UNIVERSITAS INDONESIA

IDENTIFIKASI FITUR PRIMITIF SECARA OTOMATIS PADA *PART* BERBASIS MODEL FASET

TESIS

MUIZUDDIN AZKA

0906651416

FAKULTAS TEKNIK

PROGRAM STUDI TEKNIK MESIN

DEPOK

JULI 2012

UNIVERSITAS INDONESIA

IDENTIFIKASI FITUR PRIMITIF SECARA OTOMATIS PADA *PART* BERBASIS MODEL FASET

TESIS

Diajukan sebagai salah satu syarat untuk memperoleh gelar Magister Teknik

MUIZUDDIN AZKA

0906651416

FAKULTAS TEKNIK

DEPARTEMEN TEKNIK MESIN

PROGRAM PASCA SARJANA

DEPOK

JULI 2012

HALAMAN PERNYATAAN ORISINALITAS

Tesis ini adalah karya saya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar.

Nama	**:**	**Muizuddin Azka**
NPM	**:**	**0906651416**
Tanda Tangan	**:**	
Tanggal	**:**	16 Juli 2012

HALAMAN PENGESAHAN

Tesis ini diajukan oleh :

Nama	:	Muizuddin Azka
NPM	:	0906651416
Program Studi	:	Departemen Teknik Mesin
Judul	:	Identifikasi Fitur Primitif secara Otomatis pada *Part* Berbasis Model Faset

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Magister Teknik pada Program Studi Teknik Mesin, Fakultas Teknik, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing	: Dr. Ir. Gandjar Kiswanto, M.Eng.	(.....)
Penguji	: Prof. Dr. Ir. Tresna P. Soemardi, M.Si., S.E.	(.....)
Penguji	: Dr. Ario Sunar Baskoro, S.T., M.T., M.Eng.	(.....)
Penguji	: Jos Istiyanto, S.T., M.T., Ph.D.	(.....)
Penguji	: Yudan Whulanza, S.T., M.Sc., Ph.D.	(.....)

Ditetapkan di : Depok

Tanggal : 16 Juli 2012

KATA PENGANTAR

Puji syukur kehadirat Allah SWT. atas keberhasilan dalam kegiatan penelitian sebagai bagian dari persyaratan kelulusan untuk program magister teknik DTM UI. Di dalam kesempatan kali ini, penulis mengucapkan terima kasih kepada :

1. Dr. Ir. Gandjar Kiswanto, M.Eng. selaku dosen pembimbing utama yang senantiasa memberikan waktu, semangat, arahan, dan dorongan dalam menyelesaikan penelitian ini.
2. Seluruh dewan penguji tesis, staf pengajar, laboran, dan karyawan di lingkungan Departemen Teknik Mesin atas bantuan akademis, teknis, dan administrasi selama ini.
3. Seluruh mahasiswa pascasarjana angkatan 2009 dan 2010 yang selalu memberikan semangat selama menjalankan penelitian ini.
4. Ir. Swarsono, M.T., Dede Lia Zariatin, S.T., M.T., Riky Adhiharto, S.T., Aida Mahmudah, S.T., KGS. M. Ismail, S.T. dan para peneliti muda lainnya khususnya di lingkungan laboratorium manufaktur atas sharing ilmu pengetahuan dan pengalaman di dalam penelitian selama ini.
5. Rekan kerja di BPPT yang selalu membantu dari segi dukungan moril dan administrasi selama menjalankan tugas belajar.
6. Terkhusus istri Ratna Nurmayni dan putri Talita Aisha Syakira tercinta yang senantiasa memberikan dukungan dan semangatnya serta keluarga besar M. Soleh Rosyidi dan Agus Saparudin atas do'a restu yang diberikan selama ini.

Dengan selesainya kegiatan penelitian ini, penulis banyak memperoleh bekal kemampuan untuk dipergunakan di dalam kegiatan penelitian selanjutnya. Demikian yang dapat penulis sampaikan dan atas kesalahan yang pernah penulis lakukan, mohon penulis haturkan maaf yang sedalam-dalamnya.

Depok, Juli 2012

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

==========

Sebagai sivitas akademika Universitas Indonesia, saya yang bertanda tangan di bawah ini :

Nama : Muizuddin Azka

NPM : 0906651416

Program Studi : Pasca Sarjana Teknik Mesin

Departemen : Teknik Mesin

Fakultas : Teknik

Jenis Karya : Tesis

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Nonekslusif (*Non-exclusive Royalty Free Right*)** atas karya ilmiah saya yang berjudul :

Identifikasi Fitur Primitif secara Otomatis pada *Part* Berbasis Model Faset

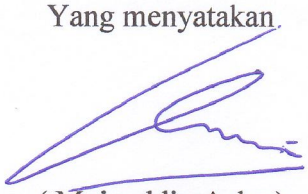
beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Nonekslusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/format-kan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan tugas akhir saya selama mencantumkan nama saya sebagai penulis/pencipta dan pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok

Pada tanggal : 16 Juli 2012

Yang menyatakan



(Muizuddin Azka)

ABSTRAK

Nama : Muizuddin Azka

Program Studi : Program Pasca Sarjana / Magister Teknik DTM UI

Judul : Identifikasi Fitur Primitif secara Otomatis pada *Part* Berbasis Model Faset

Teknologi pengenalan fitur telah berkembang seiring dengan proses pengintegrasian CAD/CAPP/CAM. Aplikasi pendeteksian fitur secara otomatis dengan berbasis model faset diharapkan mampu untuk membantu mempercepat aktivitas perancangan proses manufaktur semisal seting *tool* yang akan digunakan atau proses *machining* yang dibutuhkan pada berbagai fitur yang berbeda. Fitur suatu *part* pada bidang dapat dideteksi dengan mengaplikasikan metode *slicing* dan teknik pengelompokan faset triangulasi yang berdekatan. Identifikasi jenis fitur ini dikembangkan dengan aturan berdasarkan arah vektor normal pada grup suatu fitur tersebut. Untuk mengidentifikasi fitur pada berbagai bidang suatu *part*, dengan bantuan transformasi rotasi akan merotasi *part* sehingga bidang vektor normal terluar seakan berada pada posisi bidang referensi. Hasil penelitian ini menunjukkan bahwa dengan pendekatan ini dapat mengidentifikasi fitur primitif secara otomatis meliputi : ***pocket***, ***cylindrical*** dan ***profile*** pada berbagai bidang suatu *part*.

Kata kunci : pengenalan fitur, pengelompokan, model faset.

ABSTRACT

Name : Muizuddin Azka

Major : Post Graduate / Master of Engineering DTM UI

Title : Automatic Part Primitive Feature Identification Based on Faceted Models

Feature recognition technology has been developed along with the process of integrating CAD/CAPP/CAM. Automatic feature detection applications based on faceted models expected to speed up the manufacturing process design activities such as setting tool to be used or required machining process in a variety of different features. Features of a part in the plane can be detected by applying method of slicing and the technique of grouping adjacent triangles. Identify type of feature is developed by the rule based on normal vector direction of the group a feature. In order to identify features of various planes of a part, rotation transformation will help to rotate as if normal vector plane is on the outer part of the reference plane. The results showed that this approach could identify the primitive features automatically covered : pocket, cylindrical and profile in various plane on part.

Keywords : feature recognition, grouping, faceted models.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

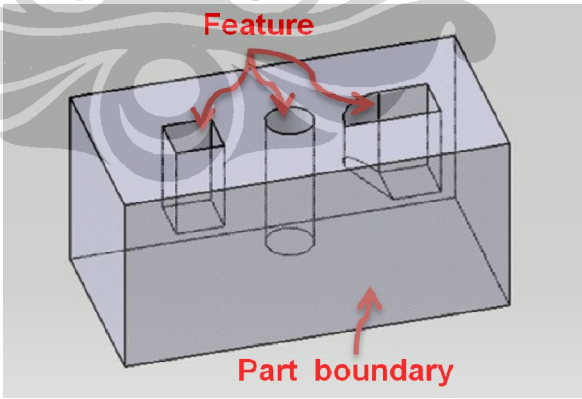
Pengenalan fitur (*feature recognition*) secara otomatis merupakan sebuah pengembangan teknologi terbaru untuk mengintegrasikan CAD/CAPP/CAM. Tata cara pengenalan fitur perlu dilakukan dalam perancangan proses produksi agar langkah-langkah yang berkaitan dengan aktivitas perancangan proses dapat dibuat. Berbagai kaidah telah dibangun untuk mengembangkan algoritma pengenalan fitur secara otomatis, secara sederhana deteksi fitur dapat didefinisikan sebagai proses ekstraksi fitur dari model CAD suatu *part.* Aplikasi pengenalan fitur dapat digunakan dalam proses mendesain fitur, menganalisa fitur, dan memanufaktur fitur seperti pada CAD, CAE, dan CAM.

Deteksi pengenalan fitur pada sebuah *part* merupakan bagian terpenting dari optimasi perancangan proses pada CAPP dan CAM. Prinsip dasar dari pengenalan fitur adalah untuk mempercepat sistem pengambilan keputusan pada aktivitas perancangan proses manufaktur. Hal ini akan membantu menentukan proses apa yang dibutuhkan, *tool* apa yang digunakan, dan lainnya. Sehingga, pengenalan fitur pada *part* atau model akan menentukan langkah selanjutnya secara efisien. Identifikasi fitur akan memberi arti konseptual pada karakteristik *part* dalam bentuk yang dikenali dan bermakna, hal ini agar dapat menangkap makna pada sebuah *part.*

Didalam penelitian sebelumnya, V.B. Sunil dan S.S. Pande [1] mendesain dan mengimplementasikan sistem pengenalan fitur secara otomatis *freeform surface* model CAD pada *sheet metal* yang direpresentasikan dalam format STL. Metodologi yang dikembangkan memiliki tiga langkah utama yaitu : *preprocessing* model STL, segmentasi area dan pengenalan fitur secara otomatis. Sementara Risal Abu dan Masine [2] mendeteksi pengenalan fitur secara otomatis dengan model B-rep menggunakan pendekatan yang berdasarkan sifat-sifat tertentu sebuah fitur seperti jumlah bidang (TNOF), jumlah sudut (TNOE), jenis bidang (TOF), dan sifat-sifat fitur seperti *protrusion/depression* (P/D).

G. Kiswanto dan A. Yahya [3] mengembangkan identifikasi *Close Bounded Volume* (CBV) pada model faset 3D kompleks melalui metode *Paired Normal Vectors Bucketing* (PNVB) untuk perencanaan *tool path* pada proses permesinan awal (*roughing*) *multi axis*. Raymond dkk. [4] mencoba algoritma untuk mengidentifikasikan fitur dengan menerapkan representasi *octree* dari model B-rep untuk mendukung logika geometrik yang diperlukan untuk mencari lokasi fitur pada proses perakitan.

Mangesh P. Bhandarkar dan Rakesh Nagi [5] menerapkan algoritma pengenalan fitur yang berorientasi pada sistem ekstraksi dengan menggunakan model STEP. Pendekatan sekuensial digunakan untuk klasifikasi yang ada pada fitur kemudian dilakukan proses ekstraksi untuk bentuk model prismatik yang dilakukan dengan perancangan operasi *milling*. C. Weber, S. Hahmann, H. Hagen [6] menyajikan teknik baru untuk mendeteksi fitur yang tajam pada *point cloud* geometri menggunakan *gauss map clustering* dan proses seleksi iterasi yang lebih tepat. G. Kiswanto dan A. Yahya [7] mencoba pengembangan algoritma untuk menentukan titik kontak pahat (*cutter contact point*) baik untuk permesinan awal (*roughing*) maupun akhir (*finishing*). G. Kiswanto, R. Widyanto and P.E. Kreshna [10] mengembangkan identifikasi fitur *surface* pada bentuk *flat, saddle, convex*, dan *concave* dengan mengkombinasikan metode Gauss Bonnet dan Spherical Image. Contoh fitur pada suatu *part* dapat dilihat pada gambar 1.1 berikut ini.



Gambar 1.1 Fitur pada suatu *part*

1.2 Perumusan Masalah

Untuk mempercepat sistem pengambilan keputusan pada aktivitas perancangan proses manufaktur dibutuhkan pengidentifikasian fitur secara otomatis yang mana :

1. Menghemat waktu proses manufaktur dibandingkan dengan tanpa pengenalan fitur sehingga menjadi lebih efisien.
2. Mengurangi tenaga operator yang harus mengoperasikan mesin pada seting *tool* yang digunakan dan atau proses *machining* yang dibutuhkan.
3. Menghindari resiko kesalahan dalam proses manufaktur yang disebabkan oleh pergantian operator.

1.3 Tujuan Penelitian

Penelitian pengenalan fitur ini mempunyai tujuan untuk mengembangkan algoritma identifikasi fitur primitif berbasis model faset secara otomatis pada berbagai bidang suatu *part* sebagai bagian dari pengembangan CAD/CAE/CAM serta sarana informasi proses *machining* yang dibutuhkan dan atau *tool* yang akan digunakan pada perancangan proses manufaktur yang efisien.

1.4 Batasan Masalah

Dengan mempertimbangkan waktu, tenaga dan cakupan penelitian, maka didalam penelitian ini ditetapkan suatu batasan masalah sebagai berikut :

1. Data yang digunakan berupa file STL yang didapat dengan menggunakan software CAD 3D.
2. Algoritma yang dikembangkan dengan mendeteksi fitur pada suatu model *part* dengan proses *slicing* dan *adjacent triangles* pada suatu bidang.
3. Penggunaan transformasi rotasi untuk membantu merotasi bidang suatu *part* sehingga seakan berada pada bidang referensi sebagai suatu proses untuk dilakukan proses deteksi fitur.
4. Penentuan jenis fitur primitif pada suatu model *part*, meliputi : *pocket, cylindrical,* dan *profile* dengan menggunakan karakteristik arah vektor normal pada fitur tersebut.

1.5 Metodologi Penelitian

Didalam melakukan penelitian identifikasi fitur primitif secara otomatis, metodologi yang dilakukan adalah sebagai berikut :

1. Studi literatur.
2. Pengembangan algoritma identifikasi fitur primitif.
3. Simulasi GUI pada pemrograman.

1.6 Sistematika Penulisan Laporan

Laporan penelitian ini disusun dengan sistematika penulisan sebagai berikut :

BAB 1. PENDAHULUAN

Bab ini berisi tentang ide, perumusan masalah, tujuan penelitian, batasan masalah dan metodologi penelitian

BAB 2. PENGENALAN FITUR DAN MODEL FASET

Bab ini berisi tentang dasar teori, konseptual dan notasi yang menjadi landasan didalam pemikiran pada pembuatan sistem dan algoritma.

BAB 3. PENGEMBANGAN ALGORITMA PENGENALAN FITUR

Bab ini berisi tentang kerangka pemikiran dan konsep dalam pengembangan algoritma yang dijalankan.

BAB 4. HASIL IMPLEMENTASI ALGORITMA DAN ANALISA

Bab ini berisi tentang implementasi algoritma dan analisa hasil. Pada bab ini banyak berisi tentang hasil identifikasi fitur dan simulasi GUI.

BAB 5 KESIMPULAN DAN SARAN PENELITIAN LEBIH LANJUT

Bab ini berisi tentang kesimpulan sebagai hasil dari tujuan penelitian dengan saran untuk dilakukan penelitian selanjutnya.

DAFTAR PUSTAKA

LAMPIRAN

Lampiran banyak diisi dengan hasil pemrograman komputer.

BAB II
PENGENALAN FITUR DAN MODEL FASET

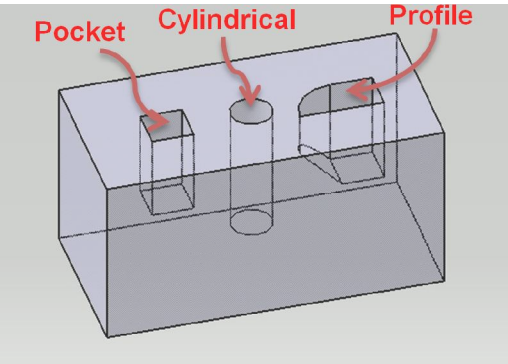
2.1 Pengenalan Fitur

2.1.1 Definisi Fitur

Istilah fitur mempunyai makna yang ambigu, tidak ada definisi yang khusus untuk itu, artinya tergantung pada konteks yang digunakan. Sebuah fitur dapat digambarkan sebagai bagian dari model *part* 3D yang sangat penting dalam pengenalan produk seperti desain, simulasi teknik, atau manufaktur. Dengan cara yang sama perakitan dibuat dari beberapa *part*, dan *part* terdiri dari fitur yang berbeda. Proses ekstraksi fitur dari *part* model CAD ini disebut pengenalan fitur (*feature recognition*). Fitur yang diekstraksi dapat menjadi desain fitur, analisis fitur atau manufaktur fitur. Jadi, Pengenalan fitur fungsinya khusus untuk wilayah penerapan pada CAD, CAE dan CAM.

