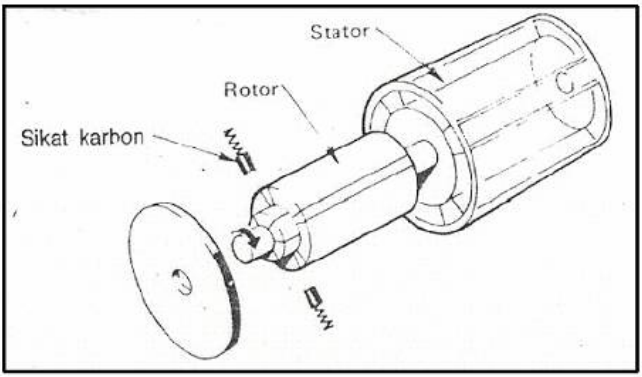


Gambar 7.5 Motor DC

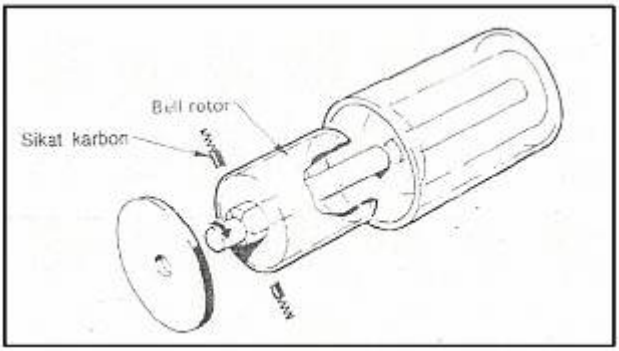


Gambar 7.6 Prinsip Motor DC

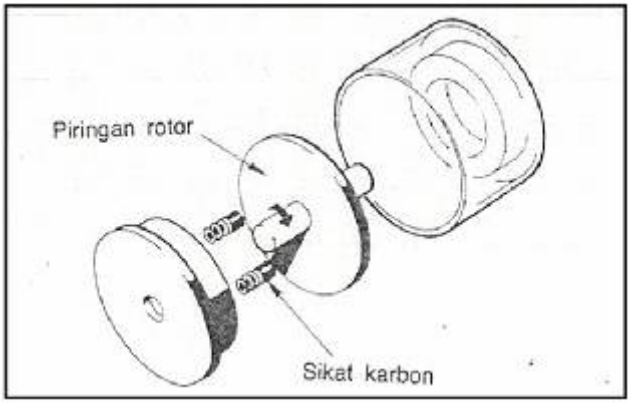
Jenis motor dikenali berdasarkan pengaturan listrik di dalamnya dan kontruksi fisiknya. Tiga cara dasar membentuk motor listrik:



Gambar 7.7 Konstruksi Motor Standar



Gambar 7.8 Konstruksi Motor Bell

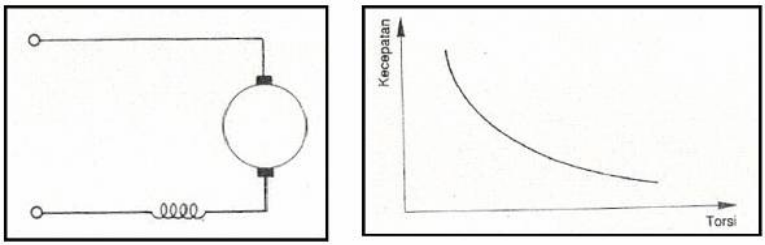


Gambar 7.9 Konstruksi Motor Disc

Motor magnet permanen

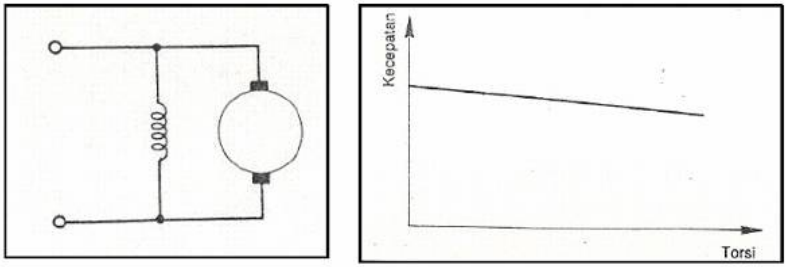
Motor di mana medan magnet di dalam stator dihasilkan oleh magnet permanen. Kekuatan medan magnetnya terbatas sehingga membatasi pula besar torsi yang mampu dihasilkannya. Macam motor magnet permanen:

1. Motor dengan lilitan seri (motor universal): bekerja baik dengan DC maupun AC, berputar lambat jika dikenai beban berat (torsi tinggi)



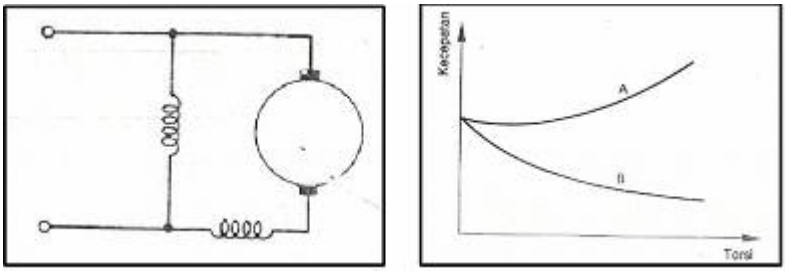
Gambar 7.10 Motor Lilitan Seri

2. Motor lilitan paralel: kecepatannya tidak terlalu terpengaruh oleh perubahan torsi yang terjadi, tetapi hanya dipengaruhi perubahan tegangan yang dikenakan kepada rotor.



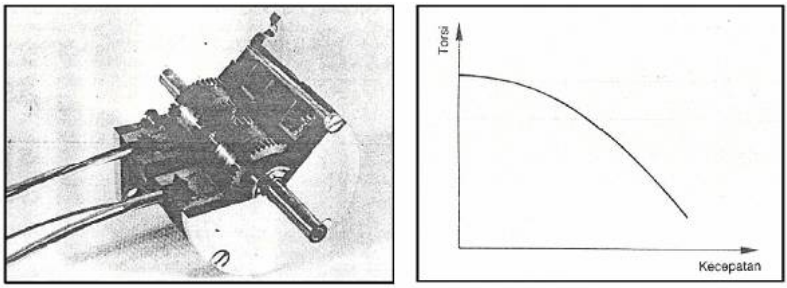
Gambar 7.11 Motor Lilitan Paralel

3. Motor dengan lilitan gabungan: medan magnet dalam stator dihasilkan melalui 2 koil yang terpisah.



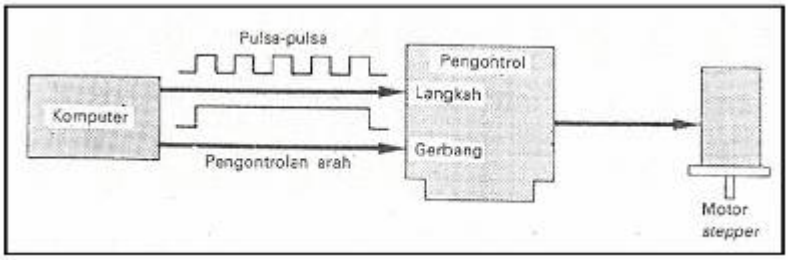
Gambar 7.12 Motor Lilitan Gabungan

Motor Stepper



Gambar 7.13 Motor Stepper

Prinsip Motor Stepper



Gambar 7.14 Prinsip Motor Stepper

Pengoperasian motor jenis ini berdasarkan pulsa-pulsa listrik. Setiap kali mengirim pulsa ke pengontrol elektronik, maka motor akan bergerak ‘selangkah’, yaitu satu putaran sudut kecil. Ukuran langkah tergantung pada perancangan motor dan dapat sekecil 1,5 derajat atau maksimal 30 derajat. Kecepatan pengiriman pulsa maksimum 2000 pulsa per detik.

7.3.3 Komponen Dasar Sebuah Robot

1. Manipulator
   - Mekanik
   - Penyangga gerakan (*appendage*)
   - *Base* (pondasi/ landasan robot)
2. *Controller*

   Adalah jantung dari robot untuk mengontrol (MP, RAM, ROM, sensor dan lain-lain).
3. *Power Supply*

   Sumber tenaga yang dibutuhkan oleh robot, dapat berupa energi listrik, energi tekanan cairan (hidrolik), atau energi tekanan udara (*pneumatic*).
4. *End Effector*

   Untuk memenuhi kebutuhan dari tugas robot atau si pemakai.

7.4 Tingkatan Teknologi Robot

1. Robot Teknologi Rendah

Robot teknologi rendah digunakan dalam lingkungan industri untuk pekerjaaan seperti mesin pemasang & pelepas, penanganan material, operasi pengepresan dan operasi perakitan sederhana. Karakteristik robot teknologi rendah:

- Siku, memiliki 2 sampai dengan 4 pergerakan siku dan biasanya robot teknologi rendah merupakan robot *non servo*.
- Beban kerja, untuk jenis robot teknologi rendah berkisar 3 sampai dengan 13,6 kg.
- Waktu siklus: waktu yang diperlukan sebuah robot untuk bergerak dari satu posisi ke posisi berikutnya. Di mana waktu siklus ini tergantung atas 2 faktor yaitu: beban kerja dan panjang lengan *manipulator*. Robot teknologi rendah biasanya memiliki waktu siklus yang cukup tinggi yaitu 5 sampai dengan 10 detik.
- Ketelitian, seberapa dekat sebuah robot dapat menggerakan *manipulator*-nya sesuai dengan titik yang telah diprogramkannya. Erat hubungannya dengan ketelitian yaitu keseragaman. Keseragaman menggambarkan seberapa sering sebuah robot melakukan program yang sama, mengulangi gerakannya pada titik yang telah diberikan. Baik ketelitian dan keseragaman sangat penting dalam sistem operasi berbagai robot. Untuk robot teknologi rendah ketelitiannya berkisar 0,050 – 0,025 mm.
- Aktuasi: metode pergerakan siku suatu robot. Aktuasi dapat dicapai dengan menggunakan *pneumatic*, hidrolik, maupun elektrik. Untuk robot yang

berteknologi rendah biasanya menggunakan motor listrik karena harganya murah dan operasinya mudah dikendalikan. Robot teknologi menengah umumnya digunakan untuk pekerjaaan mengambil dan meletakkan dan mesin pemasang & pelepas. Robot teknologi menengah memiliki kerumitan yang lebih tinggi.

2. Robot Teknologi Menengah

Karakteristik robot teknologi menengah:

- Siku, robot teknologi menengah memiliki jumlah siku yang lebih banyak dibandingkan dengan robot teknologi rendah dan memiliki baterai kerja yang lebih besar. Lengan robot ini juga memiliki kekuatan manuver yang lebih untuk memanipulasi. Siku Robot teknologi menengah berjumlah 5 sampai dengan 6 pergerakan siku.
- Beban kerja, beban kerja untuk jenis robot teknologi menengah berkisar 68 sampai dengan 150 kg. Dengan bertambahnya kemampuan beban kerja maka robot ini mampu menggantikan pekerja dalam situasi di mana mengangkat bagian yang berat secara konstan ketika diperlukan.
- Waktu siklus, robot teknologi menengah memiliki waktu siklus yaitu: dalam pergerakan siku sepanjang

25 sampai dengan 65 dapat ditempuh dalam waktu 1,0 detik. Semakin tinggi kompleksitas pekerjaan dan makin berat beban kerja yang diberikan maka makin bersar pula nilai waktu siklus yang diperoleh.

- ❖ Ketelitian, dengan bertambahnya jumlah siku akan juga berpengaruh dengan meningkatnya ketelitian. Untuk robot teknologi menengah ketelitiannya berkisar 0,2 sampai dengan 1,3 mm.
- ❖ Aktuasi, untuk robot yang berteknologi menengah digerakkan oleh 2 tipe motor yaitu: listrik atau hidrolik. Alasan menggunakan 2 tipe motor karena beban kerja yang berat.

3. Robot Teknologi Tinggi

Robot teknologi tinggi digunakan dalam lingkungan industri untuk pekerjaaan yang kompleksitasnya tinggi. Karakteristik Robot teknologi tinggi:

- ❖ Siku, memiliki 8 sampai dengan 10 pergerakan siku dan biasanya robot teknologi tinggi memiliki jenis pekerjaan yang komplek dan manuver gerakan yang beragam.
- ❖ Beban kerja, beban kerja untuk jenis robot teknologi tinggi berkisar 150 sampai dengan 250 kg.
- ❖ Waktu siklus, karena bertambahnya gerakan dan kompleksitas kerja yang tinggi maka waktu siklus

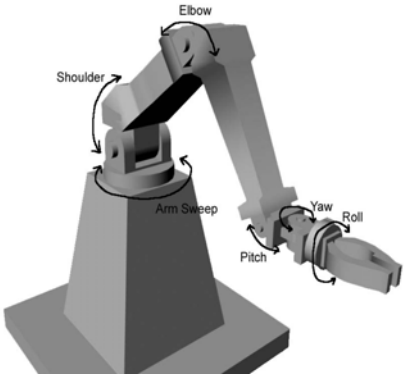
untuk robot teknologi tinggi berkisar 10 sampai dengan 25 detik.

- ❖ Ketelitian, dengan bertambahnya jumlah siku akan juga berpengaruh dengan meningkatnya ketelitian. Untuk robot teknologi tinggi ketelitiannya berkisar 1,5 sampai dengan 3,0 mm.
- ❖ Aktuasi, untuk robot yang berteknologi tinggi biasanya digerakkan oleh 3 tipe aktuator motor yaitu: listrik, hidrolik dan *pneumatic*.

7.5 Geometri Robot dan Istilah-Istilahnya

Degrees of Freedom (DOF) adalah setiap titik sumbu gerakan mekanik pada robot, tidak terhitung untuk *end effector*. Degrees Of Movement (DOM) adalah kebebasan/kemampuan untuk melakukan sebauh gerakan. Sebagai contoh, robot dengan 6 derajat kebebasan:

1. *Base Rotation* (dudukan untuk berputar)
2. *Shoulder Flex* (lengan atas/pundak)
3. *Elbow Flex* (lengan bawah)
4. *Wrist Pitch* (pergelangan angguk)
5. *Wrist Yaw* (pergelangan sisi)
6. *Wrist Roll* (pergelangan putar)



Gambar 7.15 Robot dengan 6 Derajat Kebebasan

Joint memungkinkan terjadinya gerakan pada dua bagian tubuh robot, sedangkan *link* menghubungkan tiap-tiap *joint*.

Tipe-tipe *joint*:

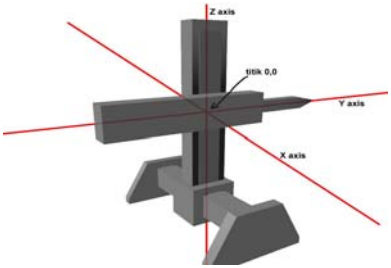
1. *Linear joint*, gerakan antara *in* & *out*, *link* adalah gerakan linear (tipe L-Joint).
2. *Orthogonal joint*, ini juga *linear joint*. Tetapi antara *in* & *out*, *Link*-nya saling tegak lurus (tipe O-Joint).
3. *Rotational joint*, merupakan penghubung di mana perputaran terjadi tegak lurus terhadap *in* & *out Link* (tipe R-Joint).
4. *Twisting joint*, mengakibatkan gerakan berputar, tapi putaran paralel dengan *in* & *out Link* (tipe T-Joint).
5. *Revolving joint*, *input Link*, paralel dengan *axis* perputaran dari *joint*. *Output* tegak lurus dengan putaran.

7.6 Konfigurasi Robot

Dikarenakan robot mempunyai bermacam-macam bentuk dan ukuran, sehingga memiliki beragam kemampuan gerakan. Secara fisik, ada beberapa konfigurasi yang dapat dibentuk, yaitu:

1. Konfigurasi Koordinat Kartesian

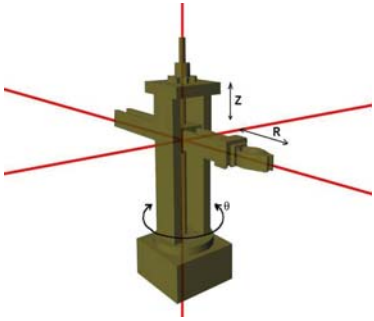
   Sistem koordinat kartesian berbasis akan 3 sumbu atau bidang, yaitu sumbu x, y dan z.



Gambar 7.16 Konfigurasi Koordinat Kartesian

2. Konfigurasi Koordinat Silinder

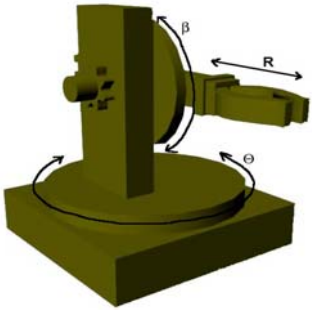
   Sistem koordinat silinder memiliki 3 derajat kebebasan (DOF) atau 3 axis, yang terdiri atas θ (*theta*) mewakili sumbu putar, sumbu *z* mewakili gerakan naik-turun serta sumbu *R* yang mewakili gerakan memanjang atau memendek.



Gambar 7.17 Konfigurasi Koordinat Silinder

3. Konfigurasi Koordinat Polar

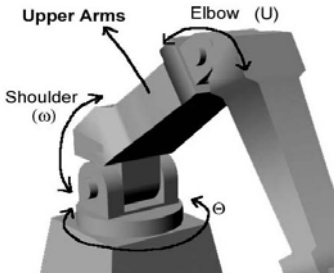
   Konfigurasi koordinat polar/simetrikal juga memiliki 3 sumbu yaitu *θ (theta), β (beta)*, dan *R*. dikatakan system simetrikal karena ruang gerakdari robot merupakan *sphere* (bola).



Gambar 7.18 Konfigurasi Koordinat Polar

4. Sistem Koordinat *Articulate*

   Sistem koordinat *articulate* didefinisikan dengan 3 sumbu, yakni *θ (theta), upper arm (w)* dan *elbow (U)*. Sumbu ini memberikan kefleksibelan lebih besar.



Gambar 7.19 Konfigurasi Koordinat *Articulate*

5. SCARA (Selective Compliance Assembly Robot Arm)

   Sistem sumbu yang mirip koordinat *articulate* tetapi berbasis pada gerakan horizontal. Memiliki kemampuan untuk *"insektion"*, salah satu sistem sumbu yang mungkin dari SCARA adalah seperti pada Gambar 7.20.



Gambar 7.19 Konfigurasi SCARA

7.7 Spesifikasi Teknis yang Lain

Sebagai tambahan konfigurasi fisik dari robot dan kemampuan gerak dasar, ada beberapa spesifikasi teknik yang lain di mana menjelaskan tentang efesiensi dan efektivitas dalam unjuk kerja pada robot.

Beberapa spesifik teknik adalah sebagai berikut:

1. **Work Volume,** arti kata *Work Volume* (area kerja) mengacu pada di mana robot itu dapat bekerja. Secara teknis dapat dikatakan adalah di mana ujung bagian masih digerakkan di bawah kontrol.

   Work Volume diperhitungkan dari:

   - Konfigurasi Fisik
   - Ukuran
   - Jangkauan Lengan
   - Hubungan/Joint Manipulator

   Fungsi mengetahui *Work Volume*:

   - *Lay Out*
   - Waktu Produksi
   - Area Kerja dan *Safety*
   - Program

2. **Precision of Movement,** ada tiga jenis kategori pada keakuratan gerakan dari ujung robot pada suatu penerapan, yaitu:

   - *Spatial Resolution*

Dapat diartikan sebagai gerakan terkecil yang masih dapat dikontrol oleh si pemrogram, sehingga *spatial resolution* adalah jumlah dari resolusi kontrol dengan ketidakakuratan mekanik.

- *Accuracy* (akurasi)

  Adalah kemampuan dari ujung robot untuk mencapai titik yang dituju. Dengan kata lain akurasi adalah setengah resolusi spasial.

- *Repeatability* (pengulangan)

  Adalah kemampuan dari ujung robot untuk mencapai titik yang sebelumnya dikontrol. *Repeatability* umumnya lebih kecil dari akurasi.

3. **Weight Carrying Capacity**, adalah kemampuan robot untuk memindahkan beban. Merupakan faktor untuk berbagai macam keperluan, yaitu:
   - Jenis tugas
   - Jenis barang
   - Produktivitas
4. **Type of Drive System**, ada tiga jenis dasar penggerak robot:
   - *Hydraulic*

     Menggunakan fluida/oli, kurang dalam segi kebersihan, beresiko kebakaran.

- *Pneumatic*

  Menggunakan tekanan udara merupakan jenis yang termurah, terpraktis dan *fixed points*.

- *Electric*

  Yang dimaksud adalah motor listrik. Ada dua jenis motor, yaitu motor DC dan motor *stepper*. Ciri khasnya adalah kecepatan.

  Selain penggerak yang telah disebutkan, untuk mencapai presisi, kecepatan serta gerakan yang diinginkan, robot selalu dilengkapi dengan *gear* dan *cam*.

5. **Effectors**, memiliki tujuan untuk melaksanakan tugas tertentu. Faktor-faktor yang penting dalam *end effector* adalah sebagai berikut:

   - Tugas
   - *Design*
   - Kontrol program
   - Ukuran area kerja
   - Waktu siklus
   - Keselamatan kerja

   Jenis *end effector* dapat dikelompokkan dalam dua jenis yaitu *gripper* dan *tooling*.

   Jenis-Jenis *Gripper*:

   - *Mechanical Gripper*

     Fungsi : Untuk menjepit objek

Tipe : *Inside – Outside*

Driver : *Gear – Pneumatic*

Perhitungan : Ukuran, gaya, kecepatan dan CG

- *Magnetic Gripper*

  Fungsi : Permukaan halus dan rapuh

  Tipe : *Single - Double*

  Driver : *Pneumatic*

  Perhitungan : Daya hisap, luas 'cups', tipe permukaan

- *Vaccum Gripper*

  Fungsi : Permukaan datar dan metal

  Tipe : *Single - Double*

  Driver : *Electric*

  Perhitungan : Beban, panas dan arus listrik

Jenis-Jenis *Tooling*:

- *Drilling*
- *Painting*
- *Welding*
- *Surfacing*

7.8 Sistem Kontrol

Sistem kontrol otomatis pertama adalah *governor sentrifugal* untuk pengontrolan kecepatan mesin uap yang dibuat oleh James Watt pada abad ke-18.

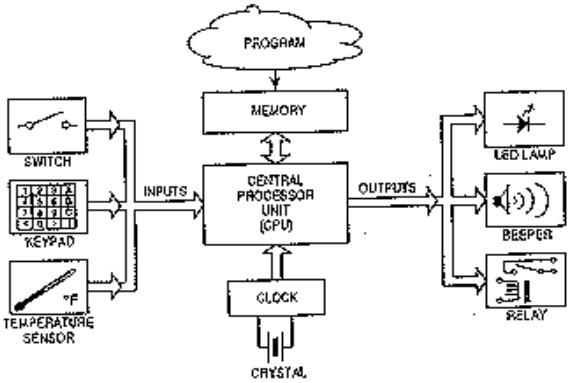
1922 : Minorsky membuat kontroler otomatis untuk mengemudikan kapal dengan cara menentukan kestabilan dari persamaan diferensial.

1932 : Nyquist mengembangkan suatu prosedur yang relatif sederhana untuk menentukan kestabilan sistem lup tertutup pada basis respon lup terbuka terhadap masukan lunak (*steady state*) sinusoida.

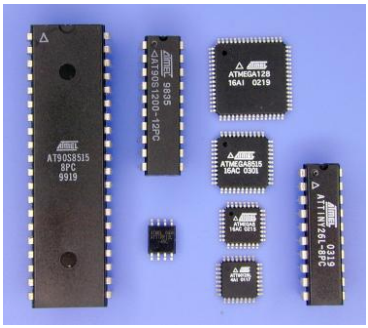
1934 : Hazen, yang memperkenalkan istilah servomekanisme untuk sistem kontrol posisi, membahas desain servomekanisme relai yang mampu mengikuti dengan baik masukan yang berubah.

1940-an : metode respon frekuensi untuk mendesain sistem kontrol linear berumpan-balik yang memenuhi persyaratan performansi.

Akhir tahun **1940** hingga awal tahun **1950,** metoda tempat kedudukan akar dalam desain sistem kontrol. Teori kontrol modern dapat dikatakan menuju pada kontrol optimal. Penerapan teori kontrol modern dalam bidang non-teknik seperti biologi, ekonomi, kedokteran dan sosiologi sekarang banyak dilakukan dan hasil-hasil yang menarik dan berarti akan dapat diperoleh di masa datang.



Gambar 7.20 Konsep Mikrokontroler



Gambar 7.21 Contoh Mikrokontroler

7.8.1 Jenis Robot Kontrol

Ada beberapa jenis pengatur gerakan pada robot, di antaranya:

1. *Limite Sequence Robot*

   Ciri-ciri: paling sederhana, paling murah, umumnya menggunakan *driver pneumatic*, operasinya *pick* & *place*.

2. *Point to Point*

   Ciri-ciri: lebih canggih dari *Limite Sequence Robot*, menyimpan titik-titik dari langkah robot, menggunakan driver hydraulic, motor elektronik.

3. *Countouring*

   Ciri-ciri: peningkatan *point to point*, *speed* & *countour*, menggunakan *driver hydraulic*.

4. *Line Tracker*

   Ciri-ciri: untuk benda bergerak, senior dan program, menggunakan *driver hydraulic*.

5. *Intelligent Robot*

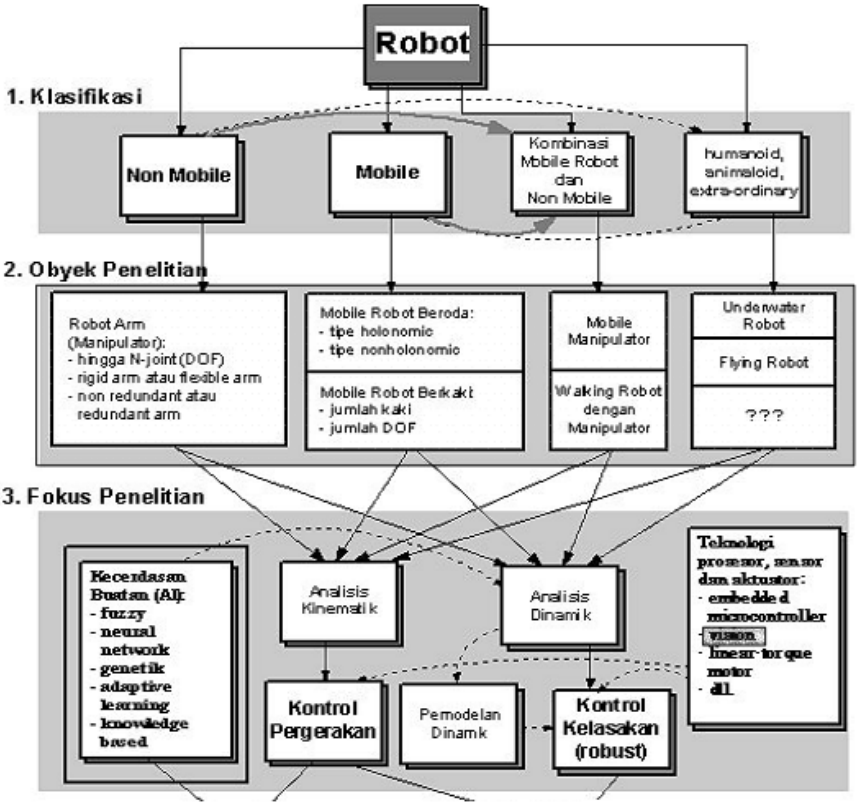
   Ciri-ciri: dapat bereaksi dengan lingkungan, dapat mengambil keputusan, *advance* I/O dan sensor.

7.8.2 Bagian-Bagian Kontrol Robot

Kontrol pada robot dapat dikelompokan dari level rendah, menengah dan tinggi. Secara detail adalah sebagai berikut:

- *Low Technology Controllers*, mungkin dapat diprogram untuk praktis atau tidak praktis. Tidak ada internal *memory*.
- *Medium Technology Controllers*, mempunyai 2 sampai 4 sumbu bergerak dan memiliki mikroprosesor serta memori (terbatas). Tetapi I/O-nya terbatas, *delay* setiap gerakan serta dapat diprogram jika kerja telah lengkap.

- *High Technology Controllers*, memiliki memori yang besar serta punya mikroprosesor dan co-mikroprosesor. Bermacam-macam I/O, *reprogram* dalam waktu singkat, mempunyai sampai dengan 9 *axis*. Dalam *controller*-nya ada 5 bagian penting, yaitu *power supply*, *interface, axis drive board*, *option boards* dan mikroprosesor.



Gambar 7.22 Diagram Penelitian Robot

7.9 Aplikasi Robot di Industri/Manufaktur

Di sini akan mengulas bagaimana robot diaplikasikan pada industri. Pertama kita lihat kondisi pada industri sehingga diperlukan robot, yaitu:

- Kondisi yang berbahaya
- Pekerjaan yang berulang dan membosankan
- Bagian yang sulit dibawa
- Operasi dengan banyak shift

General Electric Co. memberikan kriteria untuk survei penggunaan robot, yaitu:

- Operasi berulang dan sederhana dibutuhkan
- *Cycle time* lebih besar dari 5 detik
- *Part* dapat dipindahkan pada lokasi dan orientasi tepat
- Berat *part* memadai
- Satu atau dua orang dapat digantikan dalam 24 jam

Aplikasi robot pada industri:

1. Material transfer
    - *Pick* & *Place*
    - *Palleting*
    - *Depalletizing*
    - *Line Tracking*
2. *Machine loading*
    - *Die Casting*
    - *Injection* (*plastic*) *molding*

- *Transfer* (*plastic*) *molding*
- *Hot forging*
- *Up setting or upset forging*
- *Stamping press operation*
- *Machining operation*

3. *Welding*
    - *Spot welding*
    - *Arc welding*
4. Spray coating
5. Processing operations
    - Finishing
    - Bubut
6. *Assembly*
7. *Inspection*

7.10 Tipe Robot

AIBO

AIBO merupakan singkatan dari AI roBOt, yang artinya robot dengan intelegensia buatan. Di Jepang sendiri, "aibo" berarti sahabat. Versi terbaru dari robot anjing AIBO hadir memberikan hiburan dengan desain yang futuristik, AIBO ERS-220. Robot ini mempunyai 16 motor yang memungkinkannya dapat berjalan, bermain bola, duduk, dan berbaring. Lalu dengan sensor penglihatan dan pendengaran, 21 lampu mengelilingi bagian

kepala, dan dibagian atas kepala terdapat lampu yang mengekspresikan berbagai emosi dan insting untuk menghibur pemiliknya. ERS-220 memiliki kemapuan wireless LAN sehingga kita dapat mengedalikan dari jauh.



Gambar 7.23 AIBO

ASIMO

ASIMO adalah singkatan dari Advanced Step in Innovative Mobility dan telah datang ke Jakarta pada tanggal 19-27 Juli yang lalu di pameran Gaikindo. Dengan tinggi 120 cm, robot ini memiliki sistem komputerisasi dan sensor-sensor yang dapat mengatur setiap gerakannya dan memungkinkan bertingkah laku seperti gerakan manusia. ASIMO dapat melangkah naik dan turun tangga, melambaikan tangan, melakukan langkah dansa, serta berbicara dalam berbagai bahasa. Pengembangan teknologi

robotika mendapat perhatian dari para peneliti Jepang, bahkan mereka juga meminta pemerintahnya untuk melakukan investasi, dengan tujuan di beberapa tahun mendatang dapat diciptakan mesin yang memiliki intelejensia buatan layaknya anak kecil.



Gambar 7.24 ASIMO

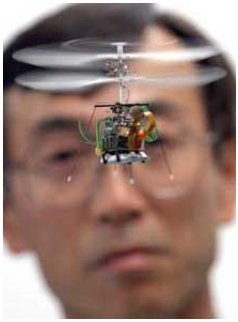
Insinyur Honda menciptakan ASIMO dengan 26 derajat kebebasan yang membantu berjalan dan melaksanakan banyak tugas manusia. Satu derajat Kebebasan adalah kemampuan untuk bergerak ke kiri dan ke kanan atau ke atas dan ke bawah. Derajat kebebasan ini dibuat seperti halnya sambungan otot pada manusia untuk pergerakan yang maksimum dan fleksibel. ASIMO mempunyai dua derajat kebebasan pada lehernya, enam

pada setiap lengannya dan enam pada setiap kakinya. Material pada badannya, adalah struktur magnesium *alloy*, dikombinasikan dengan komputer kuat dalam ransel di punggungnya dan 26 servo motor di seluruh badannya untuk membantu ASIMO berjalan dan bergerak dengan lembut dengan mudah. ASIMO dirancang untuk beroperasi di lingkungan kita, di mana kita harus menjangkau sesuatu, mengambil sesuatu dan melakukan navigasi untuk berjalan berkeliling,serta memanjat tangga misalnya. itu adalah mengapa ASIMO mempunyai dua lengan dan dua kaki sering dipanggil dengan robot humanoid. Sesungguhnya, ASIMO hanyalah robot humanoid yang dapat berjalan dengan bebas dan memanjat tangga. Kemampuan dasar Ini adalah penting, sebab lingkungan kita yang penuh dengan permukaan tidak seimbang, rintangan dan tangga rumah, untuk dapat mampu dengan mudah berfungsi dan dapat membantu manusia.

Robot Terbang Terkecil Hadir di Jepang

Seiko Epson Corp memperkenalkan robot terbang mikro, yang merupakan robot terbang termungil dan paling ringan di dunia. Robot yang merupakan pengembangan model sebelumnya ini bisa dikendalikan dari jauh menggunakan komputer secara *wireless* (menggunakan *bluetooth*), dan dilengkapi kamera kecil yang mampu mengirimkan foto-foto tanpa perlu bantuan kabel.

Robot terbang kecil bernama *Micro Flying Robot* ini diharapkan bisa dipakai dalam berbagai bidang, misalnya untuk melakukan pengamatan dan pencarian di wilayah-wilayah sempit maupun daerah berbahaya, kata Epson. Robot terbang yang berbentuk seperti helikopter mini ini dilengkapi microcontroller 32-bit dan dua motor ultrasonik berukuran kecil guna memutar baling-baling dalam dua arah berbeda sehingga robot bisa terbang. Model baru saat ini dengan lebar 136 milimeter, tinggi 85 mm dan berat 12,3 gram dengan baterai (8,6 gram tanpa baterai). Saat ini sang robot baru bisa terbang selama tiga menit. Namun perusahaan pembuatnya berencana mengembangkan kemampuannya sehingga ia bisa digunakan untuk tugas-tugas nyata. Adapun *Micro Flying Robot* sebenarnya adalah penerus robot terbang sebelumnya. Robot terdahulu memiliki keterbatasan terbang karena ia harus disambungkan dengan sumber tenaga menggunakan kabel dan harus berada dalam jangkauan mata pengendali saat terbang.



Gambar 7.25 *Micro Flying Robot*

Macam-Macam Robot Lainnya

LandShark: Robot Peace Maker



Misi robot yang didesain oleh Black-i Robotics ini adalah untuk perdamaian dunia - robot ini mampu menjinakkan bom. Perusahaan pembuatnya didirikan oleh Brian Hart setelah dia kehilangan anak laki-lakinya dalam perang Irak. Robot ini cukup rumit dan didesain khusus sebagai penjinak bom dan menyelamatkan prajurit yang terluka dalam perang.

Robot Pelayan dari Jepang



Assistant Robt (AR) asli Jepang ini lumayan tangguh untuk mengerjakan pekerjaan-pekerjaan seperti mencuci, ngepel, dan membersihkan dapur. Tapi jangan senang dulu, produsennya bilang butuh satu atau dua dekade untuk memproduksi massal robot ini.

QRIO: Robot yang bisa menari



QRIO ("Quest for cuRIOsity") adalah sebuah robot humanoid bipedal untuk pertunjukan yang dijual oleh Sony mengikuti kesuksesan mainan AIBO sebelumnya. QRIO tingginya sekitar 60 cm dengan berat 7,3 kg. QRIO mampu mengenali suara dan wajah sehingga mampu mengingat orang lain. QRIO begerak dengan kecepatan 23 cm/detik dan masuk dalam buku *Guinness World Records* (2005 *edition*) sebagai robot bipedal pertama (dan tercepat) yang mampu berlari. Generasi keempatnya memiliki baterai yang mampu bertahan hingga 1 jam. *Programmer*-nya membutuhkan waktu tiga minggu untuk memprogram koreografinya.

Murataseiko-chan: Robot Sepeda Satu Roda

Produsen alat elektronik Murata Manufacturing Co., Ltd., pencipta robot "Murata Boy" yang populer, membuat robot dengan kemampuan menyeimbangkan dirinya sendiri yang diberi nama “Murata Seiko-chan.” Tidak hanya menyeimbangkan dirinya saat mundur dan maju, tapi juga mendeteksi halangan dengan sensor-sensornya dan bergerak mendahului atau memutarinya. Robot dengan tinggi 50 cm dan berat 5 kg ini

dilengkapi dengan *bluetooth* dan kamera yang mengirim sinyal *video live*.



Gambar 7.26 Murataseiko-chan

EMA: Robot Jepang yang Bisa Berciuman

EMA (Short for Eternal, Maiden, Actualization) adalah robot humanoid "hot" pertama dari Jepang. Didesain oleh Sega, tingginya 38 cm, robot montok yang suka mencium, bernyanyi, dan bergoyang, menggunakan sensor-sensor infra merah dan bertenaga baterai, gadis kecil ini akan mencium orang yang berada di dekatnya, menjadi apa yang desainernya sebut "Love Mode". Dia juga kemana-mana membawa kartu nama.



Gambar 7.27 Murataseiko-chan

Bab 8

Tabu Search

8.1 Sejarah Tabu Search

Tabu search berasal dari sebuah bahasa daerah yaitu bahasa Polinesia yang artinya suatu hal yang tidak boleh disentuh karena 'sakralnya'. *Tabu search* adalah sebuah metode optimasi, yaitu metode yang mencari sebuah solusi optimal untuk menyelesaikan masalah. *Tabu search* berbasis pada *local search,* artinya pencarian sebuah solusi optimal, bergerak dari satu solusi ke solusi lainnya. Hal ini dilakukan dengan cara memilih solusi terbaik dari solusi sekarang di mana solusi itu tidak termasuk solusi terlarang atau tabu. Tujuan utama dari *tabu search* adalah, agar tidak melakukan pencarian ulang pada solusi yang telah pernah ditelusuri sebelumnya. Hal ini dilakukan dengan sebuah struktur memori yang mencatat jejak proses pencarian. Hal ini dilakukan demi keefisienan dalam proses pencarian dengan tidak melakukan pencarian ulang.

Struktur memori yang digunakan dalam *tabu search* dinamakan *tabu list*. *Tabu list* menyimpan atribut-atribut dari proses pindah (*move*) yang dilakukan sebelumnya. *Tabu search* menggunakan *tabu list* agar sebuah proses pencarian tidak mengalami sebuah *cycling* pada sebuah solusi yang sama. Dan

tabu list juga digunakan untuk menuntun proses pencarian menelusuri daerah solusi yang belum ditelusuri, hal ini dilakukan dengan atribut-atribut yang sebelumnya harus dipenuhi dalam *tabu list*. Dengan atribut ini, proses pencarian bisa saja mengalami *cycling* pada solusi yang sama, dan juga bisa mendapatkan solusi yang baru dengan cara menelusuri daerah solusi yang belum ditelusuri. Tanpa strategi ini, *local search* yang sudah menemukan optimal solusi, bisa terjebak di optimal solusi tersebut dan iterasi-iterasi berikutnya.

Kinerja algoritma *tabu search* sangat bergantung pada panjang pendeknya *tabu list*, secara empirik, ukuran *tabu list* untuk menghasilkan solusi yang baik akan semakin besar seiring dengan membesarnya ukuran masalah. Namun tidak ada aturan baku untuk menentukan panjang pendeknya *tabu list*. Pada tiap iterasi, dipilih solusi baru yang terbaik dari *neighborhood* dan tidak tergolong sebagai tabu. Kualitasnya tidak harus lebih baik dari solusi sebelumnya, namun apabila memiliki nilai fungsi objektif yang lebih baik dari solusi sebelumnya, maka bisa disebut sebagai solusi terbaik yang baru. Dalam *tabu search* juga ada yang disebut dengan kriteria aspirasi, yaitu sebuah penanganan terhadap sebuah *move* yang menghasilkan sebuah solusi yang baik, namun *move* nya bersifat tabu. Dalam hal ini, jika *move* tersebut memenuhi kriteria aspirasi yang telah ditentukan sebelumnya, maka status tabu tersebut bisa diabaikan untuk membentuk solusi berikutnya.

8.2 Definisi dan Algoritma Tabu Search

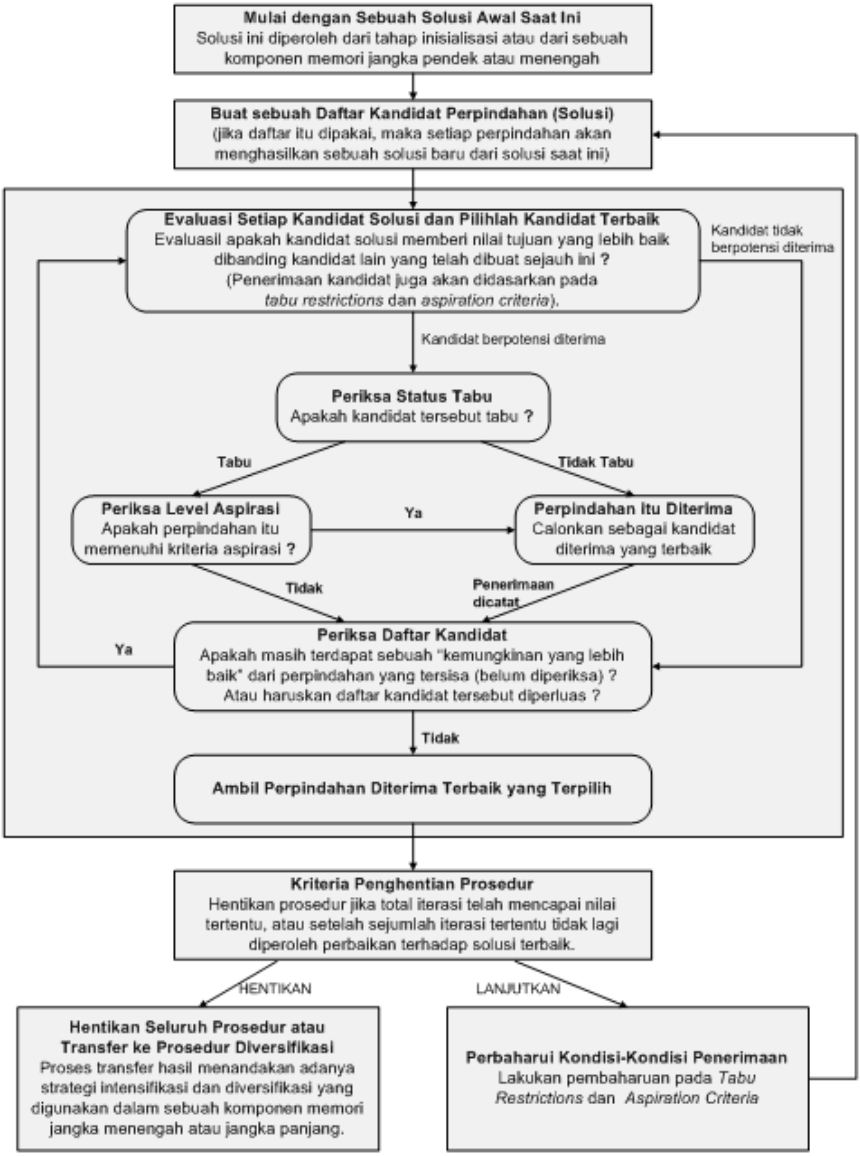
Tabu search pertama kali diperkenalkan oleh Fred W. Glover sekitar tahun 1986. Glover menyatakan bahwa *tabu search* adalah salah satu prosedur metaheuristik tingkat tinggi untuk penyelesaian permasalahan optimisasi kombinatorial. *Tabu search* ini dirancang untuk mengarahkan metode-metode lain (atau komponen proses *tabu search* itu sendiri) untuk keluar atau menghindari dari masuk dalam solusi optimal yang bersifat lokal. Kemampuan *tabu search* dalam menghasilkan solusi yang mendekati optimal telah dimanfaatkan dalam beragam permasalahan klasik dan praktis dari berbagai bidang mulai bidang penjadwalan hingga bidang telekomunikasi.

Glover mengatakan bahwa prosedur *tabu search* ini dapat ditemukan dalam tiga pola *(scheme)* utama.

1. Pola pertama adalah adanya penggunaan struktur memori berbasiskan atribut-atribut fleksibel yang dirancang untuk membolehkan sebuah kriteria evaluasi dan hasil pencarian di masa lalu dieksploitasi lebih mendalam. Pola ini menjadikan *tabu search* berbeda dengan aplikasi lain yang menggunakan struktur memori yang *rigid* (kaku) atau tanpa menggunakan struktur memori (seperti *simulated annealing*).
2. Pola kedua adalah penggunaan mekanisme atau kondisi yang dapat membatasi atau membebaskan suatu proses pencarian yang sedang berlangsung. Pola kedua ini dikenal sebagai mekanisme *tabu restriction* dan *aspiration criteria*.

3. Pola ketiga adalah pelibatan suatu fungsi memori dengan rentang waktu yang berbeda yakni berupa memori jangka pendek (*short term memory*) dan memori jangka panjang (*long term memory*) untuk menjalankan strategi intensifikasi dan diversifikasi dalam proses pencarian solusi. Strategi intensifikasi adalah strategi pencarian yang mengarahkan/mengfokuskan pencarian pada suatu area tertentu, sedangkan strategi diversifikasi adalah strategi pencarian yang mengarahkan pencarian pada area baru (yang belum dikunjungi).

Skema umum *Tabu Search* disajikan pada Gambar 8.1. Pemilihan kandidat terbaik didasarkan nilai fungsi tujuan. Pemeriksaan nilai fungsi tujuan lebih didahulukan sebelum pemeriksaan status *tabu*. Apabila nilai fungsi tujuan sebuah kandidat lebih baik dari yang lain, maka kandidat tersebut berpotensi untuk diterima sehingga perlu diperiksa status tabunya. Urutan pemeriksaan nilai fungsi tujuan kemudian status *tabu* memberikan kemungkinan proses penyelesaian program yang lebih cepat. Pemilihan kandidat solusi terbaik yang dilakukan oleh *Tabu Search* menggunakan prinsip *global-best strategy* (GB) bukan *first-best strategy* (FB). GB adalah strategi di mana algoritma akan mengganti solusi terbaik saat ini dengan solusi terbaik yang ada pada *neighborhood*. Adapun FB adalah strategi di mana algoritma akan mengganti solusi terbaik saat ini secara langsung jika solusi yang lebih baik ditemukan.



Gambar 8.1 Skema Algoritma *Tabu Search*

Tabu Search umumnya tidak menggunakan pembentukan kandidat solusi secara acak sebagaimana *simulated annealing* dan *genetic algorithm*. Pemilihan kandidat solusi dalam *Tabu Search* juga tidak dilakukan secara probabilistik sebagaimana *ant colony system*, *simulated annealing* dan *genetic algorithm*. Karakteristik ini menjadikan solusi yang dihasilkan *tabu search* akan sama setiap kali dilakukan proses pencarian solusi terhadap suatu permasalahan. Karakteristik ini juga menjadi salah satu keunggulan *tabu search* dibandingkan *ant colony system*, *simulated annealing* dan *genetic algorithm*.

Dengan kata lain, Pencarian Tabu *(Tabu Search)* didasarkan pada konsep pencarian "cerdas" yaitu menggunakan suatu bentuk memori untuk mengingat titik-titik pencarian sebelumnya dan pelarangan mencari kembali pada kumpulan *(set)* pencarian tersebut. *Tabu search* memiliki:

1. *Adaptive Memory*, yang mengingat beberapa informasi selama proses pencarian. Hal ini bisa meningkatkan tingkat efisiensi.
2. *Responsive Exploration*, yang mencari beberapa jalur yang termasuk kategori tidak-begitu-baik *(not-so-good path)* yang kadang-kadang memperoleh lebih banyak informasi daripada hanya mencari jalur yang bagus *(good path)* saja.

Jika solusi potensial yang telah dikunjungi sebelumnya (dalam jangka waktu yang singkat yang telah ditentukan) atau telah melanggar aturan, maka akan ditandai sebagai *"taboo"*, jadi algoritma tersebut tidak mempertimbangkan suatu

'kemungkinan' berulang-ulang kali. *Tabu Search* selalu berusaha mencari cara yang baru dan lebih efisien dalam mengambil keuntungan dari penggabungan mekanisme *adaptive memory* dan *responsive exploration* sebelumnya.

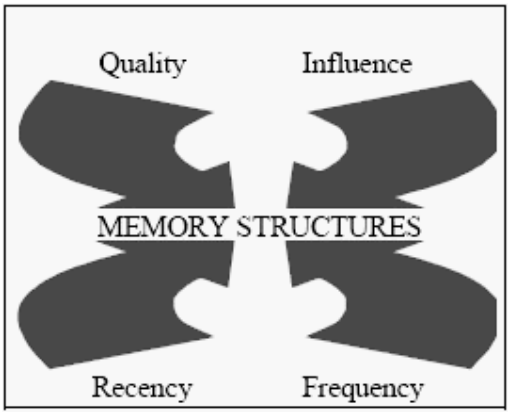
Algoritma *Tabu Search* secara garis besar dapat ditampilkan sebagai berikut. Solusi akhir adalah *best,* dengan *cost* sebesar GlobalMin.

```
0   Tetapkan:
    • X=matriks input berukuran nxm
    • MaxItr=maksimum iterasi
1   S=bangkitkan solusi secara random
2   GlobalMin=FCost(S)
3   Best=S
4   TabuList=[]
5   Kerjakan dari k=1 sampai MaxItr:
6       BestSoFar=FCost(s)
7       BestMove=S
8       Kerjakan dari i=1 sampai (n-1)
9           kerjakan dari j=i sampai n:
10              L=Tukar(S[i],S[j])
11              Cost=FCost(L)
12              Jika (L ∉ TabuList) atau (Cost < GlobalMin), kerjakan:
13                  Jika (Cost < BestSoFar), kerjakan
14                  BestSoFar=Cost
15                  BestMove=L
16  S=BestMove
17  Tambahkan S ke TabuList
18  Jika BestSoFar < GlobalMin, kerjakan:
19      GlobalMin=BestSoFar
20      Best=BestMove
```

8.3 Struktur Memori dan Komponen Tabu Search

Struktur memori dalam *Tabu Search* bekerja dengan merujuk pada empat prinsip dasar, yang terdiri atas *recently, frequency, quality* dan *influence.*

- *Recently-based* dan *frequency-based* saling melengkapi satu sama lain dan mempunyai karakteristik penting.
- *Quality* merujuk pada kemampuan untuk membedakan kualitas dari solusi yang ditemui selama proses pencarian. Dalam kerjanya, *quality* menjadi dasar dari pembelajaran *incentive-based* di mana dorongan diberikan untuk menguatkan aksi yang dapat mencari solusi yang baik. Penalti diberikan untuk melemahkan aksi yang dapat menyebabkan solusi yang tidak baik.
- *Influence,* didasarkan pada akibat dari pemilihan yang dibuat selama proses pencarian, tidak hanya pada kualitas tetapi juga pada strukturnya.



Gambar 8.1 Empat Prinsip Dasar Memori dalam *Tabu Search*

Penggunaan memori pada *tabu search* adalah secara atributif dan eksplisit. Atributif memori digunakan untuk menyimpan informasi tentang atribut solusi yang berubah karena berpindah dari satu solusi ke solusi yang lain, sedangkan eksplisit memori menyimpan solusi secara lengkap, termasuk solusi yang baik yang diketahui selama proses. Pada tipe memori atributif menyimpan informasi tentang atribut dari solusi yang berubah selama pergerakan dari satu solusi ke solusi yang lain. Contohnya, dalam *graph* atau jaringan, atribut dapat berupa kumpulan *node* yang ditambah, dikurangi atau dipindahkan selama mekanisme pergerakan.

Komponen-komponen pada *Tabu Search* bisa diuraikan sebagai berikut:

- **Representasi solusi**: setiap solusi fisibel pada suatu permasalahan optimasi harus direpresentasikan secara unik.
- **Fungsi cost**: setiap fungsi akan memetakan setiap solusi fisibel (S) ke nilai *cost*-nya.
- **Move**: menggerakkan *pattern* melalui jalur-jalur yang ditentukan oleh kriteria-kriteria tertentu.
- **Basic Neighborhood:** suatu fungsi yang memetakan setiap solusi fisibel (S) ke solusi-solusi yang lainnya.
- **Tabu List**: didefinisikan sebagai *set* $T = \{T_1, T_2, ..., T_s\}$. T_j adalah elemen ke-j dari T yang berisi nilai objektif yang berkaitan. Apabila s bervariasi dengan cara yang asiklis, maka *looping* akan dapat dihindari.

- **Tabu Tenure**: *Tabu tenure* s di-*set* secara dinamis.
- **Stopping Rules**: aturan/syarat untuk menghentikan perhitungan, bisa dipilih dengan *infinite* artinya proses mencoba segala yang mungkin, atau bisa dengan kriteria bila solusi yang diperoleh tidak mengalami peningkatan setelah mengalami iterasi berulang sebanyak t_f atau bisa dengan aturan-aturan yang lain.

8.4 Penerapan Tabu Search

Tabu Search dapat diaplikasikan pada berbagai bidang kehidupan.

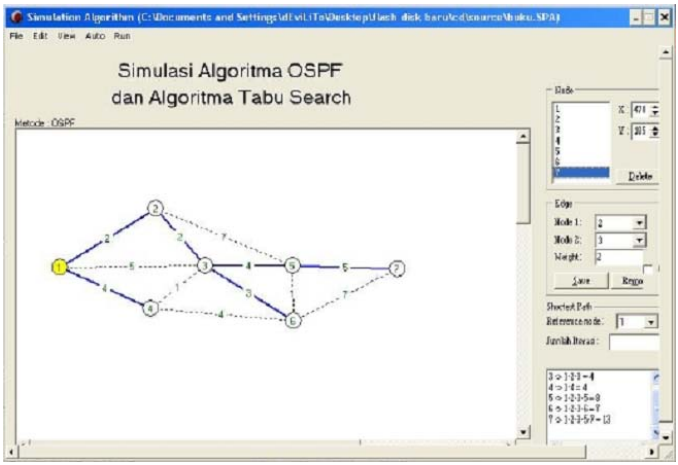
Tabel 8.1 Aplikasi pada *Tabu Search*

Scheduling	**Telecommunications**
Flow time cell manufacturing *Heterogonous processor scheduling* *Workforce planning* *Classroom scheduling* *Machine scheduling* *Flow shop scheduling* *Sequencing and batching*	*Call routing* *Bandwidth packing* *Hub facility location* *Path assignment* *Customer discount planning* *Failure immuce architecture* *Synchronous optical networks*
Design	**Productions, inventory, and Investment**
Computer aided design *Fault tolerant networks* *Transport network design* *Architectural space planning*	*Flexible manufacturing* *Just in time production* *Capicated MRP* *Part selection* *Multi item inventor planning*

Diagram coherency *Fixed change network design* *Irregular cutting problems*	*Volume discount acquisition* *Fixed mix investment*
Location and allocation	**Routing**
Multicommodity location/allocation *Quadratic assignment* *Quadratic semi assignment* *Multilevel generalized assignment* *Lay-out planning* *Off-shore oil exploration*	*Vehicle routing* *Capicated routing* *Time window routing* *Multi-mode routing* *Mixed fleet routin* *Traveling salesman* *Traveling purchaser*
Logic and artificial Intelligence	**Graph optimization**
Maximum satisfiability *Probabilistic logic* *Clustering* *Pattern recognition/classification* *Data integrity* *Neural network Training and design*	*Graph partioning* *Graph coloring* *Clique partioning* *Maximum clique problems* *Maximum planner graph* *P-median problems*
Technology	**General combinatorial optimization**
Seismic inversion *Electrical power distribution* *Engineering structural design* *Minimum volume ellipsoids* *Space station construction* *Circuit cell placement*	*Zero-one programming* *Fixed charge optimization* *Nonconvex nonlinear programming* *All-or-none networks* *Bilevel programming* *General mixed integer optimization*

Berikut ini contoh program aplikasi yang termasuk ke dalam *routing* (penjaluran) atau *traveling salesman.*

a. Program Simulasi Algoritma OSPF *(Open Sortest Path First)* dan Algoritma *Tabu Search*



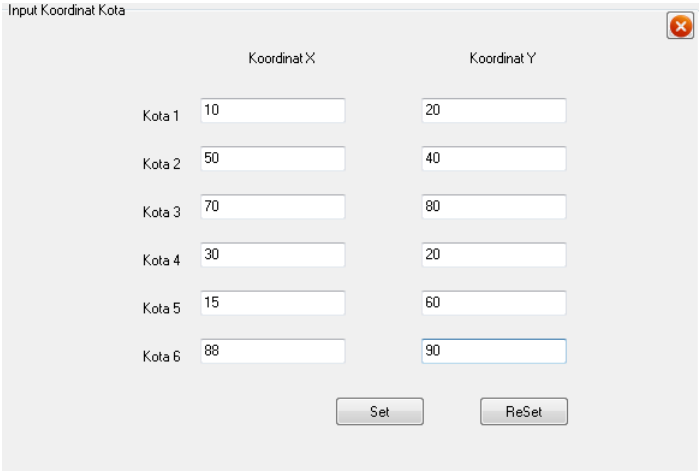
Gambar 8.2 Aplikasi Simulator Algoritma OSPF dan *Tabu Search*

Pada Gambar 8.2, terlihat hasil proses dari algoritma *Tabu Search*. Bagian sebelah kanan adalah hasil dari pemrosesan. *Tabu Search* tersebut melakukan tujuh kali proses iterasi. Setiap iterasi didapatkan hasil. Untuk iterasi pertama, hasilnya akan langsung masuk pada *Tabu List.* Untuk iterasi selanjutnya, hasil akan dibandingkan terlebih dahulu, apakah hasil tersebut lebih baik dari *Tabu List* atau tidak. Jika lebih baik, hasil iterasi terakhir itu yang dipakai

menggantikan isi *Tabu List.* Jika sebaliknya, program akan melanjutkan iterasi berikutnya. Program akan berhenti setelah semua iterasi selesai dilakukan.

b. Program Pencari Rute Terdekat

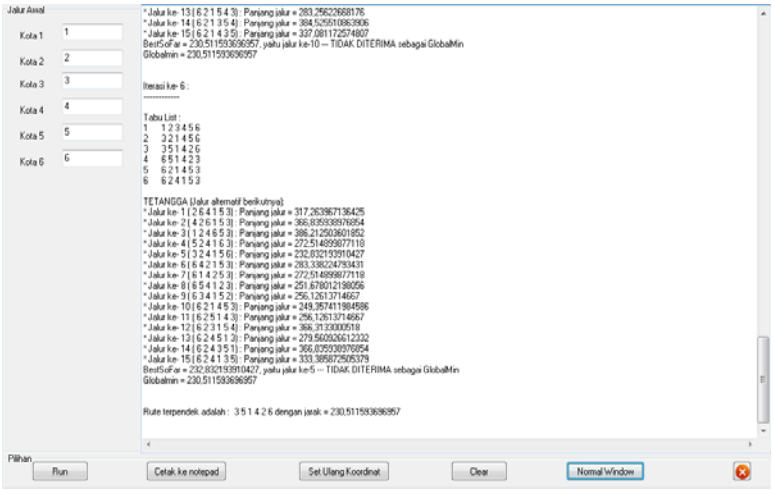
Aplikasi dimulai dengan *input* koordinat dari kota 1 sampai dengan kota 6 kemudian simpan dengan menekan tombol 'Set'. Maka tampilan utama dari program aplikasi akan tampil seperti pada Gambar 8.3 dan 8.4. Aplikasi berikut menggunakan bahasa Delphi dalam proses pembuatannya.



Gambar 8.3 Meng-*input* Koordinat Kota

Pengguna bisa memasukkan jalur awal pada kotak pengisian kota 1 sampai 6 berdasarkan *input* koordinat yang sudah di-*set*. Kemudian klik tombol 'Run' untuk

memulai proses pencarian rute terpendek. Perhatikan Gambar 8.4.



Gambar 8.4 Tampilan Hasil Perhitungan

Pada hasil perhitungan, terdapat iterasi sebanyak 6 proses dengan hasil *Tabu List* yang berbeda pada setiap iterasi. Untuk setiap iterasi, ditampilkan kemungkinan-kemungkinan jalur yang berdekatan (tetangga) sebagai jalur alternatif. Setiap iterasi akan diambil nilai yang terbaik *(BestSoFar)* yang bisa diterima atau tidak sebagai *GlobalMin* berdasarkan nilai yang ada di *Tabu List* sebelumnya. Hasil akhir akan ditampilkan rute terpendek dengan jaraknya. Proses setiap iterasi untuk inputan kota di atas dituliskan seperti berikut:

T A B U S E A R C H

Koordinat Kota:

Kota Ke -	X	Y
1	10	20
2	50	40
3	70	80
4	30	20
5	15	60
6	88	90

Iterasi ke- 1:

Tabu List 1: 1 2 3 4 5 6 dengan *cost*: 388,002357477953

TETANGGA (Jalur alternatif berikutnya):

* Jalur ke-1 (2 1 3 4 5 6): Panjang jalur = 386,130505490182
* Jalur ke-2 (3 2 1 4 5 6): Panjang jalur = 251,678012198056
* Jalur ke-3 (4 2 3 1 5 6): Panjang jalur = 368,000292789747
* Jalur ke-4 (5 2 3 4 1 6): Panjang jalur = 360,872267942862
* Jalur ke-5 (6 2 3 4 5 1): Panjang jalur = 367,469546452378

* Jalur ke-6 (1 3 2 4 5 6): Panjang jalur = 384,307057408525
* Jalur ke-7 (1 4 3 2 5 6): Panjang jalur = 360,872267942862
* Jalur ke-8 (1 5 3 4 2 6): Panjang jalur = 366,835938976854
* Jalur ke-9 (1 6 3 4 5 2): Panjang jalur = 325,259532861921
* Jalur ke-10 (1 2 4 3 5 6): Panjang jalur = 387,368750002429
* Jalur ke-11 (1 2 5 4 3 6): Panjang jalur = 325,259532861921
* Jalur ke-12 (1 2 6 4 5 3): Panjang jalur = 384,525510863906
* Jalur ke-13 (1 2 3 5 4 6): Panjang jalur = 386,397362851677
* Jalur ke-14 (1 2 3 6 5 4): Panjang jalur = 251,678012198056
* Jalur ke-15 (1 2 3 4 6 5): Panjang jalur = 371,695592859175

BestSoFar = 251,678012198056, yaitu jalur ke-2 --- DITERIMA sebagai GlobalMin

Globalmin = 251,678012198056

Iterasi ke- 2:

Tabu List:

1 1 2 3 4 5 6

2 3 2 1 4 5 6

TETANGGA (Jalur alternatif berikutnya):

* Jalur ke-1 (2 3 1 4 5 6): Panjang jalur = 334,019479980902

* Jalur ke-2 (1 2 3 4 5 6): Panjang jalur = 388,002357477953

* Jalur ke-3 (4 2 1 3 5 6): Panjang jalur = 386,212503601852

* Jalur ke-4 (5 2 1 4 3 6): Panjang jalur = 276,658948172245

* Jalur ke-5 (6 2 1 4 5 3): Panjang jalur = 249,357411984586

* Jalur ke-6 (3 1 2 4 5 6): Panjang jalur = 300,093737637908

* Jalur ke- 7 (3 4 1 2 5 6): Panjang jalur = 276,658948172245

* Jalur ke- 8 (3 5 1 4 2 6): Panjang jalur = 230,511593696957

* Jalur ke- 9 (3 6 1 4 5 2): Panjang jalur = 273,148507352641

* Jalur ke- 10 (3 2 4 1 5 6): Panjang jalur = 232,832193910427
* Jalur ke- 11 (3 2 5 4 1 6): Panjang jalur = 273,148507352641
* Jalur ke-12 (3 2 6 4 5 1): Panjang jalur = 366,3133000518
* Jalur ke-13 (3 2 1 5 4 6): Panjang jalur = 283,971832268954
* Jalur ke-14 (3 2 1 6 5 4): Panjang jalur = 388,002357477953
* Jalur ke-15 (3 2 1 4 6 5): Panjang jalur = 337,796778162

BestSoFar = 230,511593696957, yaitu jalur ke-8 --- DITERIMA sebagai GlobalMin
Globalmin = 230,511593696957

Iterasi ke- 3:

Tabu List:

1 1 2 3 4 5 6
2 3 2 1 4 5 6
3 3 5 1 4 2 6

TETANGGA (Jalur alternatif berikutnya):

* Jalur ke-1 (5 3 1 4 2 6): Panjang jalur = 333,385872505379
* Jalur ke-2 (1 5 3 4 2 6): Panjang jalur = 366,835938976854
* Jalur ke- 3 (4 5 1 3 2 6): Panjang jalur = 366,3133000518
* Jalur ke-4 (2 5 1 4 3 6): Panjang jalur = 256,12613714667
* Jalur ke-5 (6 5 1 4 2 3): Panjang jalur = 232,832193910427
* Jalur ke-6 (3 1 5 4 2 6): Panjang jalur = 279,560926612332
* Jalur ke-7 (3 4 1 5 2 6): Panjang jalur = 256,12613714667
* Jalur ke-8 (3 2 1 4 5 6): Panjang jalur = 251,678012198056
* Jalur ke-9 (3 6 1 4 2 5): Panjang jalur = 272,514899877118
* Jalur ke-10 (3 5 4 1 2 6): Panjang jalur = 249,357411984586
* Jalur ke-11 (3 5 2 4 1 6): Panjang jalur = 272,514899877118
* Jalur ke-12 (3 5 6 4 2 1): Panjang jalur = 386,212503601852
* Jalur ke-13 (3 5 1 2 4 6): Panjang jalur = 283,338224793431

* Jalur ke-14 (3 5 1 6 2 4): Panjang jalur = 366,835938976854
* Jalur ke-15 (3 5 1 4 6 2): Panjang jalur = 317,263967136425

BestSoFar = 232,832193910427, yaitu jalur ke-5 --- TIDAK DITERIMA sebagai GlobalMin
Globalmin = 230,511593696957

Iterasi ke-4:

Tabu List:
1 1 2 3 4 5 6
2 3 2 1 4 5 6
3 3 5 1 4 2 6
4 6 5 1 4 2 3

TETANGGA (Jalur alternatif berikutnya):
* Jalur ke-1 (5 6 1 4 2 3): Panjang jalur = 335,25772449315
* Jalur ke-2 (1 5 6 4 2 3): Panjang jalur = 368,000292789747
* Jalur ke-3 (4 5 1 6 2 3): Panjang jalur = 367,469546452378

* Jalur ke-4 (2 5 1 4 6 3): Panjang jalur = 256,841742733863
* Jalur ke-5 (3 5 1 4 2 6): Panjang jalur = 230,511593696957
* Jalur ke-6 (6 1 5 4 2 3): Panjang jalur = 281,432778600103
* Jalur ke-7 (6 4 1 5 2 3): Panjang jalur = 256,841742733863
* Jalur ke-8 (6 2 1 4 5 3): Panjang jalur = 249,357411984586
* Jalur ke-9 (6 3 1 4 2 5): Panjang jalur = 272,963648102817
* Jalur ke-10 (6 5 4 1 2 3): Panjang jalur = 251,678012198056
* Jalur ke-11 (6 5 2 4 1 3): Panjang jalur = 272,963648102817
* Jalur ke-12 (6 5 3 4 2 1): Panjang jalur = 387,368750002429
* Jalur ke-13 (6 5 1 2 4 3): Panjang jalur = 284,943219419707
* Jalur ke-14 (6 5 1 3 2 4): Panjang jalur = 368,000292789747
* Jalur ke-15 (6 5 1 4 3 2): Panjang jalur = 318,868961762701

BestSoFar = 249,357411984586, yaitu jalur ke-8 --- TIDAK DITERIMA sebagai GlobalMin

Globalmin = 230,511593696957

Iterasi ke- 5:

Tabu List:

1 1 2 3 4 5 6

2 3 2 1 4 5 6

3 3 5 1 4 2 6

4 6 5 1 4 2 3

5 6 2 1 4 5 3

TETANGGA (Jalur alternatif berikutnya):

* Jalur ke-1 (2 6 1 4 5 3): Panjang jalur = 333,570731755203

* Jalur ke-2 (1 2 6 4 5 3): Panjang jalur = 384,525510863906

* Jalur ke-3 (4 2 1 6 5 3): Panjang jalur = 387,368750002429

* Jalur ke-4 (5 2 1 4 6 3): Panjang jalur = 275,053953545969

* Jalur ke-5 (3 2 1 4 5 6): Panjang jalur = 251,678012198056

* Jalur ke-6 (6 1 2 4 5 3): Panjang jalur = 299,644989412209

* Jalur ke-7 (6 4 1 2 5 3): Panjang jalur = 275,053953545969
* Jalur ke-8 (6 5 1 4 2 3): Panjang jalur = 232,832193910427
* Jalur ke-9 (6 3 1 4 5 2): Panjang jalur = 271,27665536487
* Jalur ke-10 (6 2 4 1 5 3): Panjang jalur = 230,511593696957
* Jalur ke-11 (6 2 5 4 1 3): Panjang jalur = 271,27665536487
* Jalur ke-12 (6 2 3 4 5 1): Panjang jalur = 367,469546452378
* Jalur ke-13 (6 2 1 5 4 3): Panjang jalur = 283,25622668176
* Jalur ke-14 (6 2 1 3 5 4): Panjang jalur = 384,525510863906
* Jalur ke-15 (6 2 1 4 3 5): Panjang jalur = 337,081172574807

BestSoFar = 230,511593696957, yaitu jalur ke-10 --- TIDAK DITERIMA sebagai GlobalMin

Globalmin = 230,511593696957

Iterasi ke- 6:

Tabu List:

1 1 2 3 4 5 6

2 3 2 1 4 5 6

3 3 5 1 4 2 6

4 6 5 1 4 2 3

5 6 2 1 4 5 3

6 6 2 4 1 5 3

TETANGGA (Jalur alternatif berikutnya):

* Jalur ke-1 (2 6 4 1 5 3): Panjang jalur = 317,263967136425

* Jalur ke-2 (4 2 6 1 5 3): Panjang jalur = 366,835938976854

* Jalur ke-3 (1 2 4 6 5 3): Panjang jalur = 386,212503601852

* Jalur ke-4 (5 2 4 1 6 3): Panjang jalur = 272,514899877118

* Jalur ke-5 (3 2 4 1 5 6): Panjang jalur = 232,832193910427

* Jalur ke-6 (6 4 2 1 5 3): Panjang jalur = 283,338224793431

* Jalur ke-7 (6 1 4 2 5 3): Panjang jalur = 272,514899877118

* Jalur ke-8 (6 5 4 1 2 3): Panjang jalur = 251,678012198056

* Jalur ke-9 (6 3 4 1 5 2): Panjang jalur = 256,12613714667

* Jalur ke-10 (6 2 1 4 5 3): Panjang jalur = 249,357411984586
* Jalur ke-11 (6 2 5 1 4 3): Panjang jalur = 256,12613714667
* Jalur ke-12 (6 2 3 1 5 4): Panjang jalur = 366,3133000518
* Jalur ke-13 (6 2 4 5 1 3): Panjang jalur = 279,560926612332
* Jalur ke-14 (6 2 4 3 5 1): Panjang jalur = 366,835938976854
* Jalur ke-15 (6 2 4 1 3 5): Panjang jalur = 333,385872505379

BestSoFar = 232,832193910427, yaitu jalur ke-5 --- TIDAK DITERIMA sebagai GlobalMin

Globalmin = 230,511593696957

Rute terpendek adalah: 3 5 1 4 2 6 dengan jarak = 230,511593696957

c. Penerapan Algoritma TS untuk Penjadwalan Job Shop

Untuk mengaplikasikan algoritma *tabu search* ke dalam *job shop*, ada yag perlu didefinisikan terlebih dahulu:

- **Tahap pembuatan solusi awal**

Tahap ini bisa dilakukan dengan pencarian solusi dengan cara acak atau dengan cara heuristik lainnya. Pembuatan solusi *job shop* secara acak membuka peluang lebih besar

untuk dihasilkannya *graph* jadwal *job shop* yang mengandung *cycle*, terutama pada kasus uji berukuran besar, sehingga proses pembuatan jadwal awalperlu diulang beberapa kali.