Ada beberapa tahap yang berbeda dalam metode pengenalan fitur yaitu : Pertama, mengekstrak seperangkat bidang atau elemen geometris / topologi lainnya yang berpotensi membentuk suatu fitur. Langkah ini melibatkan pembentukan *sub set* dalam model *part*. Ketika suatu *part* direpresentasikan dalam bentuk model *boundary representation* (B-rep), elemen ini berupa : *face*, *edge* dan *vertex*. Kedua, mendeskripsikan masing-masing fitur untuk dikenali sebagai kombinasi dari elemen geometrik / topografi. Langkah ini setara dengan menciptakan sebuah *library* dari bentuk *template* yang sesuai dengan setiap fitur yang dikenal. Ketiga, membandingkan pola yang diekstraksi dengan *set* yang telah ditetapkan sebagai *template* fitur.

Cakupan pengenalan fitur cukup luas, terdapat metode yang dapat diklasifikasikan berdasarkan perbedaan kriteria. Dua kriteria penting tersebut adalah jenis file input sistem pengenalan fitur dan area aplikasi fitur yang diidentifikasi. Pertama, teknik pengenalan fitur mengambil input file netral CAD 3D seperti file STEP atau file STL. Kedua, fitur yang diidentifikasi merupakan manufaktur fitur. Gambar 2.1 menunjukkan suatu *part* yang memiliki beberapa fitur khas proses permesinan.

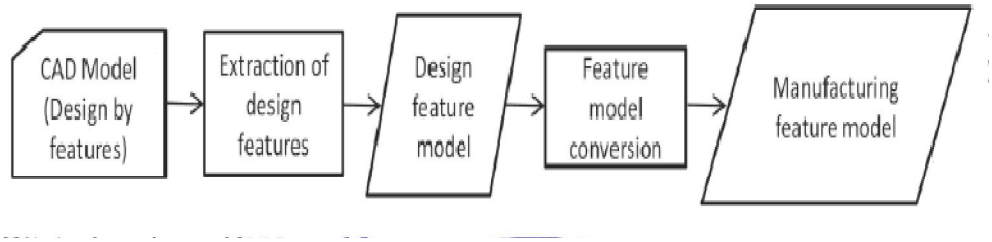


Gambar 2.1 Jenis fitur pada *part*

2.1.2 Taksonomi Pengenalan Fitur

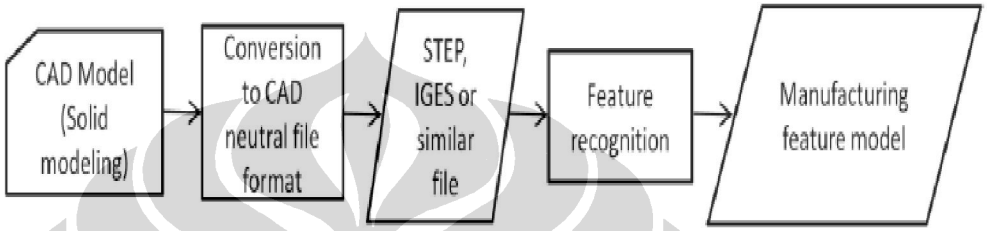
Sebelum membahas pada klasifikasi yang berbeda dari teknik pengenalan fitur, penting untuk memahami bahwa ada dua paradigma yang berkaitan dengan teknologi fitur. Yang pertama adalah fitur berbasis desain atau desain fitur, dan yang kedua adalah pengenalan fitur. Perbedaan utama pada keduanya adalah cara model CAD tersebut dibuat, paradigma ini harus diputuskan pada awal pemodelan dalam sistem CAD. Terdapat beberapa keuntungan yang melekat serta keterbatasan pada kedua pendekatan ini.

Fitur berbasis desain membutuhkan model CAD yang harus dibuat dari apa yang dikenal sebagai desain fitur. Model CAD dilakukan proses konversi ke model desain fitur. Berikutnya dan langkah terakhir adalah mengubah model desain fitur dengan model manufaktur fitur. Proses ini disebut pemetaan fitur atau konversi model fitur. Gambar 2.2 menunjukkan blok diagram dari sistem fitur berbasis desain.



Gambar 2.2 Blok diagram sistem fitur berbasis desain

Pengenalan fitur, di sisi lain tidak ada pembatasan model CAD pada cara *part* tersebut dibuat dalam sistem CAD. Setelah *part* dibuat, selanjutnya dikonversi ke dalam format file CAD yang netral seperti STL, STEP atau IGES. File ini berfungsi sebagai input untuk algoritma pengenalan fitur dalam mengekstraksi dan mengenali manufaktur fitur yang terdapat pada *part.* Gambar 2.3 menunjukkan blok diagram dari sistem pengenalan fitur.

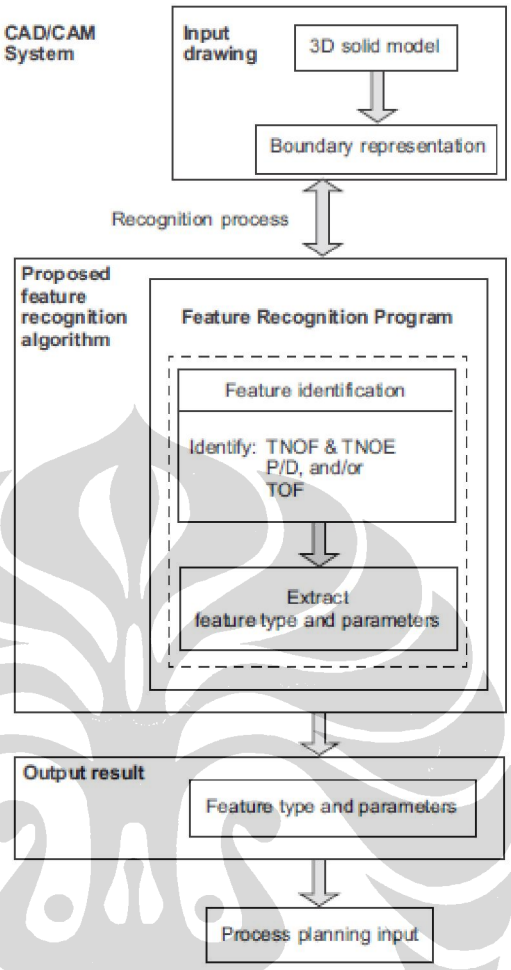


Gambar 2.3 Blok diagram sistem pengenalan fitur

Untuk membandingkan teknik yang sudah ada dan untuk *benchmark* metode yang baru dikembangkan, sangat membantu untuk mengklasifikasikan semua metode pengenalan fitur. Ada beberapa kriteria berdasarkan teknik ini yang dapat diklasifikasikan, yang terpenting adalah prinsip dasar identifikasi jenis fitur pada model *part.*

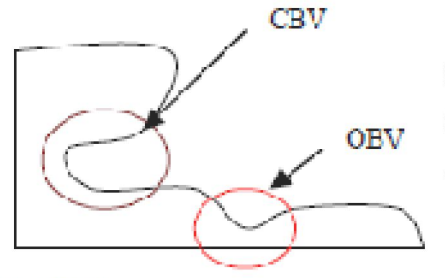
2.1.3 Teknik Pengenalan Fitur

Risal Abu dan Masine [2] mendeteksi pengenalan fitur secara otomatis dengan model *boundary representation* (B-rep) menggunakan pendekatan yang berdasarkan sifat-sifat tertentu sebuah fitur seperti jumlah bidang (TNOF), jumlah sudut (TNOE), jenis bidang (TOF), dan sifat-sifat fitur seperti *protrusion/depression* (P/D). Algoritma ini dapat dijelaskan pada diagram yang ditunjukkan gambar 2.4 berikut ini.



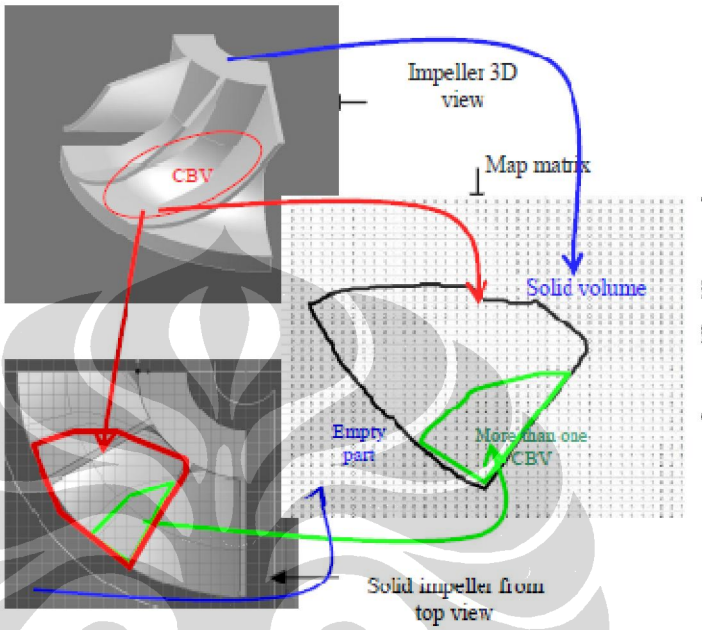
Gambar 2.4 Diagram pengenalan fitur B-rep

G. Kiswanto dan A. Yahya [3] mengembangkan identifikasi *Close Bounded Volume* (CBV) pada model faset 3D kompleks melalui metode *Paired Normal Vectors Bucketing* (PNVB) untuk perencanaan *tool path* pada proses permesinan awal (*roughing*) *multi axis.* Berikut ilustrasi CBV pada gambar 2.5.



Gambar 2.5 *Closed and open bounded volume* pada model

Selanjutnya model dipetakan pada *map matrix*, daerah yang merupakan CBV adalah daerah yang memiliki daerah *boundary* yang bertumpuk seperti diperlihatkan pada gambar 2.6 berikut ini.



Gambar 2.6 Proses *mapping* pada model

G. Kiswanto, R. Widyanto dan P.E. Kreshna [10] mengembangkan identifikasi fitur *surface* pada bentuk *flat, saddle, convex,* dan *concave* dengan mengkombinasikan metode Gauss Bonnet dan Spherical Image. Tabel kombinasi tersebut dapat dilihat pada tabel 2.1 di bawah ini.

Tabel 2.1 Kombinasi Gauss-Bonnet dan Spherical Image untuk menentukan bentuk *surface*

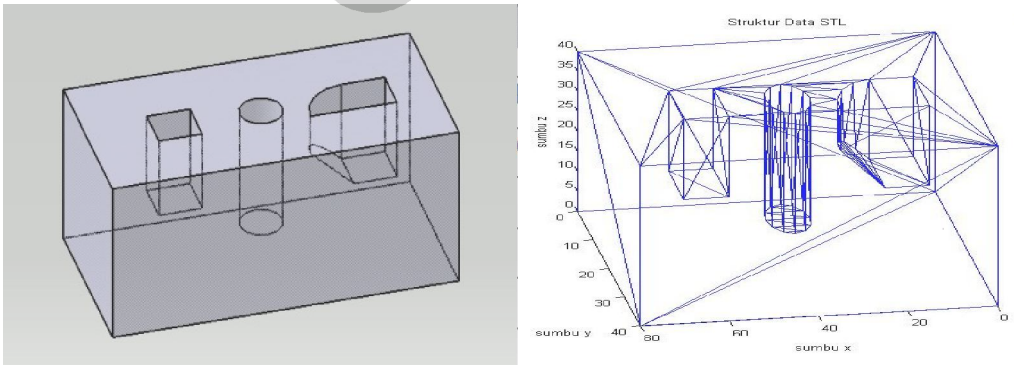
Surface Shape	Estimation method	
	Gauss-Bonnet	Spherical Image
Flat	+/- 0	+/- 1
Saddle	< 0	+/- 1
Convex	> 0	> 1
Concave	> 0	< 1

2.2 Struktur Data Model Faset

2.2.1 Definisi Model Faset

Model faset 3D adalah salah satu representasi bentuk atau model pada bidang 3 dimensi. Representasi lain yang dapat digunakan untuk model 3D adalah parametrik *surface* dan model *solid*, dimana hal ini digambarkan oleh persamaan bidang sehingga mendapat tampilan model pada ruang 3 dimensi. Model faset 3D digambarkan oleh titik-titik yang menyusun sebuah model / permukaan bidang. Titik-titik ini dihubungkan dengan bentuk kumpulan segitiga-segitiga yang merupakan diskritisasi dari obyek permukaan bidang. Jika dilihat dari perspektif umum, model faset 3D terbentuk dari kumpulan segitiga-segitiga, dimana setiap segitiga memiliki informasi berupa vektor normal bidang pada posisi tertentu dan tiga buah *vertex* yang merupakan titik penyusun pada model faset yang bersangkutan.

Model faset 3D dapat dibuat menggunakan CAD (*computer aided design*), ataupun dilakukan *scanning* terhadap obyek material nyata. Model yang dibuat dari sistem CAD, dilakukan dengan merancang sebuah bentuk benda, dan hasilnya dikonversi menjadi sebuah file yang berekstensi STL. Hasil dari proses konversi adalah sebuah pendekatan diskritisasi dari obyek tersebut. Proses konversi ini merupakan fitur standar yang dimiliki oleh sistem CAD. Sedangkan model yang didapat melalui *scanning* dibuat menggunakan alat scanner 3D dan akan menghasilkan titik-titik penyusun model faset dalam bidang 3 dimensi. Hal ini direpresentasikan dengan segitiga-segitiga yang menghubungkan titik-titik tersebut. Proses pembentukan segitiga disebut triangulasi.

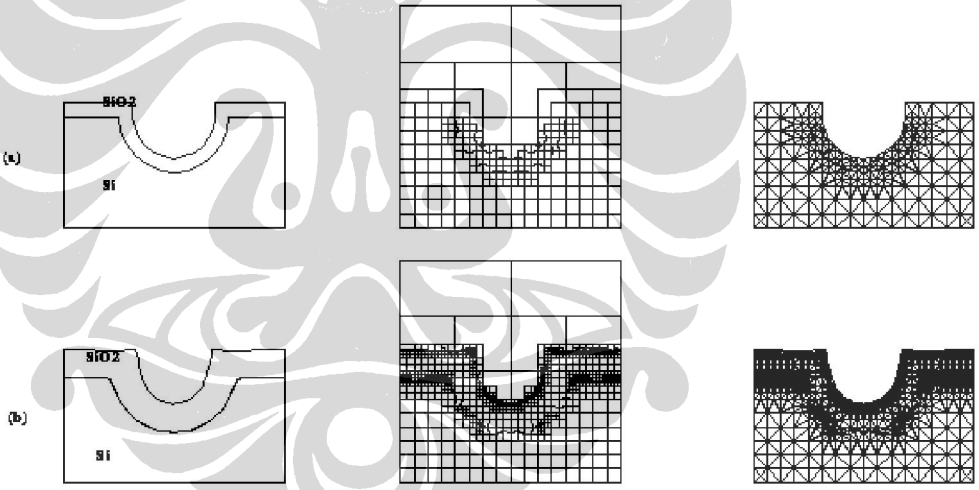


Gambar 2.7 Model faset 3D

Data model faset 3D seperti pada gambar 2.7 di atas, tersimpan dalam bentuk file yang berekstensi STL. File ini berfungsi untuk menyimpan informasi sebuah model faset 3D. Informasi ini berupa segitiga-segitiga penyusun model faset dengan vektor normal segitiga dan koordinat *vertex* segitiga pada bidang 3D sebagai propertinya. STL merupakan kependekan dari *stereolithography*.