- **Tahap evaluasi fungsi biaya dan penentuan lintasan kritis**

Dihitunglah nilai ES (*earliest start time*) dan LS (*lastest start time*) dari setiap operasi, menggunakan CPM. *Makespan* jadwal adalah nilai dari LS dan ES pada komputasi terakhir.

- **Tahap pembuatan neighbour baru**

Neighbourhood dari sebuah jadwal ialah himpunan jadwal baru yang dapat diperoleh dengan menerapkan fungsi transisi terhadap jadwal tersebut.

- **Tahap pengelolaan tabu list**

Kinerja algoritma *tabu search* sangat bergantung pada panjang pendeknya *tabu list*, secara empirik, ukuran *tabu list* untuk menghasilkan solusi yang baik akan semakin besar seiring dengan membesarnya ukuran masalah. Namun tidak ada aturan baku untuk menentukan panjang pendeknya *tabu list*.

- **Tahap pendefinisian kriteria aspirasi**

Aturan dasar yang digunakan dalam kriteria aspirasi pada model algoritma TS dalam penelitian ini adalah untuk tiap kandidat *move* yang merupakan anggota dari *tabu list* dihitung panjang lintasan kritis dari solusi yang

dihasilkannya. Jika panjang lintasan kritisnya (yang mewakili nilai *makespan* untuk solusi baru tersebut) lebih kecil dari solusi terbaik yang telah ditemukan sebelumnya, maka status tabu dari *move* tersebut dihapuskan dan *move* itu digunakan untuk membuat solusi berikutnya.

8.5 Kelebihan dan Kekurangan Tabu Search

Gendreau dkk (1998) menyatakan bahwa *Tabu Search* adalah pendekatan yang paling efektif untuk pemecahan masalah penentuan rute kendaraan. Kelebihan *tabu search* terletak pada struktur memori yang fleksibel. Struktur memori itu akan membolehkan pencarian terus dilakukan meskipun solusi yang diperoleh saat ini tidak ada yang lebih baik dari solusi terbaik yang telah diperoleh. Struktur memori tersebut juga mampu menjaga agar proses pencarian tidak jatuh pada lokal optimal yang pernah muncul pada pencarian sebelumnya. Adanya struktur memori fleksibel ini yang membedakan *Tabu Search* dengan *branch and bound* yang menggunakan struktur memori kaku atau *simulated annealing* yang tidak menggunakan struktur memori.

Kelebihan dari *Tabu Search* sendiri adalah ukuran dari tabu *list* yang tetap ataupun *random* dan perlunya *setting* parameter *tabu list* terkait seberapa besar ukuran *tabu list* yang optimal. Selain itu, *tabu search* memiliki daerah pencarian yang lebih luas daripada algoritma lainnya, sehingga algoritma ini dapat mencari kombinasi urutan *job* yang menghasilkan

makespan lebih kecil daripada algoritma lainnya. *Makespan* adalah jumlah waktu yang dibutuhkan untuk menyelesaikan seluruh proses pada semua *part* yang dijadwalkan mulai dari saat pemrosesan *part* pertama sampai *part* terakhir selesai diproses. Semakin kecil *makespan,* semakin cepat algoritma ini bekerja.

Tabu search bermanfaat untuk pembuatan aplikasi penjadwalan kuliah, penjadwalan mesin pabrik, simulasi algoritma *Tabu Search* sebagai protokol *routing* pada jaringan, pencarian rute terpendek *(Traveling Salesman Problem)* sehingga pegawai bisa lebih cepat dalam memberikan pelayanan, dan lain sebagainya berdasarkan Tabel 8.1.

Berdasarkan hasil studi literatur, analisis dan contoh implementasi yang sudah dijabarkan sebelumnya, dapat diambil kesimpulan sebagai berikut:

- Algoritma *Tabu Search* yang merupakan metode optimasi menggunakan *sort-term memory* untuk menghindari terjebak dalam nilai optimum lokal ternyata cukup efektif digunakan dalam pembuatan jadwal kuliah misalnya.
- Pada pengaplikasian algoritma untuk rute terpendek, sistem yang dibuat dapat meminimalkan rute pengiriman sehingga biaya transportasi yang dikeluarkan lebih sedikit dari jalur semula yang dibentuk secara acak.
- Untuk *job* kecil maupun besar, sebaiknya menggunakan algoritma *Tabu Search* karena lebih praktis daripada menjadwalkan secara manual.

- Algoritma *Tabu Search* memang memiliki *makespan* yang lebih kecil tetapi waktu untuk menjelajahi daerah pencarian ini lama. Apabila diinginkan *makespan* yang lebih kecil, maka algoritma *tabu search* merupakan pilihan yang terbaik. Namun sebagai konsekuensinya harus menyediakan waktu yang lebih lama untuk menjelajahi daerah pencarian.
- Algoritma *tabu search* merupakan metode yang fleksibel untuk mencari solusi dari sebuah permasalahan. Algoritma ini tidak akan terpaku pada sebuah solusi terbaik yang telah didapatkannya karena algoritma ini akan terus mencari solusi pada daerah solusi yang baru sampai menemukan solusi terbaik (solusi optimal) dari solusi sebelumnya dan tidak bersifat tabu (terlarang, mungkin karena *move* yang sudah ditelusuri). Apabila solusi yang paling terbaik (solusi optimal) berada pada suatu *move* yang tabu, maka algoritma ini akan menghapus status tabu tersebut dan menjadikan solusi tersebut solusi yang optimal.
- Algoritma ini sangat cocok digunakan untuk memecahkan masalah penjadwalan, seperti penjadwalan mengajar dosen dan pembagian ruangan. Algoritma ini akan menyediakan solusi-solusi penjadwalan yang tepat sehingga solusi tersebut bisa disesuaikan dengan kondisi kesiapan dari dosen untuk mengajar dan ruangan mana yang siap untuk digunakan.

Bab 9

Algoritma Koloni Semut

9.1 Pengantar

Dalam dunia pemrograman, setiap *programmer* selalu berusaha untuk mendesain algoritma secara teliti yang bertujuan agar algoritma tersebut dapat menyelesaikan suatu masalah yang ada. Masalah lain muncul ketika algoritma tersebut membutuhkan waktu *running* yang lama untuk memberikan hasil yang maksimal atau waktu *running* yang singkat tetapi hasilnya kurang presisi, dan algoritma optimasi muncul sebagai jawaban akan masalah tersebut dimana dengan algoritma optimasi kemungkinan untuk mendapatkan waktu *running* yang cepat dan hasil yang optimal dapat terwujud.

Saat ini algoritma optimasi dikelompokkan berdasarkan karakteristiknya, salah satunya bernama *swarm intelligence*. *swarm intelligence* berarti kecerdasan yang dihasilkan dari adanya tingkah laku kawanan atau kelompok. Dan algoritma koloni semut masuk dalam kategori *swarm intelligence* karena menggunakan kecerdasan tingkah laku koloni semut sebagai dasar pembuatan algoritma.

9.2 Sejarah

Berikut adalah kronologis waktu terbentuknya algoritma koloni semut.

1959 : Pierre-Paul Grass menemukan teori Stigmergy untuk menjelaskan perilaku bangunan sarang rayap.

1983 : Deneubourg dan rekan-rekannya mempelajari perilaku kolektif dari semut.

1988 : Moyson Manderick memiliki artikel tentang organisasi diri di antara semut.

1989 : karya Goss, Aron, Pasteels di Deneubourg dan perilaku kolektif semut Argentina, yang akan memberikan ide algoritma optimisasi koloni semut.

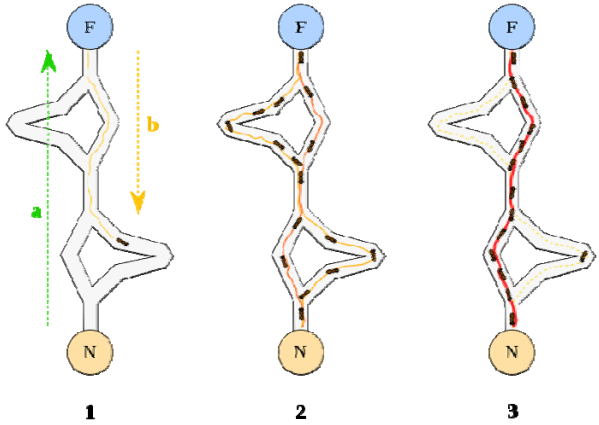
1989 : pelaksanaan model perilaku untuk makanan oleh Ebling dan rekan-rekannya.

1991 : M. Dorigo mengusulkan Sistem Ant tesis doktoralnya (yang diterbitkan pada tahun 1992 dengan V. Maniezzo dan A. Colorni). Laporan teknis diterbitkan lima tahun kemudian. Ini algoritma pertama bertujuan untuk mencari sebuah jejak optimal di dalam sebuah *graph* didasari pada perilaku semut-semut yang mencari jalan di antara koloninya dan sumber makanan.

9.3 Perilaku Semut

Ketika mencari makan maka semut-semut akan bergerak ke segala arah secara acak, kemudian apabila mereka menemukan makanan maka semut-semut tersebut akan

meninggalkan jejak berupa *pheromone*, yaitu sejenis zat kimia yang dihasilkan oleh tubuhnya yang baunya hanya dapat dikenali oleh sesama jenisnya saja dan akan menguap dalam jangka waktu tertentu, jadi semakin dekat jarak tempuh koloni dengan sumber makanan maka jejak *pheromone* akan semakin kuat dan mempunyai kerapatan tinggi dibanding dengan jejak *pheromone* dari sumber makanan yang letaknya jauh karenya yang letaknya jauh tentu membutuhkan waktu yang lebih lama untuk dapat dicapai yang akan membuat *pheromone* itu menguap sebelum semut-semut lain sempat mencium baunya. Adanya penguapan ini membantu semut untuk mengekplorasi ruang yang lebih besar dan membantu dalam menemukan jalur yang optimal di mana yang dimaksud jalur optimal adalah yang jarak antara sumber makanan dengan koloninya lebih dekat.



Gambar 9.1 Simulasi Koloni Semut

Ant Colony Optimization (ACO) atau dikenal dengan algoritma Koloni Semut adalah sebuah metodologi yang dihasilkan melalui pengamatan terhadap semut. Pada algoritma ini, semut berfungsi sebagai agen yang ditugaskan untuk mencari solusi terhadap suatu masalah optimisasi. Secara informal, ACO bekerja sebagai berikut: pertama kali, sejumlah *x* semut ditempatkan pada sejumlah *y* titik berdasarkan beberapa aturan inisialisasi (misalkan secara acak). Setiap semut membuat sebuah tur dengan menerapkan sebuah aturan transisi status secara berulang kali. Selagi membangun turnya, seekor semut juga memodifikasi jumlah *pheromone* (= feromon, sejumlah informasi yang ditinggalkan oleh semut di tempat yang dilalui dan menandai jalur tersebut) pada ruas-ruas yang dikunjunginya dengan menerapkan aturan pembaruan *pheromone* lokal. Setelah semua semut mengakhiri tur mereka, jumlah *pheromone* yang ada pada ruas-ruas dimodifikasi kembali (dengan menerapkan aturan pembaruan *pheromone* global). Dalam membuat tur, semut 'dipandu' oleh informasi heuristik (mereka lebih memilih ruas-ruas yang pendek) dan oleh informasi *pheromone*. Sebuah ruas dengan jumlah *pheromone* yang tinggi merupakan pilihan yang sangat diinginkan. Kedua aturan pembaruan *pheromone* itu dirancang agar semut cenderung untuk memberi lebih banyak *pheromone* pada ruas-ruas yang harus mereka lewati. ACO dimanfaatkan untuk menyelesaikan masalah penjadwalan, *protein folding*, *data mining* dan lain-lain.

9.4 Karakteristik Optimasi Koloni Semut

1. Aturan transisi status

   Seekor semut yang ditempatkan pada titik *t* memilih untuk menuju ke titik *v*, kemudian diberikan bilangan pecahan acak *q* dimana $0 \leq q \leq 1$, q_0 adalah sebuah parameter yaitu probabilitas semut melakukan eksplorasi pada setiap tahapan, dimana ($0 \leq q0 \leq 1$) dan $p_k\ (t,v)$ adalah probabilitas di mana semut *k* memilih untuk bergerak dari titik *t* ke titik *v*.

2. Aturan pembaruan *pheromone* lokal

   Selagi melakukan tur dan mencari tujuan (*food*), semut mengunjungi ruas-ruas dan mengubah tingkat *pheromone* pada ruas-ruas tersebut dengan menerapkan aturan pembaruan *pheromone* lokal.

3. Aturan pembaruan pheromone global

   Pada sistem ini, pembaruan *pheromon*e secara global hanya dilakukan oleh semut yang membuat tur terpendek sejak permulaan percobaan. Pada akhir sebuah iterasi, setelah semua semut menyelesaikan tur mereka, sejumlah *pheromon*e ditaruh pada ruas-ruas yang dilewati oleh seekor semut yang telah menemukan tur terbaik (ruas-ruas yang lain tidak diubah). Tingkat *pheromone* itu diperbarui dengan menerapkan aturan pembaruan *pheromone* global.

9.5 Algoritma *Ants Colony Optimization* (ACO)

Dalam mencari jalur optimal yang digunakan semut untuk menemukan tujuan, diperlukan beberapa langkah:

1. Tentukan terlebih dahulu banyaknya semut dalam proses tersebut, lalu tentukan titik awal masing-masing semut. Selanjutnya menentukan *pheromone* awal masing-masing semut.
2. Tentukan titik selanjutnya yang akan dituju, ulangi proses sampai semua titik terlewati. Jika titik yang dimaksud bukanlah titik yang akan dilalui, maka kembali ke titik sebelumnya.
3. Apabila telah mendapatkan titik yang dituju, *pheromone* masing-masing pada titik tersebut diubah. Perubahan *pheromone* tersebut dinamakan perubahan *pheromone* lokal.

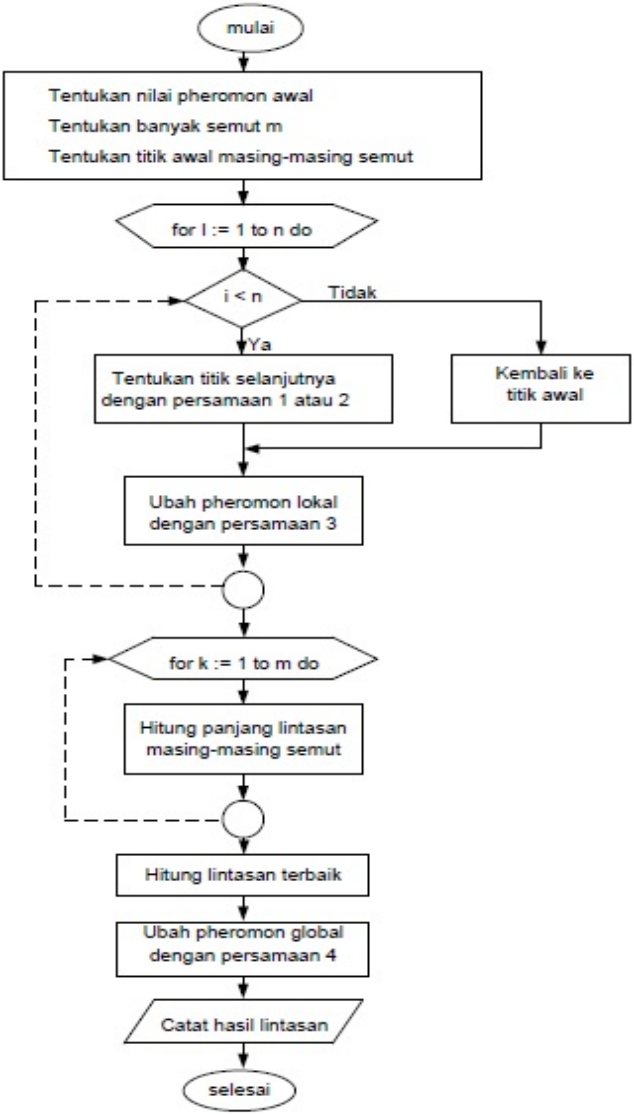
$$\Delta\tau(t,v) = \frac{1}{L_{nn}.c} \tag{9.1}$$

L_{nn} = panjang tur yang diperoleh

c = jumlah lokasi

$\Delta\tau(t,v)$ = perubahan *pheromone*

4. Hitung panjang lintasan masing-masing semut.
5. Kemudian didapatkan panjang lintasan yang minimal.
6. Ubah *pheromone* pada titik-titik yang termuat dalam lintasan tersebut.
7. Setelah semua proses telah dilalui, maka akan didapatkan lintasan dengan panjang lintasan yang minimal.



Gambar 9.2 *Flowchart* Algoritma Koloni Semut

Berikut adalah *pseudocode* dari algoritma koloni semut:

1. /* *Initialization phase* */

```
For each pair (t,v) τ(t,v):= τ0
End-for
For k:=1 to m do
      Let tk1 be the starting city for ant k
      Jk(tk1):= {1, ..., n} - rk1
      /* Jk (tk1) is the set of yet to be
visited cities for ant k in city tk1 */
      tk:= tk1
      /* tk is the city where ant k is located
*/
End-for
```

2. /* *This is the phase in which ants build their tours. The tour of ant k is stored in Tourk.* */

```
For i:=1 to n do
      If i<n then
For k:=1 to m do
      Choose the next city vk according to
Eq.(1) and Eq.(2)
      Jk(sk):= Jk (tk) - vk
      Tourk (i):=(tk ,vk)
End-for
Else
For k:=1 to m do
      /* In this cycle all the ants go back to the
initial city tk1 */
      vk := tk1
      Tourk (i):=(tk ,vk)
End-for
```

```
        End-if
        /* In this phase local updating occurs and
pheromone is updated using Eq. (4)*/
        For k:=1 to m do
                τ(tk ,vk):=(1-ρ)τ(tk ,vk)+ ρτ0
                tk := vk /* New city for ant k */
        End-for
        End-for
```

3. `/* In this phase global updating occurs and pheromone is updated */`

```
        For k:=1 to m do
                Compute Lk
                /* Lk is the length of the tour done by
        ant k */
        End-for
        Compute Lbest
        /* Update edges belonging to Lbest using Eq.
(3) */
        For each edge (t,v)
                τ(tk ,vk):=(1-α)τ( tk ,vk) + α (Lbest)-1
        End-for
```

4. `If (End_condition = True) then Print shortest of Lk`

```
        Else
        goto Phase 2
```

Bab 10

Algoritma Immune

10.1 Pengantar

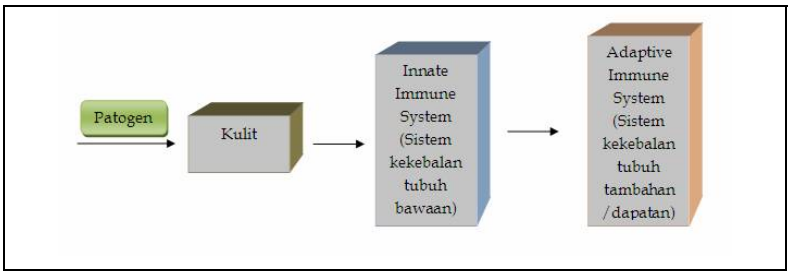
Sekarang ini banyak berkembang algoritma-algoritma yang disebut sebagai *bio-inspired algorithm* yaitu sebuah algoritma komputasi yang terinspirasi oleh mekanisme alamiah makhluk hidup seperti misalnya *neural network* yang didasari oleh mekanisme jaringan saraf, algorima genetika yang didasari dari sistem evolusi, sistem koloni semut dan lainnya. Metode-metode tersebut telah dapat diaplikasikan secara luas untuk berbagai permasalahan teknik sampai ekonomi dan keuangan. Salah satu metode dalam *bio-inspired algorithm* yang baru berkembang adalah *Artificial Immune System*.

Artificial Immune System adalah metode yang terinspirasi dari cara kerja sistem kekebalan tubuh pada mamalia. Metode ini telah diaplikasikan untuk pengenalan pola dan klasifikasi, optimasi, analisa data, keamanan komputer dan robotika.

10.2 Sistem Kekebalan Tubuh Manusia

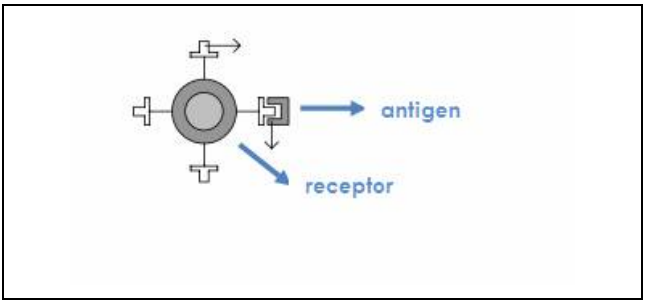
Algoritma *immune* merupakan bagian *Artificial Immune Systems* (AIS) yang terinspirasi dari proses dan mekanisme sistem kekebalan tubuh. Secara sederhana, sistem kekebalan

tubuh adalah sistem organ yang berfungsi melindungi tubuh dari ancaman *pathogen* (penyebab penyakit) dan zat beracun yang dapat mengganggu kestabilan tubuh (termasuk bakteri dan virus). Arsitekturnya seperti rangkaian lapisan yang melindungi tubuh, dengan kulit menjadi penghalang utama terhadap infeksi. Sekali *pathogen* telah memasuki tubuh, mereka ditangani dengan sistem kekebalan tubuh bawaan dan kemudian oleh sistem kekebalan tubuh tambahan. Namun, sistem kekebalan tubuh tidak bisa mendeteksi secara langsung adanya *pathogen* yang masuk ke dalam tubuh, melainkan dengan cara mendeteksi melalui bagian dari *pathogen* yang disebut dengan *antigen*. Jika terdeteksi adanya *pathogen*, sistem kekebalan tubuh bertugas untuk mengeleminasi dari tubuh. Agar proses eliminasi *pathogen* berlangsung baik dan benar, sistem kekebalan tubuh harus mampu membedakan antara *antigen* pada *pathogen* yang disebut dengan *nonself-antigen* dengan *antigen* sel-sel tubuh yang disebut dengan *self-antigen*. Sistem kekebalan tubuh ini yang diadaptasi menjadi Sistem Kekebalan Buatan (*Artificial Immune System*).



Gambar 10.1 Tingkat Pertahanan dalam Sistem Kekebalan Tubuh

Sel-sel kekebalan tubuh yang fungsinya khusus mendeteksi adanya pathogen adalah *lymphocytes*. Ada dua tipe *lymphocytes* yaitu *B-cells* dan *T-cells*. Kedua tipe *lymphocytes* mempunyai molekul pada permukaan sel-nya, yang disebut sebagai receptor, yang berfungsi untuk mengikat molekul *antigen* dari *pathogen*. *Receptor* dari *T-cells* disebut dengan TCR sedangkan receptor *B-cell* disebut dengan BCR atau biasa dikenal dengan antibodi. Proses deteksi *pathogen* oleh *B-cells* dan *T-cells* menggunakan prinsip komplemen struktur bentuk *receptor* (lihat Gambar 10.2).

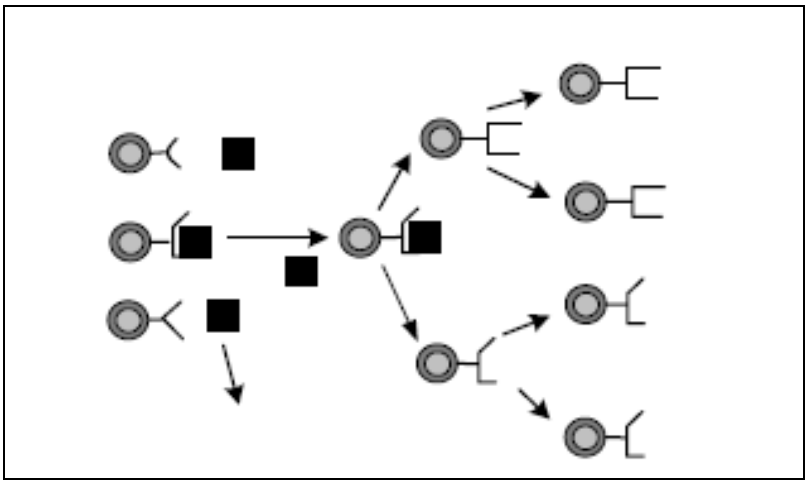


Gambar 10.2 Proses Deteksi Antigen oleh *Receptor*

Agar *receptor* mampu mengikat antigen maka *receptor* harus mempunyai struktur bentuk yang berkomplemen terhadap struktur *antigen*. Untuk menyatakan kekuatan ikatan antara *antigen* dengan *receptor* digunakan istilah *affinitty*. Semakin besar nilai *affinitty* menunjukkan semakin kuatnya ikatan antara *antigen* dan *receptor*, demikian pula sebaliknya jika nilai *affinity* kecil menunjukkan ikatan *receptor* dengan *antigen* yang lemah. Melalui proses yang dinamakan *affinitty maturation, receptor*

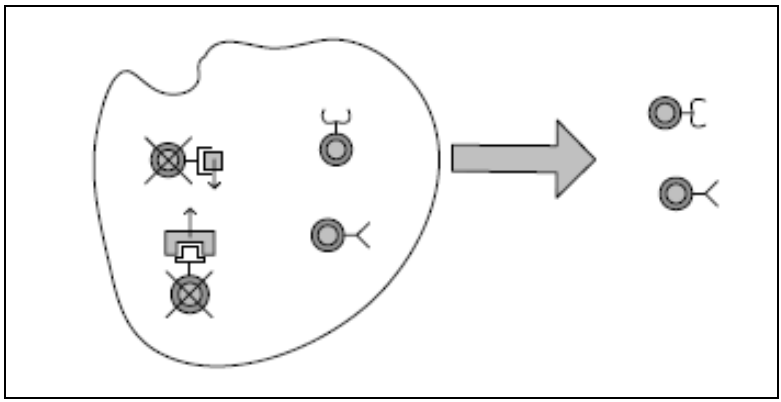
melakukan adaptasi terhadap bentuk struktur antigen dengan cara mengikat struktur, *antigen-antigen* yang telah dikenali tubuh. Dari banyak sel-sel *lymphocytes* hanya *lymphocytes* yang receptornya mampu mengikat *antigen* saja yang dipertahankan tetap hidup sedangkan yang tidak mampu mengikat antigen akan mati. Jika ada sel *lymphocytes* yang dapat mengikat *antigen* maka sel ini akan diperbanyak dengan cara *mitosis*. Karena adanya mutasi sel-sel hasil perbanyakan tidak semuanya sama persis dengan sel induk tetapi akan ada sedikit perbedaan dalam struktur bentuk *receptor*. Keseluruhan proses *affinity maturation* dan perbanyakan sel ini disebut dengan *clonal selection* (Gambar 10.3).

Selain bertugas untuk mendeteksi *pathogen*, sel *lymphocytes* juga harus bisa membedakan *self-antigen* dengan *nonself-antigen*. Metode yang digunakan berlawanan dengan *clonal selection* yaitu dengan melatihkan *lymphocytes* untuk mengikat *self-antigen*. Jika struktur *lymphocytes* berkomplemen atau dapat mengikat *self-antigen* maka *lymphocytes* tersebut akan dimatikan sedangkan *lymphocytes* yang tidak dapat mengikat *self-antigen* akan tetap dipertahankan hidup.



Gambar 10.3 Proses *Clonal Selection*

Dengan kata lain jika *affinity* dibawah suatu nilai ambang ikatan, maka *lymphocytes* tetap dipertahankan hidup sedangkan jika nilai *affinity* diatas batas ikatan maka *lymphocytes* tersebut akan dimatikan. Proses ini disebut dengan *negative selection* dan nama organ tempat terjadinya proses *negative selection* adalah *thymus* (lihat Gambar 4). *Lymphocytes* yang tetap hidup selanjutnya disebut sebagai *detector* akan masuk dalam aliran darah dan bersirkulasi ke seluruh tubuh untuk mendeteksi adanya nonself-antigen yang masuk tubuh.



Gambar 10.4 Proses *Negative Selection* dalam *Thymus*

10.3 Artificial Immune System (AIS)

AIS merupakan sistem komputasi yang diinspirasikan oleh teori imunologi, mengamati fungsi, prinsip, dan mekanisme kekebalan yang diaplikasikan dalam pemecahan masalah. Aplikasi dari AIS mencakup pengamanan dan jaringan komputerisasi, deteksi kesalahan dan penyimpangan, optimisasi, analisis data dan penjadwalan.

Forrest dkk telah menggunakan *r-contiguous bit rule* dan membandingkan masalah pengamanan sistem komputer dan memperkenalkan algoritma *negative selection*. Dasgupta dan Forrest memperkenalkan aplikasi algortima *negative selection* dari AIS untuk mendeteksi hal-hal baru pada data time series. Dasgupta dan Forrest juga memperkenalkan penggunaan AIS daam mendeteksi kerusakan mesin. Metode ini diinspirasikan dari algoritma *negative selection* yang memungkinkan untuk membedakan *self-antigen* dengan *nonself-antigen*. De castro dan

Timmis memperkenalkan penggunaan AIS dalam pengenalan pola (*pattern recognition*). De castro dan Von Zuben mengaplikasikan algoritma *clone selection* untuk menyelesaikan masalah optimasi multimodal, penugasan *pattern recognition* dan masalah perjalanan *salesman*. Hart dkk menggunakan pendekatan AIS untuk menyelesaikan masalah penjadwalan *job shop*.

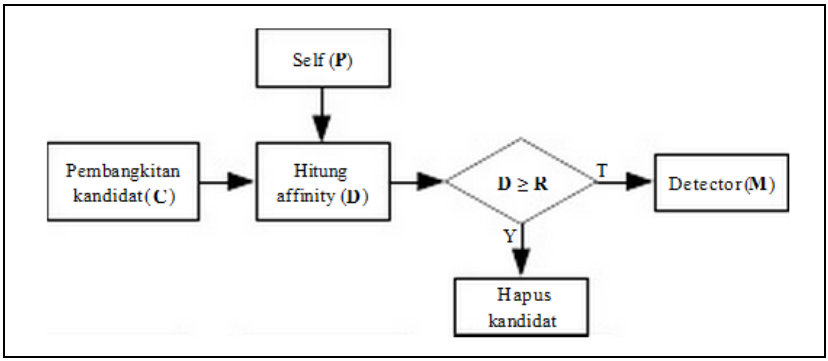
Terdapat tiga teori utama pada sistem imun yang mendasari lahirnya bidang *Artificial Immune System,* yaitu teori Jaringan Imun (*Immune Network*) oleh Jerne, teori *Negative Selection*, teori *Clonal Selection* oleh Burnet. *Immune Network Theory* menerangkan bahwa sel imun tetap beraktivitas dalam ketidakadaan patogen, bahkan antibodi dan sel imun saling mengenali dan merespon satu dengan yang lain sehingga menyebabkan kegiatan sel menjadi sangat dinamis. Pada teori ini, pengenalan pada antigen menghasilkan aktivasi jaringan sedangkan pengenalan dari antibodi yang satu terhadap yang lain menghasilkan supresi jaringan.

a. Algoritma Negative Selection

Mekanisme *negative selection* direpresentasikan sebagai proses pembangkitan *detector* yang dipergunakan untuk mendeteksi pola *nonself-antigen*. Misalkan terdapat pola *self* (P), melalui proses *negative selection* akan dihasilkan *detector* (M) yang bertugas untuk mengidentifikasi pola *nonself*. Langkah-

langkah dalam algoritma *negative selection* adalah sebagai berikut (Gambar 10.5):

1. Bangkitkan kandidat *detector* (C) secara acak dengan cara representasi yang sama dengan pola *self-antigen*, maksudnya jika komponen dalam pola *self-antigen* direpresentasikan menggunakan bilangan integer maka bilangan acak yang dibangkitkan juga dalam bentuk bilangan integer.
2. Hitung *affinity* (D) antara pola *self-antigen* P dengan pola kandidat C, jika nilai *affinity* lebih besar dari suatu nilai *affnity threshold* (R) maka hapus pola kandidat dan jika kurang dari *affinity* maka simpan pola kandidat ini sebagai pola *detector* M.



Gambar 10.5 Proses *Negative Selection*

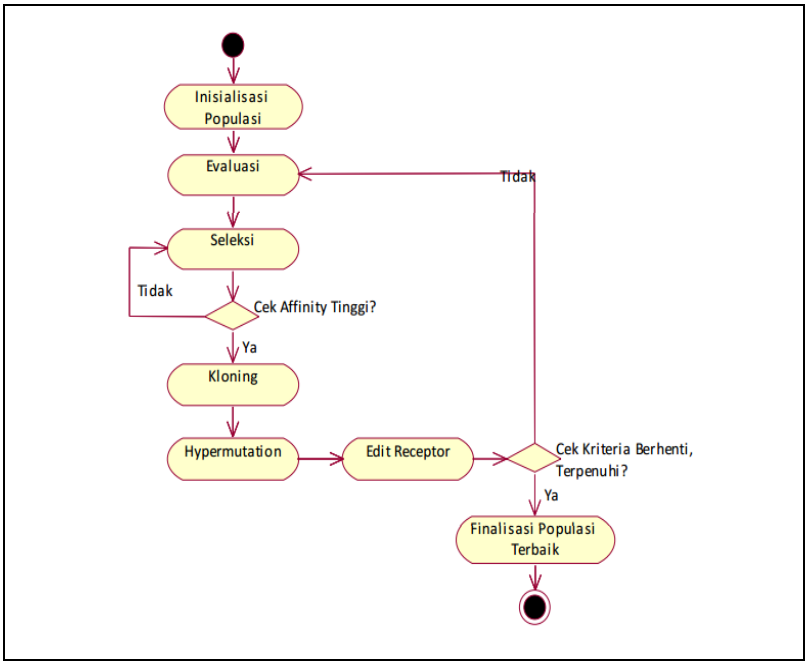
b. Algoritma Clonal Selection

Tahun 2002 DeCastro dan Von Zuben mengemukakan hasil *immune engineering* dari prinsip seleksi clonal, yang diberi nama algoritma CLONALG. Prinsip *clonal selection* adalah seluruh proses pengenalan antigen, proliferasi (perbanyakan) sel, dan diferensiasi menjadi memori sel sesuai teori seleksi clonal dari Burnet. CLONALG pada awalnya diimplementasikan untuk permasalahan learning dan optimasi. Prinsip dasar CLONALG adalah menghasilkan populasi antibodi N, masing-masing menetapkan solusi acak untuk proses optimasi. Pada setiap iterasi, beberapa antibodi yang ada terbaik dipilih, kloning, dan bermutasi untuk membangun suatu calon populasi baru. Antibodi baru kemudian dievaluasi dan persentase tertentu dari antibodi terbaik ditambahkan ke populasi asli. Prosentase antibodi terburuk dari generasi sebelumnya kemudian diganti dengan yang baru secara acak.

Tabel 10.1 Gambaran Umum Algoritma CLONALG

Tahap	Nama Tahap	Deskripsi
1	Inisialisasi	Inisialisasi populasi dari individual (N) secara random.
2	Evaluasi	Terdapat pola yang akan dikenali (P), untuk setiap pola tentukan kecocokan (*affinity*) setiap elemen dalam populasi terhadap pola tersebut.
3	Seleksi dan Kloning	Pilih sejumlah n dari N elemen dengan *affinity* tertinggi, bangkitkan *copy* dari individual ini dengan

		proposional terhadap *affinity* masing-masing dengan antigen.
4	*Hypermutation*	Mutasikan seluruh *copy* dengan kecepatan yang proposional terhadap *affinity* masing-masing dengan pola masukan
5	Editing *Receptor*	Tambahkan individual yang termutasi ke dalam populasi dan seleksi ulang sejumlah d dari individu yang matured sebagai sel memori.
6		Ulangi tahap 1-5 sampai dengan kriteria terminasi terpenuhi.



Gambar 10.6 *Flowchart* CLONALG Model *Clonal Selection*

Bab 11

Algoritma Genetika

11.1 Pengantar

Beragamnya makhluk hidup yang ada di bumi ini, terkadang atau seringkali membuat kita terkesima, bahkan kita yang merupakan spesies *homo sapiens*, merupakan bentuk kehidupan terkompleks dan menurut sebagian ahli, merupakan produk sempurna dari seleksi alam dan evolusi. Manusia (*homo sapiens*) itu sendiri memiliki perbedaan, baik secara fisik, maupun tingkah laku antara yang satu dengan yang lain. Begitu pula dengan organisme lain, walaupun secara sekilas beberapa diantara mereka terlihat serupa/sama, namun secara biologis merupakan spesies yang berbeda, bahkan secara genetika, antara satu individu dengan individu yang lain, seringkali terdapat perbedaan fisik dan tingkah laku, sekalipun berada dalam satu spesies. Menurut teori evolusi yang diungkapkan oleh Charles Darwin, keanekaragaman ini terjadi akibat adanya adaptasi yang dilakukan oleh masing-masing individu terhadap perubahan ekosistem tempat di mana mereka tinggal. Spesies yang mampu beradaptasi, akan bertahan dan meneruskan keturunan sehingga dapat memastikan kehidupan generasi spesies berikutnya, seperti

halnya buaya, komodo dan kecoa yang sudah ada sejak jutaan tahun yang lalu, dalam beberapa hal, spesies yang beradaptasi dengan lingkungan yang berbeda dari sebelumnya, dalam waktu lama, akan berevolusi menjadi spesies lain, yang pada umumnya mengarah kepada bentuk (morfologi), fungsi organ tubuh (fisiologi) dan tingkah laku yang lebih sesuai dengan ekosistem barunya, seperti halnya beruang panda yang merupakan spesies hasil evolusi dari beruang Siberia. Sementara, spesies yang tidak mampu bertahan terhadap perubahan ekosistem yang terjadi, akan punah, seperti halnya *homo neanderthalensis*, *dinosaurus* dan hewan-hewan purbakala lainnya. Hal-hal inilah yang menjadi inspirasi bagi John Holland di tahun 1975 untuk memperkenalkan **Algoritma Genetika** (AG), yang merupakan algoritma pencarian dan optimasi dari sekumpulan solusi (*search space*) yang dipilih secara acak. Gagasan ini kemudian dikembangkan oleh muridnya, David Goldberg di tahun 1989. Algoritma Genetika secara teknis adalah program komputer yang mensimulasikan proses evolusi alamiah, memakai analogi secara langsung dari kebiasaan alamiah, yaitu seleksi alam. Algoritma Genetika bekerja dengan sebuah populasi yang terdiri atas individu-individu, yang masing-masing merepresentasikan sebuah solusi bagi kemungkinan bagi persoalan yang ada. Dalam hal ini, individu dilambangkan dengan sebuah nilai *fitness* yang akan digunakan untuk mencari solusi optimal dari permasalahan yang ada. Fungsi *fitness* dibutuhkan untuk memeriksa hasil optimasi yang menandakan gambaran solusi yang sudah

dilakukan pengkodean. Selama berjalan, generasi awal (induk) harus digunakan untuk reproduksi, pindah silang dan mutasi untuk menciptakan keturunan. Algoritma Genetika yang dirancang dengan baik, akan menghasilkan solusi yang optimal bagi permasalahan yang diberikan.

Konsep Algoritma Genetika sudah dicetuskan 35 tahun yang lalu. Kini Algoritma Genetika, digunakan untuk banyak keperluan, ada sekitar lebih dari 40 aplikasi di dunia nyata yang telah menerapkan algoritma ini, di antaranya adalah:

1. Pencocokan Kata (AG sebagai algoritma pencarian),
2. Bioinformasi,
3. *Recurrent Neural Networks*,
4. Studi mengenai *Fuzzy Rule Base*,
5. Studi mengenai *Robotic Behaviour*,
6. CAD,
7. *Audio watermark detection*,
8. *Wireless Sensor*,
9. *Software Product Line Engineering*,
10. Analisa Lingual termasuk Induksi Tata Bahasa dan aspek lain dari NLP seperti *word sense disambiguation*, dan lain-lain.

11.2 Definisi Algoritma Genetika

John Holland (1975) menyatakan bahwa setiap masalah yang berbentuk adaptasi (alami maupun buatan) dapat diformulasikan ke dalam terminologi genetika. Kemudian

muridnya, Goldberg (1989) mendefinisikan algoritma ini genetika ini sebagai suatu algoritma pencarian berdasarkan pada mekanisme seleksi alam dan genetika alam.

"A genetic algorithm (GA) is a search technique used in computing to find exact or approximate solutions to optimization and search problems. Genetic algorithms are categorized as global search heuristics. Genetic algorithms are a particular class of evolutionary algorithms (EA) that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover."

Algoritma Genetika adalah algoritma yang dikembangkan dari proses pencarian solusi optimal menggunakan pencarian acak, ini terlihat pada proses pembangkitan populasi awal yang menyatakan sekumpulan solusi yang dipilih secara acak. Algoritma ini memanfaatkan proses seleksi alamiah yang dikenal dengan proses evolusi.

Algoritma Genetika (AG) adalah suatu algoritma pencarian yang berbasis pada mekanisme seleksi alam dan genetika. Algoritma genetika merupakan salah satu algoritma yang sangat tepat digunakan dalam menyelesaikan masalah optimasi yang kompleks, yang sulit dilakukan oleh metode konvensional.

Secara definitif, penulis menyimpulkan bahwasanya AG (algoritma genetika) adalah algoritma pencarian dan optimisasi yang berbentuk adaptasi, bersifat mencari kemungkinan-kemungkinan dari sekumpulan solusi yang ada, untuk mencari solusi yang paling optimal berdasarkan fungsi *fitness* yang telah

ditentukan. Metode-metode yang digunakan terinspirasi dari genetika alamiah.

Algoritma genetika memiliki perbedaan yang mendasar dengan metode pencarian solusi optimum model matematika kalkulus, perbedaan tersebut dijelaskan sebagai berikut.

Tabel 11.1 Perbedaan Algoritma Genetika dan Model Matematika Kalkulus

Algoritma Genetika	Model Matematika Kalkulus
Pencarian solusi menggunakan informasi langsung dar hasil transfer tiap-tiap parameternya ke suatu fungsi yang dapat mewakili tujuan dari proses optimasi yang sedang dilakukan	Pencarian solusi menggunakan prosedur-prosedur matematis dan prosedur-prosedur turunan
Proses pencarian solusi dilakukan pada sekumpulan titik pencarian dengan titik acuan yang sembarang	Proses pencarian solusi dilakukan pada suatu titik pencarian dengan titik acuan yang sudah ditentukan
Bersifat propabilistik	Bersifat deterministik

Secara garis besar algoritma genetika memiliki banyak kesamaan dengan mekanisme genetika alami dan seleksi alam, baik dalam tahapan prosesnya meupun definisi dari istilah-istilah atau terminologi yang digunakan. Terminologi algoritma

genetika dan genetik alami bisa dilihat sebagai berikut:

Tabel 11.2 Algoritma Genetika dan Genetika Alami

Algoritma Genetika	**Genetika Alami**
String	Kromosom
Posisi *string*	Lokus
Karakter	Gen
Nilai karakter	Alel
Struktur	Genotip
Kode struktur	fenotip

Secara alami semua organisme terdiri atas sel yang tersusun atas sekumpulan kromosom.Kromosom terbentuk dari sekumpulan *gen,* membuat suatu kesatuan yang tersusun dalam rangkaian linear. Setiap gen mempunyai letak tersendiri di dalam kromosom, disebut *lokus.* Gen tersusun dari DNA yang membawa sifat keturunan. Setiap gen menyandikan protein tertentu suatu sifat. Contoh: gen warna mata binatang dengan posisi lokus 10. Bagian tertentu dari gen di dalam *genome* disebut *genotip.* Berdasarkan sifat individu yang menunjukkan perbedaan gen dan berada pada bagian yang berbeda disebut *alel.*

Algoritma genetika didasari oleh bidang genetika natural dan ilmu komputer, maka istilah-istilah yang digunakan akan berupa campuran dari disiplin kedua ilmu tersebut. Adapun

penjelasan dari istilah-istilah yang digunakan dapat dilihat tabel berikut.

Tabel 11.3 Istilah Algoritma Genetika

No	Algoritma genetika	Penjelasan	Definisi
1	Kromosom (*string*, individual)	Solusi (pengkodean)	Struktur yang mengkodekan preskripsi yang menspesifikasikan bagaimana organisme dikonstruksikan
2	Gen-gen (bit-bit)	Bagian dari solusi	Bagian dari kromosom yang berupa sejumlah struktur individu
3	Locus	Posisi dari gen	
4	Alleles	Nilai gen	
5	Phenotype	Solusi yang diuraikan	Organisme yang dihasilkan sari sekumpulan kromosom
6	Genotype	Solusi yang disandikan	Sekumpulan kromosom-kromosom yang lengkap

Algoritma genetik merupakan teknik *search stochastic* yang berdasarkan mekanisme seleksi alam dan genetika natural. Yang membedakan algoritma genetik dengan berbagai algoritma

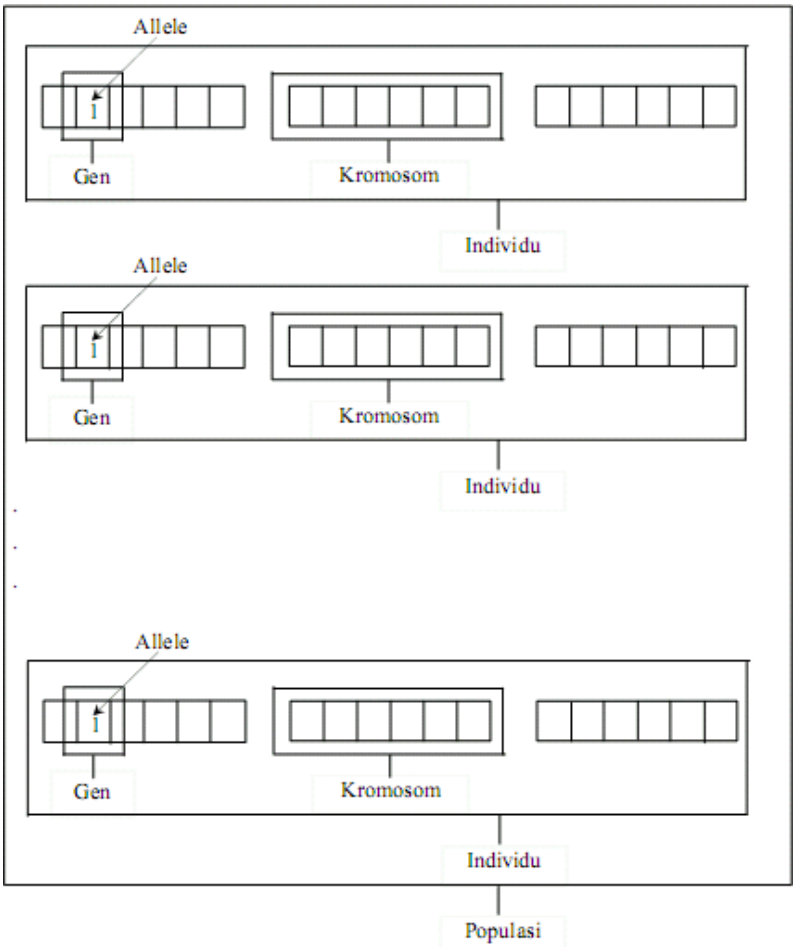
konvensional lainnya adalahbahwa algoritma genetik memulai dengan suatu himpunan penyeleksian acak awal yang disebut *populasi*.Setiap individu di dalam populasi disebut *kromosom,* yang merepresentasikan suatu penyelesaian terhadap masalah yang ditangani. Sebuah kromosom terdiri atas sebuah *string* yang berisi berbagai simbol, dan biasanya, tetapi tidak mutlak, *string* tersebut berupa sederetan bit-bit biner "0" dan "1". Sebuah kromosom tumbuh atau berkembang baik melalui berbagai iterasi yang berulang-ulang, dan disebut sebagai *generasi*. Pada setiap generasi, berbagai kromosom yang dihasilkan akan dievaluasi menggunakan suatu pengukuran *fitness*.

11.3 Definisi Penting AG

Di dalam AG, terdapat beberapa definisi penting yang digunakan, yang sebagian besar diambil dari terminologi genetika untuk memfilosofikan bagian-bagian dari permasalahan Algoritma Genetika itu sendiri.

- **Genotype (Gen)**, sebuah nilai yang menyatakan satuan dasar yang membentuk suatu arti tertentu dalam satu kesatuan gen yang dinamakan kromosom. Dalam algoritma genetika, gen ini bisa berupa nilai biner, *float*, integer maupun karakter, atau kombinatorial.
- **Allele**, sebuah nilai dari gen.
- **Kromosom**, gabungan gen-gen yang membentuk nilai tertentu.

- **Individu**, menyatakan satu nilai atau keadaan yang menyatakan salah satu solusi yang mungkin dari permasalahan yang diangkat. Individu bisa dikatakan sama dengan kromosom, yang merupakan kumpulan gen. Gen ini bisa biner, *float*, dan kombinatorial.
- **Populasi**, merupakan sekumpulan individu yang akan diproses bersama dalam satu siklus proses evolusi.
- **Generasi**, menyatakan satu satuan siklus proses evolusi di mana populasi awal dibangun secara acak sedangkan populasi selanjutnya merupakan hasil evolusi kromosom-kromosom melalui iterasi.
- **Nilai Fitness**, menyatakan seberapa baik nilai dari suatu individu atau solusi yang didapatkan. Nilai *fitness* ini yang dijadikan acuan dalam mencapai nilai optimal dalam algoritma genetika. Algoritma genetika bertujuan mencari individu dengan nilai *fitness* yang paling tinggi. Dalam TSP, karena TSP bertujuan meminimalkan jarak, maka nilai fitnessnya adalah inversi dari jarak.
- **Mutasi**, operator untuk memodifikasi kromosom.



Gambar 11.1 Istilah dalam Algoritma Genetika

11.4 Perbedaan Algoritma Genetika dengan Algoritma Konvensional

Goldberg di tahun 1989, mengemukakan bahwa Algoritma Genetika mempunyai karakteristik-karakteristik yang perlu diketahui, sehingga dapat terbedakan dari prosedur pencarian atau optimasi yang lain, yaitu:

1. AG bekerja dengan pengkodean dari himpunan solusi berdasarkan parameter yang telah ditetapkan dan bukan parameter itu sendiri.
2. AG melakukan pencarian di dalam suatu populasi dari sejumlah individu yang merupakan solusi permasalahan, bukan hanya dari sebuah individu.
3. AG merupakan informasi fungsi objektif (*fitness*), sebagai cara untuk mengevaluasi individu yang mempunyai solusi terbaik, bukan turunan dari suatu fungsi.
4. AG menggunakan aturan-aturan transisi peluang, bukan aturan-aturan deterministik.

Variabel dan parameter yang digunakan pada Algoritma Genetika adalah:

1. Fungsi *fitness* (fungsi tujuan) yang dimiliki oleh masing-masing individu untuk menentukan tingkat kesesuaian individu tersebut dengan kriteria yang ingin dicapai.
2. Populasi jumlah individu yang dilibatkan pada setiap generasi.

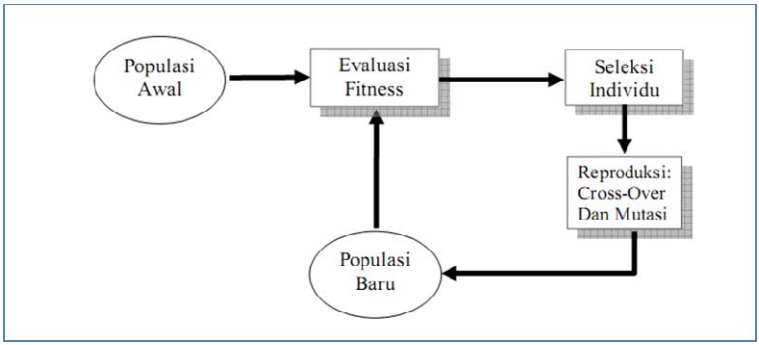
3. Probabilitas terjadinya persilangan (*crossover*) pada suatu generasi.
4. Probabilitas terjadinya mutasi pada setiap individu.
5. Jumlah generasi yang akan dibentuk yang akan menentukan lama penerapan AG.

11.5 Definisi Nilai *Fitness*

Nilai *fitness* adalah nilai yang menyatakan baik tidaknya suatu solusi (individu). Nilai *fitness* ini yang dijadikan acuan dalam mencapai nilai optimal dalam AG. AG bertujuan mencari individu dengan nilai *fitness* yang paling tinggi, dengan kata lain, semakin besar nilai *fitness* yang dihasilkan, semakin baik pula solusi yang dihasilkan. Walaupun pada mulanya semua nilai *fitness* berkemungkinan kecil (karena populasi awal dihasilkan secara random), sebagian akan lebih tinggi dari yang lain. Kromosom dengan nilai *fitness* yang tinggi, akan memberikan probabilitas yang tinggi untuk bereproduksi pada generasi selanjutnya. Sehingga untuk setiap generasi pada proses evolusi, fungsi *fitness* yang mensimulasikan seleksi alam, akan menekan populasi ke arah yang meningkat.

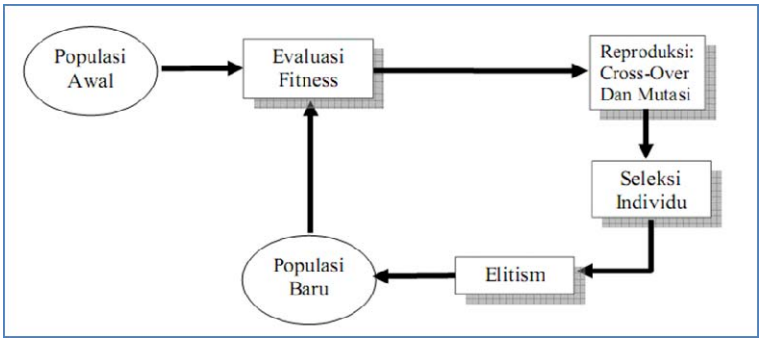
11.6 Siklus Algoritma Genetika

Untuk menjelaskan tahapan-tahapan yang dilakukan di dalam Algoritma Genetika, Goldberg di tahun 1989, membuat suatu gambaran siklus.



Gambar 11.2 Siklus Algoritma Genetika

Yang kemudian diperbarui oleh Zbigniew Michalewicz.



Gambar 11.3 Siklus Algoritma Genetika yang Diperbarui

Terdapat 6 komponen utama dalam algoritma genetika, yaitu:

a. Teknik Pengkodean

Menurut Gen dan Chang di tahun 2000, Pengkodean adalah suatu teknik menyatakan populasi awal sebagai calon solusi suatu

masalah ke dalam suatu kromosom, sebagai suatu kunci pokok persoalan ketika menggunakan AG. Gen dan Chang juga menyatakan, berdasarkan jenis simbol yang digunakan sebagai nilai suatu gen, metode pengkodean dapat diklasifikasikan sebagai berikut: pengkodean, bilangan real, bilangan bulat dan struktur data. Selain itu, kita juga dapat merepresentasikan gen dalam bentuk daftar aturan, elemen permutasi atau elemen lain untuk diimplementasikan oleh operator genetika. Dengan demikian, kromosom dapat direpresentasikan dengan menggunakan:

- *String* bit : 100101, dst,
- *Array* bilangan real : 23,65,-33,12,-77,dst,
- Elemen permutasi : E2, E10, E5, dst,
- Daftar aturan : R1, R2, R3, dst
- Struktur lainnya.

b. Nilai *Fitness*

Nilai *fitness* adalah nilai yang menyatakan baik tidaknya suatu solusi (individu). Nilai *fitness* ini yang dijadikan acuan dalam mencapai nilai optimal dalam AG. AG bertujuan mencari individu dengan nilai *fitness* yang paling tinggi, dengan kata lain, semakin besar nilai *fitness* yang dihasilkan, semakin baik pula solusi yang dihasilkan. Walaupun pada mulanya semua nilai *fitness* berkemungkinan kecil (karena populasi awal dihasilkan secara *random*), sebagian akan lebih tinggi dari yang lain. Kromosom dengan nilai *fitness* yang tinggi, akan memberikan

probabilitas yang tinggi untuk bereproduksi pada generasi selanjutnya. Sehingga untuk setiap generasi pada proses evolusi, fungsi *fitness* yang mensimulasikan seleksi alam, akan menekan populasi ke arah yang meningkat.

c. Seleksi

Seleksi digunakan untuk memilih individu-individu mana saja yang akan dipilih untuk proses kawin silang dan mutasi. Induk yang baik (nilai *fitness* baik) akan menghasilkan keturunan yang baik pula. Semakin tinggi nilai *fitness* suatu individu, semakin besar kemungkinannya untuk terpilih. Perlu diperhatikan bahwasanya tahapan seleksi sangat dipengaruhi oleh nilai *fitness*. Masing-masing individu/solusi akan mendapatkan probabilitas reproduksi yang tergantung pada nilai objektif dirinya sendiri terhadap nilai objektif dari semua individu dalam wadah seleksi tersebut. Kemampuan AG untuk memproduksi kromosom yang lebih baik secara progresif tergantung pada penekanan selektif (*selective pressure*) yang diterapkan ke populasi. Penekanan selektif dapat diterapkan dengan dua cara.

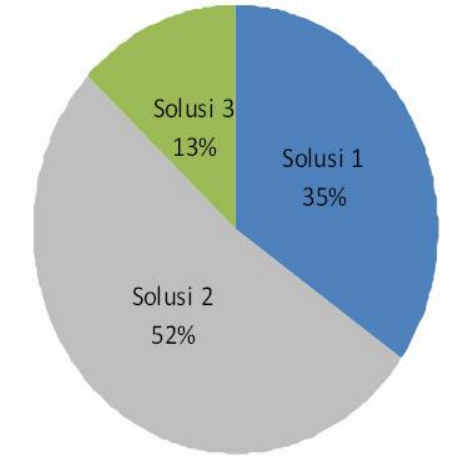
1. Membuat lebih banyak kromosom anak yang dipelihara dalam populasi dan memilih hanya kromosom-kromosom terbaik bagi generasi berikut. Walaupun induk dipilih secara acak, metode ini melakukan penekanan selektif kepada individu anak tersebut
2. Memilih induk yang lebih baik ketika membuat keturunan baru.

Bertujuan untuk memberikan kesempatan reproduksi yang lebih besar bagi anggota populasi yang paling fit. Ada beberapa metode seleksi dari induk, antara lain:

1. **Roulette wheel selection,** istilah lainnya adalah *stochastic sampling with replacement*. Individu-individu dipetakan dalam suatu segmen garis secara berurutan sedemikian hingga tiap-tiap segmen individu memiliki ukuran yang sama dengan ukuran *fitness*-nya. Sebuah bilangan *random* dibangkitkan dan individu yang memiliki segmen dalam kawasan segmen dalam kawasan bilangan *random* tersebut akan terseleksi. Proses ini berulang hingga didapatkan sejumlah individu yang diharapkan. Metode roda *roulette*, metode ini paling sering digunakan, karena mudah digunakan. Model pengacakannya seragam. Berikut merupakan contoh penentuan nilai *fitness* suatu populasi dengan 3 kromosom.

Tabel 11.4 Metode Seleksi

Solusi	Nilai *Fitness*	Persentase	Ruang Roulette
Solusi 1	8	34,78%	1-35
Solusi 2	12	52,17%	36-87
Solusi 3	3	13,04%	88-100



Gambar 11.4 Persentase Metode Seleksi

Misal dibangkitkan 3 bilangan random dari 1 sampai 100, muncul 39, 55 dan 13. 39 menunjuk individu 2, 55 menunjuk individu 2 dan 13 menunjuk individu 1.

2. **Rank-based fitness assignment**

   Populasi diurutkan menurut nilai objektifnya. Nilai fitness dari tiap-tiap individu hanya tergantung pada posisi individu tersebut dalam urutan dan tidak dipengaruhi oleh nilai objektifnya.

3. **Stochastic universal sampling**

   Memiliki nilai bias nol dan penyebaran yang minimum. Individu-individu dipetakan dalam suatu segmen garis secara berurut sedemikian hingga tiap-tiap segmen individu memiliki ukuran yang sama dengan ukuran fitnessnya

seperti halnya pada seleksi roda *roulette*. Kemudian diberikan sejumlah *pointer* sebanyak individu yang ingin diseleksi pada garis tersebut. Andaikan N adalah jumlah individu yang akan diseleksi, maka jarak antar *pointer* adalah 1/N dan posisi *pointer* pertama diberikan secara acak pada *range* [1, 1/N].

4. Local selection

Setiap individu yang berada di dalam *constraint* tertentu disebut dengan nama lingkungan lokal. Interaksi antar individu hanya dilakukan di dalam wilayah tersebut. Lingkungan tersebut ditetapkan sebagai struktur dimana populasi tersebut terdistribusi. Lingkungan tersebut juga dapat dipandang sebagai kelompok pasangan-pasangan yang potensial. Langkah pertama adalah menyeleksi separuh pertama dari populasi yang berpasangan secara *random*. Kemudian lingkungan baru tersebut diberikan pada setiap individu yang terseleksi. Jarak antara individu dengan struktur akan sangat menentukan ukuran lingkungan. Individu yang terdapat dalam lingkungan dengan ukuran yang lebih kecil, akan lebih terisolasi dibandingkan dengan individu yang terletak pada lingkungan dengan ukuran yang lebih besar.

5. Truncation selection

Seleksinya adalah buatan. Digunakan oleh populasi yang jumlahnya sangat besar. Individu-individu diurutkan berdasarkan nilai fitnessnya. Hanya individu yang terbaik

saja yang akan diseleksi sebagai induk. Parameter yang digunakan adalah suatu nilai ambang *trunk* yang mengindikasikan ukuran populasi yang akan diseleksi sebagai induk yang berkisar antara 50% - 10%. Individu-individu yang ada di bawah nilai ambang ini tidak akan menghasilkan keturunan.

6. **Tournament selection**

   Ditetapkan suatu nilai tour untuk individu-individu yang dipilih secara random dari suatu populasi. Individu-individu yang terbaik dalam kelompok ini akan diseleksi sebagai induk. Parameter yang digunakan adalah ukuran tour yang

11.7 Mutasi

Mutasi adalah tahapan mengubah nilai dari satu atau beberapa gen dalam suatu kromosom. Operator ini bertujuan untuk menggantikan gen yang hilang dari populasi akibat proses seleksi yang memungkinkan munculnya kembali gen yang tidak muncul pada inisialisasi populasi.

Operasi *crossover* yang dilakukan pada kromosom dengan tujuan untuk memperoleh kromosom-kromosom baru sebagai kandidat solusi pada generasi mendatang dengan nilai *fitness* yang baik, dan lama kelamaan menuju solusi optimum yang diinginkan. Namun, jika di dalam proses pemilihan kromosom-kromosom cenderung terus pada kromosom yang memiliki nilai *fitness* yang tinggi saja, konvergensi prematur (mencapai solusi optimal yang bersifat lokal) sangat mudah terjadi. Untuk

menghindari terjadinya konvergensi prematur tersebut dan tetap menjaga perbedaan (*diversity*) kromosom-kromosom dalam populasi, selain melakukan penekanan selektif yang lebih efisien, kita dapat pula menerapkan operasi mutasi. Operasi mutasi dilakukan dengan mutasi kromosom anak dengan menambahkan nilai random yang sangat kecil, dengan probabilitas yang rendah. Peluang mutasi (p_m) didefinisikan sebagai persentase dari jumlah total gen pada populasi yang mengalami mutasi. Peluang mutasi ini, mengendalikan banyaknya gen baru yang akan dimunculkan untuk evaluasi. Bila peluang mutasi terlalu kecil, akan ada banyak gen potensial yang tidak dievaluasi. Bila peluang mutasi terlalu besar, maka akan terjadi banyak gangguan yang menyebabkan anak akan kehilangan kemiripan dengan induknya. Beberapa jenis mutasi yang dikenal di antaranya:

1. Mutasi dalam Pengkodean Biner, adalah operasi yang sederhana, karena kita hanya menginversi nilai bit pada posisi tertentu yang dipilih secara acak atau terencana pada kromosom.

Tabel 11.5 Mutasi Pengkodean Biner

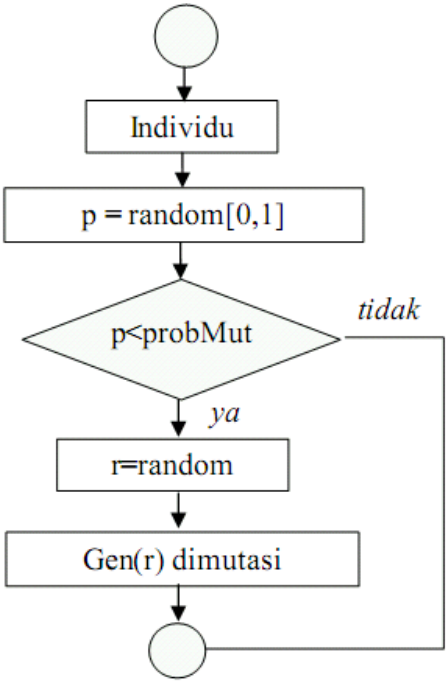
Kromosom Sebelum Mutasi	1 0 0 1 0 ***1*** 1 1
Kromosom Sesudah Mutasi	1 0 0 1 0 ***0*** 1 1

2. Mutasi dalam Pengkodean Permutasi, memiliki proses berbeda dengan mutasi dalam pengkodean biner. Untuk pengkodean permutasi, umumnya cara yang dilakukan

adalah menukarkan nilai antara dua gen dari kromosom yang dipilih secara acak.