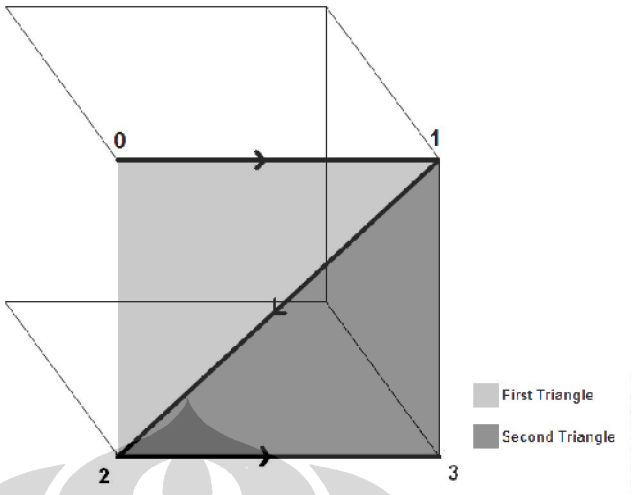
2.2.2 Data STL File

Model faset 3D merupakan model berbasis triangulasi segitiga. *Triangular mesh* merupakan salah satu proses pemodelan berbasis elemen hingga dimana sebuah model akan digeneralisasi menjadi elemen kecil-kecil yang berbentuk segitiga, dimana apabila terdapat bidang persegi yang datar maka dalam triangulasi segitiga akan dipresentasikan oleh 2 bidang segitiga yang berhimpit. Adapun visualisasinya dapat dilihat dalam gambar 2.8 berikut ini :



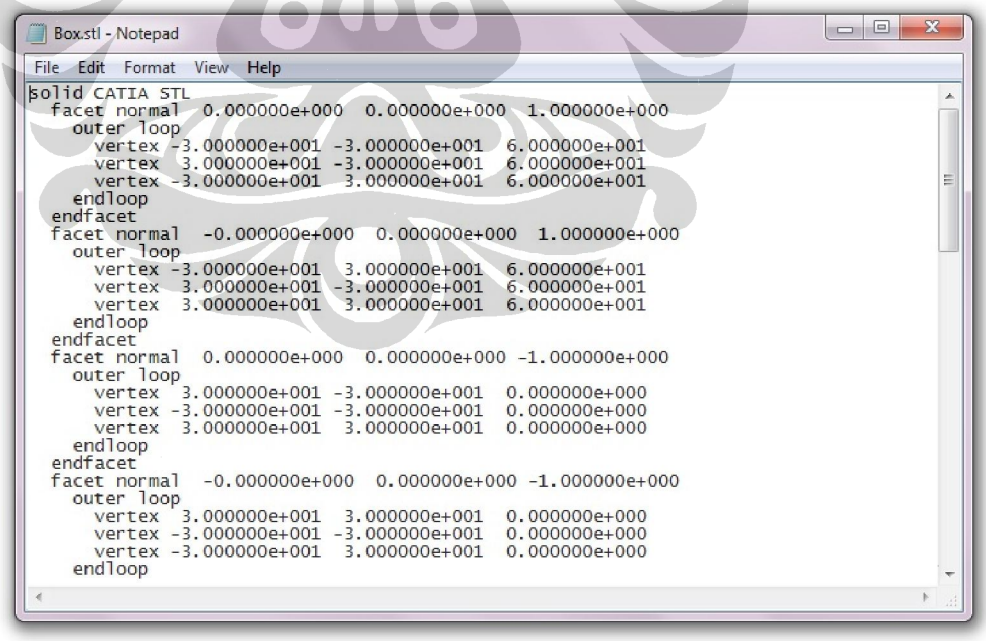
Gambar 2.8 Visualisasi triangulasi

Struktur data yang dihasilkan oleh model faset dalam sistem CAD memiliki dua data utama yaitu *list index segitiga* dan *list index vertex*. Proses pembuatan *list* ini didasarkan pada urutan segitiga file ekstensi STL dari model faset 3D. Adapun ilustrasi model faset 3D yang tersusun dari segitiga-segitiga pembentuknya dapat dilihat pada gambar 2.9 berikut ini.



Gambar 2.9 Ilustrasi model faset 3D

Data yang diperoleh dari sistem CAD berupa STL file yang kemudian akan diolah menjadi struktur data model faset berupa *list index segitiga* dan *list index vertex*. Format STL file ini adalah yang paling umum digunakan karena lebih mudah dibaca dan dimengerti, serta dapat dibuka di *text editor*. Gambar 2.10 berikut ini adalah contoh isi dari file berekstensi STL.



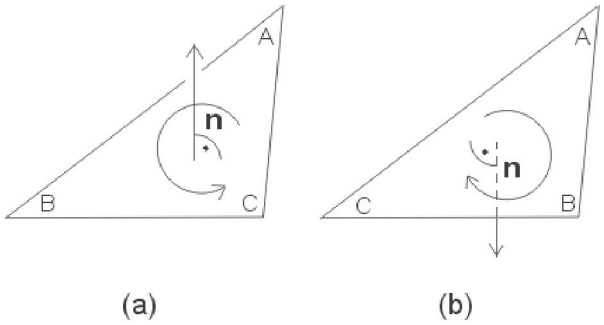
Gambar 2.10 Contoh isi file STL (ASCII)

Dapat dilihat bahwa file berformat STL menyimpan obyek *surface* 3D dalam bentuk segitiga yang tersusun dari tiga buah *vertex* yang berada pada lokasi tertentu dalam sistem koordinat 3D. Berikut adalah penjelasan mengenai isi dari file STL di atas,

1 Kata SOLID menandakan dimulainya penggambaran atau penyimpanan model faset hingga ditutup dengan kata ENDSOLID.
2 Kata FACET NORMAL menandakan bahwa akan dibangun sebuah permukaan yang berbentuk segitiga dengan nilai vektor normal berada pada kata setelah kata FACET NORMAL hingga bertemu dengan kata ENDFACET yang berarti sebuah permukaan segitiga telah terbentuk beserta informasi urutan dan letak *vertex*, serta vektor normal dari segitiga.
3 Kata OUTER LOOP menandakan dimulainya loop dari koordinat *vertex-vertex* yang membangun segitiga hingga bertemu dengan kata ENDLOOP.
4 Kata VERTEX merupakan *vertex* penyusun sebuah segitiga yang sebelumnya telah didefinisikan dengan OUTER LOOP. Informasi yang berada setelah kata VERTEX adalah posisi *vertex* pada sistem koordinat.

2.2.3 Aturan Arah Vektor Normal

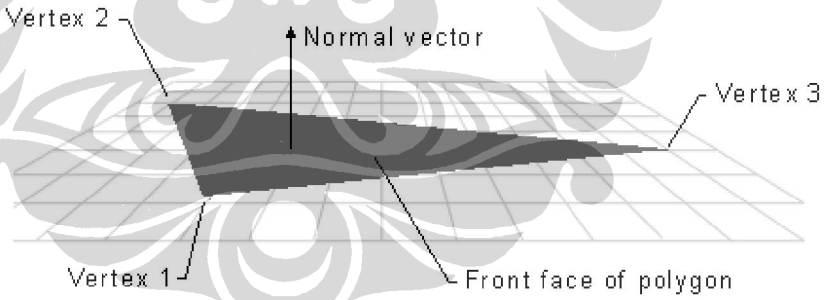
Arah vektor normal segitiga akan sesuai dengan urutan pengambilan *vertex*, dalam mencari vektor normal mengikuti kaidah tangan kanan. Jika putaran searah dengan jarum jam (*clockwise*), maka vektor normal akan menuju bidang. Sebaliknya, jika putaran berlawanan arah jarum jam (*counter clockwise*), maka vektor normal keluar dari bidang, untuk gambaran yang lebih jelas dapat dilihat pada gambar 2.11 di bawah ini :



Gambar 2.11 Aturan kaidah tangan kanan

Arah vektor normal bergantung pada urutan *vertex* yang membangun sebuah segitiga. Pada gambar 2.11(a) di atas, urutan pembentukan *vertex* berlawanan arah jarum jam (*counter clockwise*), maka perkalian silang (*cross product*) antara vektor-vektor yang menghubungkan *vertex* tersebut akan menghasilkan sebuah vektor normal yang arahnya keluar dari bidang, sedangkan pada gambar 2.11(b), urutan pembentukan *vertex* searah jarum jam (*clockwise*), maka perkalian silang (*cross product*) antara vektor-vektor yang menghubungkan *vertex* akan menghasilkan sebuah vektor normal yang arahnya masuk ke bidang.

Pembentukan bidang datar berfungsi untuk menghubungkan titik-titik penyusun bidang dan membentuk sebuah permukaan. Bentuk segitiga adalah bentuk yang paling sesuai untuk fungsi tersebut sebab pada ruang 3 dimensi, bentuk yang memiliki 3 buah *vertex* ini selalu konsisten pada penempatan titik, arah bidang, dan vektor normal bidang. Bidang datar lain selain segitiga, memiliki kemungkinan terjadi ketidak-konsistenan antara arah bidang dengan penempatan titik, sebab sangat mungkin terjadi bidang tersebut memiliki lebih dari satu vektor normal untuk sebuah bidang datar akibat dari letak titik-titik penyusunnya yang tidak konsisten.



Gambar 2.12 Arah vektor normal segitiga

Gambar 2.12 di atas adalah arah vektor normal pada sebuah segitiga dalam ruang 3 dimensi yang selalu konsisten. Vektor normal pada sebuah bidang yang bukan segitiga (memiliki lebih dari tiga *vertex*), misalnya pada sebuah ruang 3 dimensi terdapat 4 buah titik yang akan disusun menjadi sebuah bidang datar segiempat beraturan, maka arah vektor normal dapat berjumlah lebih dari satu bergantung pada koordinat titik-titik penyusunnya yang dipilih sebagai acuan dalam menentukan arah vektor normal.

2.2.4 Normalisasi Sumbu

Normalisasi sumbu adalah penyesuaian letak model faset dalam ruang 3D sehingga nilai minimum dan maksimum koordinat x, y, dan z ditempatkan pada titik yang kita inginkan. Agar normalisasi sumbu ini dapat dilakukan, maka nilai koordinat-koordinat pada x minimum, x maximum, y minimum, y maximum, z minimum, dan z maximum harus ditentukan. Dari nilai-nilai minimum dan maximum tersebut, dapat dibentuk sebuah batas model faset.

Normalisasi sumbu dilakukan dengan memindahkan nilai-nilai x, y, z sehingga salah satu sudutnya berada pada titik pusat (0,0,0). Hal ini memiliki arti bahwa *vertex-vertex* penyusun model faset dipindahkan sehingga nilai koordinat dari x minimum, y minimum, dan z minimum berada pada titik (0,0,0). Pemindahan ini dilakukan dengan menggunakan perhitungan berikut ini :

Jarak perpindahan sumbu x (dx) = $x_{min} - x_0$

Jarak perpindahan sumbu y (dy) = $y_{min} - y_0$

Jarak perpindahan sumbu z (dz) = $z_{min} - z_0$

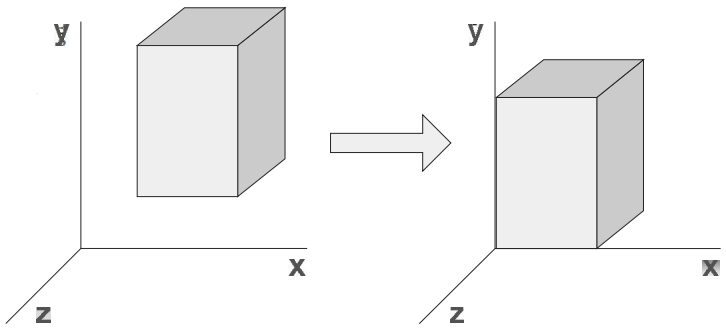
Selanjutnya pada setiap *vertex* penyusun model faset dilakukan penyesuaian nilai x, y, dan z sebagai berikut :

x’ = x – dx

y’ = y – dy

z’ = z – dz

Kegunaan normalisasi sumbu ini antara lain mempermudah pencarian *vertex* segitiga, hal ini dapat dijelaskan dengan gambar 2.13 di bawah ini :



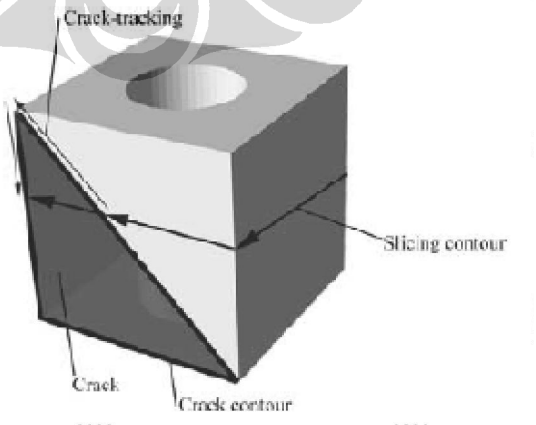
Gambar 2.13 Ilustasi normalisasi sumbu

2.3 Metode Deteksi Fitur

2.3.1 Proses *Slicing*

Salah satu algoritma yang mampu untuk mengidentifikasi segitiga-segitiga atau titik-titik perpotongan yang membentuk fitur pada suatu bidang adalah metode *slicing*. Secara sederhana dapat dijelaskan bahwa metode *slicing* dilakukan dengan mengiriskan z = c dengan model faset berbentuk prismatik atau berkontur. Irisan ini akan menghasilkan titik-titik perpotongan antara bidang z dengan model. Karena modelnya berbasis faset segitiga, maka perpotongan terjadi antara bidang z dengan sisi-sisi segitiga. Bidang *slice* yang digunakan untuk mencari perpotongan ini tentunya bukan hanya satu bidang saja melainkan bidang-bidang $z = c_i$ untuk i = 1, 2, …, k, dimana nilai c_i berkisar antara koordinat z paling rendah sampai koordinat z paling tinggi, atau diartikan $z_{min} < c_i > z_{max}$. Untuk mendapatkan nilai-nilai c_i cukup diberikan interval nilai antara c_i dan c_{i+1} yang diinginkan. Input pada proses *slicing* menentukan perpotongan sebagai interval antar bidang potong.

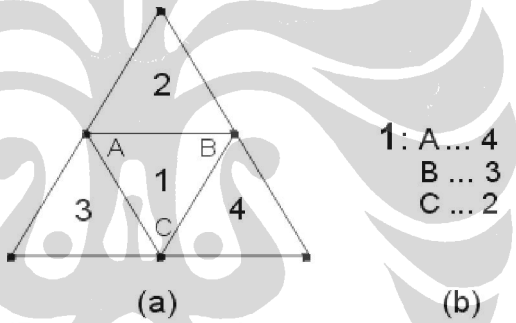
Metode *brute searching* merupakan metode yang sederhana dalam proses *slicing*, yaitu dengan melakukan pengecekan pada setiap segitiga yang ada apakah berpotongan dengan bidang z atau tidak. Jika berpotongan, *index segitiga* tersebut disimpan dalam sebuah struktur data vektor. Pengecekan ini dilakukan untuk setiap bidang z yang akan diiriskan dengan model faset 3D tersebut. Sebagai ilustrasi proses *slicing* model faset 3D, dapat dilihat pada gambar 2.14 berikut ini.



Gambar 2.14 Ilustrasi *slicing* model faset 3D

2.3.2 Identifikasi *Adjacent Triangles*

Metode yang digunakan untuk mengetahui keterhubungan (*conectivity*) faset satu dengan lainnya adalah dengan menggunakan identifikasi segitiga yang berdekatan (*adjacent triangles*). Dalam fungsi ini, dibuat informasi tentang kedekatan segitiga atau dibuat lagi jika sudah dilakukan. Untuk setiap *edge* masing-masing segitiga menemukan segitiga yang lain yang berisi *edge* yang sama atau mempunyai dua (2) *vertex* yang sama. Langkah *adjacent triangles* yang dilakukan adalah dengan mencari segitiga yang mempunyai *edge* yang sama sampai tidak ditemukan lagi *edge* yang sama dengan yang lainnya. Faset yang berdekatan yaitu bila terdapat *edge* yang sama atau dua(2) *vertex* yang sama pada faset yang berbeda. Segitiga yang berdekatan dapat dilihat pada gambar 2.15 berikut ini.



Gambar 2.15 Segitiga yang berdekatan

2.3.3 Transformasi Rotasi

Didalam aljabar linear, matriks rotasi adalah matriks yang digunakan untuk melakukan rotasi dalam ruang Euclidean. Sebagai contoh matriks

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

pusat rotasi dalam bidang xy-cartesian berlawanan dengan sudut θ yang berasal dari koordinat sistem cartesian. Untuk melakukan rotasi menggunakan matriks rotasi R, posisi setiap titik harus diwakili oleh kolom vektor v, yang berisi koordinat dari titik tersebut. Sebuah vektor diputar diperoleh dengan

menggunakan Rv perkalian matriks. Karena perkalian matriks tidak berpengaruh pada vektor nol (yaitu, pada koordinat asal), matriks rotasi hanya dapat digunakan untuk menggambarkan rotasi dari sistem koordinat asalnya.

Transformasi rotasi memberikan deskripsi aljabar sederhana dari rotasi tersebut, dan digunakan secara ekstensif untuk perhitungan dalam geometri, fisika, dan komputer grafis. Dalam ruang dua-dimensi, rotasi secara sederhana dapat dijelaskan oleh sudut θ rotasi, tetapi dapat juga diwakili oleh empat entri dari matriks rotasi dengan dua baris dan dua kolom. Dalam ruang 3 dimensi, setiap rotasi dapat diartikan sebagai sebuah rotasi dengan sudut yang diberikan pada sumbu tetap tunggal rotasi (Teorema Euler rotasi), dan karena itu bisa secara sederhana dijelaskan oleh sudut dan vektor dengan tiga entri. Namun, juga dapat diwakili oleh sembilan entri dari matriks rotasi dengan tiga baris dan tiga kolom. Gagasan rotasi tidak umum digunakan pada dimensi yang lebih tinggi dari 3, ada gagasan tentang perpindahan rotasi, yang dapat diwakili oleh sebuah matriks, tapi tidak terkait sumbu tunggal atau sudut.