Tabel 11.6 Mutasi Pengkodean Permutasi

Kromosom Sebelum Mutasi	1 2 4 ***3*** 5 6 7 ***9*** 8
Kromosom Sesudah Mutasi	1 2 4 ***9*** 5 6 7 ***3*** 8



Gambar 11.5 *Flowchart* Mutasi

3. Mutasi dalam Pengkodean Nilai, memiliki penerapan yang tergantung kepada jenis nilai yang digunakan. Caranya adalah memilih sembarang posisi gen pada kromosom. Nilai yang ada tersebut, kemudian ditambahkan atau dikurangkan dengan suatu nilai kecil yang diambil secara acak.

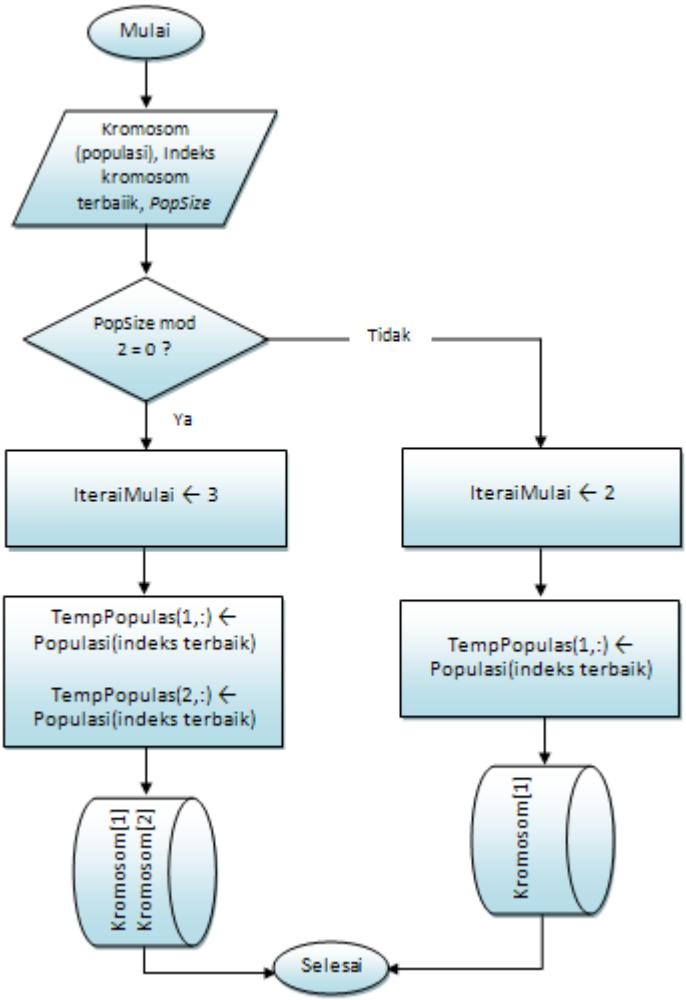
Tabel 11.7 Mutasi Pengkodean Nilai

Kromosom Sebelum Mutasi	1,12 7,10	2,45 5,66	4,11	***3,34***
Kromosom Sesudah Mutasi	1,12 7,10	2,45 5,66	4,11	***3,44***

1. Elitisme

Karena seleksi dilakukan secara acak, maka tidak ada jaminan bahwa suatu individu bernilai *fitness* tertinggi akan selalu terpilih. Kalaupun individu bernilai *fitness* tertinggi dipilih, mungkin saja individu tersebut akan rusak karena proses pindah silang. Untuk menjaga agar individu bernilai *fitness* tertinggi tersebut tidak hilang selama evolusi, perlu dibuat satu atau dua kopinya. Prosedur ini dikenal sebagai elitisme.Prosedur ini hanya digunakan pada AG berjenis *generational replacement. Input* pada prosedur ini adalah *populasi,* indeks kromosm terbaik dan ukuran populasi *PopSize*. Apabila *PopSize* bernilai genap, maka variabel Iterasi Mulai diberi nilai 3 dan kromosom terbaik akan dikopi sebanyak 2 kali yang masing-masing disimpan pada

variabel TempPopulasi. Namun jika *PopSize* bernilai ganjil maka variabel Iterasi Mulai diberi nilai 2 dan kromosom terbaik dikopi sebanyak 1 kali yag disimpan pada variabel TempPopulasi. Aturan pengkopian ini dilakukan megingat akan dilakukannya perkawinan silang antara dua induk (sepasang) dan nilai variabel IterasiMulai akan digunakan sebagai nilai awal *counter* atau iterasi pada proses pindah silang dan mutasi. Kromosom terbaik yang telah dikopi ini akan tetap dipilih sebagai salah satu kandidat induk yang akan dipindah silangkan. Nilai *fitness* terbaik ini akan dibandingkan dengan nilai *fitness* kromosom-kromosom generasi berikutnya hasil pindah silang dan mutasi. Kromosom hasil pengkopian ini pastinya akan disertakan lagi pada generasi berikutnya.



Gambar 11.6 *Flowchart* Elitisme

7. Inversion

Inversion digunakan bila representasi kromosom berbentuk *integer*atau karakter.Biasanya proses *inversion* dilakukan pada *offspring* hasil *crossover*.Pada *inversion* pun dikenal sebuah parameter yaitu *inversion probability* atau *inversion rate. Inversion probability* adalah probabilitas dari banyaknya proses *inversion* yang terjadi pada kromosom. Proses *inversion* adalah sebagai berikut. Mula-mula tentukan dua titik potong pada kromosom secara acak. Kemudian tukarposisi gen-gen yang berada diantara kedua titik potong tersebut. Penukaran posisi gen dilakukan dengan membalik urutan posisi gen diantara kedua titik potong tersebut.

A = 1	2	3	4	5	6	7	8
A' = 1	2	6	5	4	3	7	8

11.8 Operasional AG

Setelah menentukan teknik pengkodean untuk memecahkan suatu masalah dengan AG, maka tahap berikutnya adalah menggenerasikan populasi awal. Kemudian melakukan regenerasi dengan iterasi terhadap operator-operator AG, yakni seleksi, *crossover* dan mutasi.

a. Generasi Populasi Awal

Penggenerasian/pembangkitan populasi awal merupakan proses pembangkitan individu secara *random*/acak atau melalui prosedur tertentu. Ukuran untuk populasi tergantung pada masalah yang akan diselesaikan dan jenis operator genetika yang akan diterapkan. Syarat-syarat yang harus dipenuhi untuk menunjukkan suatu solusi harus benar-benar diperhatikan dalam pembangkitan setiap individunya. Ada beberapa teknik dalam pembangkitan populasi awal, di antaranya:

1. Generator bilangan acak, adalah membangkitkan bilangan secara acak untuk nilai setiap gen sesuai dengan representasi kromosom yang digunakan.
2. Memasukkan nilai awal tertentu ke dalam gen.
3. Permutasi Gen.

b. Operator Genetika

Algoritma Genetika adalah proses pencarian yang heuristik dan acak, sehingga penekanan pemilihan operator yang digunakan sangat menentukan keberhasilan AG dalam menemukan solusi terbaik terhadap suatu masalah yang diberikan. Operator genetika digunakan setelah proses evaluasi tahap pertama untuk membentuk suatu populasi baru dari generasi awal. Operator-operator tersebut adalah seleksi, reproduksi (*crossover* dan mutasi) dan terminasi.

Ada 2 operator genetika:

- Operator untuk melakukan rekombinasi, yang terdiri atas:

- Rekombinasi bernilai real
    1. Rekombinasi diskrit
    2. Rekombinasi *intermediate*
    3. Rekombinasi garis
    4. Rekombinasi garis yang diperluas
- Rekombinasi bernilai biner (*Crossover*)
    1. *Crossover* satu titik
    2. *Crossover* banyak titik
    3. *Crossover* seragam

➢ Mutasi
- Mutasi bernilai real
- Mutasi bernilai biner

c. Reproduksi

Langkah selanjutnya adalah membangkitkan generasi kedua dari solusi melalui operator seleksi dan mutasi. Untuk setiap solusi yang dihasilkan, sepasang "induk" solusi dipilih untuk reproduksi, dari populasi yang ada sebelumnya. Dengan menghasilkan suatu solusi "anak" baru menggunakan metode *crossover* dan mutasi, solusi baru yang diciptakan tersebut akan memiliki banyak kesamaan karakteristik dengan "induk"nya. "Induk-induk" baru dipilih lagi dari setiap "anak" dan proses berlanjut hingga suatu populasi solusi dengan ukuran generasi yang tepat telah dibangkitkan. Walaupun metode reproduksi yang berdasarkankan kepada "induk" lebih cenderung karena terinspirasi oleh biologi, beberapa penelitian menyarankan agar

lebih dari 2 “induk” digunakan untuk mereproduksi kromosom berkualitas bagus. Proses ini secara final menghasilkan generasi dengan kromosom yang berbeda dengan generasi awal. Umumnya, rata-rata *fitness* akan meningkat di tiap generasi barunya.

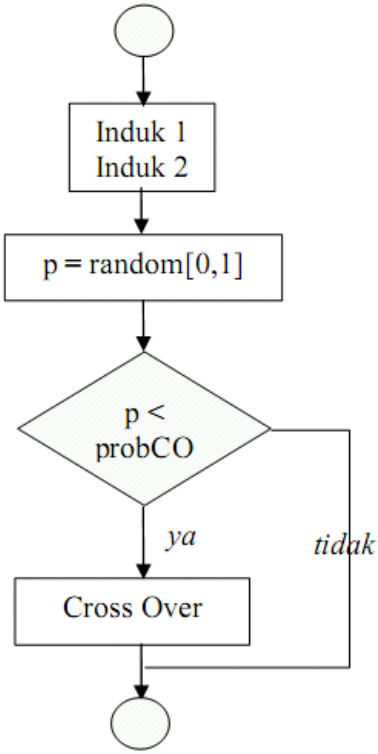
d. Crossover

Crossover (perkawinan silang) pada dasarnya bertujuan menambah keanekaragaman solusi dalam satu populasi dengan penyilangan antar solusi yang diperoleh dari populasi sebelumnya. Dengan kata lain, *crossover* menghasilkan titik solusi baru dalam ruang pencarian yang siap diuji. Tidak semua pasangan induk mengalami proses *crossover*, banyaknya pasangan induk yang mengalami *crossover* ditentukan dengan nilai probabilitas *crossover*. Jika pindah silang tidak dilakukan, berarti nilai dari induk akan diturunkan ke anaknya.

Beberapa jenis *crossover* yang dikenal adalah:

1. *Crossover* satu titik, dilakukan dengan memisahkan suatu *string* menjadi 2 bagian dan selanjutnya salah satu bagian dipertukarkan dengan salah satu bagian dari *string* lain yang telah dipisahkan dengan cara yang sama. *Crossover* satu titik dan banyak titik biasanya dipakai untuk representasi kromosom dalam biner.

Kromosom induk 1	0011 0101
Kromosom induk 2	1010 0000
Keturunan 1	0011 0000
Keturunan 2	1010 0101



Gambar 11.7 *Flowchart* Crossover

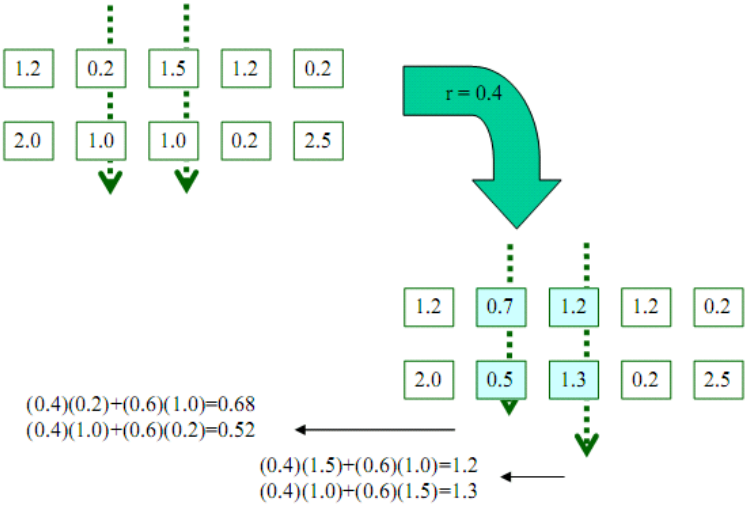
2. *Crossover* banyak titik, proses *crossover* ini dilakukan dengan memilih dua atau lebih titik *crossover* kromosom, keturunan kemudian dibentuk dengan barisan bit dari awal kromosom sampai titik *crossover* pertama disalin dari induk pertama, bagian dari titik *crossover* pertama dan kedua disalin dari orangtua kedua, kemudian selebihnya disalin dari orangtua pertama lagi.

Kromosom induk 1	01 001 000
Kromosom induk 2	11 111 101
Keturunan 1	01 111 000
Keturunan 2	11 001 101

3. *Crossover* aritmatika, umumnya digunakan untuk kromosom dengan representasi nilai pecahan. *Crossover* ini dilakukan dengan membangkitkan bilangan acak r yakni, 0<r<1. Ditentukan pula, posisi gen yang dilakukan *crossover* menggunakan bilangan *random*. Nilai baru pada gen anak mengikuti rumus berikut:

$$x_1'(k) = r.x_1(k) + (1-r).x_2(k) \quad (11.1)$$

$$x_2'(k) = r.x_2(k) + (1-r).x_1(k) \quad (11.2)$$



Gambar 11.8 *Crossover*

4. *Terminasi*

   Operator-operator AG tersebut akan terus dilakukan secara iteratif, hingga mencapai suatu kondisi terminasi, di mana AG berhenti beriterasi. Berikut merupakan kondisi terminasi yang umumnya digunakan:

   1. Solusi yang ditemukan telah mencapai kriteria minimum.
   2. Tercapainya jumlah generasi yang telah direncanakan sebelumnya.
   3. Tercapainya budget/anggaran (waktu komputasi/uang) yang telah dianggarkan.
   4. Urutan tertinggi dari *fitness* suatu solusi sedang mencapai atau telah mencapai kondisi di mana tingkat

diversifikasi rendah, sehingga iterasi-iterasi berikutnya tidak akan memproduksi hasil yang lebih baik.

5. Inspeksi manual dari *programmer*.

11.9 Kendali Parameter Algoritma Genetika

Kendali parameter AG penting untuk mengatur operator-operator seleksi, yang berguna untuk menentukan performa kinerja AG dalam memecahkan suatu masalah. Terdapat dua parameter dasar di dalam AG, yakni probabilitas *crossover* (Pc) dan probabilitas mutasi (Pm). Probabilitas *crossover* menyatakan seberapa sering proses *crossover* akan terjadi antara dua kromosom induk. Jika tidak terjadi *crossover*, satu induk dipilih secara random dengan probabilitas yang sama dan diduplikasi menjadi anak. Jika terjadi *crossover*, keturunan dibuat dari bagian-bagian kromosom induk. Jika probabilitas *crossover* 100%, maka keseluruhan keturunan dibuat dengan *crossover*. Jika probabilitas *crossover* 0%, maka seluruh generasi baru dibuat dari salinan kromosom-kromosom populasi lama, hal ini belum tentu menghasilkan populasi yang sama dengan populasi sebelumnya karena adanya penekanan selektif. Para ahli mengungkapkan bahwasanya angka probabilitas *crossover* sebaiknya cukup tinggi minimal 60%. Probabilitas mutasi menyatakan seberapa sering bagian-bagian kromosom akan dimutasikan. Jika tidak ada mutasi, keturunan diambil/disalin langsung setelah *crossover* tanpa perubahan. Jika mutasi dilakukan, bagian-bagian kromosom diubah. Jika probabilitas

mutasi 100%, keseluruhan kromosom akan berubah. Jika probabilitas mutasi 0%, tidak ada kromosom yang berubah. Umumnya, probabilitas mutasi di-*set* dengan angka kecil, yakni angka/allele=1/(panjang kromosom) dengan tujuan mendapatkan rata-rata satu mutasi per kromosom. Para ahli menyarankan probabilitas mutasi adalah antara 0,5% sampai dengan 1%, untuk menghindari konvergensi prematur. Parameter penting lainnya adalah banyaknya kromosom dalam satu populasi. Jika terlalu sedikit, AG mempunyai kemungkinan yang sedikit untuk melakukan *crossover* dan hanya sebagian kecil dari ruang pencarian yang dieksplorasi. Jika terlalu banyak, AG cenderung menjadi lambat dalam menemukan solusi. Ukuran yang sering digunakan adalah 20 sampai dengan 30, walaupun ada juga yang menggunakan 50 sampai dengan 100. Beberapa penelitian menunjukkan bahwa banyaknya kromosom tergantung dari jenis pengkodean, yaitu ukuran barisan yang dikodekan, bila ukuran kromosom 32 bit, maka ukuran populasinya 32.

11.10 Kritik terhadap Algoritma Genetika

Berikut merupakan kritik-kritik tentang Algoritma Genetika dibandingkan dengan algoritma pencarian/optimisasi lainnya:

1. Evaluasi oleh fungsi *fitness* yang berulang untuk masalah-masalah rumit sering menjadi bagian yang menghambat dan membatasi algoritma-algoritma evolusioner buatan. Menemukan solusi optimal di dalam

masalah dimensional yang sangat kompleks, masalah-masalah multimodal sering membutuhkan evaluasi-evaluasi fungsi *fitness* yang mahal. Dalam permasalahan dunia nyata, seperti masalah optimisasi struktural, satu fungsi evaluasi membutuhkan waktu beberapa jam untuk menyelesaikan simulasi.

2. Perbandingan nilai *fitness*, antara mana yang lebih baik, hanya berlaku bagi solusi-solusi yang dipopulasikan, sehingga kondisi terminasi terkadang tidak jelas.
3. Pada banyak masalah, AG cenderung mencapai konvergensi prematur, yakni mencapai optimasi lokal daripada optimasi global. AG tak dapat mengenal perbedaan tersebut. Hal ini bisa dicegah dengan menggunakan fungsi *fitness* yang berbeda, menaikkan jumlah rasio mutasi atau dengan menggunakan teknik seleksi yang menjaga diversifikasi solusi-solusi di dalam populasi.
4. AG sulit beroperasi pada kumpulan data yang dinamis, karena gen-gen mulai berkonvergensi lebih awal menuju solusi yang tidak lagi valid untuk data-data berikutnya. AG tak bisa secara efektif menyelesaikan masalah-masalah dimana hanya ukuran *fitness* adalah ukuran antara benar/salah, karena tidak ada cara untuk mengkonvergensi solusi.

5. Untuk tujuan optimisasi tertentu, algoritma optimisasi lainnya mungkin menemukan solusi yang lebih baik dibandingkan AG.

11.11 *Word Matching* dengan Algoritma Genetika

Sebuah kata ditentukan sebagai target, misalnya: "HELLO". Bila setiap huruf diberi nilai dengan nilai urut alfabet, maka targetnya bisa dinyatakan sebagai besaran numerik:

Target=[8 5 12 12 15]

Komputer akan membangkitkan kata dengan jumlah huruf yang sama dengan target secara acak, terus-menerus hingga diperoleh kata yang sama dengan kata target. Misal kata yang muncul pertama kali adalah:

RGAFD = 18 7 1 6 4

ZYAVE = 26 25 1 22 5

ERTLO = 5 18 20 12 15

ATVBC = 1 20 22 2 3

DTZQL = 4 20 26 17 12

GEQCT = 7 5 17 3 20

Selanjutnya kita perhatikan nilai *fitness* dari permasalahan ini, Nilai *fitness* adalah inversi dari perbedaan antara nilai kata yang muncul (individu) dan target yang ditentukan. Dalam hal ini, kita bandingkan *string* yang muncul dengan *string* yang menjadi target. Salah satu *string* awal yang muncul adalah RGAFD dan targetnya HELLO. Maka, nilai perbedaannya:

$g_i - t_i$

= |18-8| + |7-5| + |1-12| + |6-12| + |4-15|

= 10+2+11+6+11 = 40

➔ *Fitness* = (26)(5) - 40 = 130-40 = 90

Dengan rumus:

Fitness (k) = (jumlah gen * 26) – $\sum_i^n = g_i t_i$

g_i adalah gen ke *i* dari individu

t_i adalah target ke *i*

Setelah melakukan pengujian nilai *fitness*, didapati masing-masing nilai *fitness* solusi-solusi yang muncul sebagai berikut:

RGAFD = 18 7 1 6 4 *Fitness*: 90

ZYAVE = 26 25 1 22 5 *Fitness*: 61

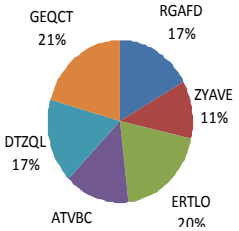
ERTLO = 5 18 20 12 15 *Fitness*: 106

ATVBC = 1 20 22 2 3 *Fitness*: 76

DTZQL = 4 20 26 17 12 *Fitness*: 89

GEQCT = 7 5 17 3 20 *Fitness*: 110

Kemudian diseleksi untuk menentukan orangtua yang dipilih untuk melakukan *crossover* dan mutasi, dalam hal ini dipakai metode roda *roulette*. Total nilai *fitness* adalah 585, maka setelah ditentukan bagian roda masing-masing solusi, ilustrasi roda *roulette* menjadi seperti berikut:



Gambar 11.9 Roda *Roulette*

Setelah dilakukan pemutaran roda *roulette* tersebut, terpilih 6 solusi:

ERTLO

GEQCT

ERTLO

GEQCT

DTZQL

GEQCT

Kemudian dilakukan penentuan probabilitas *crossover*, yakni P_c=0,9, dan nilai *random* untuk masing masing p adalah:

ERTLO-GEQCT = 0,92

ERTLO-GEQCT = 0,87

DTZQL-GEQCT = 0,6

Bila $p<P_c$, maka *crossover* dilakukan. Kemudian salah satu *crossover* yang dilakukan adalah:

D T Z Q L *fitness*: 89
G E Q C T *fitness*: 110

Menjadi:
D T Q C T *fitness*: 92
G E Z Q L *fitness*: 107

Setelah operasi *crossover* selesai, dilakukanlah operasi mutasi, dengan P_m=0,2%, kemudian untuk masing-masing nilai p adalah:
ERTLO = 0,19
GEQCT = 0,42
DTQCT = 0,52
GEZQL = 0,13
DTZCT = 0,8
GEQQL = 0,2

Bila $p<P_m$, maka mutasi dilakukan. Kemudian salah satu mutasi yang dilakukan adalah:

ERTLO *fitness*: 106
dengan titik mutasi yakni 1 dan nilai geser adalah +5 menjadi
JRTLO *fitness*: 113

Setelah iterasi pertama dilakukan didapati generasi kedua yaitu:
JRTLO *fitness*: 113

GEQCT *fitness*: 110
DTQCT *fitness*: 92
GEZOL *fitness*: 109
DTZCT *fitness*: 83
GEQQL *fitness*: 111

Proses operasi seleksi, *crossover* dan mutasi seperti di atas dilakukan terus menerus secara iteratif, hingga kondisi terminasi terpenuhi. Diharapkan di setiap generasi nilai *fitness* akan bertambah.

Bab 12

Jaringan Saraf Tiruan

12.1 Definisi Jaringan Saraf Tiruan

> *"Sebuah jaringan saraf adalah sebuah prosesor didistribusikan massal paralel yang terdiri dari pengolahan sederhana unit, yang mana juga memiliki kecenderungan alami untuk menyimpan pengalaman pengetahuan dan membuatnya tersedia untuk digunakan"* (Simon Haykin, 1999).

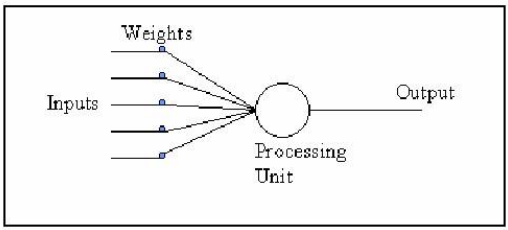
Sedangkan pengertian lain adalah "Salah satu bagian ilmu komputer yang membuat agar komputer dapat melakukan pekerjaan seperti dan sebaik yang dilakukan oleh manusia" (Kusumadewi, 2003).

Istilah buatan disebabkan jaringan saraf tiruan (JST) diimplementasikan menggunakan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama proses pembelajaran. Menurut Siswanto "JST merupakan sistem pengolahan informasi yang didasari filosofi struktur perilaku saraf makhluk hidup" .

JST bekerja meniru kerja otak manusia dari sudut: 1) Pengetahuan yang diperoleh *network* dari lingkungan, melalui suatu proses pembelajaran. 2) Kekuatan koneksi antar unit yang

disebut *synaptic weight*, berfungsi untuk menyimpan pengetahuan yang telah diperoleh dari jaringan tersebut. Karakteristik lain yang penting dari JST, yang sama dengan sistem saraf biologis adalah toleransi kesalahan.

Sistem saraf biologis memberikan toleransi kesalahan dalam mengenali banyak sinyal input yang agak berbeda dengan sembarang sinyal *input* lain yang pernah dilihat sebelumnya. Ciri utama yang dimiliki JST adalah kemampuan untuk belajar, diartikan sebagai proses penyesuaian parameter berbobot karena *output* yang diinginkan tergantung pada harga pembobotan interkoneksi yang dimiliki sel. Proses ini akan dihentikan, jika nilai kesalahan sudah danggap cukup kecil untuk semua pasangan data latihan. Jaringan yang sedang melakukan proses belajar disebut berada dalam tahap pelatihan, baru setelah tahap ini selesai terdapat tahap pengujian suatu objek. Secara mendasar, sistem pembelajaran merupakan proses penambahan pengetahuan pada *neural network* (NN) yang sifatnya kontinuitas sehingga pada saat digunakan pengetahuan tersebut akan dieksploitasikan secara maksimal dalam mengenali suatu objek. Neuron adalah bagian dasar dari pemrosesan suatu NN.



Gambar 12.1 Bentuk Dasar Neuron

- *Input* merupakan masukan yang digunakan baik saat pembelajaran maupun dalam mengenali suatu objek.
- *Weight*, beban yang selalu berubah setiap kali diberikan *input* sebagai proses pembelajaran.
- *Processing Unit* merupakan tempat berlangsungnya proses pengenalan suatu objek berdasarkan pembebanan yang diberikan.
- *Output*, keluaran dari hasil pengenalan suatu objek.

12.2 Keuntungan Penggunaan *Neural Network*

- ❖ Perangkat yang mampu untuk mengenali suatu objek secara non-linear.
- ❖ Mempermudah pemetaan *input* menjadi suatu hasil tanpa mengetahui proses sebenarnya.
- ❖ Mampu melakukan pengadaptasian terhadap pengenalan suatu objek.
- ❖ Perangkat yang memiliki toleransi terhadap suatu kesalahan dalam pengenalan suatu objek.
- ❖ *Neural Network* mampu diimplementasikan pada suatu *hardware* atau perangkat keras.
- ❖ Perangkat yang mampu diimplementasikan secara paralel.

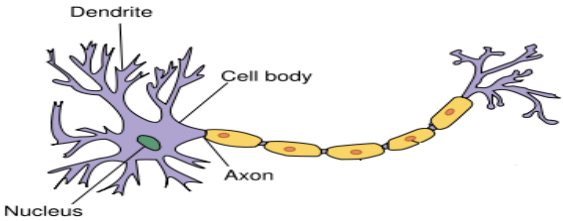
Jaringan saraf tiruan juga didefinisikan sebagai suatu sistem pemrosesan informasi yang mempunyai karakteristik menyerupai jaringan saraf manusia. Jaringan saraf tiruan tercipta

sebagai suatu generalisasi model matematis dari pemahaman manusia *(human cognition)* yang didasarkan atas asumsi berikut:

- Pemrosesan informasi terjadi pada elemen sederhana yang disebut neuron.
- Isyarat mengalir di antara sel saraf/neuron melalui suatu sambungan penghubung.
- Setiap sambungan penghubung memiliki bobot yang bersesuaian. Bobot ini akan digunakan untuk menggandakan/mengalikan isyarat yang dikirim melaluinya.
- Setiap sel saraf akan menerapkan fungsi aktivasi terhadap isyarat hasil penjumlahan berbobot yang masuk kepadanya untuk menentukan isyarat keluarannya.

12.3 Jaringan Saraf Manusia

Neuron adalah satuan unit pemroses terkecil pada otak, bentuk sederhana sebuah neuron yang oleh para ahli dianggap sebagai satuan unit pemroses tersebut digambarkan sebagai berikut. Jaringan saraf manusia merupakan suatu hal yang unik. Terkadang ada suatu masalah yang dapat dipecahkan oleh manusia, akan tetapi tidak dapat dipecahkan oleh komputer. Hal ini terjadi dikarenakan pada otak manusia terdiri atas jaringan saraf yang sangat kompleks.



Gambar 12.2 Jaringan Saraf Manusia

Jaringan saraf manusia terdiri atas tiga komponen utama, yaitu:

- *Soma* (Badan Sel)

  Soma (Badan sel) merupakan bagian terbesar dari sel saraf pada manusia. Fungsi utama dari komponen ini adalah menerima rangsangan dari *Dendrit* hingga kemudian meneruskannya ke *Axon*.

- *Dendrit*

  Dendrit merupakan bagian sel saraf pendek yang bercabang-cabang. Fungsi utama dari komponen ini adalah menerima rangsangan dan menghantarkannya ke *Soma* (Badan Sel).

- *Axon* (Neurit)

  Axon (neurit) berfungsi untuk membawa rangsangan dari badan sel ke sel saraf lainnya. Rangsangan akan ditransmisikan melalui celah sinapsis (*Synaptic Gap*) yang disebabkan oleh proses kimiawi pada tubuh. Proses transmisi secara kimiawi ini mengubah rangsangan yang masuk dengan mengukur besarnya frekuensi dari

rangsangan yang diterima. Cara inilah yang kemudian dijadikan konsep dalam *Artifical Neural Networks*.

Jaringan otak manusia tersusun tidak kurang dari 10^{13} buah neuron yang masing-masing terhubung oleh sekitar 10^{15} buah dendrite. Fungsi dendrite adalah sebagai penyampai sinyal dari neuron tersebut ke neuron yang terhubung dengannya.Sebagai terusan keluaran, setiap neuron memiliki axon, sedangkan bagian penerima sinyal disebut *synapse* (sinapsis). Secara umum jaringan saraf terbentuk dari satu trilyun (bahkan lebih) struktur dasar neuron yang terinterkoneksi dan dan terintegrasi antara satu dengan yang lain oleh satu trilyun sinapsis sehigga dapat melaksanakan aktifitas menyimpan (*memorize*) pengetahuan (*knowledge*) secara teratur dan terus menerus sesuai dengan kebutuhan.

Dalam jaringan saraf tiruan, neuron diartikan sebagai bagian terkecil dari jaringan saraf tiruan yang berfungsi sebagai elemen pemroses. Dengan demikian neuron juga dapat dinyatakan sebagai prosesor sederhana dari sistem jaringan saraf tiruan. Neuron juga dikenal dengan sebutan *percepton* atau *Adaline.* Dalam sistem jaringan saraf tiruan, neuron akan bekerja dengan mengumpulkan sinyal dari neuron yang terhubung sebelumnya dan memprosesnya untuk menjadi masukan bagi neuron berikutnya. Neuron tersusun dari komponen-komponen sebagai berikut:

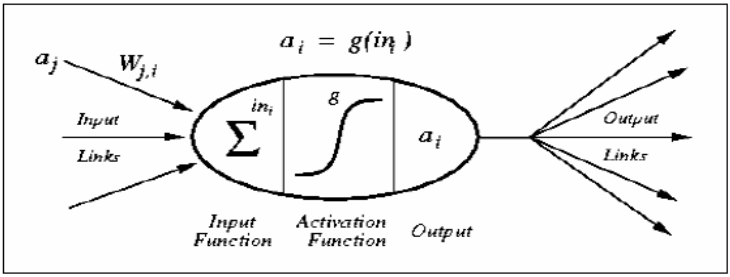
Sekumpulan penghubung atau yang dikenal dengan *synapses* atau *connection link* yang dikarakterkan dengan sebuah pembobot *(weight/strength connection)*. Sebuah penjumlah *(summing/adder)* yang berfungsi untuk menjumlahkan semua sinyal masukannya. Sebuah fungsi tidak dinamis *(non dynamical)* yang dikenali dengan sebutan fungsi aktivasi *(activation function)*.

12.4 Jaringan Saraf Tiruan (Artificial Neural Networks)

Artificial Neural Networks atau yang sering disebut juga jaringan saraf tiruan menyerupai dengan otak manusia dalam dua hal:

- Pengetahuan yang diperoleh dari lingkungan sekitar dengan melalui proses pembelajaran.
- Kekuatan hubungan antar *neuron,* sering disebut juga *synaptic weight,* yang digunakan untuk menyimpan pengetahuan yang telah didapatkan.

Jaringan saraf tiruan biasanya tersusun dari elemen-elemen pada lapisan-lapisan yang terhubung dan diberi bobot. Jaringan ini memodifikasi bobot tersebut berdasarkan serangkaian *input* yang diberikan dari luar sistem tersebut, sehingga menghasilkan *output* yang konsisten dan serupa dengan *input* yang diberikan. Biasanya, setiap elemen akan memproses berdasarkan operasi matematika yang telah diberikan pada setiap elemen. Berikut contoh model tiruan dari *Neural Networks.*



Gambar 12.3 Contoh Model Tiruan *Neural Network*

Jaringan saraf tiruan juga merupakan suatu sistem yang "*fault tolerant*" dalam dua keadaan:

- Pertama, dapat mengenali suatu sinyal *input* yang agak berbeda dari *input* sebelumnya.
- Kedua, tetap mampu bekerja meskipun beberapa *Neuron* tidak mampu bekerja dengan baik, jika ada sebuah *Neuron* yang rusak, maka *Neuron* lainnya akan menggantikan kerja dari *Neuron* yang rusak tersebut.

Akan tetapi, jaringan saraf tiruan ini tentu sangat jauh lebih sederhana daripada sistem saraf manusia yang sangat kompleks.

A. Perbandingan antara Otak Manusia dengan Jaringan Saraf Tiruan

Perbedaan utama antara otak manusia dengan jaringan saraf tiruan adalah bahwa otak manusia bisa lupa, sedangkan jaringan saraf tiruan tidak mungkin lupa. Jaringan saraf tiruan yang telah dilatih akan menanam secara "mendalam" dan

permanen informasi tersebut dalam sel-selnya. Sel-sel saraf dalam jaringan saraf tiruan tidak mungkin rusak sedangkan sel-sel saraf jaringan saraf manusia sangat mungkin rusak. Ketika sel saraf dalam jaringan saraf manusia itu rusak maka informasi yang terkandung di dalamnya akan hilang.

Data dan informasi pada manusia disimpan dalam suatu unit sel yang terstruktur dalam otak. Sementara pada jaringan saraf tiruan, data dan informasi tersimpan dalam bobot dan bisa berbentuk *file* sehingga kerusakan dapat diantisipasi dengan menggunakan *backup* atau data cadangan.

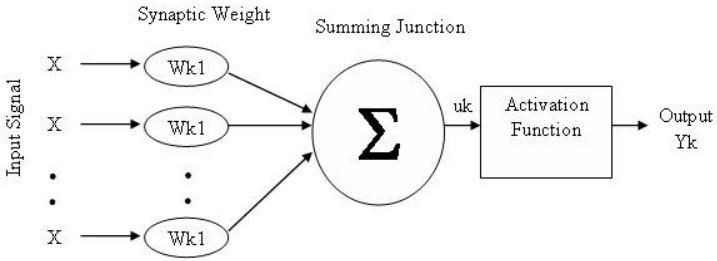
Cara belajar otak manusia dan jaringan saraf tiruan juga berbeda. Manusia belajar secara terus menerus, mengakumulasikan informasi yang diterimanya, sedangkan jaringan saraf tiruan tidak. Setelah jaringan saraf tiruan selesai melakukan proses belajar, jika akan ditambahkan informasi baru, maka jaringan saraf tiruan harus mempelajari semua informasi itu lagi dan bukan hanya informasi tambahannya saja.

Perbedaan lainnya adalah ketepatan. Jaringan saraf tiruan akan mampu menyelesaikan masalah yang sama dengan hasil yang sama sekalipun masalah tersebut diulang puluhan juta kali. Sedangan manusia memiliki keterbatasan atas pekerjaan yang bersifat berulang. Untuk puluhan proses atas pekerjaan mungkin masih akurat, tetapi untuk ratusan atau ribuan, mungkin sudah tidak akurat lagi. alam satuan panjang yang sama, jaringan saraf tiruan dapat mengirim informasi lebih cepat dari pada otak manusia. Hal ini karena jaringan saraf tiruan bekerja secara

elekrtonis sedangkan otak manusia bekerja secara kimiawi. Otak manusia berisi sekitar 10^{11} sel saraf yang berfungsi untuk memproses informasi yang masuk. Sedangkan pada jaringan saraf buatan terdiri atas 108 transistor. Sebuah prosesor pada komputer dapat menjalankan sebuah instruksi tunggal maupun ganda. Sedangkan CPU konvensional dapat menyelesaikan beberapa ratus juta operasi tiap detiknya.

B. Cara Kerja *Neural Networks*

Neuron merupakan sebuah unit penting dalam operasi di dalam sistem jaringan saraf buatan. Gambar 12.4 menunjukkan sebuah bentuk model dasar dari neuron yang digunakan di dalam sistem jaringan saraf buatan.



Gambar 12.4 Bentuk Model Dasar Neuron

Pada Gambar 12.4 terdapat tiga elemen dasar dari neuron:

- Satu set *synapses* atau jaringan yang saling terhubung

Masing-masing ditandai dengan berat dan kekuatan dari *synapses* itu sendiri. Secara khusus, sebuah sinyal *X1* pada masukan *Synapses1* terhubung ke *Neuronk* kemudian dikalikan dengan berat *wk1Synaptic*. Tidak seperti *synapses* di otak, bobot sinaptik dari *Neuron* buatan mungkin terletak pada rentang yang meliputi negatif serta nilai-nilai positif.

- Sebuah *adder*

  Digunakan untuk menjumlahkan masukan dari sinyal *input* yang dihitung dengan masing-masing *Neuron*.

- Sebuah fungsi aktivasi

  Digunakan untuk membatasi amplitudo *output* dari *Neuron.* Fungsi ini disebut juga dengan *squashing function.* Ini akan membatasi kisaran amplitudo dari sinyal keluaran untuk beberapa nilai yang dibatasi. Biasanya, rentang amplitudo dinormalisasi dari *output* dari *neuron* ditulis sebagai interval unit tertutup [0,1] atau [-1,1].

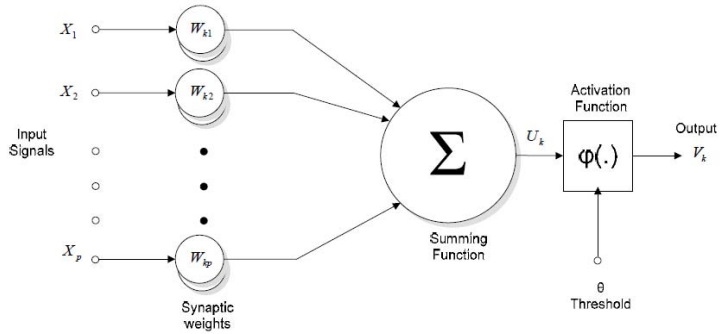
Berdasarkan Gambar 12.4, secara singkat cara kerja dari sistem jaringan syaraf buatan adalah sebagai berikut:

1. Satu set *synapses input* membawa aktivasi dari *neuron* lain.
2. Sebuah unit pengolahan merangkum masukan dan kemudian menerapkan fungsi aktivasi non-linear (yaitu *squashing*/transfer/*thresholdFunction*).

3. *Line output* mengirimkan hasilnya ke *neuron* lain.

C. Komponen

Satu sel saraf terdiri atas tiga bagian, yaitu fungsi penjumlah (*summing function*), fungsi aktivasi (*activation function*), dan keluaran (*output*).



Gambar 12.5 Model Neuron

Jika dilihat, neuron buatan di atas mirip dengan sel neuron biologis. Informasi (*input*) akan dikirim ke neuron dengan bobot tertentu. *Input* ini akan diproses oleh suatu fungsi yang akan menjumlahkan nilai-nilai bobot yang ada. Bobot adalah hubungan antar neuron-neuron. Hasil penjumlahan kemudian akan dibandingkan dengan suatu nilai ambang (*threshold*) tertentu melalui fungsi aktivasi setiap neuron. Apabila *input* tersebut melewati suatu nilai ambang tertentu, maka neuron tersebut akan diaktifkan, jika tidak, maka neuron tidak akan diaktifkan. Apabila neuron tersebut diaktifkan, maka neuron

tersebut akan mengirimkan *output* melalui bobot-bobot *output*-nya ke semua neuron yang berhubungan dengannya, sehingga dapat disimpulkan bahwa neuron terdiri atas 3 elemen pembentuk, yaitu:

1. Himpunan unit-unit yang dihubungkan dengan jalur koneksi. Jalur-jalur tersebut memiliki bobot yang berbeda-beda. Bobot yang bernilai positif akan memperkuat sinyal dan yang bernilai negatif akan memperlemah sinyal yang dibawa. Jumlah, struktur dan pola hubungan antar unit-unit tersebut akan menentukan arsitektur jaringan.
2. Suatu unit penjumlah yang akan menjumlahkan *input-input* sinyal yang sudah dikalikan dengan bobotnya.
3. Fungsi aktivasi yang akan menentukan apakah sinyal dari *input* neuron akan diteruskan ke neuron lain atau tidak.

Setiap pola-pola informasi *input* dan *output* yang diberikan ke dalam JST diproses dalam neuron. Neuron-neuron tersebut berkumpul di dalam lapisan-lapisan yang disebut *neuron layers.* Lapisan-lapisan penyusun JST tersebut dapat dibagi menjadi 3, yaitu:

1. Lapisan *Input*

   Unit-unit di dalam lapisan *input* disebut unit-unit *input*. Unit-unit *input* tersebut menerima pola *input* data dari luar yang menggambarkan suatu permasalahan.

2. Lapisan Tersembunyi

   Unit-unit di dalam lapisan tersembunyi disebut unit-unit tersembunyi, di mana *output*-nya tidak dapat secara langsung diamati.

3. Lapisan *Output*

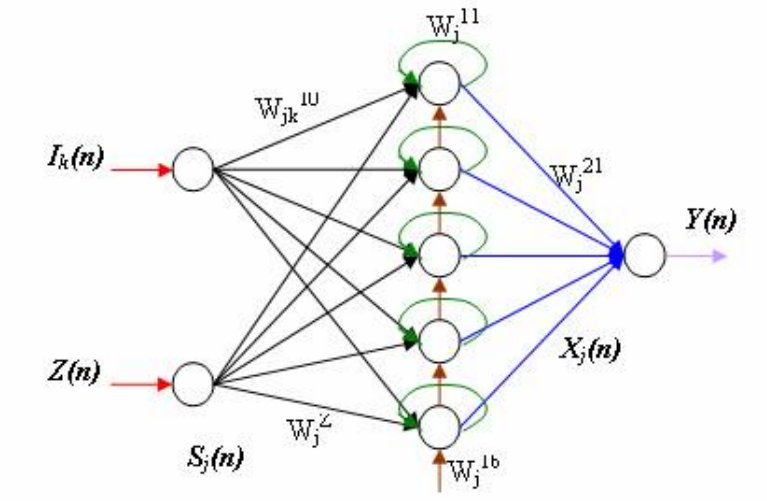
   Unit-unit di dalam lapisan *output* disebut unit-unit *output*. *Output* dari lapisan ini merupakan solusi JST terhadap suatu permasalahan.

Pada jaringan saraf, neuron-neuron akan dikumpulkan dalam lapisan-lapisan (*layer*) yang disebut dengan lapisan neuron *(neuron layers)*. Biasanya neuron-neuron pada satu lapisan akan dihubungkan dengan lapisan-lapisan sebelum dan sesudahnya (kecuali lapisan *input* dan lapisan *output*). Informasi yang diberikan pada jaringan saraf akan dirambatkan lapisan ke lapisan, mulai dari *input* sampai ke lapisan *output* melalui lapisan yang lainnya, yang dikenal dengan lapisan tersembunyi *(hidden layer),* tergantung pada algoritma pembelajarannya, bisa jadi informasi tersebut akan dirambatkan secara mundur pada jaringan lapisan.

D. Arsitektur

Dari beberapa jenis JST, di sini dijelaskan tentang JST *Diagonal Recurrent Neural Network* (JST DRNN). Struktur JST DRNN ini dapat dilihat pada Gambar 12.6, di mana terdapat aksi umpan balik di lapisan tersembunyinya. Informasi berupa umpan

balik ini diberikan supaya model kaya akan informasi dinamik dari sistem.



Gambar 12.6 Struktur Diagonal *Recurrent Neural Network*

Pada Gambar 12.6, *input* ditunjukkan dengan $I_k(n)$ dan z(n). W_{jk}^{10}, W_j^{Z}, W_j^{11}, W_j^{1b}, W_j^{21}, $X_j(n)$ dan Y(n) secara berurutan menunjukkan bobot masukan 1, masukan 2, diagonal, bias, *output*, keluaran lapisan *hidden*, dan keluaran JST. Model matematik dari arsitektur DRNN tersebut dapat dituliskan sebagai berikut:

$$S_j(n) = \sum_{k=1}^{N} W_{jk}^{10} I_k(n) + W_j^{Z} Z(n) + W_j^{11} X_j(n-1) + Wj^{1b} \qquad (12.1)$$

$S_j(n)$ adalah penjumlahan dari perkalian *input* dengan masing-masing bobotnya dan bobot bias.

$$X_j(n) = f(S_j(n))$$

(12.2)

di mana f adalah fungsi aktivasi dengan

$0 \leq k \leq N$, N jumlah *neuron* lapisan masukan

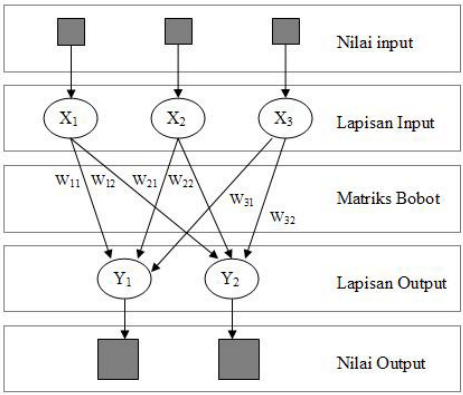
$0 \leq j \leq M$, M jumlah *neuron* lapisan tersembunyi

$$Y(n) = \sum_{j=0}^{M} W_j^{21} Xj(n) \qquad (12.3)$$

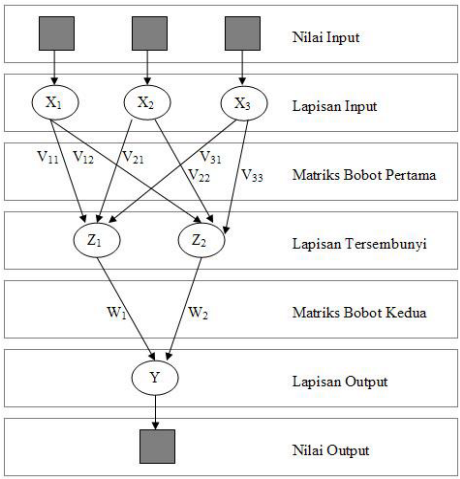
Y(n) adalah penjumlahan dari perkalian bobot *output* dan *output* lapisan sebelumnya pada setiap *node*. Secara umum, arsitektur jaringan dibagi menjadi 3, yaitu:

1. Jaringan saraf dengan lapisan tunggal (*single layer net*)
   Jaringan dengan lapisan tunggal hanya memiliki satu lapisan dengan bobot-bobot terhubung. Jaringan ini hanya menerima *input* kemudian secara langsung akan mengolahnya menjadi *output* tanpa harus melalui lapisan tersembunyi. Arsitektur jaringan ini hanya terdiri atas satu lapisan *input* dan satu lapisan *output*, tanpa lapisan tersembunyi.
2. Jaringan saraf dengan banyak lapisan (*multilayer net*)
   Jaringan dengan banyak lapisan memiliki satu atau lebih lapisan yang terletak di antara lapisan *input* dan lapisan

output (memiliki satu atau lebih lapisan yang tersembunyi). Umumnya, ada lapisan yang berbobot yang terletak antara 2 lapisan yang bersebelahan. Jaringan ini dapat menyelesaikan permasalahan yang lebih sulit daripada lapisan dengan lapisan tunggal.



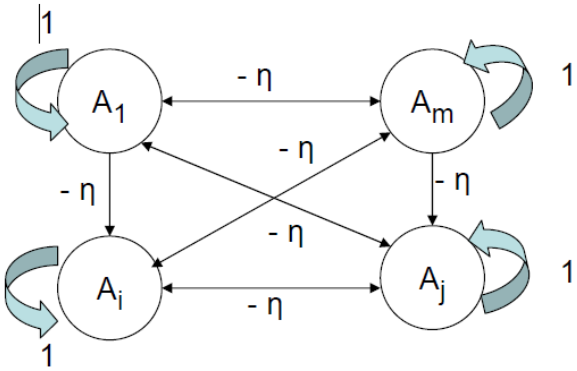
Gambar 12.7 Jaringan Saraf Lapisan Tunggal



Gambar 12.8 Jaringan Saraf Banyak Lapisan

3. Jaringan saraf dengan lapisan kompetitif (*competitive layer net*)

   Jaringan saraf dengan lapisan kompetitif ini adalah jaringan yang mempunyai minimal satu *feedback loop*. Sebagai contoh, suatu *competitive layer net* bisa terdiri atas satu lapisan neuron tunggal dengan masing-masing neuron memberikan kembali *output*-nya sebagai *input* data pada semua neuron yang lain. Tidak ada *self-feedback loops* dalam jaringan. Arsitektur ini memiliki bentuk yang berbeda, di mana antara neuron dapat saling dihubungkan.



Gambar 12.9 Jaringan Saraf Lapisan Kompetitif

E. Fungsi Aktifasi

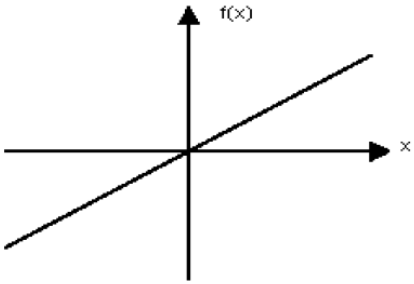
Fungsi Aktivasi adalah sebuah proses pemetaan sinyal *output* oleh suatu neuron untuk menghasilkan *output*. Apakah

sinyal dari *input* suatu neuron akan diteruskan ke neuron lain atau tidak ditentukan menggunakan fungsi aktivasi. Beberapa FA yang biasa digunakan dalam JST adalah:

1. Fungsi Linear (Identitas)

   Fungsi linear memiliki nilai *output* sama dengan nilai *input*-nya.

   Persamaan: $y = f(x) = x$ (12.4)



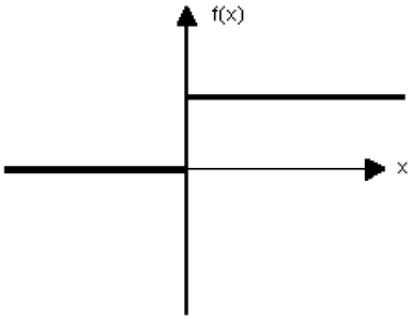
Gambar 12.10 Fungsi Linear

2. Fungsi *Threshold* (*Hard Limit*)

   Fungsi ini digunakan untuk mengkonversikan *input* dari suatu variabel yang bernilai kontinu ke suatu *output* biner (0, 1 atau -1).

   Biner

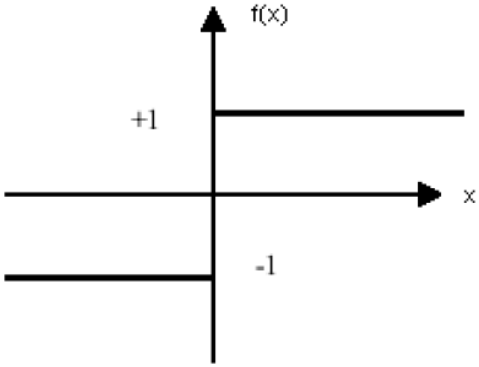
   Persamaan: $y = f(x) = \begin{cases} 0, \text{ jika } x < 0 \\ 1, \text{ jika } x \geq 0 \end{cases}$ (12.5)



Gambar 12.11 Fungsi Biner

Bipolar

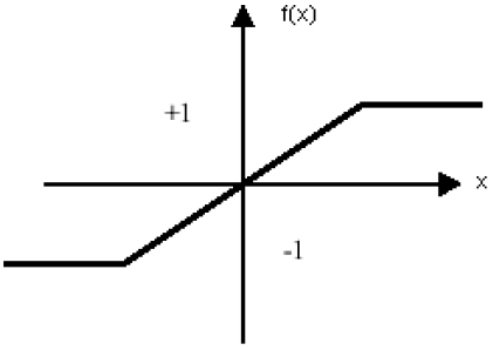
Persamaan: $y = f(x) = \begin{cases} 1, \text{ jika } x \geq 0 \\ -1, \text{ jika } x < 0 \end{cases}$ (12.6)



Gambar 12.12 Fungsi Bipolar

3. Fungsi *Symetric Saturating Linear*

Persamaan: $y = f(x) = \begin{cases} 1; \text{ jika } x \geq 1 \\ x; \text{ jika } -1 \leq x \leq 1 \\ -1; \text{ jika } x \leq -1 \end{cases}$ (12.7)



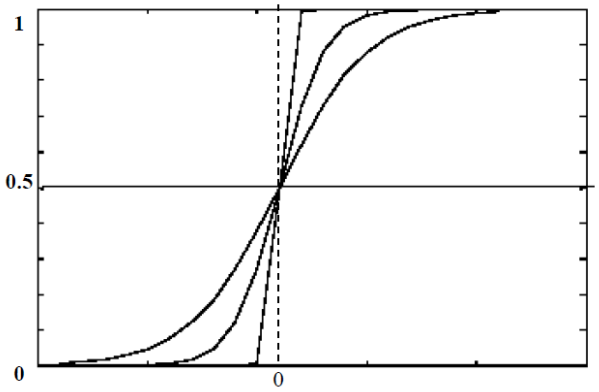
Gambar 12.13 Fungsi *Symetric Saturating Linear*

4. Fungsi Biner Sigmoid

   Fungsi ini digunakan untuk jaringan saraf yang dilatih dengan menggunakan metode *backpropagation*. Fungsi sigmoid biner memiliki nilai pada *range* 0 sampai 1. Oleh karena itu, fungsi ini sering digunakan untuk jaringan saraf yang membutuhkan nilai *output* yang terletak pada interval 0 sampai 1. Namun, fungsi ini dapat juga digunakan oleh jaringan saraf yang nilai *output*-nya 0 atau 1.

   Persamaan:

$$y = f(x) = \frac{1}{1+e^{-ax}} \text{ untuk } 0 \le f(x) \le 1 \qquad (12.8)$$



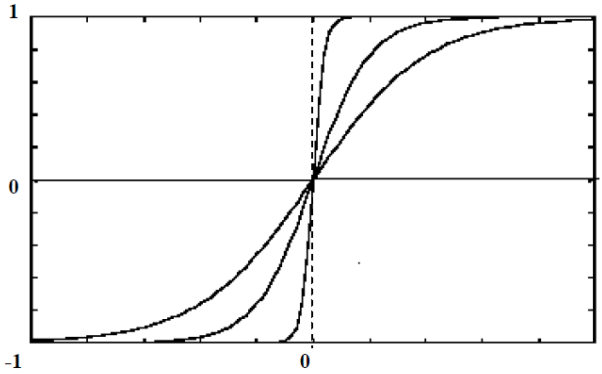
Gambar 12.14 Fungsi Biner Sigmoid

5. Fungsi Bipolar Sigmoid

   Fungsi sigmoid bipolar hamper sama dengan fungsi sigmoid biner, hanya saja *output* dari fungsi ini memiliki *range* antara 1 sampai -1.

   Persamaan:

$$y = f(x) = \frac{e^{ax} - e^{-ax}}{e^{ax} + e^{-ax}} \text{ untuk } -1 \le f(x) \le 1 \qquad (12.9)$$



Gambar 12.15 Fungsi Bipolar Sigmoid

12.5 Algoritma Pembelajaran

A. Proses Belajar

Terdapat sangat banyak ide dan definisi yang berhubungan dengan "belajar". Di sini, belajar dalam konteks JST diartikan sebagai berikut:

Belajar adalah suatu proses dimana parameter-parameter bebas JST diadaptasikan melalui suatu stimulus berkelanjutan oleh lingkungan di mana jaringan berada. Jenis belajar ditentukan oleh pola dimana pengubahan parameter dilakukan, sehingga, dalam proses belajar terdapat kejadian-kejadian sebagai berikut:

- JST dirancang oleh lingkungan
- JST mengubah dirinya sebagai hasil stimulus (rangsangan) ini

- JST memberikan respon dengan cara yang baru kepada lingkungan, disebabkan perubahan yang terjadi dalam struktur internalnya sendiri.

1. *Supervised Learning* (Belajar dengan Pengawasan)
   Supervised atau *active learning* adalah proses balajar yang membutuhkan guru. Guru di sini adalah sesuatu yang memiliki pengetahuan tentang lingkungan. Guru bisa direpresentasikan sebagai sekumpulan sampel *input-output*. Pembangunan pengetahuan dilakukan oleh guru dengan memberikan respon yang diinginkan kepada JST. Respon yang diinginkan tersebut merepresentasikan aksi optimum yang dilakukan oleh JST. Parameter-parameter jaringan berubah-ubah berdasarkan vektor latih dan sinyal kesalahan (sinyal kesalahan adalah perbedaan antara keluaran JST dan respon yang diinginkan). Proses perubahan ini dilakukan secara berulang-ulang, selangkah demi selangkah, dengan tujuan agar JST bisa memiliki kemampuan yang mirip dengan gurunya. Dengan kata lain, JST dilatih untuk dapat memetakan sekumpulan sampel *input-output* dengan akurasi yang tinggi.
2. *Unsupervised Learning* (Belajar Tanpa Pengawasan)
   Sesuai dengan namanya, *unsupervised* atau *self-organized learning* tidak membutuhkan guru untuk memantau proses balajar. Dengan kata lain, tidak ada sekumpulan *input-output* atau fungsi tertentu untuk dipelajari oleh jaringan.

Salah saru contoh *unsupervised learning* adalah *competitive learning*. Sebagai contoh, kita bisa menggunakan JST yang terdiri atas dua lapisan, satu lapisan masukan dan satu lapisan kompetitif. Lapisan masukan menerima data yang disediakan. Lapisan kompetitif terdiri atas neuron-neuron yang saling bersaing untuk meraih "kesempatan" memberikan respon ke ciri khas yang berisi data masukan. Dalam bentuk paling sederhana, jaringan beroperasi berdasarkan strategi "*winner-takes-all*".

Model-Model pembelajaran dalam ANN:

1. Algoritma Pembelajaran Hebb

Pada metode ini, pembelajaran dilakukan dengan cara memperbaiki nilai bobot sedemikian rupa sehingga jika ada 2 neuron yang terhubung dan keduanya pada kondisi 'hidup' (*on*) pada saat yang sama, maka bobot antara keduanya dinaikkan.

Algoritma:

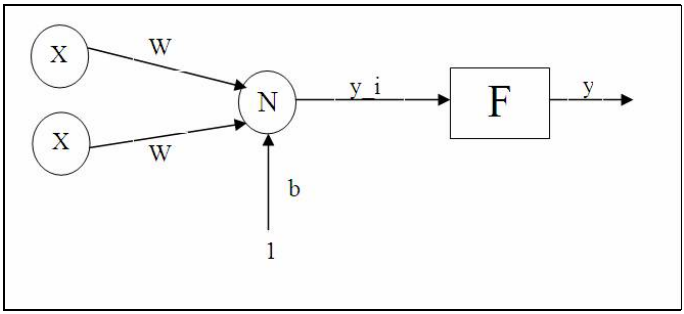
0. Inisialisasi semua bobot:
   $w_{ij} = 0$; dengan $i = 1,2,\ldots,n$; dan $j = 1,2,\ldots,m$.
1. Untuk setiap pasangan *input-output* (s-t), lakukan langkah-langkah sebagai berikut:
   a. *Set input* dengan nilai sama dengan vektor *input*:
      $x_i = s_i$; $(i = 1,2,\ldots,n)$
   b. *Set output* dengan nilai sama dengan vektor *output*:
      $y_j = t_j$; $(j = 1,2,\ldots,m)$

c. Perbaiki bobot:

$w_{ij} = w_{ij} + x_i y_j$

(i = 1,2,…,n; dan j = 1,2,…,m)

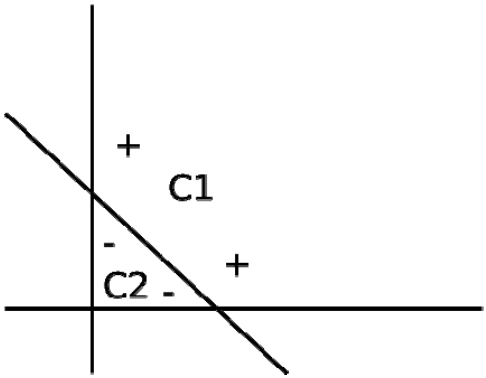
Dengan catatan bahwa nilai bias selalu 1.



Gambar 12.16 Algoritma Pembelajaran Hebb

2. Algoritma Perceptron

Perceptron termasuk salah satu bentuk jaringan saraf yang sederhana. Perceptron biasanya digunakan untuk mengklasifikasikan suatu tipe pola tertentu yang sering dikenal dengan pemisahan secara linear. Pada dasarnya, perceptron pada jaringan saraf dengan satu lapisan memiliki bobot yang bisa diatur. Algoritma yang digunakan oleh aturan perceptron ini akan mengatur parameter-parameter bebasnya melalui proses pembelajaran. Fungsi aktivasi ini dibuat sedemikian rupa sehingga terjadi pembatasan antara daerah positif dan daerah negatif.



Gambar 12.17 Algoritma Perceptron

Algoritma:

S adalah vektor masukan dan t adalah target keluaran.

α adalah laju pemahaman (*learning rate*) yang ditentukan.

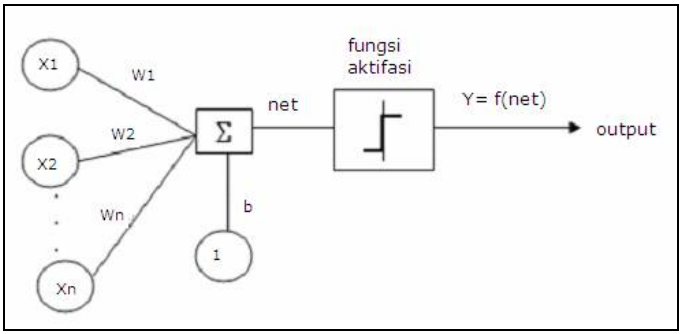
θ adalah *threshold* yang ditentukan.

a. Inisialisasi semua bobot dan bias (umumnya $w_i = b_0$)
   Tentukan laju pemahaman (α). Untuk penyederhanaan biasanya α diberi nilai = 1

b. Selama ada elemen vektor masukan yang respon unit keluarannya tidak sama dengan target, lakukan:
   1. *Set* aktivasi unit masukan $x_i = s_i$ ($i = 1,\ldots,n$)
   2. Hitung respon unit keluaran: net = $\Sigma\, x_i w_i + b$

$$f(\text{net}) = \begin{cases} 1 & \text{jika net} > \theta \\ 0 & \text{jika } -\theta \le \text{net} \le \theta \\ -1 & \text{jika net} < -\theta \end{cases} \qquad (12.10)$$

3. Perbaiki bobot pola yang mengandung kesalahan (y ≠ t) menurut persamaan :
   w_i(baru) = w_i(lama) + Δw(i=1,…n) dengan Δw=αt x_i
   b(baru) = b(lama) + Δb dengan Δb = αt



Gambar 12.18 Diagram Algoritma Perceptron

Ada beberapa hal yang perlu diperhatikan dalam algoritma tersebut:

a. Iterasi dilakukan terus hingga semua pola memiliki keluaran jaringan yang sama dengan targetnya (jaringan sudah memahami pola). Iterasi tidak berhenti setelahsemua pola dimasukan seperti yang terjadi pada model Hebb.
b. Perubahan bobot hanya dilakukan pada pola yang mengandung kesalahan (keluaran jaringan ≠ target). Perubahan tersebut merupakan hasil kali unit masukan dengan target laju pemahaman. Perubahan bobot hanya akan terjadi kalau unit masukan ≠ 0.

3. Algoritma Pembelajaran Backpropagation

Backpropagation merupakan algoritma pembelajaran yang terawasi dan biasanya digunakan oleh perceptron dengan banyak lapisan untuk mengubah bobot-bobot yang terhubung dengan neuron-neuron yang ada pada lapisan tersembunyinya. Algoritma *backpropagation* menggunakan *error output* untuk mengubah nilai bobot-bobotnya dalam arah mundur (*backward*). Unutk mendapatkan *error* ini, tahap perambatan maju (*forward propagation*) harus dikerjakan terlebih dahulu. Pada saat perambatan maju, neuron-neuron diaktifkan dengan menggunakan fungsi aktivasi yang dapat dideferensiasikan seperti sigmoid.

Algoritma:

Step 0: Inisialisasi semua bobot dengan bilangan acak kecil.

Step 1: Jika kondisi penghentian belum dipenuhi, lakukan langkah 2 – 9.

Step 2: Untuk setiap pasang data pelatihan, lakukan langkah 3 – 8.

Fase I

Step 3: Tiap unit masukan menerima sinyal dan meneruskannya ke unit tersembunyi di atasnya.

Step 4: Hitung semua keluaran di unit tersembunyi z_j (j=1, 2,...,p).

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij},$$
$$z_j = f\left(z\_in_j\right) \tag{12.11}$$

Step 5: Hitung semua keluaran jaringan di unit keluaran y_k (k = 1, 2,...,m).

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk},$$
$$y_k = f\left(y\_in_k\right) \tag{12.12}$$

Fase II

Step 6: Hitung faktor δ unit keluaran berdasarkan kesalahan di setiap unit keluaran y_k (k = 1, 2, ...,m).

$$\delta_k = (t_k - y_k)\, f'(y\_in_k), \tag{12.13}$$
$$\Delta w_{jk} = \alpha\, \delta_k\, z_j,$$

Hitung perubahan bobot w_{jk} dengan laju pemahaman α

$$\Delta w_{0k} = \alpha\, \delta_k \tag{12.14}$$

δ_k merupakan unit kesalahan yang akan dipakai dalam perubahan bobot layar di bawahnya.

Step 7: Hitung faktor δ unit tersembunyi berdasarkan kesalahan di setiap unit tersembunyi z_j (j=1,2,...,p).

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk} \tag{12.15}$$

Faktor δ unit tersembunyi.

$$\delta_j = \delta\_in_j f'(z\_in_j) \tag{12.16}$$

Hitung suku perubahan bobot v_{ji}.

$$\Delta v_{ij} = \alpha \delta_j x_i \quad j = 1,2,\ldots,p; i = 1,2,\ldots,n \tag{12.17}$$

Fase III

Step 8: Hitung semua perubahan bobot. Perubahan bobot garis yang menuju ke unit keluaran, yaitu:

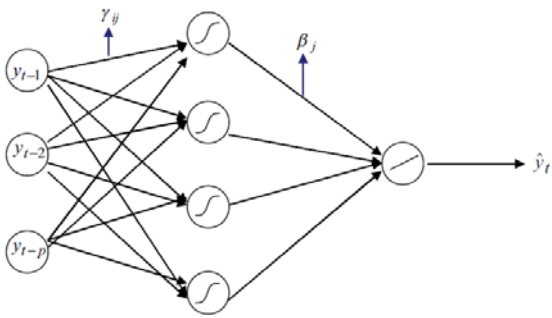
w_{jk}(new) = w_{jk}(old) + Δw_{jk} (12.18)

(k = 1,2,…,m; j = 0,1,…,p)

Perubahan bobot garis yang menuju ke unit tersembunyi, yaitu:

v_{ij}(new) = v_{ij}(old) + Δv_{jk} (12.19)

(j = 1,2,…,p; i = 0,1,…,n)



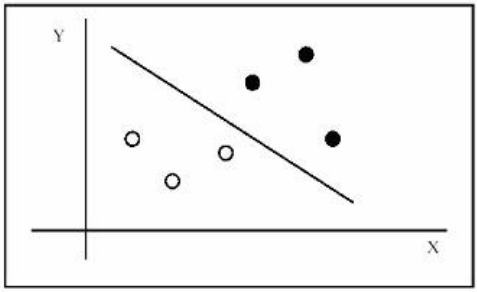
Gambar 12.19 Diagram Algoritma *Backpropagation*

Backpropagation dapat diaplikasikan dengan baik adalah bidang peramalan (*forecasting*). Secara umum, masalah peramalan dapat dinyatakan dengan sejumlah data runtun waktu (*time series*) yang bertujuan untuk memperkirakan berapa harga x_n+1 berdasarkan data $x_1,x_2,...,x_n$. Langkah-langkah membangun struktur jaringan untuk peramalan sebagai berikut.