Tiga dasar rotasi beserta matriks rotasi yang memutar vektor sumbu x, y, atau z, dalam tiga dimensi dapat ditunjukkan pada rumusan berikut ini :

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

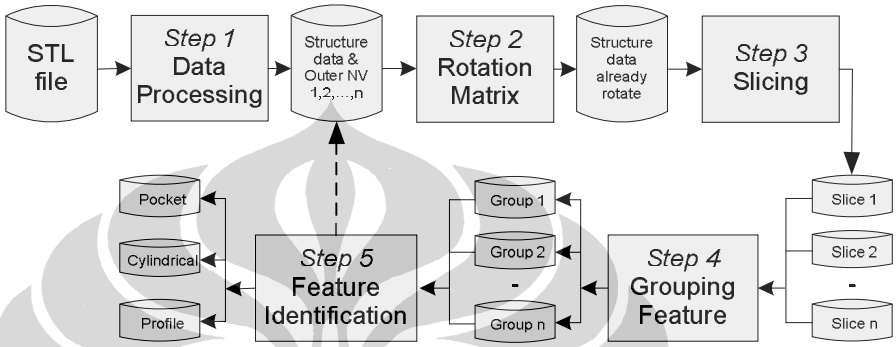
Sedangkan untuk rotasi tiga (3) dimensi pada sumbu x, y, dan z menggunakan perumusan sebagaimana berikut ini :

$$\begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}$$

BAB III
PENGEMBANGAN ALGORITMA PENGENALAN FITUR

3.1 Kerangka Pemikiran

Perencanaan untuk melakukan identifikasi fitur model faset 3D dilakukan dengan tahapan algoritma pada gambar 3.1 berikut ini.



Gambar 3.1 Alur algoritma identifikasi fitur

Tahapan algoritma identifikasi fitur dimulai dengan pengolahan struktur data dan pendeteksian vektor normal terluar pada file STL yang berisi informasi mengenai data faset dan tiga *vertex* pembentuk masing-masing segitiga. Proses selanjutnya adalah transformasi rotasi, dimana struktur data model *part* dirotasi menuju sumbu z atau sejajar bidang xy dari vektor normal terluar. Kemudian, melakukan proses *slicing*, model faset 3D dipotong atau diiris dengan bidang *slice* pada sumbu arah z atau sejajar bidang xy. Interval *slicing* dibuat seragam yang prosesnya dimulai dari bawah antara z min s/d z max model faset. Hasil *slicing* ini kemudian disimpan dalam data vektor.

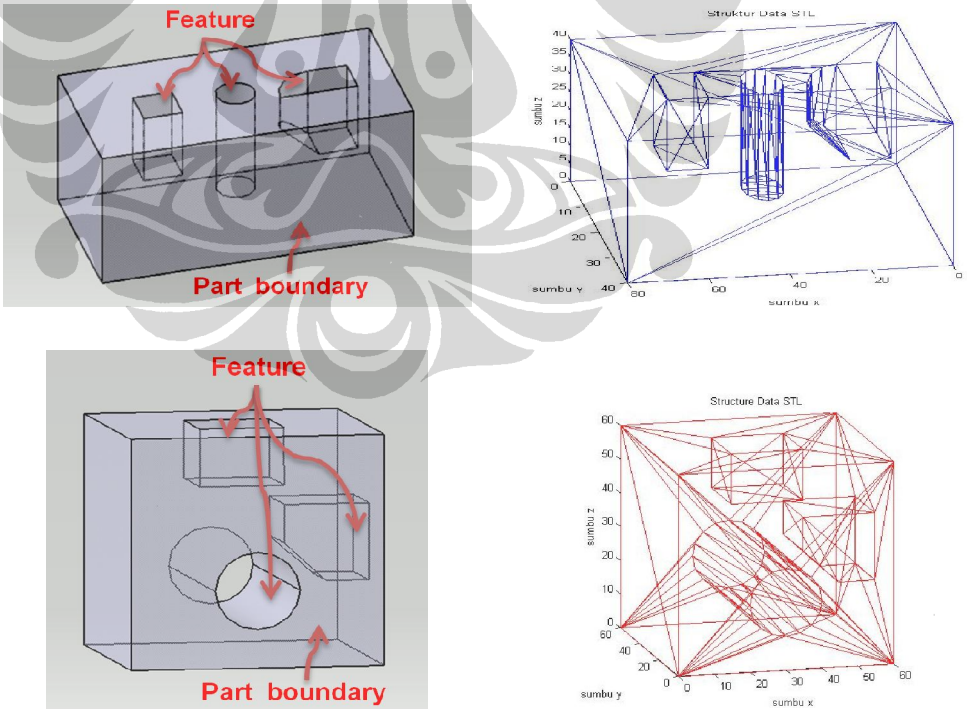
Tahapan proses berikutnya adalah *grouping feature*, dengan mengelompokkan seluruh faset *slice* yang selanjutnya dilakukan *sorting* jika terdapat faset yang sama sehingga berisi faset-faset yang *unique*. Faset-faset tersebut lalu dikelompokkan lagi berdasarkan faset yang berdekatan (*adjacent triangles*) sehingga terbentuk beberapa grup. Salah satu grup tersebut merupakan *boundary group* yang harus dihilangkan dengan pengecekan koordinat maksimum *vertex* pada masing-masing grup, sehingga terbentuk Group Feature. Tahapan terakhir yaitu deteksi fitur dengan karakteristik arah vektor normal. Langkah-langkah tersebut diulang sebanyak jumlah arah vektor normal terluar.

3.2 Pengolahan Struktur Data Model Faset

3.2.1 Pembuatan Model *Part* 3D

Model yang diambil dari sebuah sistem *Computer Aided Design* (CAD) memuat titik koordinat 3 dimensi. Perlu metode penghubung yang terintegrasi pada model *part* sehingga dapat dibaca oleh sistem yang lain. Oleh sebab itu, model *part* 3D yang dibuat, diubah menjadi data yang berformat *Stereolithography* (STL). File STL ini menyimpan informasi obyek *surface* dalam bentuk model faset 3D yang tersusun dari satu atau lebih segitiga.

Pendeteksian terhadap bentuk suatu model menjadi elemen yang sangat penting, sebab fitur akan memberi konsep informasi terhadap karakteristik komponen serta metode pengerjaannya. Sedangkan untuk desainer, fitur suatu model dapat merepresentasikan fungsi, rencana *assembly* serta merepresentasikan bagian dari *part* lain terhadap satu kesatuan benda. Gambar 3.2 di bawah merupakan model *part* 3D pada sistem CAD yang diubah menjadi format STL file yang mengandung informasi vektor segitiga dan *vertex*.



Gambar 3.2 Model *part* - faset 3D

3.2.2 Penyusunan Struktur Data

Dari model yang telah dibuat, data perlu dikelompokkan dan diberikan *index* agar mempermudah dalam proses pendeteksian. Pemberian *index* ini dilakukan terhadap segitiga maupun terhadap *vertex* yang terkandung dalam setiap segitiga. Pemberian *index* ini juga bertujuan untuk menghindari adanya penyimpanan *vertex* segitiga yang sama dengan lebih dari satu kali. Informasi struktur data pada model faset 3D yang dibutuhkan antara lain :

a. *Index segitiga*; untuk identifikasi seluruh faset yang ada dalam STL. Setiap *index segitiga* memiliki tiga *index vertex* sebagai pembentuknya.
b. *Index vertex*; untuk identifikasi seluruh *vertex* yang membentuk segitiga. Masing-masing *index vertex* memiliki nilai *koordinat vertex*.
c. *Koordinat vertex*; adalah nilai koordinat seluruh *vertex* dalam x,y,z.

Database struktur data model faset dapat digambarkan pada tabel 3.1 *index segitiga* dan tabel 3.2 *index vertex* di bawah ini.

Tabel 3.1 Struktur Tabel *Index Segitiga*

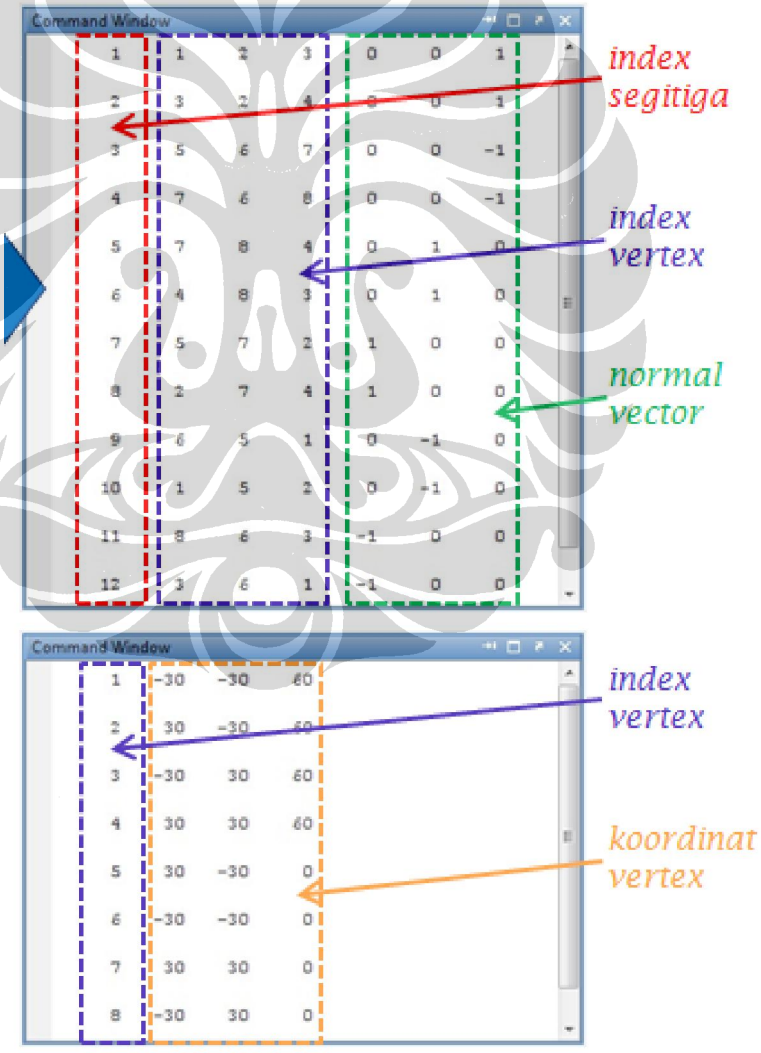
Index Segitiga	Index Vertex 1	Index Vertex 2	Index Vertex 3
1	1	2	3
2	3	2	4
dst.	…	…	…

Tabel 3.2 Struktur Tabel *Index Vertex*

Index Vertex	Koordinat X	Koordinat Y	Koordinat Z
1	40	0	40
2	25	10	40
dst.	…	…	…

Ketiga informasi di atas menjadi *database* struktur data pada model faset 3D. Untuk menentukan lokasi sebuah segitiga terhadap bidang, cari *index*-nya, kemudian tentukan tiga *index vertex*-nya, dan dapatkan koordinat (x,y,z) dari ketiga *vertex*-nya. Setelah informasi mengenai *index segitiga, index vertex,* dan *koordinat vertex* sudah teridentifikasi kemudian disimpan dalam data *array.*

Misalkan sebuah kubus (*box*) yang memiliki enam sisi, maka dalam pembentukan struktur data akan tersusun berupa : 12 segitiga faset dan 8 titik *vertex* pembentuk model *solid.* Apabila program pengolahan data STL file dijalankan, maka di dalam *command window* akan muncul informasi mengenai *index segitiga, index vertex,* dan *koordinat vertex* seperti gambar 3.3 berikut ini.

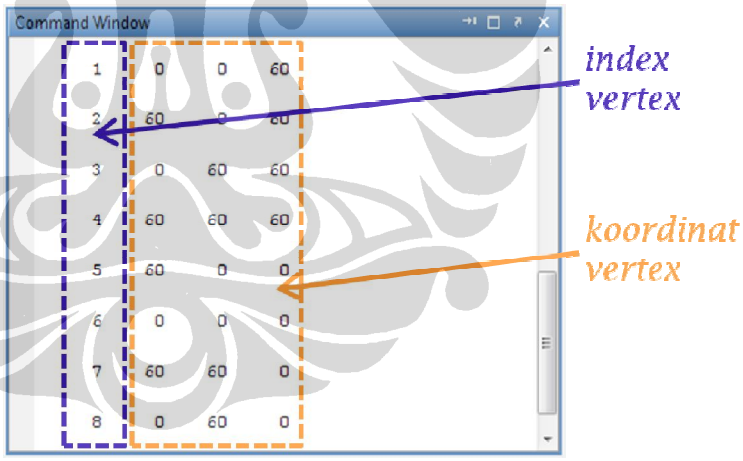


Gambar 3.3 Struktur data kubus model faset 3D

3.2.3 Normalisasi Model Faset

Untuk mempermudah alur perhitungan dan membuat nilai dari semua koordinat pada posisi normal koordinat, dengan kata lain penyesuaian letak model faset dalam ruang 3D sehingga nilai minimum dan maksimum koordinat x, y, dan z berada pada titik yang diinginkan. Agar normalisasi sumbu ini dapat dilakukan, maka nilai-nilai koordinat x minimum, x maksimum, y minimum, y maksimum, z mimimum, dan z maksimum harus disimpan. Dari nilai-nilai minimum dan maksimum tersebut, dapat dibentuk sebuah *bounding box* yang membatasi model faset 3D dan membatasi daerah perhitungan (*bounded*).

Normalisasi sumbu dilakukan dengan memindahkan *bounding box*, sehingga salah satu sudutnya berada pada titik pusat (0,0,0). Ini berarti *vertex-vertex* penyusun model faset dipindahkan sehingga nilai dari x minimum, y minimum, dan z minimum berada pada titik (0,0,0). Struktur data *index vertex* setelah proses normalisasi mengalami perubahan posisi koordinat yang dapat dilihat pada gambar 3.4 berikut ini.



Gambar 3.4 Struktur data *index vertex* normalisasi

3.3 Metode Yang Dikembangkan

3.3.1 Transformasi Rotasi

Transformasi rotasi digunakan untuk memutar model faset 3D menuju arah z-axis atau bidang xy agar mampu mendeteksi fitur pada berbagai bidang, tidak hanya pada bidang xy saja. Hal ini karena proses *slicing* atau pemotongan model faset yang dilakukan berdasarkan pada interval arah z-axis atau bidang xy. Model faset yang memiliki fitur pada berbagai bidang, dirotasi dengan perbandingan sudut antara arah vektor normal terluar pada masing-masing bidang dengan arah z-axis atau bidang xy [0 0 1]. Setelah sudut terhadap z-axis diketahui, selanjutnya adalah melakukan rotasi model faset 3D tersebut. Adapun persamaan transformasi rotasi ditunjukkan sebagimana berikut ini.

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

$$\begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}$$

Model *part* dirotasi berdasarkan sudut antara arah bidang vektor normal terluar terhadap bidang xy atau arah z-axis [0 0 1]. Setelah sudut tersebut diketahui, langkah selanjutnya adalah merotasi model *part* 3D berdasarkan persamaan di atas pada perkalian silang antara rotasi x atau rotasi y dan rotasi z dengan besar sudut yang telah didapat sebelumnya sehingga model *part* pada arah bidang vektor normal terluar berputar dan berubah menjadi arah bidang xy, begitu seterusnya sebanyak jumlah bidang vektor normal terluar.

3.3.2 Algoritma *Slicing*

Algoritma *slicing* digunakan untuk mendapatkan perpotongan antara sebuah model 3D dengan bidang datar. Bidang yang dipilih adalah bidang z, yaitu bidang yang tegak lurus dengan sumbu koordinat z (atau bidang yang sejajar dengan bidang yang dibentuk oleh sumbu koordinat xy). Bidang-bidang yang digunakan sebagai pemotong model bergantung kepada interval antar dua bidang berdekatan serta koordinat z minimum dan maksimum. Misalkan jika $z_{min} = 0$ dan $z_{max} = 10$, sedangkan interval antar dua bidang *slicing* interval =2, bidang *slice* pertama adalah seperdua dari interval *slicing*, maka ada lima bidang, masing-masing z = {1, 3, 5, 7, 9}.

Untuk mendapatkan titik-titik potong sebuah bidang dengan model faset 3D, informasi yang harus sudah dimiliki adalah :

a. *Index segitiga*; untuk mengidentifikasi semua segitiga atau faset yang ada dalam STL. Masing-masing *index segitiga* memiliki tiga *index vertex* sebagai pembentuknya.
b. *Index vertex*; untuk mengidentifikasi semua *vertex* yang menjadi titik-titik pembentuk segitiga. Masing-masing *index vertex* menunjuk ke satu nilai *koordinat vertex*.
c. *Koordinat vertex*; yaitu nilai koordinat semua *vertex* dalam (x,y,z).