1. Transformasi Data.
2. Pembagian Data.
3. Perancangan Struktur Jaringan yang Optimum.
4. Pemilihan Koefisien Pemahaman dan Momentum.
5. Memilih dan Menggunakan Struktur Jaringan yang Optimum.
6. Pemilihan jaringan optimum dan penggunaannya untuk peramalan.

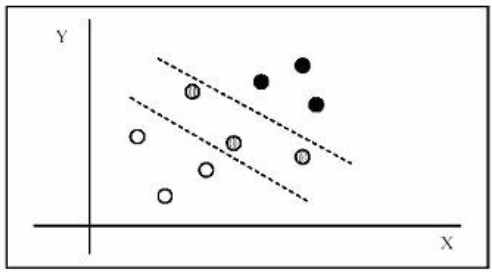
4. *Multi Layer Perceptron* (MLP)

MLP merupakan model JST yang paling banyak digunakan dalam edukasi. Sesuai dengan karakteristik *neural network*, pada dasarnya *Multi layer Perceptron* memiliki kecenderungan yang sama dengan jenis neural network lainnya, namun setiap jenis memiliki karakteristik masing-masing, seperti halnya *Single layer Neural Network*, biasanya hanya digunakan untuk memberikan solusi yang sifatnya hanya sederhana saja, sebagai contoh berikut ini.



Gambar 12.20 Penggunaan *Single Layer Neural Network*

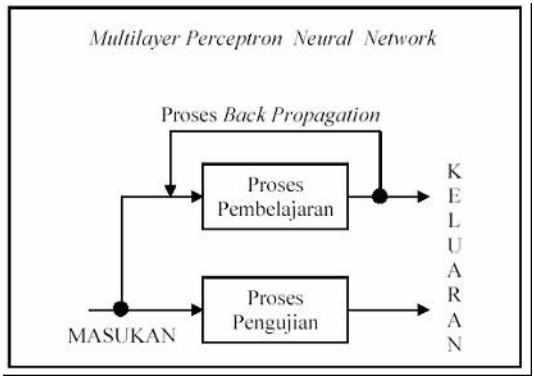
Gambar 12.20 menunjukkan bahwa *single layer neural network* digunakan untuk menganalisis dua bagian yang berbeda saja, yaitu agar dapat mengetahui posisi lingkaran hitam dan lingkaran yang berwarna putih. Lain halnya dengan dengan kondisi pada gambar berikut.



Gambar 12.21 Penggunaan *Multi Layer Perceptron Neural Network*

Pada Gambar 12.20, menunjukkan bahwa dengan karakteristik *Single Layer Neural Network* yang hanya mampu mendeteksi dua daerah saja membuat kasus ini sulit

untuk dapat diselesaikan. *Multi Layer Perceptron Neural Network* adalah jenis *neural network* yang memiliki kemampuan untuk mendeteksi atau melakukan analisis untuk permasalahan yang sifatnya cukup atau bahkan sangat kompleks, seperti pada masalah pemrosesan bahasa, pengenalan suatu pola serta pemrosesan suatu *image* atau gambar. Adapun proses yang terjadi pada *Multi Layer Perceptron Neural Network*, adalah sebagai berikut:



Gambar 12.22 Proses *Multi Layer Perceptron Neural Network*

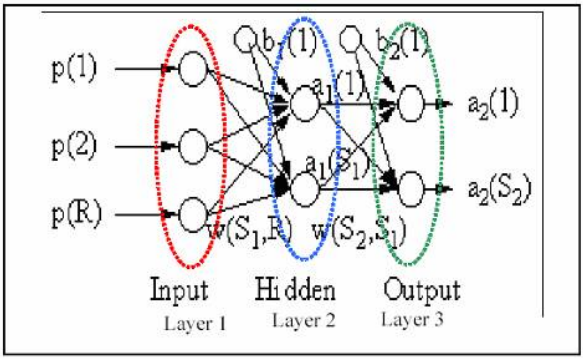
a. Masukan

Proses ini merupakan bagian dari sistem kerja secara keseluruhan, karena proses masukan digunakan untuk menunjang pada proses pembelajaran serta proses pengujian. Pada proses ini, masukan diklasifikasikan berdasarkan keinginan dari pembuat, di mana bentuk masukan dapat

berupa nilai *logic* atau bilangan biner (1 atau 0) atau juga bisa berupa nilai angka atau bilangan *real* (120.3 atau 100) bahkan dapat melakukan proses dengan menggunakan bilangan negatif.

b. Proses Pembelajaran

Pada bagian ini merupakan sisi kelebihan dari metoda *neural network*, dimana setiap permasalahan yang akan dihadapi dapat dipecahkan dengan melalui tahapan pembelajaran, seperti halnya otak manusia yang secara sifat biologis, memiliki kemampuan untuk mempelajari hal-hal yang baru. Memang pada dasarnya, *neural network* ini dibuat untuk dapat mempelajari sesuatu hal yang baru sehingga mampu melakukan analisis tanpa menggunakan suatu rumusan yang baku. Proses pembelajaran ini sangat mempengaruhi sensitifitas kemampuan dalam melakukan penganalisaan, semakin banyak bahan atau masukan sebagai pembelajaran maka akan semakin mudah dan sensitif dalam melakukan analisa. Biasanya untuk membahas hal-hal yang cukup kompleks, *Multi Layer Perceptron Neural network* memiliki *hidden neuron* yang digunakan untuk mengimbangi setiap permasalahan yang akan dihadapi, umumnya untuk melakukan analisis pada hal-hal yang rumit, rancangan *neural network* yang dibuat minimal memiliki tiga *layer* seperti pada Gambar 12.23, namun hal ini tergantung pada tingkat kompleksitas yang dihadapi.



Gambar 12.23 *Multi Layer Perceptron Neural Network* 3 *Layer*

c. Proses Perhitungan saat pembelajaran

Proses ini melibatkan dua faktor penting, yaitu masukan dan keluaran yang ditentukan. Keluaran tersebut merupakan bagian dari sistem atau metoda pembelajaran yang dinamakan "Supervised Learning", dengan demikian setiap masukan memiliki keluaran yang nantinya akan dijadikan sebagai acuan pembelajaran. Hal inilah yang membuat *Neural Network* melakukan analisis, selain banyaknya masukan yang diberikan, proses pembelajaran yang dilakukan secara berulang pun akan menunjang kemampuan *Neural Network* saat menganalisis.

d. Keluaran

Bagian ini merupakan proses yang digunakan untuk mengetahui seberapa besar pengaruh pembelajaran terhadap keluaran yang diinginkan, jika hasil yang diinginkan kurang sesuai maka kemungkinan yang terjadi adalah:

- Variabel masukan (informasi yang diberikan) kurang menunjang
- Kurangnya *layer* pada rancangan keseluruhan
- Jumlah neuron yang terlalu sedikit

Namun tidak menutup kemungkinan karena ketidaksesuaian penerapan saat proses pembelajaran dilakukan juga dapat mempengaruhi proses pembelajaran. Hal lainnya yang dapat mempengaruhi proses pencapaian target adalah pemilihan metoda *back propagation*, yang dijelaskan berikutnya.

5. Back Propagation *Multilayer Perceptron Neural Network*

Back Propagation adalah istilah dalam penggunaan metoda MLP-NN untuk melakukan proses *update* pada nilai vektor *weight* dan *bias*. Adapun bentuk metoda *weight* ini memiliki beberapa macam, antara lain adalah sebagai berikut.

- Gradient Descent Back Propagation (GD)
- Gradient Descent Back Propagation dengan Momentum (GDM)
- Variable Learning Rate Back Propagation dengan Momentum (GDX)
- Conjugate Gradient Back Propagation (CGP)

Gradient Descent Back Propagation (GD)

Metoda ini merupakan proses *update* untuk nilai *weight* dan *bias* dengan arah propagasi fungsinya selalu menurunkan nilai *weight*

sebelumnya. Bentuk vektor *weight* tersebut berlaku seperti metoda berikut.

$$W_{k+1} = W_k - \alpha \, . \, g_k \qquad (12.20)$$

Dimana α, merupakan *learning rate* serta g, merupakan *gradient* yang berhubungan dengan nilai *error* yang diakibatkan oleh *weight* tersebut.

Gradient Descent Back Propagation dengan Momentum

Penggunaan Momentum pada Metoda ini memberikan nilai tambah dimana hasil *update* diharapkan tidak berhenti pada kondisi yang dinamakan "Local Minimum", sehingga proses penelusuran hingga mencapai nilai minimum yang paling puncak dalam pengertian nilai *error* yang paling kecil dapat tercapai. Adapun bentuk metoda penggunaan Momentum ini adalah seperti dibawah ini.

$$W_{k+1} = W_k - \alpha \, . \, g_k + \mu \, . \, W_{k-1} \qquad (12.21)$$

Variabel Learning Rate Back Propagation dengan Momentum

Penggunaan metoda ini bertujuan untuk mempercepat waktu penyelesaian sehingga proses mendapatkan nilai error yang paling kecil dapat tercapai dengan cepat serta penelusuran yang lebih singkat. Sebaliknya jika nilai yang digunakan dalam praktisnya maka hasil yang didapatkan biasanya akan

memperlambat proses penelusuran nilai *error* yang paling kecil. Dalam penggunaan metoda ini para peneliti biasanya menggunakan cara memperbesar nilai dari *Variabel Learning Rate* saat hasil yang dicapai jauh dari target dan sebaliknya saat hasil yang dicapai dekat dengan nilai target. Secara perhitungan metoda ini memang tidak begitu jauh dari metoda yang telah dijelaskan sebelumnya, namun perbedaannya adalah seperti berikut.

$$W_{k+1} = W_k - \alpha_{k+1} \,.\, g_k + \mu \,.\, W_{k-1} \tag{12.22}$$

$$\alpha_{k+1} = \beta \,.\, \alpha_k \tag{12.23}$$

$$\beta = \begin{cases} 0.7 & \text{jika nilai } new\ error > 1.04\ (old\ error) \\ 1.05 & \text{jika nilai } new\ error < 1.04\ (old\ error) \end{cases} \tag{12.24}$$

Conjugate Gradient Back Propagation (CGX)

Conjugate Gradient Back Propagation memiliki perbedaan dibandingkan dengan metoda GD yaitu pada saat melakukan proses *update*, di mana untuk metoda GD proses tersebut dilakukan setiap penggunaan rumus sedangkan pada proses CGX, *update* dilakukan setiap iterasi dilakukan. Berikut ini merupakan proses *update* nilai *weight*.

Di mana: $$W_{k+1} = W_k + a.p_k \tag{12.25}$$

$$p_k = -g_k + \beta_k . p_{k-1} \tag{12.26}$$

$$\beta = \frac{\Delta g_{k-1}^T . g_k}{g_{k-1}^T g_{k-1}} \tag{12.27}$$

dan $$g_{k-1}^{T} = g_{k}^{T} - g_{k-1}^{T} \quad (12.28)$$

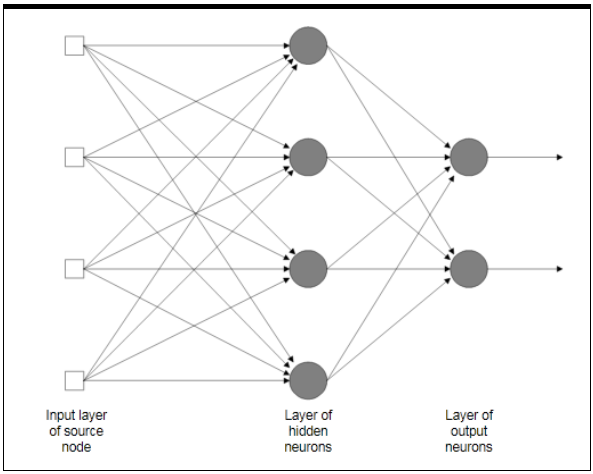
Quasi-Newton Back Propagation (BPGS)

Metoda *Newton* ini merupakan improvisasi dari metoda CGX, dimana pencapaian nilai konvigurasi dapat dilakukan lebih cepat. Metoda yang digunakan adalah sebagai berikut:

$$W_{k+1}^{-1} = W_{k} - A_{k} . g_{k} \quad (12.29)$$

A_k merupakan Hessian Matrix untuk nilai *weight* dan *Bias*.

Arsitektur dan proses belajar yang sederhana sangat memudahkan untuk dipelajari. Arsitektur JST Propagasi Balik diilustrasikan oleh Gambar 12.24.



Gambar 12.24 MLP Dengan Empat Node pada *Input Layer*, Satu *Hidden Layer* dengan Empat *Node* dan Dua *Node* pada *Output Layer*

Banyak algoritma pelatihan yang diusulkan untuk melatih MLP. Salah satu yang populer adalah algoritma pelatihan *back propagation* atau propagasi balik. Sesuai dengan namanya, algoritma ini melakukan dua tahap perhitungan, yaitu: perhitungan maju untuk menghitung galat antara keluaran aktual dan target; dan perhitungan mundur yang mempropagasikan balik galat tersebut untuk memperbaiki bobot-bobot sinaptik pada semua neuron yang ada. Berikut ini adalah langkah-langkah dari algoritma pelatihan propagansi balik.

Algoritma Pelatihan JST Propagasi Balik

1. Definisi masalah, misalkan matriks masukan (P) dan matriks target (T).
2. Inisialisasi, menentukan bentuk jaringan dan menetapkan nilai-nilai bobot sinaptik (W1 dan W2) dan *learning rate* (lr).
3. Pelatihan Jaringan.
4. Langkah-langkah sebelumnya adalah untuk satu kali siklus pelatihan (satu *epoch*). Proses pelatihan diulang sampai jumlah *epoch* tertentu atau telah tercapai yang diinginkan.
5. Hasil akhir pelatihan jaringan adalah didapatkannya bobot-bobot W1 dan W2 yang kemudian disimpan untuk pengujian jaringan.

Pada prakteknya, perancangan arsitektur JST Propagasi Balik sangat tergantung pada masalah yang akan diselesaikan. Untuk himpunan masukan berdimensi besar atau jumlah kelas keluaran yang dinginkan besar, maka diperlukan jumlah node pada hidden layer yang lebih banyak.Atau diperlukan lebih dari satu hidden layer, tetapi tentu saja ada batas optimumunya untuk kedua parameter tersebut.

6. JST Probabilistik

JST Probabilistik merupakan model yang dibentuk berdasarkan penaksir fungsi padat peluang. Model ini memberikan unjuk kerja pengklasifikasian yang sangat baik dan cepat dalam pelatihan karena dilakukan hanya dalam satu tahap pelatihan. Suatu parameter penghalus tunggal, σ, mengendalikan jaringan dari pengaruh tiap pola pada penaksiran fungsi padat peluang.

(* Tahap Pertama *)
For setiap pola ρ_i
begin
$w_i = \rho_i$;
Bentuk unit pola dengan masukan vektor bobot w_i;
Hubungkan unit pola pada unit penjumlah untuk masing-masing kelas;
end;

Tentukan konstanta $|C_k|$ untuk setiap unit penjumlah;

(* Tahap ke dua *)

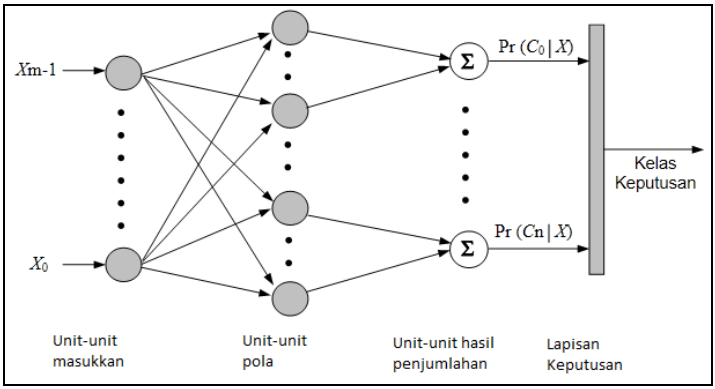
For setiap pola ρ_i

begin k= kelas ρ_i;

Cari jarak, d_i, dengan pola terdekat pada kelas k;

d_{tot} [k] = d_{tot} [k] + d_i;

end;



Gambar 12.25 Arsitektur JST Probabilistik

Berdasarkan pembahasan sebelumnya, maka perbedaan mendasar antara MLP dengan JST Probabilistik adalah sebagai berikut:

Tabel 12.1 Perbandingan MLP dan JST Probabilistik

MLP	JST Probabilistik
Jumlah neuron tetap, tetapi	Jumlah neuron bertambah

bobot-bobot sinaptiknya berubah	sebanding dengan banyaknya vektor *input*, tetapi bobot-bobotnya sinaptiknya tetap.
proses pelatihan memerlukan waktu yang lama karena membutuhkan banyak iterasi pengubahan bobot sampai mencapai *steady state*	Proses pelatihan memerlukan waktu yang sangat singkat, karena hanya dilakukan satu tahap penelitian saja.
Memerlukan memori yang kecil karena hanya menyimpan bobot-bobot sinaptik hasil penelitian	Memerlukan memori yang besar sebanding dengan jumlah vektor pola latihnya.

A. Implementasi

Secara garis besar implementasi JST dibagi menjadi empat kelompok besar:

1. Data Filtering

- Contoh pengenalan suara yang antar muka pada animasi ekspresi wajah secara real, aplikasi *neural network* pada pengenalan pola tanda tangan.

2. Peramalan

- Contoh deteksi kanker atau penyakit, aplikasi hopfield *neural network* untuk prakiraan cuaca.

3. Klasifikasi

- menggunakan nilai *input* tertentu

- Contoh klasifikasi bau aroma (hidung bionik), klasifikasi kadar udara bersih, identifikasi jenis plankton.

4. Data Association

- menggunakan nilai *input* dan mengenal data yang mengalami kesalahan.
- Contoh klasifikasi masyarakat miskin, klasifikasi curah hujan, aplikasi hopfield *neural network* untuk prakiraan cuaca.

Perkembangan aplikasi teknologi JST sangat pesat dan telah diterapkan pada hampir semua bidang keilmuan. Seperti pada bidang komunikasi multimedia, sistem tenaga listrik, kedokteran, ekonomi, dan industri. Secara garis besar implementasi JST dibagi menjadi empat kelompok besar yaitu data *filtering*, peramalan, klasifikasi (menggunakan nilai *input* tertentu), dan data *association* (menggunakan nilai *input* dan mengenal data yang mengalami kesalahan).

Pada bidang kedokteran, komunikasi multimedia, biologi, dan industri, JST diperlukan dalam permasalahan pengolahan citra sebagai alat bantu penambilan keputusan dan alat bantu identifikasi. Penerapan dalam teknik sistem tenaga, berkaitan dengan masalah pengaturan atau kendali sistem, serta sebagai alat prediksi atau estimasi (yang biasanya menggunakan metode statistik). Dalam bidang ekonomi dan manajemen, JST juga berguna sebagai alat bantu prediksi. Berikut beberapa contoh aplikasi yang berkaitan dengan JST:

- Dalam Data *Filtering*
  - Pengenalan suara yang diantar-mukakan pada animasi ekspresi wajah secara waktu nyata
  - Aplikasi *multilayer* perceptron *neural network* pada sistem pendeteksi gangguan (ids) berdasarkan anomali suatu jaringan
  - Aplikasi *neural network* pada pengenalan pola tanda tangan
- Dalam Peramalan
  - Peramalan cuaca
  - Menemukan saham terbaik di pasar bursa saham
  - Deteksi kanker
  - Deteksi kelainan otak
  - Aplikasi *hopfield neural network* untuk prakiraan cuaca
- Dalam Klasifikasi
  - Klasifikasi kualitas *printed circuit board* (PCB)
  - Identifikasi jenis plankton
  - Analisis pengenalan teks cetak
  - Klasifikasi bau aroma (hidung bionik)
  - Klasifikasi kadar timbal yang terdapat pada air kali
  - Klasifikasi kadar udara bersih
- Dalam Data *Association*
  - Klasifikasi masyarakat miskin
  - Klasifikasi curah hujan
  - Pelepasan beban (*load shedding*)

Contoh Aplikasi pada Komunikasi Multimedia, yaitu Pengenalan Suara yang diantarmukakan pada animasi Ekspresi Wajah secara Real-Time

Kemajuan teknologi multimedia yang berkembang pesat seiring perkembangan komputer, serta kian banyaknya penggunaan sistem komunikasi berbasis multimedia. Kombinasi teks, audio, gambar, dan video menjadi media untuk mengirim informasi secara efisien dan efektif. Secara umum, kombinasi dari sinyal wicara ataupun akustik dengan sinyal gambar dari video lebih banyak memberikan informasi daripada sinyal wicara saja. Sebagai contoh, suara manusia jika disertakan wajahnya akan lebih mudah memahami informasi yang disampaikan. Sedangkan informasi yang disediakan hanya dengan sinyal suara kadang masih membingungkan untuk dianlisa dan divisualisasikan.

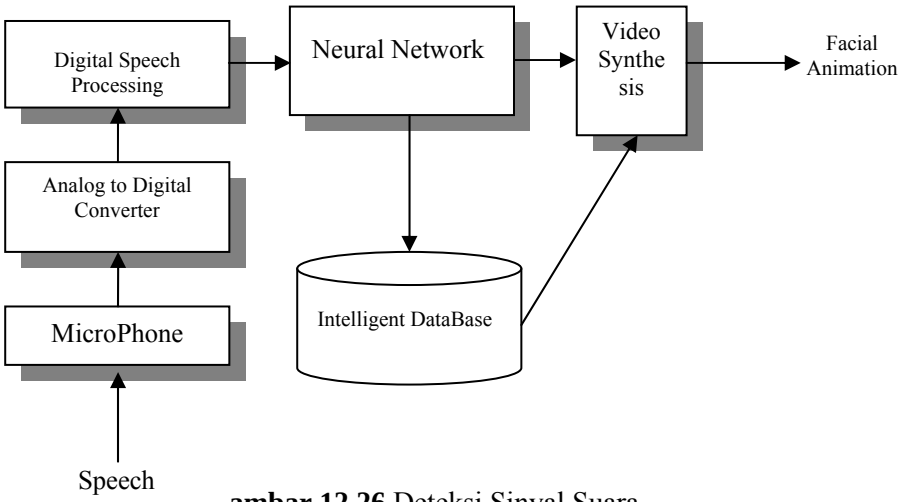
Di lingkungan industri, hampir semua strategi dalam komunikasi *audio-visual* dapat ditujukan untuk meningkatkan produktifitasnya. Pada komunikasi *audio-visual*, sistem dirancang agar mampu memahami sinyal suara yang muncul dengan bentuk visualisasi ekspresi wajah manusia. Pada sistem ini sinyal suara diolah dengan metode *signal processing*, kemudian sinyal akan dikodekan dan disinkronisasi dengan animasi wajah yang disintesis dengan teknik komputer grafik atau *image processing*. Sedangkan untuk model *coding*, gerakan mulut sebagai parameternya

akan diukur pada *encoder*. Teknik *coding* yang digunakan adalah teknik *backpropagation neural network.* Metode tersebut, dapat melakukan sinkronisasi antara sinyal masukan berupa sinyal suara dan animasi ekspresi wajah manusia (gerakan mulut) secara *real time*. Berikut merupakan garis besar langkah dan batasan yang digunakan adalah:

1. Teknik*backprogration neural nertwork* digunakan untuk mengkonversi dari sinyal suara kedalam visualisasi ekspresi wajah.
2. Karakteristik propagasi tidak dimasukkan dalam sistem.
3. Derau pada sinyal masukan suara diabaikan karena derau ini dianggap kecil sehingga tidak mempengaruhi sistem, sekitarnya dalam kondisi sunyi.

Arsitektur Sistem

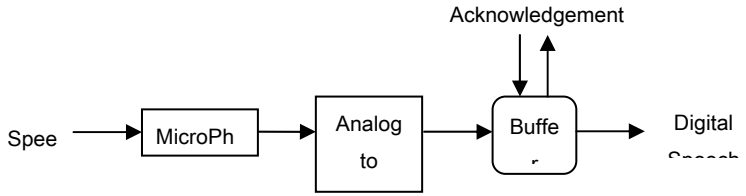
Sistem yang akan diimplementasikan untuk mendeteksi sinyal suara dan mengkonversi ke dalam bentuk animasi ekspresi akan digambarkan sebagai berikut.



ambar 12.26 Deteksi Sinyal Suara

Konversi dari Sinyal Analog ke Sinyal Digital (ADC)

Sinyal suara akan dimasukan melalui **Microphone** yang berfungsi mengubah sinyal suara analog menjadi sinyal tegangan analog, kemudian diubah menjadi sinyal digital, diagram bloknya seperti berikut.



Gambar 12.27 ADC

Buffer data merupakan tempat untuk menampung data (sinyal suara digital), untuk menghindari kehilangan data akibat kecepatan pencuplikan data yang tidak sama dengan peralatan pengambilan data. Sinyal *acknowledgement* (ACK) merupakan media komunikasi yang berada diantara peralatan ADC dengan peralatan untuk mengambil data, sehingga saat dilakukan pengambilan data digital berupa sinyal suara, maka terlebih dahulu akan mengirimkan sinyal ACK untuk menanyakan kesiapan pengambilan data ke sistem.

Digital Speech Processing

Pada blok ini, proses yang terjadi adalah mengubah sinyal suara berbasis ranah waktu (*Time domain*) menjadi sinyal berbasis ranah frekuensi (*Frequency domain*) bertujuan untuk mempermudah penganalisahan. Metode yang digunakan adalah *Fast Fourier Transform* (FFT) dengan 128 poin masukan, dengan pertimbangan kecepatan dan kemudahan dalam merealisasikan.

Jaringan Saraf Tiruan untuk Pengolahan Citra

Blok JST berfungsi untuk pengenalan suara (*Speech Recognition*) sekaligus mengidentifikasinya. Data yang masuk, terlebih dahulu harus dinormalisasi untuk mempercepat proses komputasi, sehingga target yang diinginkan cepat tercapai. Metode untuk normalisasi sebagai berikut, dengan N merupakan masukan data yang mempunyai nilai A_i, i bernilai 1 sampai N,

maka nilai data yang telah dinormalisasi tersebut adalah (μ adalah mean dari N masukan data)

$$A = \frac{A_i - \mu}{\sum_{j=1}^{N} A_j} \tag{12.30}$$

Intelligence Database

Database diperlukan untuk mengkonversi keluaran dari JST yang berupa angka-angka rumit menjadi data yang mudah dipahami serta memberikan informasi data yang diperlukan oleh blok selanjutnya. Selain itu, blok ini juga digunakan untuk mengontrol blok selanjutnya agar dapat menghasilkan gambar yang sesuai dengan masukan suara.

Video Synthesis

Blok terakhir ini berfungsi untuk mengubah gambar diam (*Still Image*) menjadi gambar bergerak seperti video. Pada proses ini, identifikasi dilakukan untuk kasus berupa wajah seseorang. Penggunaan proses *maping* untuk memisahkan bagian atau arena mana yang merupakan daerah *Rigid* dan *Non-Rigid*. Setelah proses *maping* maka dilakukan proses *Image Warping* untuk sinteksis gambar agar menjadi sederatan gambar, kemudian gambar ini akan dipakai sebagai video yang ditampilkan pada monitor. Bagian *maping* dari suatu wajah ada sepuluh area, dimana masing-masing area akan digerakkan dengan metode transformasi yang meliputi rotasi, translasi, dan perskalaan

(*scaling*), sehingga pengenalan suara berhasil di-*interface*-kan pada suatu animasi ekspresi wajah secara *real-time*, seperti tujuan penulis.

12.6 Keuntungan dan Kerugian *Neural Networks*

a. Keuntungan dari *Neural Networks*

Penggunaan *Neural Network* menawarkan kemampuan yang berguna, yaitu:

1. Nonlinearity

Nonlinearity merupakan jaringan saraf buatan nonlinear. *Neural Networks* membuat koneksi dalam dari saraf nonlinear. Nonlinear merupakan bagian yang penting dalam mekanisme fisik yang bertanggung jawab atas bagian sinyal *input* (contoh: sinyal suara) diturunkan menjadi nonlinear.

2. Input-Output Mapping

Jaringan dipresentasikan dengan contoh-contoh yang diambil secara acak dari beberapa data dan berat *synaptic* (paramater bebas) yang dimodifikasi untuk meminimalisasi perbedaan antar respon yang diinginkan dan respon dari jaringan yang dihasilkan oleh sinyal *input*. Hal ini diulang hingga mencapai kondisi dimana tidak ada lagi yang mengalami perubahan yang signifikan dalam berat *synaptic*. Kemudian jaringan tersebut belajar dari contoh-contoh dengan konstruksi *input-output mapping*.

3. Adaptivity

Neural Network mempunyai kemampuan dasar untuk mengadaptasi berat synaptic untuk berubah dalam tiap lingkungannya. Biasanya, *Neural Network* dapat dilatih dengan mudah untuk menghadapi perubahan dalam lingkungan yang tidak statik. Menurut aturan umum, dapat dikatakan bahwa semakin sering kita adaptasi dalam membuat suatu sistem, lambat laun dapat menjamin bahwa sistem tetap stabil.

4. Evidental Response

Neural Network dapat didesain untuk menyediakan informasi yang tidak hanya sebatas pola biasa yang dipilih, akan tetapi juga kepercayaan dalam pembuatan keputusan. Informasi terkini dapat digunakan untuk menolak pola yang ambigu.

5. Contextual Information

Struktur dan titik aktivasi dari *Neural Network* menjadi tempat untuk mepresentasikan pengetahuan yang ada, sehingga, menjadikan setiap neuron dalam jaringan sangat berpotensial dipengaruhi dengan kegiatan global dari semua neutron yang lain di dalam suatu jaringan.

6. Fault tolerance

Neural Network yang diimplementasikan dalam bentuk *hardware* memiliki potensial untuk menurunkan *Fault Tolerance* atau kemampuan untuk menghitung secara konsisten.

7. VLSI implementability

Sejumlah besar paralel alami dari *Neural Network* membuat *Neural Network* berpotensial cepat untuk menghitung tugas-tugas

tertentu. Fungsi yang sama ini membuat *Neural Network* cukup baik untuk implementasi menggunakan teknologi Very-Large-Scale-Intergrated (VLSI). Di antara salah satu kegunaan dari VLSI itu sendiri adalah menyediakan alat dari penangkap sifat yang benar-benar kompleks dalam bentuk hierarki.

8. Uniformity of Analysis and Design

Pada dasarnya, *Neural Network* cocok secara universal untuk prosesor informasi. Fungsi ini membuktikan diri sendiri dalam berbagai cara:

- Neuron dalam bentuk satu sama lain dapat mempresentasikan sejumlah bagian-bagian yang seama ke seluruh *Neural Network*.
- Persamaan ini memungkinkan untuk membagi teori dan algoritma belajar dalam aplikasi-aplikasi berbeda dari *Neural Network*.

9. Neuro-biological Analog

Neurobiologist melihat *Artificial Neural Network* sebagai alat riset untuk mengartikan *Neurobilogical*. Di sisi lain, para ahli menggunakan *neurobiology* untuk ide baru dalam menyelesaikan masalah yang lebih kompleks.

b. Kerugian dari *Neural Network*

- *Neural Network* memiliki beberapa kelemahan, yaitu:
- Mengecilkan kelebihan-kelebihan yang memerlukan banyak perhitungan.

- Hubungan individu di antara variabel *input* dan variabel *output* tidak dibangun dari pertimbangan rancangan sehingga model cenderung.
- Menjadi kotak hitam atau *input-output* tanpa berbasis analitis.
- Ukuran contoh harus besar.

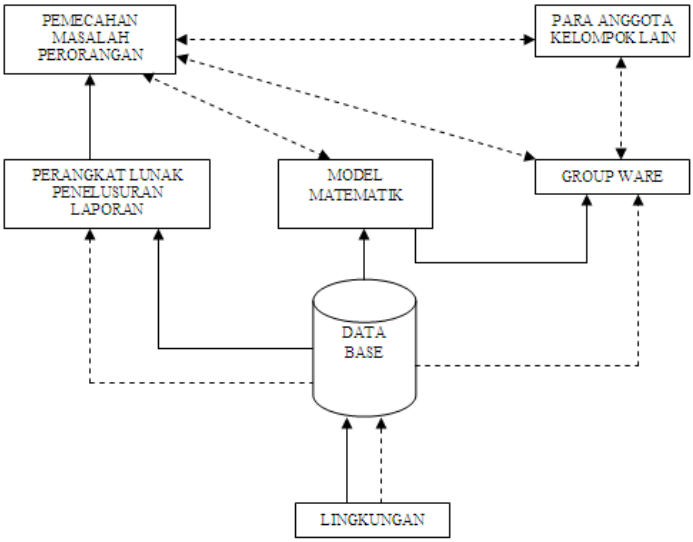
Bab 13

Decision Support System (DSS)

13.1 Pengantar

Bagaimana cara mengambil keputusan? Jawaban atas pertanyaan ini akan mempengaruhi perancangan sistem informasi di dalam komputer untuk mendukung proses dalam pengambilan keputusan (Decision Support System - DSS). Kekuatan yang memprakarsai proses pengambilan keputusan dapat berupa ketidakpuasan terhadap keadaan saat itu atau manfaat yang diharapkan dari keadaan yang baru. Dalam hal ketidakpuasan, kekuatan yang memprakarsai adalah penemuan masalah sedangkan dalam hal manfaat yang diharapkan kekuatan yang memprakarsai berasal dari penyelidikan untuk mendapat kesempatan.

Proses pengambilan keputusan dapat ditinjau dari sudut kegiatan yang terus-menerus didorong oleh tujuan mengubah sistem (perusahaan, departemen, keluarga, dan sebagainya) dari keadaannya yang sekarang menjadi keadaan yang diinginkan dengan menggunakan suatu sistem yang disebut sistem penunjang keputusan.