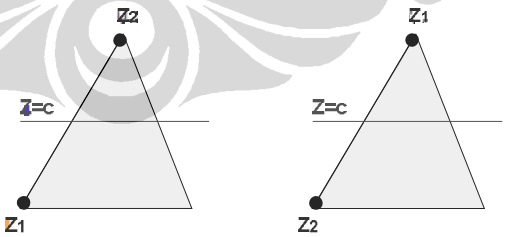
Untuk menentukan lokasi sebuah segitiga terhadap bidang, cari *index*-nya, kemudian tentukan tiga *index vertex*-nya, dan dapatkan koordinat (x,y,z) dari ketiga *vertex*-nya. Setelah informasi mengenai *index segitiga*, *index vertex*, dan *koordinat vertex* sudah teridentifikasi dan tersimpan dalam data *array*, selanjutnya adalah mencari *index segitiga* yang berpotongan dengan bidang potong z. Metode pencarian ini dilakukan dengan metode *brute searching*, metode ini dilakukan dengan mengecek setiap segitiga yang ada apakah berpotongan dengan bidang z atau tidak. Jika berpotongan, *index segitiga* tersebut disimpan dalam sebuah struktur data *array*. Pengecekan ini dilakukan untuk setiap bidang z yang akan diiriskan dengan model faset. Adapun algoritma *slicing* dapat ditunjukkan sebagaimana berikut ini.

```
Algorithm: Slicing
Input: STL File
Output: Facet Slice File

Write_structure_data_facet
Define_interval
For(every_facet)
  If(Zmin<Zslice<Zmax)
    Save_the_data_facetslice
    Zslice=Zslice+interval
  End
  If(Zslice>Zmax_or_Zslice<Zmin)
    Break
  End
End
```

Metode ini dilakukan dengan mengecek setiap segitiga yang ada apakah berpotongan dengan bidang z atau tidak. Jika berpotongan, *index segitiga* tersebut disimpan dalam sebuah struktur data *array*. Pengecekan ini dilakukan untuk setiap bidang z yang akan diiriskan dengan model faset 3D. Pendekatan untuk pengecekan setiap bidang *slice* dilakukan dengan persyaratan sebagai berikut :

- Jika letakZ_1 < z < letakZ_2, atau
- Jika letakZ_1 > z > letakZ_2



Gambar 3.5 Titik perpotongan segitiga

Gambar 3.5 di atas, menggambarkan bahwa titik perpotongan bidang Zslice = c, berada pada posisi lebih besar dari koordinat titik Zminimum atau diartikan (> Zmin) dan kurang dari koordinat titik Zmaksimum (< Zmax) atau diartikan Zmin < Zslice < Zmax.

3.3.3 Algoritma *Grouping Feature*

Algoritma yang digunakan untuk mengetahui keterhubungan (*conectivity*) faset satu dengan lainnya adalah dengan menggunakan algoritma segitiga yang berdekatan (*adjacent triangles*). Langkah yang dilakukan untuk mendapatkan Group Feature antara lain : Mengelompokkan faset *slice* kemudian *sorting* bila ada yang sama. Mencari faset yang bertetanggaan dengan aturan bila terdapat *vertex* yang sama pada faset yang berbeda. Membuang grup yang merupakan *part boundary* dengan mengecek koordinat yang terluar.

Dalam pengelompokan faset *slice* dilakukan dengan membaca data vektor hasil *slicing* kemudian dibandingkan dengan data tersebut apakah sudah ada dalam *array*, jika belum diambil, tetapi kalau ada tidak diambil. Data-data faset tersebut disimpan dalam Group Matrix. Selanjutnya adalah mendapatkan Group Adjacent yang memiliki faset yang berdekatan. Sebelum mengelompokkan faset ke dalam Group Adjacent dilakukan pengurutan faset yang memiliki salah satu *vertex* yang sama untuk mencegah terjadinya kesalahan pengelompokkan.

Setelah terbentuk matrik yang berisi faset-faset yang memiliki *vertex* yang sama, selanjutnya dilakukan pengelompokkan dalam matrik Group Adjacent. Kemudian, untuk mendapatkan Group Feature yang diinginkan dilakukan penghapusan grup yang merupakan *part boundary* dengan mengecek koordinat *vertex* pada masing-masing grup, lalu hapus grup yang memiliki koordinat terluar sehingga didapatkan Group Feature. Adapun algoritma *grouping feature* dapat dilihat sebagaimana berikut ini.

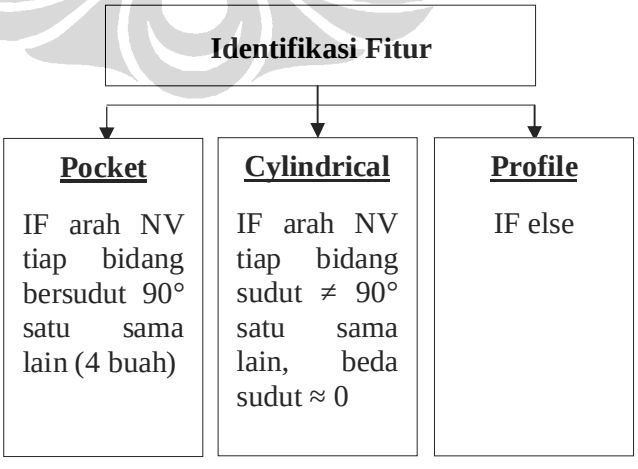
```
Algorithm: Grouping Feature
Input: Facet Slice File
Output: Group Feature File

Write_data_facetslice
For(every_facetslice)
  If(data_facetslice_unique)
    Save_the_data_GroupMatrix
  End
End
```

```
Write_data_GroupMatrix
For(every_facet_GroupMatrix)
  If(adjacent_triangles)
    Save_the_data_GroupAdjacent
  End
End
Write_data_GroupAdjacent
For(every_GroupAdjacent)
  Find(group_part_boundary)
    Eliminate_group_part_boundary
  End
End
```

3.3.4 Aturan Identifikasi Fitur

Dengan megetahui arah vektor normal pada Group Feature, maka dapat dibuat aturan untuk mengidentifikasikan jenis fitur pada grup tersebut. Langkah yang dilakukan adalah dengan mengidentifikasikan vektor normal pada masing-masing Group Feature, jika ada arah vektor normal yang sama pada suatu grup maka eliminasi sehingga terdapat arah vektor normal yang berbeda pada setiap Group Feature. Sudut antara masing-masing bidang pada Group Feature dapat diketahui dengan perkalian silang (*cross product*) antara vektor normal satu dengan lainnya yang berdekatan. Aturan identifikasi fitur dapat dilihat pada gambar 3.6 di bawah ini.



Gambar 3.6 Aturan identifikasi fitur

Gambar di atas menunjukkan bahwa jika arah vektor normal tegak lurus satu sama lain atau membentuk sudut 90° yang terdiri dari empat (4) bidang maka dinamakan “Pocket”, jika arah vektor normal tidak tegak lurus atau sudut ≠ 90° satu sama dan mempunyai beda sudut sama dengan atau mendekati 0 maka dinamakan “Cylindrical”, dan jika bukan keduanya dinamakan “Profile”. Identifikasi fitur ini sebagai informasi untuk membantu mempercepat aktivitas perancangan proses manufaktur. Adapun algoritma identifikasi fitur dapat ditunjukkan sebagaimana berikut ini.

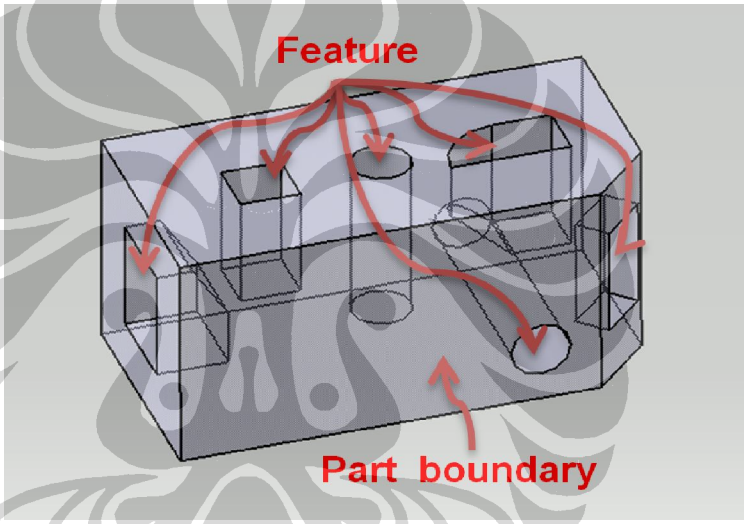
```
Algorithm: Rule of Feature Identification
Input: GroupFeature
Output: Type of Feature

Reading_the_data_groupfeature
Define_normal_vector
For(NV_every_group)
  If(Θ=90° and sumNV=4)
    as “POCKET”
  Elseif(Θ≠90° and ΔΘ≈0)
    as “CYLINDRICAL”
  Else
    as “PROFILE”
  End
End
```

BAB IV
HASIL IMPLEMENTASI ALGORITMA DAN ANALISA

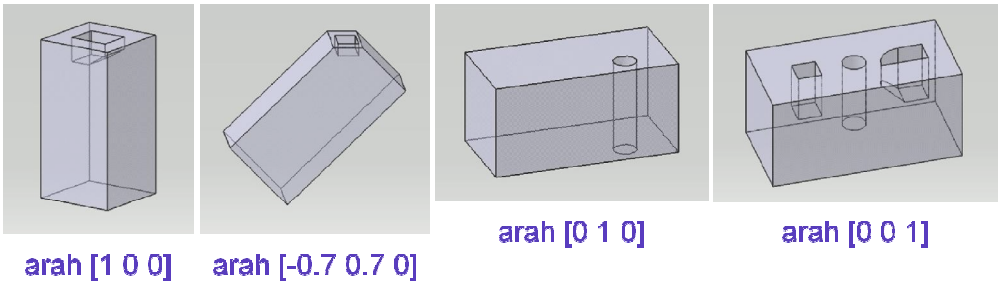
4.1 Hasil Penelitian

Metode yang dijelaskan di atas, diimplementasikan pada model *part* 3D. Misalkan model terdiri dari tiga (3) buah fitur yaitu : *pocket*, *cylindrical*, dan *profile* pada arah [0 0 1], fitur *cylindrical* pada arah [0 1 0], fitur *profile* pada arah [1 0 0], serta fitur *pocket* pada arah [-0.707 0.707 0] yang dapat dilihat pada gambar 4.1 sebagaimana berikut ini.



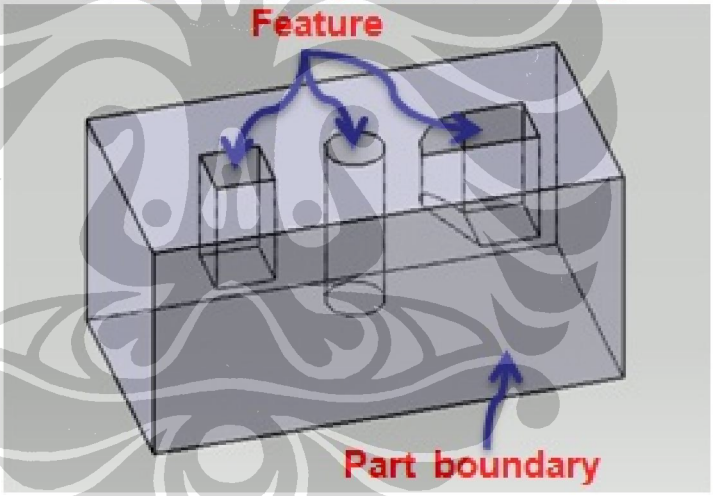
Gambar 4.1 Model CAD 3 Dimensi

Setelah model CAD 3 dimensi di-ekspor ke dalam format STL, selanjutnya dilakukan pengolahan struktur data sehingga didapat *list index segitiga* dan *list index vertex*. Kemudian, pendeteksian arah vektor normal terluar sehingga didapatkan bidang yang kemungkinan terdapat fitur. Arah vektor normal tersebut ditentukan sudutnya terhadap arah z-axis atau [0 0 1] untuk selanjutnya dilakukan rotasi. Dengan menggunakan transformasi rotasi, bidang yang terdeteksi arah vektor normal-nya, dirotasi menuju arah z-axis seperti diilustrasikan pada gambar 4.2. Setelah model faset 3D pada suatu bidang dirotasi, kemudian dilakukan proses *slicing*, *grouping feature*, dan identifikasi jenis fitur. Langkah ini dilakukan berulang-ulang sesuai dengan jumlah bidang arah vektor normal yang terdeksi kemudian dilakukan proses identifikasi fitur.



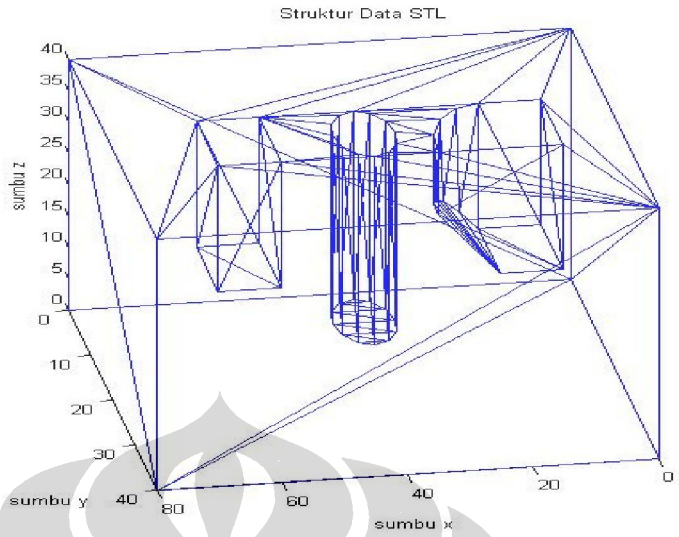
Gambar 4.2 Ilustrasi transformasi rotasi

Untuk mempermudah pemahaman identifikasi fitur berbagai bidang pada model di atas, dimisalkan salah satu bidang yang terdeteksi adalah bidang xy atau arah z-axis [0 0 1] sehingga dapat disederhanakan seperti model pada gambar 4.3 di bawah ini, dikarenakan fitur yang berada pada posisi bidang selain bidang xy bila dilakukan proses *slicing* dan *grouping feature* akan menjadi bagian dari grup *part boundary* yang nantinya akan dihilangkan.



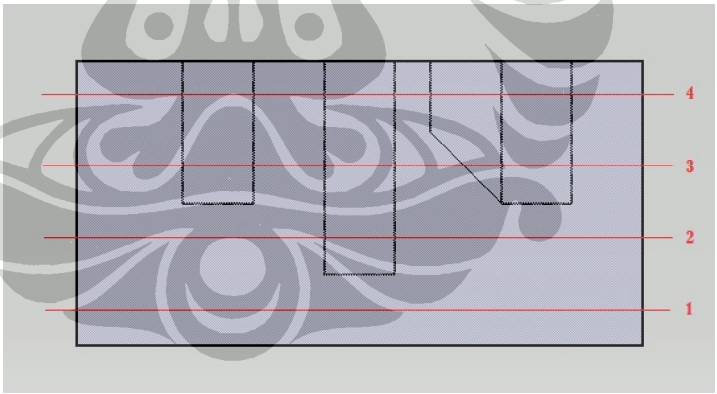
Gambar 4.3 Model fitur bidang XY

Model *part* di atas diubah dalam bentuk STL file yang terdiri dari informasi faset, vektor normal, serta *vertex-vertex* pembentuknya yang diikuti koordinat x, y, dan z masing-masing *vertex*. Informasi tersebut diolah menjadi struktur data yang terdiri dari *list index segitiga* dan *list index vertex*. Struktur data tersebut dapat divisualisasikan menjadi bentuk model faset 3D sebagaimana terlihat pada gambar 4.4, dimana model terdiri dari susunan segitiga yang membentuk menyerupai model *part*.



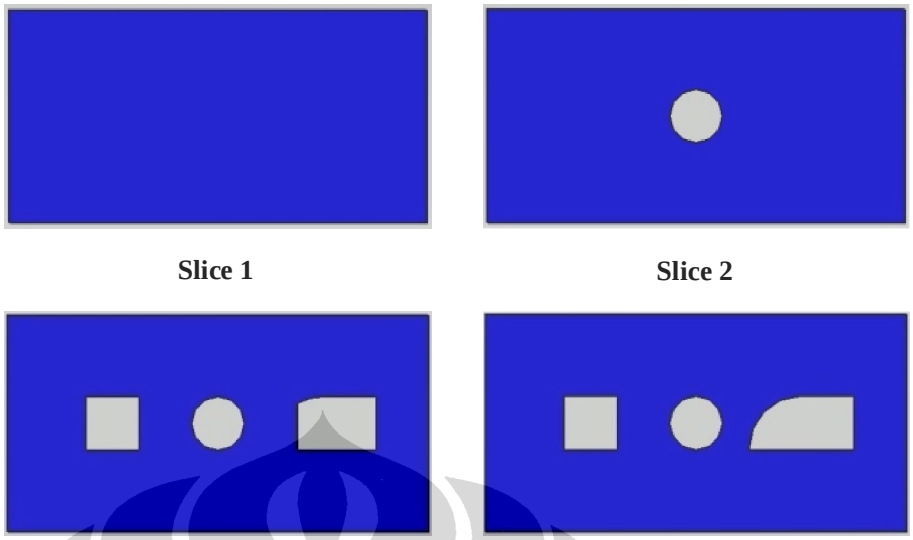
Gambar 4.4 Model faset 3D bidang XY

Gambar 4.5 merupakan ilustrasi proses *slicing* model faset yang diiris pada bidang XY sebanyak empat (4) kali dengan Zminmax = 40 dan interval 10. Penentuan *slicing* pertama dilakukan dengan nilai seperdua dari interval *slicing*. Besaran interval mempengaruhi keakuratan data.



Gambar 4.5 View *slicing* bidang XY

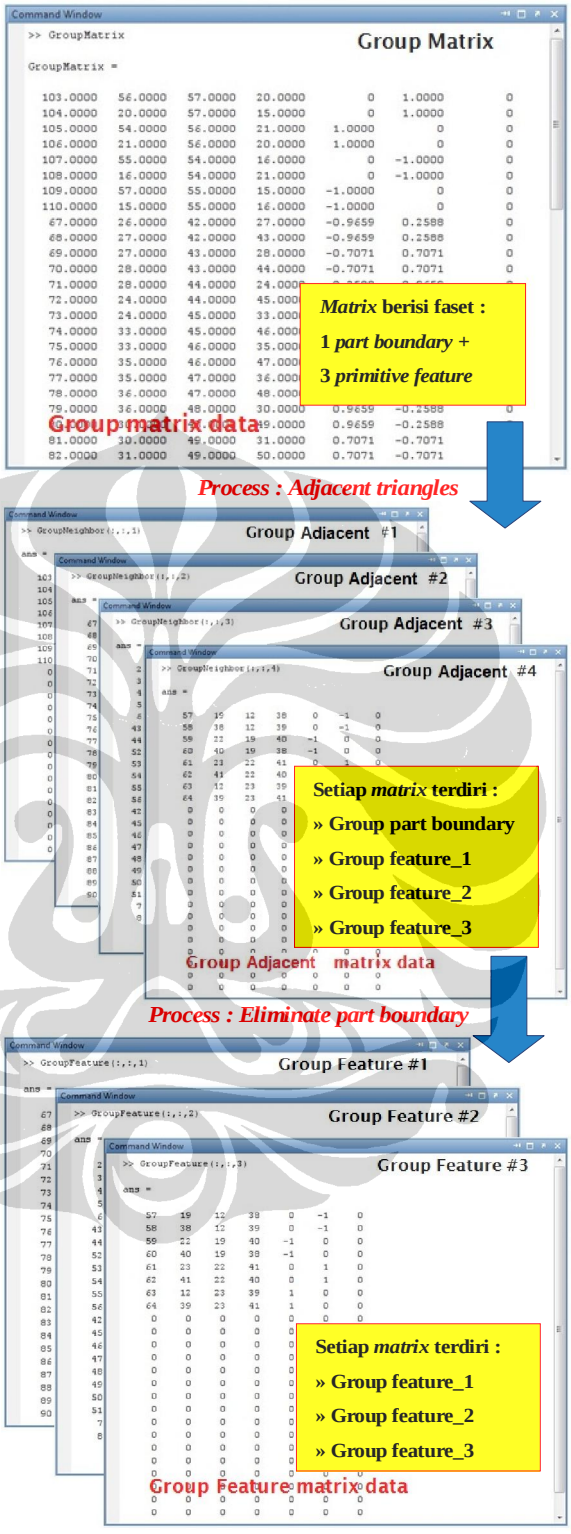
Hasil *slicing* tiap interval dapat diilustrasikan pada gambar 4.6. Dapat dilihat bahwa pada *slice* 1, segitiga yang berpotongan adalah *part boundary*. Pada s*lice* 2, yang berpotongan adalah *part boundary* dan *cylindrical*. Selanjutnya pada *slice* 3, yang berpotongan adalah *part boundary*, *pocket*, *cylindrical*, dan *profile* bagian bawah. Kemudian pada *slice* 4, yang berpotongan adalah *part boundary*, *pocket*, *cylindrical*, dan *profile* bagian atas.



Gambar 4.6 View hasil *slicing* bidang XY

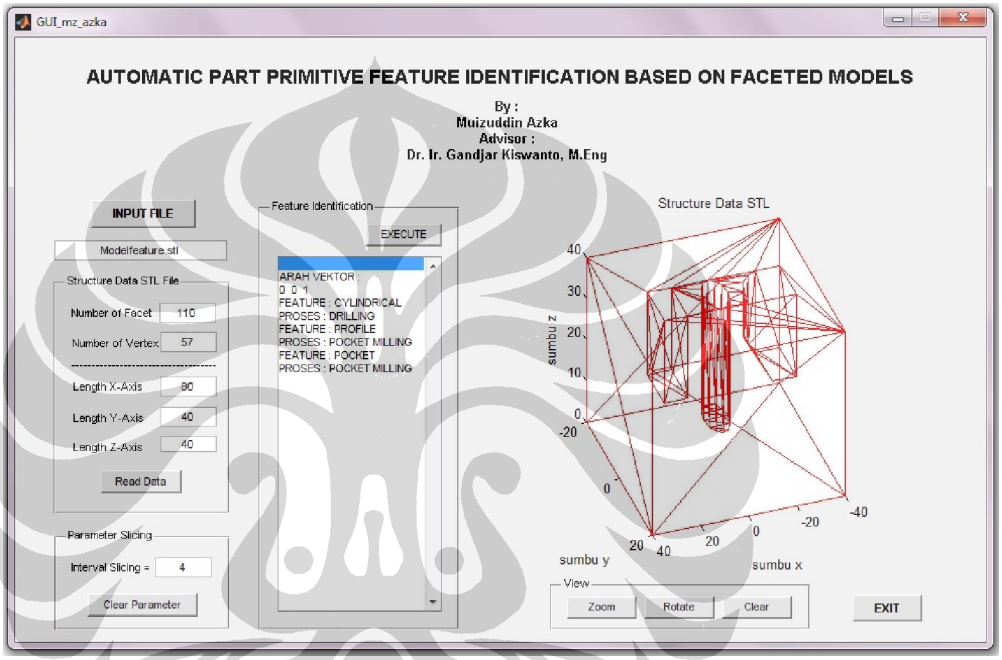
Data faset *slice* tiap interval tersebut kemudian disimpan dalam *array* yang selanjutnya akan digunakan untuk langkah pembuatan Group Feature. Faset-faset hasil *slicing* merupakan sekumpulan segitiga yang apabila dihubungkan antara faset satu dengan faset lainnya yang berdekatan akan membentuk karakteristik tertentu dalam proses identifikasi fitur suatu *part.*

Gambar 4.7 di bawah menunjukkan hasil Group Feature yang didapatkan dari beberapa langkah yaitu : Langkah *sorting*; yaitu pengelompokan seluruh faset *slice* menjadi satu kesatuan matrik, jika terdapat faset yang sama diambil salah satu sehingga matrik berisi faset-faset pembentuk fitur. Langkah *adjacent triangles*; yaitu pembuatan grup yang berisi faset-faset yang memiliki sifat segitiga yang berdekatan (*adjacent triangles*) dari data Group Matrix sehingga didapatkan beberapa Group Adjacent. Langkah *eliminate part boundary*; yaitu menghilangkan Group Adjacent yang merupakan *part boundary* dengan pengecekan koordinat maksimum pada setiap grup sehingga didapatkan Group Feature pada suatu model faset 3D secara otomatis.



Gambar 4.7 Hasil *grouping feature*

Langkah terakhir adalah mengidentifikasikan jenis fitur pada Group Feature dengan menggunakan karakteristik arah vektor normal pada grup. Pada gambar 4.8 di bawah dapat dilihat bahwa fitur terdiri dari : *Cylindrical, Profile,* dan *Pocket*. Urutan jenis fitur ini mengikuti urutan pembentukan Group Feature yang kemudian dilakukan identifikasi fitur dengan menggunakan aturan karakteristik arah vektor normal pada grup tersebut.

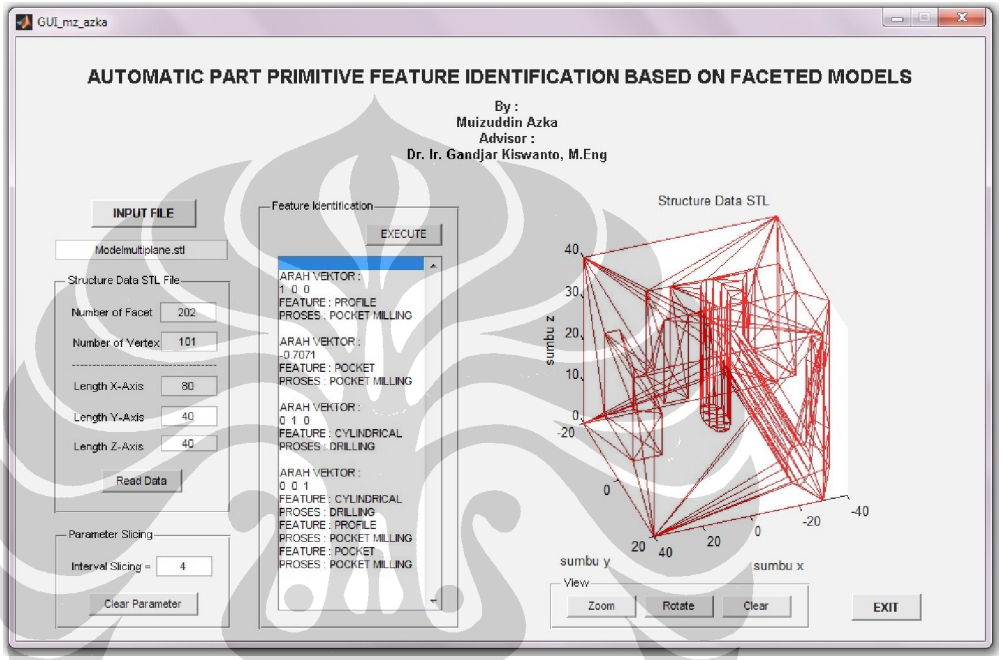


Gambar 4.8 Hasil identifikasi fitur

Pada fitur *Cylindrical*, diketahui bahwa arah vektor normal antara bidang satu dengan lainnya tidak membentuk sudut 90° dan sudut-sudut bidang mendekati 0. Sedangkan fitur *Pocket*, dapat dilihat bahwa arah vektor normal antara bidang yang satu dengan lainnya membentuk sudut 90° dan terdapat empat (4) buah bidang vektor normal. Kemudian fitur *Profile*, arah vektor normal ada yang membentuk sudut 90° dan ada juga yang tidak, maka aturan fitur yang bukan "cylindrical" atau "pocket" dinamakan "profile".

Setelah fitur pada bidang xy teridentifikasi, selanjutnya adalah identifikasi fitur pada bidang yang telah terdeteksi yaitu bidang pada arah [0 1 0], [1 0 0], dan [-0.707 0.707 0]. Dengan cara yang sama, maka model faset pada bidang

selanjutnya dirotasi ke arah bidang xy atau [0 0 1] yang kemudian dilakukan proses *slicing*, *grouping feature* dan identifikasi jenis fitur sehingga terdeteksi fitur pada bidang tersebut. Langkah tersebut diulang sampai pada bidang yang terdeteksi terakhir. Hasil identifikasi fitur pada berbagai bidang dapat ditunjukkan pada gambar 4.9 sebagaimana berikut ini.



Gambar 4.9 Hasil identifikasi fitur berbagai bidang

Gambar di atas menunjukkan bahwa fitur pada arah vektor [1 0 0] berupa *Profile*, arah vektor [-0.707 0.707 0] berupa *Pocket*, arah vektor [0 1 0] berupa *Cylindrical*, serta arah vektor [0 0 1] berupa *Cylindrical*, *Profile* dan *Pocket* yang berarti program identifikasi fitur sesuai dengan model CAD 3D yang dibuat. Hal ini menjelaskan bahwa program yang telah dibuat mampu mendeteksi fitur pada berbagai bidang dengan baik.

Hasil penelitian pengenalan fitur (*feature recognition*) ini mampu mendeteksi fitur meliputi : *pocket*, *cylindrical*, dan *profile* pada berbagai bidang suatu *part* dengan menggunakan metode transformasi rotasi, *slicing*, *grouping feature*, dan aturan identifikasi fitur sehingga mampu untuk mengidentifikasi jenis fitur primitif berbasis model faset.

4.2 Simulasi *Graphical User Interface*

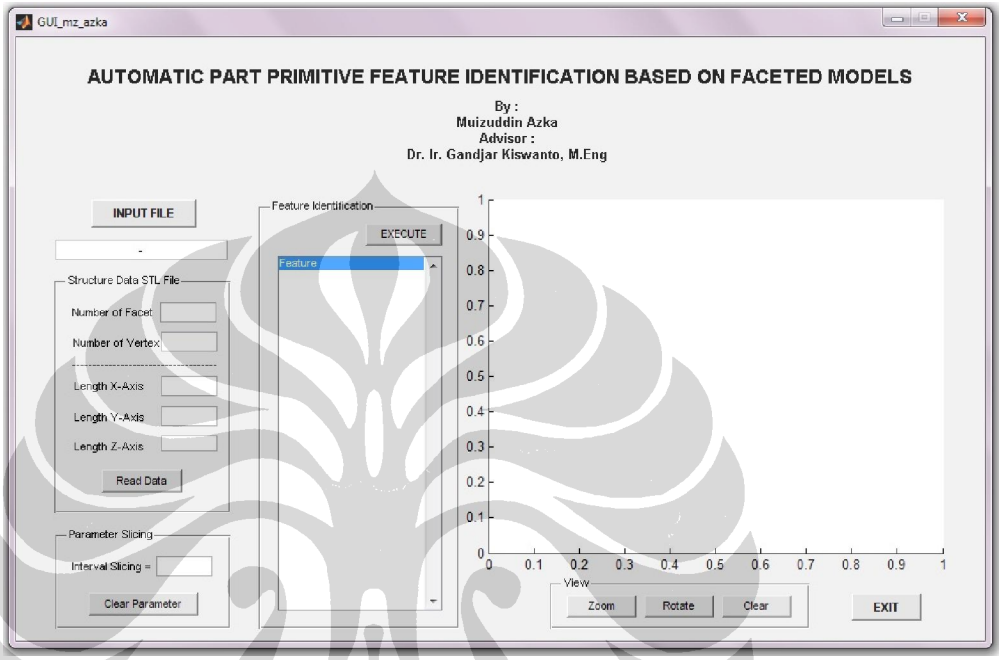
Dalam pembuatan sebuah aplikasi untuk kebutuhan dasar sangat diperlukan tampilan grafis yang sangat menunjang untuk memudahkan pengoperasian suatu sistem program dan sebagai media komunikasi antara program yang dibuat dengan *user* penggunanya. Seperti halnya aplikasi berbasis bahasa pemrograman (*programming*) lainnya, software *Matrix Laboratory* diantaranya MATLAB (dalam *platform Windows*) atau SKYLAB (dalam *platform Linux*) juga mempunyai fitur dasar dukung pembuatan aplikasi berbasis *Graphical User Interface* (GUI), dimana dalam GUI tersebut memuat sejumlah informasi yang meminta *user* untuk memasukkan inputan dan sejumlah informasi lainnya yang menginformasikan kepada *user* sebagai keluaran program.

Sama halnya dengan program pembuat aplikasi lainnya, MATLAB mempunyai kemudahan dalam pembuatan GUI dan hasil dari pembuatan GUI ini menghasilkan 2 file, yaitu : Pertama, tampilan (*figure*) yang berekstensi *.fig. Kedua, fungsi panggil (*callback*) yang berekstensi *.m. Kedua file ini mempunyai nama yang sama. Dalam pembuatan GUI, langkah-langkah yang harus dilakukan antara lain adalah :

a. Desain tampilan (*layout*) GUI yang diinginkan.
b. Komunikasi perangkat (*touch button*) dan lainnya (*other function button*).
c. Urutan program utama (*main programming*) beserta sub programnya yang diletakkan di dalam fungsi panggil (*callback*).
d. Fitur *output* dalam bentuk *output file* atau grafik koordinat.
e. Informasi *output* lainnya yang diperlukan *programming* atau *user* lainnya.
f. *Platform* dasar sistem yang digunakan (sistem operasi yang mendukung dan aplikasi *multi support* lainnya yang dibutuhkan). Kategori ini tidak selalu sama dengan bahasa pemrograman lainnya, karena MATLAB perlu dukungan *compiler* lain untuk membuat dirinya bisa menjalankan *STAND-ALONE Application*. Seperti halnya *compiler* yang digunakan yaitu *Free Command Tool* (*from Borland International Language Programming*).