Gambar 13.1 *Decision Support System*

13.2 Definisi DSS

Secara umum DSS adalah sistem berbasis komputer yang interaktif, yang membantu mengambil keputusan dengan memanfaatkan data dan model untuk menyelesaikan masalah-masalah yang terstruktur. Sedangkan secara khusus DSS adalah Sebuah sistem yang mendukung kerja seorang manajer maupun sekelompok manajer dalam memecahkan masalah semi-terstruktur dengan cara memberikan informasi ataupun usulan menuju pada keputusan tertentu. DSS mendayagunakan *resources* individu-individu secara intelek dengan kemampuan komputer untuk meningkatkan kualitas keputusan. Jadi ini

merupakan sistem pendukung yang berbasis komputer untuk manajemen pengambilan keputusan yang berhubungan dengan pemecahan masalah.

Adapun menurut para ahli definisi dari DSS adalah sebagai berikut:

- **Menurut Mann dan Watson**, Sistem Penunjang Keputusan atau DSS adalah sistem yang interaktif, membantu pengambilan keputusan melalui penggunaan data dan model-model keputusan untuk memecahkan masalah-masalah yang sifatnya semi terstruktur dan tidak terstruktur.
- **Menurut Maryam Alavi dan H.Albert Napier**, Sistem Penunjang Keputusan atau DSS adalah suatu kumpulan prosedur pemrosesan data dan informasi yang berorientasi pada penggunaan model untuk menghasilkan berbagai jawaban yang dapat membantu manajemen dalam pengambilan keputusan.
- **Menurut Litle**, Sistem Penunjang Keputusan atau DSS adalah suatu sistem informasi berbasis komputer yang menghasilkan berbagai alternatif keputusan untuk membantu manajemen dalam menangani berbagai permasalahan yang terstruktur atupun tidak terstruktur dengan menggunakan data dan model.
- **Menurut Raymond Mc Leod**, Sistem Penunjang Keputusan atau DSS adalah sistem penghasil informasi spesifik yang

ditujukan untuk memecahkan suatu masalah tertentu yang harus dipecahkan oleh manajer pada berbagai tingkatan.

Dari berbagai definisi beberapa ahli sebelumnya ada satu kesamaan tentang pengertian dari DSS yaitu merupakan suatu sistem untuk membantu pemecahan sebuah masalah. Dan pemecahan masalah tersebut dapat dipicu penyelesaiannya dengan 6 pertanyaan:

- ✓ Apa (*what*)?
- ✓ Siapa (*who*)?
- ✓ Kapan (*when*)?
- ✓ Mengapa (*why*)?
- ✓ Di mana (*where*)?
- ✓ Bagaimana (*how*)?

13.3 Mengapa Menggunakan DSS?

DSS digunakan dalam sebuah perusahaan karena berbagai hal, antara lain:

a. Perusahaan beroperasi pada ekonomi yang tidak stabil.
b. Perusahaan dihadapkan pada kompetisi dalam dan luar negeri yang meningkat.
c. Perusahaan menghadapi peningkatan kesulitan dalam hal melacak jumlah operasi-operasi bisnis.

d. Sistem komputer perusahaan tidak mendukung peningkatan tujuan perusahaan dalam hal efisiensi, profitabilitas, dan mencari jalan masuk di pasar yang benar-benar menguntungkan.
e. Adanya perubahan perilaku komputasi *end-user*. Dalam hal ini *end-user* bukanlah *programmer* sehingga mereka membutuhkan alat dan prosedur yang mudah untuk digunakan dan ini dipenuhi oleh DSS.
f. Membutuhkan informasi yang akurat dan baru secara cepat.
g. DSS sering dianggap sebagai keberhasilan dalam suatu organisasi.
h. Manajemen mengamanatkan perlunya DSS dalam organisasi.
i. Perlunya penghematan biaya operasional.

13.4 Konsep DSS

Konsep DSS dimulai akhir tahun 1960 dengan *time sharing* komputer yaitu untuk pertama kalinya seseorang dapat berinteraksi langsung dengan komputer tanpa harus melalui spesialis informasi. Istilah DSS diciptakan pada tahun 1971 oleh Anthony Gory dan Scott Morton untuk mengarahkan aplikasi komputer pada pengambilan keputusan manajemen. Konsep DSS menggunakan informasi spesifik yang ditujukan untuk membantu

manajemen dalam mengambil keputusan dengan menggunakan model sebagai dasar pengembangn alternatif yang secara interaktif dapat digunakan oleh pemakai. Dari penjelasan tersebut maka dapat diketahui bahwa DSS mempunyai karakteristik tersendiri, antara lain:

a. DSS dirancang untuk membantu pengambil keputusan dalam memecahkan masalah yang bersifat semi terstruktur ataupun tidak terstruktur,
b. Dalam proses pengolahannya, DSS mengkombinasikan penggunaan model-model/teknik-teknik analisis dengan teknik pemasukan data konvensional serta fungsi-fungsi pencari/interogasi informasi,
c. DSS dirancang sedemikian rupa, sehingga dapat digunakan dengan mudah oleh orang yang tidak memiliki dasar kemampuan pengoperasian komputer yang tinggi,
d. DSS dirancang dengan menekankan pada aspek fleksibilitas serta kemampuan adaptasi yang tinggi, sehingga mudah disesuaikan dengan kebutuhan pemakai.

A. Tujuan DSS

Bila diterapkan dalam sebuah organisasi atau perusahaan tujuan utama DSS adalah membantu manajer dan orang-orang yang terlibat dalam proses pengambilan keputusan untuk meningkatkan kemampuannya dalam memutuskan pemecahan

suatu masalah. Keputusan yang dihasilkan nantinya diharapkan dapat memenuhi batasan kognitif, waktu dan ekonomis.

Menurut Holsapple dan Winston, 1996 tujuan dari DSS adalah sebagai berikut:

a. DSS membantu pengambil keputusan dalam mengenali masalah dan kemudian memformulasikan data pendukung untuk keperluan analisis dan pengambilan tindakan.
b. DSS memfasilitasi salah satu atau semua fase pengambilan keputusan agar prosesnya berjalan secara lancar dan cepat (efektif dan efisien). Fase pengambilan keputusan itu sendiri menurut Herbert A. Simon yang ditulis oleh Mc Leod (2001) adalah:
   - ***Intellegence Activity*** yaitu proses pencarian informasi dan data dari lingkungan yang berguna bagi pemecahan masalah,
   - ***Design Activity*** yaitu menemukan, mengembangkan dan menganalisa kemungkinan dari tindakan yang akan dijadikan solusi,
   - ***Choice Activity*** yaitu memilih salah satu tindakan yang telah dianalisis pada fase sebelumnya yang kemudian dijadikan sebagai alternatif solusi.
   - ***Review Activity*** yaitu mengimplementasikan solusi.
c. DSS menjadi bantuan untuk memecahkan masalah yang semi terstruktur atau yang tidak terstruktur.

d. DSS membantu dalam memanajemen informasi atau pengetahuan. Hal ini dimungkinkan karena DSS dapat memiliki kemampuan untuk menerima, menyimpan, menggunakan, menurunkan dan mempresentasikan informasi/pengetahuan yang sesuai dengan keputusan yang akan diambil.
e. DSS mendukung penilaian manajer tanpa bermaksud untuk menggantikannya.

B. Konsep Keputusan

Pengambilan keputusan merupakan hal yang pokok bagi pemegang jabatan manajer. Karena keputusan merupakan rangkaian tindakan yang perlu diikuti dalam memecahkan masalah untuk menghindari atau mengurangi dampak negatif atau untuk memanfaatkan kesempatan di dalam perusahaan. Model sistem yang dipergunakan untuk mengambil keputusan dapat bersifat tertutup atau terbuka. *Sistem pengambilan tertutup* menganggap bahwa keputusan dipisahkan dari masukan-masukan yang tidak diketahui dari lingkungannya. Dalam sistem ini pengambil keputusan dianggap:

a. Mengetahui semua alternatif dan akibat atau hasil dari masing-masing alternatif;

b. Mempunyai suatu metode (aturan, hubungan dan sebagainya) yang memungkinkan ia membuat urutan alternatif yang lebih disukainya;
c. Memilih alternatif yang memaksimalkan sesuatu seperti keuntungan, volume penjualan atau kegunaan.

Faham pengambilan keputusan yang tertutup jelas menganggap bahwa orang yang rasional secara logis menguji semua alternatif, membuat urutan berdasarkan hasilnya yang lebih disukai, dan memilih alternatif yang mendatangkan hasil terbaik.

Sistem pengambilan keputusan terbuka adalah keputusan yang dipengaruhi oleh lingkungan dan proses pengambilan keputusan selanjutnya juga mempengaruhi lingkungan tersebut. Pengambil keputusan dianggap tidak harus logis dan sepenuhnya rasional, tetapi lebih banyak menunjukkan rasionalitas hanya dalam batas-batas yang ditentukan oleh latar belakang, penglihatan alternatif-alternatif, kemampuan untuk menangani model keputusan dan sebagainya. Mengingat tujuan model tertutup telah dirumuskan dengan baik, tujuan model terbuka sama dengan tingkat keinginan sebab model terbuka dapat berubah apabila pengambil keputusan menerima bukti keberhasilan atau kegagalan. Dibandingkan dengan ketiga

anggapan model tertutup, model keputusan terbuka menganggap bahwa pengambil keputusan:

a. Tidak mengetahui semua alternatif dan semua hasil,
b. Melakukan penyelidikan secara terbatas untuk menemukan beberapa alternatif yang memuaskan,
c. Mengambil keputusan yang memuaskan tingkat keinginannya.

Model terbuka adalah dinamis atas urutan pilihan-pilihan karena tingkatan keinginan berubah menangani perbedaan antara hasil dan tingkat keinginan.

C. Jenis-Jenis Keputusan Menurut Herbert A. Simon

Menurut Herbert A. Simon jenis-jenis keputusan dalm suatu perusahaan dibedakan menjadi 2 yaitu keputusan terprogram dan keputusan tidak terprogram. Perbedaan keputusan terprogram dan tidak terprogram terlihat dari persyaratan operasionalnya yang berlainan bagi kedua jenis keputusan tersebut. Ciri-ciri keputusan terprogram dan keputusan tidak terprogram dapat diikhtisarkan sebagai berikut.

Tabel 13.1 Keputusan Terprogram dan Tidak Terprogram

Keputusan Terprogram	Keputusan Tidak Terprogram
• Berulang • Dirumuskan dengan	• Kadang-kadang • Unik

cermat • Aturan atau algoritma keputusan bagi orang bawahan untuk digunakan	• Analisis baru untuk setiap kejadian

Dengan kata lain, keputusan terprogram adalah keputusan yang dirumuskan dengan cermat dan cukup sering diulangi sehingga aturan keputusan atau algoritma keputusan dapat dirumuskan. Aturan-aturan dapat diuraikan sebelumnya, dan karena itu aturan-aturan tersebut biasanya dapat diberi kode untuk pengolahan komputer. Penggunaan komputer untuk mengolah aturan-aturan keputusan terprogram merupakan suatu pra pemilihan oleh seorang pengambil keputusan mengenai bagaimana keputusan harus diambil untuk waktu yang akan datang.Karena pengambilan keputusan itu merupakan suatu proses yang mahal ditinjau dari sudut sumber daya yang sangat langka, waktu dan tenaga manajerial, maka keputusan terprogram merupakan suatu metode yang efisien untuk menghemat sumber daya yang langka dan untuk meningkatkan produktifitas manajer.

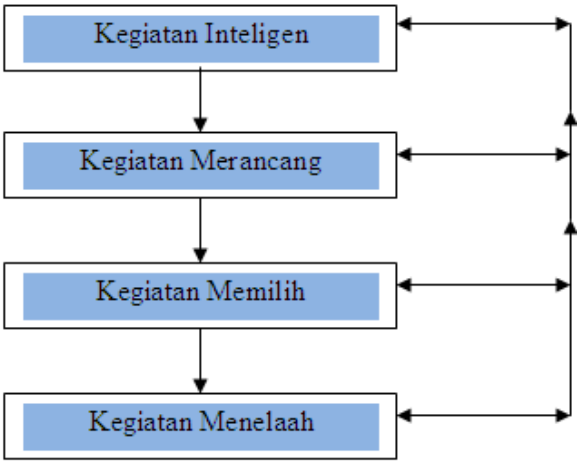
Sedangkan untuk keputusan tidak terprogram, keputusan ini tidak sering diulang atau dapat dikatakan keputusan ini sangat berbeda di setiap pengulangannya, sehingga tidak dapat

dikembangkan suatu model umum sebagai suatu dasar untuk memprogramnya.

Kegiatan pengambilan keputusan baik yang terprogram ataupun tidak terprogram dapat mengikuti proses pengambilan keputusan termasuk pemahaman, perancangan dan pemilihan. Penentuan keputusan terprogram memerlukan lebih banyak pemecahan umum daripada keputusan tidak terprogram. Untuk keputusan terprogram harus mempertimbangkan bermacam-macam kondisi sedangkan keputusan tidak terprogram hanya berhubungan dengan suatu situasi tertentu.

D. Tahapan Pengambilan Keputusan Herbert A. Simon

Ada 4 tahapan dalam pengambilan keputusan menurut Herbert A. Simon yang dapat digambarkan seperti berikut.



Gambar 13.2 Tahapan Pengambilan Keputusan

Keterangan:

- **Kegiatan Inteligen** yaitu proses pencarian informasi dan data dari lingkungan yang berguna bagi pemecahan masalah,
- **Kegiatan Merancang** yaitu menemukan, mengembangkan, dan manganalisis arah tindakan yang mungkin dapat dipergunakan. Dalam hal ini mengandung proses-proses untuk memahami masalah, untuk menghasilkan cara pemecahan masalah dan untuk menguji apakah cara pemecahan tersebut dapat dilaksanakan.
- **Kegiatan Memilih** yaitu memilih arah tindakan tertentu dari semua arah tindakan yang ada. Pilihan ditentukan dan dilaksanakan.
- **Kegiatan Menelaah** disebut juga pemahaman yaitu menyelidiki lingkungan tentang kondisi-kondisi yang memerlukan keputusan. Data mentah yang diperoleh diolah dan diperiksa untuk dijadikan petunjuk yang dapat menentukan masalahnya.

Masing-masing kegiatan tersebut saling memberi *feed back* atau umpan balik hasil keputusan. Hal ini sama seperti langkah-langkah yang disarankan Rubenstein dan Haberstroh yaitu: pengenalan masalah atau kebutuhan akan pengambilan

keputusan, analisis dan pernyataan alternatif-alternatif, pemilihan di antara alternatif-alternatif, komunikasi dan pelaksanaan keputusan, dan tindak lanjut dan umpan balik hasil keputusan.

13.5 Tingkat Pengambilan Keputusan

Pengambil keputusan mempunyai suatu cara untuk dapat memahami informasi yang menentukan efisiensi pengolahan informasinya. Pengetahuan seseorang digabungakan dengan kecakapannya mengolah informasi akan menentukan kesanggupannya mengambil keputusan. Dihadapkan dengan alternatif-alternatif, pengambil keputusan menentukan suatu tujuan, dan kemudian berusaha mencapainya dengan memilih alternatif yang terbaik berdasarkan pengetahuan yang dimilikinya.

Pengambilan keputusan merupakan suatu proses penggunaan informasi secara rasional bukan secara emosional. Dengan demikian dalam hubungan ini, kesulitan dalam pengambilan keputusan dapat diakibatkan oleh kedua-duanya.

1. Informasi yang tidak cukup, yakni informasi yang tidak benar atau tidak lengkap mengenai bermacam-macam arah tindakan alternatif yang berpengaruh pada hasil akhir,

2. Tujuan yang tidak jelas diuraikan, yakni tidak dapat menguraikan tujuan yang hasilnya lebih banyak diinginkan daripada yang lain.

Pengambilan keputusan dapat terjadi mulai dari jenis keputusan sepintas lalu yang sangat rutin (keputusan terprogram) sampai keputusan kompleks yang mempunyai pengaruh besar terhadap sistem (keputusan tidak terprogram). Untuk menggolongkannya, pengambilan keputusan dapat dibagi menjadi tiga tingkat yaitu:

1. **Pengambilan keputusan tingkat strategis**

   Keputusan yang ditandai oleh banyak ketidakpastian dan berorientasikan masa depan. Keputusan ini menentukan rencana jangka panjang yang mempengaruhi seluruh bagian perusahaan. Tujuan perusahaan ditentukan oleh beberapa strategi, oleh karena itu strategi berhubungan dengan perencanaan jangka panjang dan meliputi penentuan tujuan, penentuan kebijaksanaan, pengorganisasian, dan pencapaian keberhasilan organisasi secara menyeluruh.

2. **Pengambilan keputusan tingkat taktis**

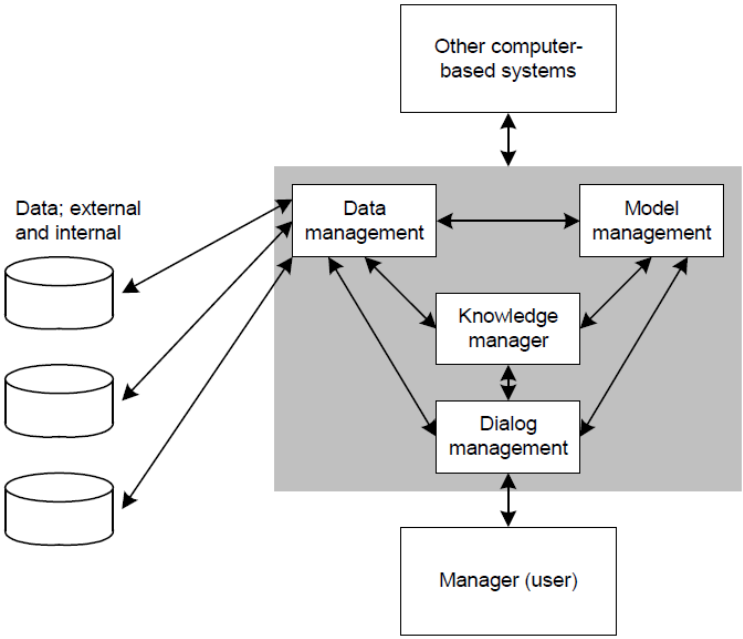
   Pengambilan keputusan tingkat taktis berhubungan dengan kegiatan jangka pendek dan penentuan sumber daya untuk mencapai tujuan. Jenis pengambilan keputusan ini berhubungan dengan bidang-bidang seperti perumusan anggaran, analisis aliran dana, penentuan tata ruang, masalah kepegawaian, perbaikan produk, serta penelitian dan pengembangan.

3. **Pengambilan keputusan tingkat teknis**

   Pada tingkat pengambilan keputusan ini standar-standar ditentukan dan hasil keputusan yang sifatnya menentukan. Pengambilan keputusan teknis adalah suatu proses untuk menjamin agar tugas-tugas khusus dapat dilaksanakan dengan cara efektif dan efisien. Pengambilan keputusan ini memerlukan diberikannya perintah-perintah khusus yang mengawasi operasi-operasinya.

13.6 Komponen DSS

Komponen yang terdapat dalam DSS adalah:



Gambar 13.3 Komponen DSS

1. **Data *Management***

   Termasuk *database*, yang mengandung data yang relevan untuk berbagai situasi dan diatur oleh *software* yang disebut *Database Management Sistem* (DBMS).

2. **Model *Management***

   Melibatkan model finansial, statistik, *management science* atau berbagai model kualitatif lainnya, sehingga dapat memberikan ke sistem suatu kemampuan analitis, dan manajemen *software* yang dibutuhkan.

3. ***Communication***

   User dapat berkomunikasi dan memberikan perintah pada DSS melalui subsistem ini. Ini berarti menyediakan antarmuka.

4. ***Knowledge Management***

   Subsistem optional ini dapat mendukung subsistem lain atau bertindak sebagai komponen yang berdiri sendiri.

13.7 Ciri, Keuntungan dan Keterbatasan DSS

1. Ciri Decision Support System

a. DSS dirancang untuk membantu pengambil keputusan dalam memecahkan masalah yang bersifat semi terstruktur ataupun tidak terstruktur.

b. Dalam proses pengolahannya, DSS mengkombinasikan penggunaan model-model/teknik-teknik analisis dengan teknik pemasukan data konvensional serta fungsi-fungsi pencari/interogasi informasi.

c. DSS dirancang sedemikian rupa, sehingga dapat digunakan dengan mudah oleh orang yang tidak

memiliki dasar kemampuan pengoperasian komputer yang tinggi.

d. DSS dirancang dengan menekankan pada aspek fleksibilitas serta kemampuan adaptasi yang tinggi, sehingga mudah disesuaikan dengan kebutuhan pemakai.
e. Menghasilkan acuan data untuk menyelesaikan masalah yang dihadapi oleh manajer yang kurang berpengalaman.
f. Fasilitas untuk mengambil data dapat memberikan kesempatan bagi beberapa manajer untuk berkomunikasi dengan lebih baik.
g. Meningkatkan produktifitas dan kontrol dari manajer.

2. Keuntungan Decision Support System

a. DSS memperluas kemampuan pengambil keputusan dalam memproses data/informasi bagi pemakainya.
b. DSS membantu pengambil keputusan dalam penghematan waktu yang dibutuhkan untuk memecahkan masalah.
c. DSS dapat menghasilkan solusi dengan lebih cepat serta hasilnya dapat diandalkan.
d. DSS mampu menyajikan berbagai alternatif.

e. DSS dapat menyediakan bukti tambahan untuk memberikan pembenaran sehingga dapat memperkuat posisi pengambil keputusan.

3. Keterbatasan DSS

a. Beberapa kemampuan manajemen dan bakat manusia yang tidak dapat dimodelkan.
b. Kemampuan terbatas pada perbendaharaan pengetahuan yang dimilikinya.
c. Proses tergantung pada perangkat lunak yang digunakan.
d. Tidak memiliki kemampuan intuisi (berpikir) seperti pada manusia.

13.8 Sistem Penunjang Keputusan Kelompok (GDSS)

Suatu sistem berbasis komputer yang mendukung kelompok-kelompok orang yang terlibat dalam suatu tugas (tujuan) bersama dan yang menyediakan *interface* bagi suatu lingkungan yang digunakan bersama atau bisa dikatakan GDSS adalah sistem pendukung keputusan kelompok yang berusaha memperbaiki komunikasi diantara para anggota kelompok dengan menyediakan lingkungan yang mendukung dan mendukung para pengambil keputusan dengan menyediakan perangkat lunak GDSS yang disebut *groupware*.

1. Nama lain dari GDSS yaitu:

a. Group Support System (GSS)
b. Computer Supported Cooperative Work (CSCW)
c. Computerized Collaborative Work Support
d. Electronic Meeting System

2. Pengaturan GDSS adalah:

a. Ruang keputusan; merupakan pengaturan untuk rapat kelompok kecil serta tatap muka. Ruangan tersebut mendukung komunikasi melalui kombinasi perabot, peralatan dan tata letak.
b. Jaringan keputusan; dalam hal ini yang dimaksud adalah LAN. Jika kelompok kecil tidak mungkin bertemu secara bertatap muka maka para abggota dapat berinteraksi melalui jaringan.
c. Pertemuan Legislatif; jika kelompok terlalu besar untuk ruang keputusan maka pertemuan legislatif diperlukan.
d. Konferensi bermedia komputer; beberapa aplikasi kantor virtual memungkinkan komunikasi antara kelompok-kelompok besar dengan anggota yang tersebar secara geografis.

DAFTAR PUSTAKA

1. Abdul Kadir & Terra Ch. Triwahyuni. *Pengenalan Teknologi Informasi.* Yogyakarta: Penerbit Andi, 2003.
2. Andri Kristanto, *Kecerdasan Buatan (Sebuah Pengantar)*, Edisi Pertama. Yogyakarta: Penerbit Graha Ilmu, 2004.
3. Anita Desiani dan M. Arhami. *Konsep Kecerdasan Buatan*. Yogyakarta: Penerbit Andi, 2006.
4. Arief Hermawan. *Jaringan Saraf Tiruan Teori dan Aplikasi.* Yogyakarta: Penerbit Andi, 2006.
5. A. Walid Salameh. A Study in Informatics and Control, Vol. 13, No.2, June 200 pp.135.
6. Castillo Enrique dan E. Álvarez. Expert Systems: uncertainty and learning. UK: Computational Mechanic Publication, 1991.
7. Douglas R. Malcolm, Jr. *Robotics An Introduction*. Boston: BostBreton Publishers, 1985.
8. Efraim Turban, Jay E. Aronson, Ting-Peng Liang. *Decision Support Systems and Intelligent Systems (Sistem Pendukung Keputusan dan Sistem Cerdas)*, Ed. Jilid 2. Yogyakarta: ANDI, 2005.

9. Dr. Ir. Endra Pitowarno, M.Eng (PENS-ITS). Introduction to Robotics. Seminar "New Concept Robotics: Robot Vision". Jakarta: Universitas Gunadarma.
10. Endra Pitowarno. *Robotika Desain, Kontrol, dan Kecerdasan Buatan*. Yogyakarta: Andi Offset, 2006.
11. Elaine Rich dan Kevin Knight. *Artificial Intelligence*. New York: Mc-Graw Hill Book Co., 1991.
12. *F. W. Glover dan M. Laguna, Tabu Search, Springer, 1998*
13. Giarratano, Joseph dan Gary Riley. *Expert System Principles and Programming (third edition)*. China Machine Press, 1998.
14. Henry Kurniawan. Simulasi Algoritma Tabu Search sebagai Protokol Routing pada Jaringan. Surabaya: STIKOMP, 2008.
15. H.P. Panggabean. Penjadwalan Job shop Statik dengan Algoritma Simulated Annealing. *Integral*, Vol. 10, No. 1, Maret 2002.
16. Jogiyanto HM. *Pengenalan Komputer*. Yogyakarta: Penerbit Andi, 1999.
17. J.E. Muñoz-Expósito, S. García-Galán, N. Ruiz-Reyes, P. Vera-Candeas. Engineering Applications of Artificial Intelligence**,** Volume 20, Issue 6, September 2007, pp. 783–793.
18. K. S. Fu dkk. Ro*botics Control, Sensing, Vision and Intelegence*. McGraw-Hill International, 1987.
19. Louis E. Frenzel Jr. *Crash Course in Artificial Intelligence and Expert System*. Indianapolis, USA: Howard W. Sams & Co.

20. Luger, George F dan William A. Stubblefield. *Artificial Intelligence Structures and Strategies for Complex Problem Solving (third edition)*. Adition Wesley Longman, 1998.
21. Martin Hewings. *Advanced Grammar In Use*, Edisi ke-6. UK: Cambridge University Press, 2001.
22. Martin T. Hagan dkk. *Neural Networks Design*. USA: PWS Publishing Company, 1996.
23. Mauridhi Hery P., Agus Kurniawan. *Supervised Neural Networks*. Yogyakarta: Graha Ilmu, 2006.
24. M. Farid Azis. *Belajar Sendiri Pemrograman Sistem Pakar*. Jakarta: PT Elex Media Komputindo, 1994.
25. Drs. Moekijat. *Pengantar Sistem Informasi Manajemen*. Bandung: PT. Remaja Rosdakarya, 1986.
26. Muhammad Arhami. *Konsep Dasar Sistem Pakar*, Ed.1. Yogyakarta: ANDI, 2004.
27. Simon Haykin. *Neural Networks A Comprehensive Foundation*, *2nd edition*. New Jersey: Prentice-Hall, Inc., 2000.
28. Sandi Setiawan. *Mengenal Network Saraf*. Yogyakarta: Penerbit Andi Offset, 1992.
29. Sandi Setiawan, *Artificial Intelligence*, Edisi pertama. Yogyakarta: Penerbit Andi Offset, 1993.
30. Siswanto. Kecerdasan Buatan. Yogyakarta: Graha Ilmu, 2010.
31. Sri Kusumadewi. *Artificial Intelligence, Teknik dan Aplikasinya*. Yogyakarta: Graha Ilmu, 2003.

32. Sri Kusumadewi. *Artificial Intelligent*, *Teknik dan Aplikasi, Edisi ke-1*. Yogyakarta: Graha Ilmu, 2003.
33. Suparman dan Marlan. *Komputer Masa Depan (Pengenalan Artificial Intelligence)*. Yogyakarta: ANDI, 2007.
34. Suparman. *Mengenal Artificial Intelligence Edisi pertama*. Yogyakarta: Penerbit Andi Offset, 1991.
35. Suyanto, ST, MSc. *Artificial Intelligence: Searching, Reasoning, Planning and Learning*. Bandung: Penerbit Informatika, 2011.
36. Tavri Niemueller dan Sumedha Deviyan, *Pemrograman Deklaratif dengan Turbo Prolog 2.0*. Jakarta: Elex Media Komputindo.
37. Tim Widyadharma. *Artificial Intelligence – An Introduction to Robotics*.
38. Tim Penerbit ANDI. *Pengembangan Sistem Pakar Menggunakan Visual Basic*, *Edisi 1*. Yogyakarta: ANDI, 2003.
39. Tjendry Harianto. *Bahasa Turbo Prolog, Edisi Pertama*. Yogyakarta: Penerbit Andi Offset, 1992.
40. Uung Ungkawa. *Bahasa Pemrograman Logika Turbo Prolog Edisi Pertama*. Yogyakarta: Penerbit Andi Offset, 1992.
41. Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springler Verlag, 1996.

Link

Dr. Arry Akhmad Arman. Orasi Ilmiah Sidang Terbuka Institut Teknologi Bandung dengan acara Peresmian Penerimaan Mahasiswa Baru ITB. [Online]. Tersedia: http://dc382.4shared.com/doc/qTk3G-uN/preview.html. [23 Agustus 2004]

Cnblogs. An Introduction To Fuzzy Control Systems. [Online]. Tersedia: http://www.cnblogs.com/T_max_Csharp/archive/2011/03/07/1973221.html. [Juni 2003]

CSGadget. Fuzzy Logic – part 4. [Online]. Tersedia: http://csgadget.wordpress.com/2010/08/18/fuzzy-logic-part-4/. [18 Agustus 2010]

Denny Hermawanto. Komputasi Sistem Kekebalan Tubuh. [Online]. Tersedia: http://id.scribd.com/doc/23176061/Komputasi-Sistem-Kekebalan-Tubuh-Artificial-Immune-System. [30 Agustus 2013]

Denny Hermawanto. Komputasi Sistem Kekebalan Tubuh. [Online]. Tersedia: http://ikc.depsos.go.id/pengantar/dennyhermawanto/denny-kskt.zip. [30 Agustus 2013]

Hadibanoe. Pengantar Immune Algorithms. [Online]. Tersedia: http://profhadibanoe.wordpress.com/2012/01/13/pengantar-immune-algorithms/. [30 Agustus 2013]

Idahceris. Fungsi Keanggotaan Logika Fuzzy. [Online]. Tersedia: http://belajaritsaja.com/sistemcerdas/fuzzy/fungsi-keanggotaan-logika-fuzzy. [14 March 2013]

Plctutor. Pengenalan Motor Listrik. [Online]. Tersedia: http://plctutor.wordpress.com/2012/03/28/pengenalan-motor-listrik/. [28 Maret 2012]

Pranoto Budi Sasongko. Cloning-Based Algorithm Dan Aplikasinya Dalam Travelling Salesperson Problem. [Online]. Tersedia: http://www.scribd.com/doc/40692839/MakalahIF2153-0708-096. [30 Agustus 2013]

Priyandari. Skema Algoritma Tabu Search. [Online]. Tersedia: http://priyandari.staff.uns.ac.id/files/2009/09/image025.gif. [30 Agustus 2013]

Radtechclass. Fluoroscopy. [Online]. Tersedia: http://www.radtechclass.com/core/fluoroscopy/. [25 Mei 2013]

Shahariz Abdul Aziz dan Jeyakody Parthiban. What do ya mean *fuzzy* ??!!. [Online]. Tersedia:

http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/sbaa/report.fuzzysets.html.

Ayi Purbasari, Iping Supriana S, dan Oerip S. Clonal Selection Algorithm:
Bio-Inspired Algorithms Sebagai Solusi Persoalan Kompleks. [Online].
Tersedia: http://yudiagusta.files.wordpress.com/2008/09/329-334-knsi2011-051-clonal-selection-algorithm.pdf. [30 Agustus 2013]

Wikipedia. Genetic Algorithm. [Online]. Tersedia: http://en.wikipedia.org/wiki/Genetic_algorithm). [30 Agustus 2013]

Algoritma Genetik. [Online]. Tersedia: http://k12008.widyagama.ac.id/ai/diktatpdf/BabIV_Algoritma_Genetika.pdf. [30 Agustus 2013]

BIOGRAFI PENULIS

Victor Amrizal lahir di Kediri, 24 Juni 1974, menyelesaikan pendidikan strata satu di STMIK Budi Luhur Jakarta program studi Teknik Informatika dan bekerja sebagai *system analyst* dan programmer di beberapa instansi/lembaga di Jakarta. Penulis menyelesaikan pendidikan strata dua di STTBI Jakarta pada program studi yang sama. Penulis sebagai staf pengajar di Universitas Muhammadiyah Prof.DR. HAMKA Jakarta dan pernah menduduki jabatan sebagai Kepala Laboratorium dan Ketua Jurusan Teknik Informatika di Fakultas Teknik Universitas Muhammadiyah Prof.DR. HAMKA Jakarta. Penulis juga sebagai staf pengajar tidak tetap di Universitas Bina Nusantara dan Universitas Paramadina Mulya. Saat ini penulis sebagai staf pengajar tetap di jurusan Teknik Informatika Fakultas Sains dan Teknologi UIN Syarif Hidayatullah Jakarta.

Qurrotul Aini lahir di Malang, 25 Maret 1973, menyelesaikan pendidikan strata satu di Universitas Brawijaya Malang Fakultas Teknik jurusan Teknik Elektro. Penulis bekerja selama dua tahun di PT. IPTN Bandung, setelah itu menjadi staf pengajar tidak tetap di salah satu perguruan tinggi swasta di Malang. Penulis menyelesaikan pendidikan strata dua di Institut Teknologi Sepuluh Nopember (ITS) Surabaya jurusan Teknik Elektro – Telekomunikasi Multimedia. Sejak tahun 2006 hingga 2008, penulis sebagai staf pengajar di Politeknik Negeri Jakarta dan salah satu perguruan

tinggi swasta. Saat ini penulis sebagai staf pengajar tetap di Fakultas Sains dan Teknologi UIN Syarif Hidayatullah Jakarta.