4.2.1 Desain *Layout*

Pembuatan desain *layout* diusahakan sekomunikatif mungkin, dimana semua yang terkandung dalam *layout* tersebut adalah bentuk yang mudah dimengerti oleh *user*, seperti contoh pada gambar 4.10 di bawah ini :



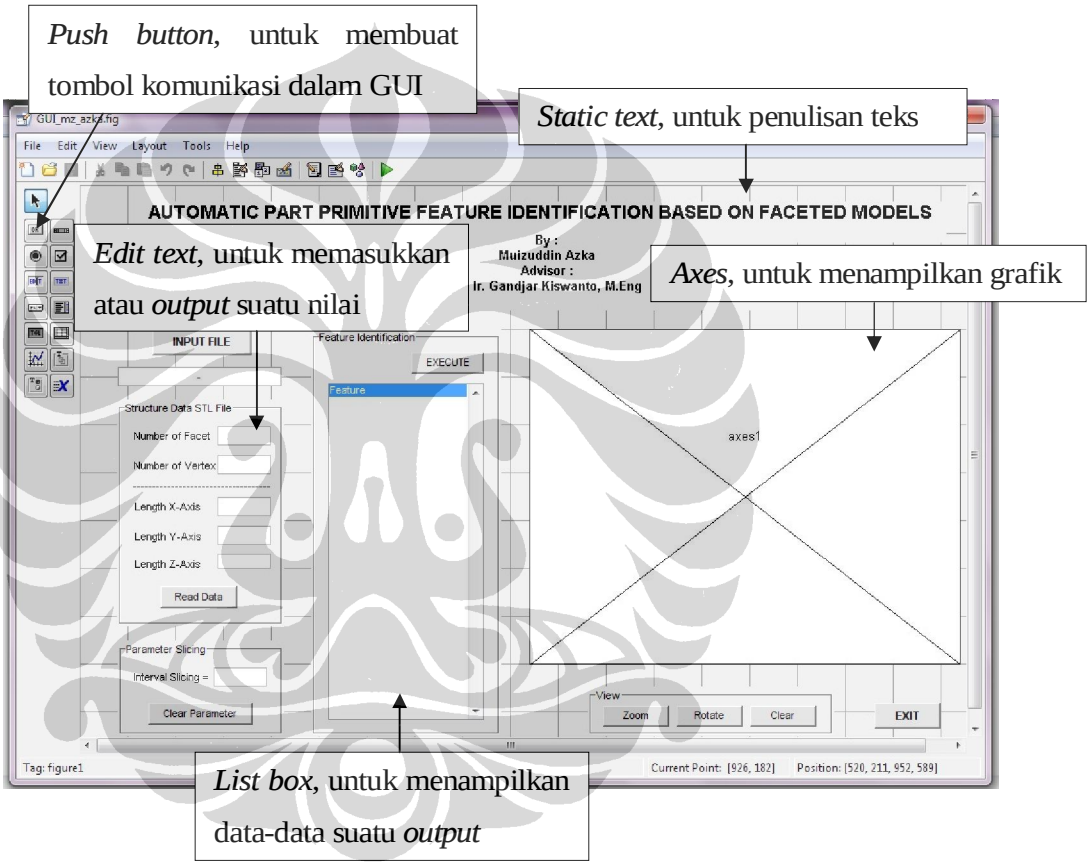
Gambar 4.10 *Layout* GUI untuk berkomunikasi operator dengan program inti

Gambar di atas, menunjukkan bahwa *push button* "INPUT FILE" digunakan untuk memasukkan STL file yang akan diproses. Sedangkan *push button* "Read Data" berguna untuk mengolah data pada *panel* "Structure Data STL File" berupa : jumlah segitiga dan *vertex* serta panjang koordinat x, y, dan z pada model faset 3D. Pada *static text* "Interval Slicing" pada *panel* "Parameter Slicing" memasukkan nilai interval pada "edit text" untuk penentuan interval *slicing* yang akan digunakan pada program utama. Jika ingin mengganti nilai interval *slicing*, dengan menekan *push button* "Clear Parameter" dan memasukkan kembali nilai interval *slicing* pada "edit text".

Push button "EXECUTE" pada *panel* "Feature Identification" berguna untuk mengeksekusi program utama sehingga dapat mengidentifikasi jenis fitur pada model faset yang ditunjukkan pada "list box" Feature Identification.

Visualisasi model faset 3D ditampilkan pada ”graphics axes” dengan dilengkapi *push button* “Zoom, Rotate, dan Clear” pada *panel* “View” untuk mengubah tampilan model faset 3D pada “graphics axes”. Untuk mengakhiri atau keluar dari *Graphical User Interface* (GUI) menggunakan *push button* “EXIT”.

Langkah-langkah yang dilakukan dalam pembuatan desain *layout Graphical User Interface* (GUI) dapat dilihat pada gambar 4.11 sebagaimana berikut di bawah ini :



Gambar 4.11 Fungsi *callback*

1. Gunakan fitur : *pushbutton, static text, edit text, list box,* dan *graphics axes* untuk membuat desain *layout* GUI seperti di atas.
2. Edit semua fitur tersebut dengan menggunakan *double click* atau klik dua (2) kali pada bagian fitur tersebut untuk penentuan identitas nama, *font,* karakter, dan lain sebagainya.
3. Buat informasi ID Tag seperti diilustrasikan pada tabel di bawah ini.

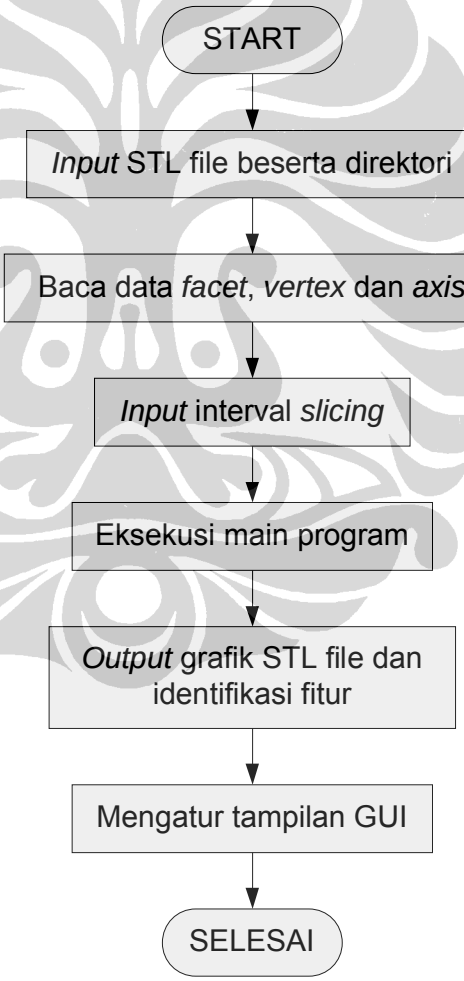
Dalam tabel 4.1 di bawah ini adalah menentukan informasi **ID Tag** dalam operator fungsi GUI yang digunakan, dimana **Tag** tersebut adalah sebuah nama sebagai pengenal dan penghubung antara program inti dengan GUI.

Tabel 4.1 Operator Tag dan Parameternya

SEBAGAI INPUT OPERATOR		
Deskripsi	**ID Tag**	**Keterangan**
Push Button (1) / Input File	namaFile	*Push button menu* untuk memasukkan STL file
Edit Text (1) / Interval Slicing	edit14	*Edit menu* untuk memasukkan nilai interval *slicing*
SEBAGAI OUTPUT OPERATOR		
Deskripsi	**ID Tag**	**Keterangan**
Edit Text (2) / Number of Facet	edit9	*Edit text* untuk *output* jumlah segitiga pada model faset
Edit Text (3) / Number of Vertex	edit10	*Edit text* untuk *output* jumlah *vertex* pada model faset
Edit Text (4) / Length X-Axis	edit11	*Edit text* untuk *output* panjang koordinat X
Edit Text (5) / Length Y-Axis	edit12	*Edit text* untuk *output* panjang koordinat Y
Edit Text (6) / Length Z-Axis	edit13	*Edit text* untuk *output* panjang koordinat Z
List Box (1) / Feature Identification	listbox1	*List box* untuk *output* identifikasi fitur
Graphics Axes (1) / Feature Identification	axes1	*Axes* untuk *output* visualisasi model faset 3D

4.2.2 Komunikasi Perangkat

Menentukan komunikasi antara program inti dan GUI merupakan bagian terpenting sebagai komunikasi perangkat keseluruhan. Dalam menentukan komunikasi antara program inti dan GUI, *programmer* harus memahami bagaimana pengurutan dalam sebuah fungsi panggil (*callback*) dalam GUI di program MATLAB dengan program inti yang dimasukkan ke dalamnya. Pertama kalinya, *programmer* menentukan alur yang dikerjakan di awal dan alur mana yang dikerjakan di akhir, biasanya sebagai *output* yang dapat digambarkan dalam sebuah diagram alir seperti pada gambar 4.12 di bawah ini :



Gambar 4.12 Diagram alir komunikasi perangkat

4.3 Pembahasan dan Analisa

Untuk membantu meningkatkan kinerja CAPP dan CAM dalam proses manufaktur, diperlukan pengenalan fitur secara otomatis. Pengenalan ini digunakan untuk merencanakan tahap selanjutnya agar lebih efisien. Penelitian ini membahas metode pengelompokan fitur dan mengenali jenis fitur tersebut dengan bantuan arah vektor normal. Fitur *pocket*, *cylindrical*, dan *profile* adalah bentuk yang secara umum sering digunakan, metode ini mampu mengidentifikasi fitur tersebut. Hasil dari penelitian ini berguna sebagai sarana informasi proses manufaktur yang dibutuhkan pada *process planning*..

BAB V
KESIMPULAN DAN SARAN PENELITIAN LEBIH LANJUT

Pada penelitian identifikasi fitur primitif secara otomatis pada *part* berbasis model faset yang telah dilakukan, maka didapat hasil kesimpulan dan saran penelitian lebih lanjut sebagai berikut :

5.1 Kesimpulan

Dari algoritma pengenalan fitur yang telah dilakukan, didapatkan kesimpulan sebagaimana berikut ini :

1. Algoritma *slicing* dan *grouping feature* mampu mendeteksi fitur secara otomatis pada bidang suatu *part*.
2. Aturan karakteristik arah vektor normal dapat digunakan untuk mengidentifikasi jenis fitur primitif, meliputi : *pocket*, *cylindrical* dan *profile* pada Group Feature yang terbentuk.
3. Transformasi rotasi dapat digunakan untuk membantu merotasi model *part* menuju bidang referensi (dalam penelitian ini bidang XY) agar dapat dilakukan proses identifikasi fitur secara otomatis pada berbagai bidang.

5.2 Saran penelitian lebih lanjut

Untuk penelitian bertema identifikasi fitur secara otomatis berbasis model faset, penulis menyarankan untuk dilakukan penelitian sebagai berikut :

1. Perlu pengembangan algoritma *slicing* lebih lanjut tanpa melakukan *input* interval *slicing* berupa metode *adaptive slicing*.
2. Perlu dikembangkan identifikasi jenis fitur yang lain, tidak hanya berupa *pocket*, *cylindrical* dan *profile*.
3. Perlu dilakukan manajemen memori dalam program yang lebih baik agar dapat dilakukan operasional program yang lebih cepat dan akurat.
4. Perlu dicoba beberapa metode lain yang belum diimplementasikan dalam penelitian ini sebagai pembanding dan pengembangan lebih lanjut.

DAFTAR PUSTAKA

[1] Pande, V.B. Sunil and S.S. "*Automatic recognition of feature from freeform surface CAD models"* Elsevier, 2008.

[2] R. Abu and M. M. Tap, "*Attribute based feature recognition for machining features*" Jurnal Teknologi vol. 46.A Universiti Teknologi Malaysia, Jun 2007.

[3] G. Kiswanto and A. Yahya, "*Development of Closed Bounded Volume (CBV) Grouping Method of Complex Faceted Model through CBV Boundaries Identification*" Proc. ICCAE 2010, 2010 The 2nd International Conference on Computer and Automation Engineering, Feb. 2010.

[4] C. W. Raymond, J. S. Corney, and D. E. R. Clark, "*Octree based recognition of assembly features,*" Proc. DETC'00, ASME 2000 Design Engineering Technical Conferences and Computer and Information in Engineering Conferences, Sept. 2000, pp. 1-10.

[5] Mangesh P. Bhandarkar, Rakesh Nagi, "*STEP-based feature extraction from STEP geometry for Agile Manufacturing*", Elsevier, 2000.

[6] C. Weber, S. Hahmann, H. Hagen, " *Sharp Feature Detection in Ploint Clouds*", IEEE International Conference on Shape Modeling and Applications (SMI), 2010.

[7] G. Kiswanto and A. Mujahid, "*Pengembangan Algoritma Cepat Penentuan Titik Kontak Pahat (cutter contact point) pada Sistem CAM Berbasis Model Facet 3D untuk Permesinan Awal (roughing) dan Akhir (finishing)*", Prociding SNTTM V, UI Jakarta, 2006.

[8] Francesco Bianconi, " *Bridging The Gap between CAD and CAE using STL files*", International Journal of CAD/CAM Vol. 2, No. 1, pp. 55-67, 2002.

[9] Pierre P. Lefebvre and Bert Lauwers, " *STL Model Segmentation for Multi-Axis Machining Operations Planning*", IEEE International Conference on Shape Modeling and Applications (SMI), 2010.

[10] G. Kiswanto, R. Widyanto and P.E. Kreshna, "*Feature Based Milling Direction of a Faceted Model Based on Intelligent Fuzzy 3D Feature Identification*", http://ui.academia.edu

LAMPIRAN

KEGIATAN KOMPUTASI

Penelitian	Automatic Part Primitive Feature Identification Based on Faceted Models
Kegiatan	Pengolahan Struktur Data Model Faset 3D beserta Visualisasi Model Grafik
Tahapan proses	Data Processing

```
% ---------------- AUTOMATIC PART PRIMITIVE FEATURE IDENTIFICATION ----------------
  ------------------------------------ BASED ON FACETED MODELS ------------------------------------
% ---------------------------------- OLEH : MUIZUDDIN AZKA, ST. ----------------------------------
% ------------------------------------------ NPM  : 0906651416 ------------------------------------------
% --------------- DOSEN PEMBIMBING : DR. IR. GANDJAR KISWANTO, M.ENG --------------
% --------- DEPARTEMEN TEKNIK MESIN FAK.TEKNIK UNIVERSITAS INDONESIA ---------
% ------------------------------------------------- 2012 -------------------------------------------------

clear all ;
clc ;

fid = fopen ('ModelFeature.stl','r');

% "PROSES PENGOLAHAN STRUCTURE DATA FACETED MODELS"

xminmax=[1000 -1000];          % asumsi awal
yminmax=[1000 -1000];
zminmax=[1000 -1000];

i = 1 ;                % variabel yang menampung index segitiga
j = 1 ;                % variabel yang menampung index vertex
count = 0 ;

arr = zeros (3);       % membentuk matrik 3x3 bernilai 0

while(1)

   aa=fgetl(fid);      % mengambil data
   if ~ ischar (aa);
      break
   end
   [a,b] = strtok (aa,' ');

   if strcmp (a,'facet');
      [c,d]= strtok (b,' ');
      for k = 4:6,
         [e,d]= strtok (d,' ');
         hihi = inline (e);
         T(i,k)= hihi(1);
      end
      ii = 0 ;
      count = 1;
```

```
   elseif strcmp (a,'vertex')

       ii = ii + 1 ;
       [a,b]= strtok (b,' ');   % a menyimpan koordinat vertex X dalam bentuk string
       [b,c]= strtok (b,' ');   % b menyimpan koordinat vertex Y dalam bentuk string
       haha = inline (a)  ;    % c menyimpan koordinat vertex Z dalam bentuk string
       huhu (1) = haha (1);
       x(ii) = haha (1);          % nilai huhu(1) = x(ii)
       haha = inline (b);        % huhu [x y z ] ->  menyimpan koordinat x y z dalam
matrix
       huhu (2) = haha (2);    % x(ii) -> menyimpan koordinat x ke ii dari vertex ke ii
setelah di inline
       y(ii) = haha (1);
       haha = inline (c);
       huhu (3) = haha (3);    % huhu (3) -> menyimpan koordinat x,y,z setiap index
vertex
       z(ii)= haha (1);

       % disp(huhu)
       v(j,1) = 0.1 ;              % berfungsi hanya untuk memberikan nilai awalan di
variabel v untuk keperluan eliminasi
       v(j,2) = 0.1 ;
       v(j,3) = 0.1 ;

       index = 0;
       for kaka = 1:j,
          if (abs(v(kaka,1)-huhu(1))<0.001 && abs (huhu(2)-v(kaka,2))<0.001 ...
                && abs (huhu(3)- v(kaka,3))<0.001)
             index = kaka;

       break;                      % if ini berfungsi untuk mengeliminasi koordinat
vertex yang sama dengan yang sudah
                                   % ada dan tersimpan di variabel nilai huhu(1),(2),(3)
yang baru didapat dari proses
                                   % strtok dikurangi dengan setiap nilai koorinat yang
sudah tersimpan di variabel v
                                   % bila hasil pengurangan TIDAK kurang dari 0.001
maka nilai huhu disimpan di variabel v
                                   % sebagai nilai baru (mengudate nilai di variabel v )
                                   % jika nilai huhu tersebut kurang dari 0.001 maka,
pada nilai kaka berapa
                                   % huhu tersebut bernilai kurang dari 0.001, nilai
kaka tersebut disimpan
                                   % di variabel index dan disimpan pada
                                   % variabel arr nilai j nya disimpan

          end
       end

       if index == 0,              % masukkan ke vertex
```

```
        %disp('masuk');
        v(j,1)=huhu(1);        % mendata koordinat vertex,
        v(j,2)=huhu(2);        % bila ada koordinat vertex yang x y z nya sama
maka akan dieliminasi
        v(j,3)=huhu(3);

        if xminmax (1) >v(j,1),
           xminmax(1) = v(j,1);
        end
        if xminmax (2) <v(j,1),
           xminmax(2) = v(j,1);
        end

        if yminmax (1) >v(j,2),
           yminmax (1) = v(j,2);    % mereplace nilai yang ada di yminmax(1)
dengan nilai v(j,2)
        end
        if yminmax(2)<v(j,2),
           yminmax (2)= v(j,2);
        end

        if zminmax (1) > v(j,3),
           zminmax (1) = v(j,3);
        end
        if zminmax (2) < v(j,3),
           zminmax (2)=v(j,3);
        end

        arr(count) = j;                % menyimpan index vertex mana saja yang
lolos dari eiminasi

     else
        arr(count)=index;

        %disp ('tidak masuk');
        j= j-1 ;
     end

     j=j+1;                          % setelah proses eliminasi dan penyimpanan
selesai lakukan increment
     count =count +1 ;               % j menunjukan jumlah index vertex

  elseif strcmp (a, 'endsolid'),
     i=i-1;
     j=j-1  ;

  elseif strcmp (a, 'endfacet')
     T(i,1)=arr(1);
     T(i,2)=arr(2);
     T(i,3)=arr(3);
     i=i+1 ;                         % jumlah segitiga
  end
end
```

```
% menentukan index segitiga dan index vertex
for i=1:i,
  % disp ([i T(i,1) T(i,2) T(i,3) T(i,4) T(i,5)  T(i,6)]);
  indexfacetOK(i,:)=([i T(i,1) T(i,2) T(i,3) T(i,4) T(i,5) T(i,6)]);
end

ListIndexSegitiga=indexfacetOK;


% menentukan index vertex dan koordinat vertex
for ii = 1:j ,
   %disp ([ii v(ii,1) v(ii,2) v(ii,3)]);
   indexvertexOK(ii,:)=([ii v(ii,1) v(ii,2) v(ii,3)]);
end

ListIndexVertex=indexvertexOK;

matrixfacetNV=indexfacetOK;
pjgidxfct=length(matrixfacetNV(:,1));
matrixvertex=indexvertexOK;

for i = 1:pjgidxfct
   axis square
   hold on
   koordX=[matrixvertex(matrixfacetNV(i,2),2) matrixvertex(matrixfacetNV(i,3),2)
matrixvertex(matrixfacetNV(i,4),2) matrixvertex(matrixfacetNV(i,2),2)];
   koordY=[matrixvertex(matrixfacetNV(i,2),3) matrixvertex(matrixfacetNV(i,3),3)
matrixvertex(matrixfacetNV(i,4),3) matrixvertex(matrixfacetNV(i,2),3)];
   koordZ=[matrixvertex(matrixfacetNV(i,2),4) matrixvertex(matrixfacetNV(i,3),4)
matrixvertex(matrixfacetNV(i,4),4) matrixvertex(matrixfacetNV(i,2),4)];
   plot3(koordX,koordY,koordZ,'r')
   title('Structure Data STL')
   xlabel('sumbu x')
   ylabel('sumbu y')
   zlabel('sumbu z')
end
```

Penelitian	Automatic Part Primitive Feature Identification Based on Faceted Models
Kegiatan	Rotasi Model Faset 3D pada Bidang Vektor Normal Terluar menuju Bidang Referensi
Tahapan proses	Transformasi Rotasi

```
% "PROSES ROTATION MATRIX"

ang=0;
for i=1:i,
indexfacet(i,:)=([i T(i,1) T(i,2) T(i,3) cos(ang*pi/180)*T(i,4)-
sin(ang*pi/180)*T(i,5)+0*T(i,6)
sin(ang*pi/180)*T(i,4)+cos(ang*pi/180)*T(i,5)+0*T(i,6)
0*T(i,4)+0*T(i,5)+1*T(i,6)]);
end
iiindexfacet(:,:,1)=indexfacet;
ListIndexSegitigaRotate=indexfacet;

for ii = 1:j,
vertex2(ii,:) = [ii cos(ang*pi/180)*v(ii,1)-sin(ang*pi/180)*v(ii,2)+0*v(ii,3)
sin(ang*pi/180)*v(ii,1)+cos(ang*pi/180)*v(ii,2)+0*v(ii,3)
0*v(ii,1)+0*v(ii,2)+1*v(ii,3)];
end

YY=vertex2;

AAAA=sortrows(YY,2);

xminmax(1)=AAAA(1,2);
xminmax(2)=AAAA(j,2);

BBBB=sortrows(YY,3);

yminmax(1)=BBBB(1,3);
yminmax(2)=BBBB(j,3);

CCCC=sortrows(YY,4);

zminmax(1)=CCCC(1,4);
zminmax(2)=CCCC(j,4);

deltax=xminmax(1)- 0;          % karena nilai dari x y dan z minmax(1)= 0
deltay=yminmax(1)- 0;          % maka delta x y dan  z == 0
deltaz=zminmax(1)- 0;
```

```
% normalisasi
xminmax(1)=xminmax(1)-deltax;
xminmax(2)=xminmax(2)-deltax;
yminmax(1)=yminmax(1)-deltay;
yminmax(2)=yminmax(2)-deltay;
zminmax(1)=zminmax(1)-deltaz;
zminmax(2)=zminmax(2)-deltaz;

zmimax(:,:,1)=[zminmax(1) zminmax(2)];

% menentukan index vertex dan koordinat vertex
for ii = 1:j ,
   YY(ii,2)=YY(ii,2)- deltax;
   YY(ii,3)=YY(ii,3)- deltay;
   YY(ii,4)=YY(ii,4)- deltaz;

   yiyi(:,:,1)=[YY(ii,2) YY(ii,3) YY(ii,4)];
   indexvertex(ii,:)=([ii YY(ii,2) YY(ii,3) YY(ii,4)]);
end
iindexvertex(:,:,1)=indexvertex;
ListIndexVertexRotate=indexvertex;

pjgvertexR=length(ListIndexVertexRotate(:,1));
ax=sortrows(ListIndexVertexRotate,2);
ay=sortrows(ListIndexVertexRotate,3);
az=sortrows(ListIndexVertexRotate,4);

xmaxx=ax(pjgvertexR,2);
xminn=ax(1,2);
ymaxx=ay(pjgvertexR,3);
yminn=ay(1,3);
zmaxx=az(pjgvertexR,4);
zminn=az(1,4);

indexnya=0;
for xy = 1:pjgvertexR
   kordxxx=az(xy,2);
   kordyyy=az(xy,3);
   koordzzz=az(xy,4);
   indexyz=az(xy,1);
   if koordzzz == zmaxx & kordxxx==xmaxx | koordzzz == zmaxx &
kordxxx==xminn | koordzzz == zminn & kordxxx==xmaxx | koordzzz == zminn
& kordxxx==xminn...
     |koordzzz == zmaxx & kordyyy==ymaxx | koordzzz == zmaxx &
kordyyy==yminn | koordzzz == zminn & kordyyy==ymaxx | koordzzz == zminn &
kordyyy==yminn...
     |kordxxx == xmaxx & kordyyy==ymaxx | kordxxx == xmaxx &
kordyyy==yminn | kordxxx == xminn & kordyyy==ymaxx | kordxxx == xminn &
kordyyy==yminn
      indexnya=indexnya+1;
      vertexouter(indexnya,1)=indexyz;
   end
end
```

```
pjgvout=length(vertexouter(:,1));
pjginfct=length(iiindexfacet(:,1));
aabb=0;
for bbcc = 1:pjginfct
XXXX=iiindexfacet(bbcc,2);
YYYY=iiindexfacet(bbcc,3);
ZZZZ=iiindexfacet(bbcc,4);
ccdd=0;
   for pjgverout = 1:pjgvout
      vout = vertexouter(pjgverout,1);
      if XXXX ==vout | YYYY ==vout | ZZZZ ==vout
         ccdd=ccdd+1;
          aabb=aabb+1;
         facetout(aabb,:)=bbcc;
         break
      end

   end
end
facetouter=facetout;
pjgfout=length(facetouter(:,1));
ddee=0;
for infout=1:pjgfout
   ddee=ddee+1;
   infctout(ddee,:)=iiindexfacet(facetouter(infout,:),:);
end
facetluarr=infctout;
pjgfctluar=length(facetluarr(:,1));
for infctluar = 1:pjgfctluar

   nvluarr=[facetluarr(infctluar,5) facetluarr(infctluar,6) facetluarr(infctluar,7)];
   for innvout = 1:pjgfctluar
      nvouter =[facetluarr(innvout,5) facetluarr(innvout,6) facetluarr(innvout,7)];
      if abs(nvluarr(1,1)-nvouter(1,1))< 10^-3  & abs(nvluarr(1,2)-nvouter(1,2)) <
10^-3  & abs(nvluarr(1,3)-nvouter(1,3))< 10^-3

         innvouter (innvout,:)=nvouter;
         break
      end
   end
end
norvecout=innvouter;
pjgnorvout=length(norvecout(:,1));
eeff=0;
for innorvout = 1:pjgnorvout
   normvector=norvecout(innorvout,:);
   if normvector(1,1)~=0 | normvector(1,2)~=0 | normvector(1,3)~=0
      eeff=eeff+1;
      normvectout(eeff,:)=normvector;
   end
end
```

```
nvrotate=normvectout;
pjgnvrot=length(nvrotate(:,1));
ffgg=0;
gghh=0;
while pjgnvrot ~= 0
   ffgg=ffgg+1;
pjgnvrotate=pjgnvrot;
nvrotsatu=nvrotate(1,:);
jmlplane=0;
for pjgnvlain = 2:pjgnvrotate
   nvrotlain=nvrotate(pjgnvlain,:);
dotproduct=nvrotsatu(1,1)*nvrotlain(1,1)+nvrotsatu(1,2)*nvrotlain(1,2)+nvrotsatu
(1,3)*nvrotlain(1,3);
   if dotproduct -(-1) < 0.00001
      gghh=gghh+1;
      nvplane(gghh,:)=nvrotsatu;
      nvrotate(pjgnvlain,:)=[];
      nvrotate(1,:)=[];
      break
   end
   if  dotproduct -(-1) > 0.00001
      jmlplane=jmlplane+1;
      if jmlplane==pjgnvrotate-1
      gghh=gghh+1;
      nvplane(gghh,:)=nvrotsatu;
      nvrotate(1,:)=[];
      end
   end
end
nvrotasi=nvrotate;
pjgnvrotasi=length(nvrotasi(:,1));
if pjgnvrotasi == 0
  pjgnvplane=length(nvplanepart(:,1));
  nvplanepart(pjgnvplane+1,:)=nvrotsatu;
  break
end
hhii=0;
nvrotplane=[];
for idxnvrot =1:pjgnvrotasi
   nvplanepersatu=nvrotate(idxnvrot,:);
   hhii=hhii+1;
   nvrotplane(hhii,:)=nvplanepersatu;
end
nvrotate=nvrotplane;
nvplanepart=nvplane;
pjgnvrot = pjgnvrotasi;
end
nvplanerot=nvplanepart;
jmlnvplane=length(nvplanerot(:,1));
idxsudut=0;
for idxnvplane = 1:jmlnvplane
   cosalfa=nvplanerot(idxnvplane,3)*1;
   if cosalfa >=0
      alfa=acos(cosalfa);
      anglez=-1*alfa;
```

```
    elseif cosalfa < 0
        alfa=(acos(cosalfa));
        anglez=-1*alfa;
    end
    cosbeta=nvplanerot(idxnvplane,1)*1;
    cosibeta=nvplanerot(idxnvplane,2)*1;
    if ( cosbeta >=0 & cosibeta >0 | cosbeta >0 & cosibeta >=0 )        % KUADRAN I
        beta=acos(cosbeta);
        anglex=beta*-1;
    elseif ( cosbeta <= 0 & cosibeta > 0 | cosbeta < 0 & cosibeta >= 0 )
% KUADRAN II
        beta=(acos(cosbeta));
        anglex=beta*-1;
    elseif ( cosbeta <= 0 & cosibeta < 0 | cosbeta < 0 & cosibeta <= 0 )
% KUADRAN III
        beta=2*pi-(acos(cosbeta));
        anglex=beta*-1;
    elseif ( cosbeta >= 0 & cosibeta < 0 | cosbeta > 0 & cosibeta <= 0 )
% KUADRAN IV
        beta=2*pi-(acos(cosbeta));
        anglex=beta*-1;
    elseif cosbeta == 0 & cosibeta == 0
        beta=0;
        anglex=beta*-1;
    end
  sudutx=anglex * 180/pi;
  sudutz=anglez * 180/pi;
  anglexx=anglex;
  anglezz=anglez;
  idxsudut=idxsudut+1;
  anglexz(idxsudut,:)=[anglexx anglezz];
end
matrixangle=anglexz;

for idxnvplane = 1:jmlnvplane
matrot=[cos(anglexz(idxnvplane,1))*cos(anglexz(idxnvplane,2))
cos(anglexz(idxnvplane,2))*sin(anglexz(idxnvplane,1))*-1
sin(anglexz(idxnvplane,2));sin(anglexz(idxnvplane,1)) cos(anglexz(idxnvplane,1))
0; -1*sin(anglexz(idxnvplane,2))*cos(anglexz(idxnvplane,1))
sin(anglexz(idxnvplane,2))*sin(anglexz(idxnvplane,1))
cos(anglexz(idxnvplane,2))];

for i=1:i,
indexfacet(i,:)=([i T(i,1) T(i,2) T(i,3)
T(i,4)*matrot(1,1)+T(i,5)*matrot(1,2)+T(i,6)*matrot(1,3)
T(i,4)*matrot(2,1)+T(i,5)*matrot(2,2)+T(i,6)*matrot(2,3)
T(i,4)*matrot(3,1)+T(i,5)*matrot(3,2)+T(i,6)*matrot(3,3)]);
end
iindexfacet(:,:,idxnvplane)=indexfacet;
ListIndexSegitigaRotate=indexfacet;

for ii = 1:j,
vertex2(ii,:) = [ii  v(ii,1)*matrot(1,1)+v(ii,2)*matrot(1,2)+v(ii,3)*matrot(1,3)
v(ii,1)*matrot(2,1)+v(ii,2)*matrot(2,2)+v(ii,3)*matrot(2,3)
v(ii,1)*matrot(3,1)+v(ii,2)*matrot(3,2)+v(ii,3)*matrot(3,3)];
end
```

```
YY=vertex2;

AAAA=sortrows(YY,2);

xminmax(1)=AAAA(1,2);
xminmax(2)=AAAA(j,2);

BBBB=sortrows(YY,3);

yminmax(1)=BBBB(1,3);
yminmax(2)=BBBB(j,3);

CCCC=sortrows(YY,4);

zminmax(1)=CCCC(1,4);
zminmax(2)=CCCC(j,4);

deltax=xminmax(1)- 0;          % karena nilai dari x y dan z minmax(1)= 0
deltay=yminmax(1)- 0;          % maka delta x y dan  z == 0
deltaz=zminmax(1)- 0;

% normalisasi
xminmax(1)=xminmax(1)-deltax;
xminmax(2)=xminmax(2)-deltax;
yminmax(1)=yminmax(1)-deltay;
yminmax(2)=yminmax(2)-deltay;
zminmax(1)=zminmax(1)-deltaz;
zminmax(2)=zminmax(2)-deltaz;

zmimax(:,:,idxnvplane)=[zminmax(1) zminmax(2)];

% menentukan index vertex dan koordinat vertex
for ii = 1:j ,
   YY(ii,2)=YY(ii,2)- deltax;
   YY(ii,3)=YY(ii,3)- deltay;
   YY(ii,4)=YY(ii,4)- deltaz;

   yiyi(:,:,1)=[YY(ii,2) YY(ii,3) YY(ii,4)];
   indexvertex(ii,:)=([ii YY(ii,2) YY(ii,3) YY(ii,4)]);
end
   iindexvertex(:,:,idxnvplane)=indexvertex;
   ListIndexVertexRotate=indexvertex;
end
```

Penelitian	Automatic Part Primitive Feature Identification Based on Faceted Models
Kegiatan	Pengirisan Model Faset 3D dengan Bidang Referensi (Plane XY) sesuai Jumlah Interval
Tahapan proses	Uniform Slicing

```
% "PROSES SLICING"

% pembagian slicing z
tinggiangkat = 1 ;
jmlhsegitiga = i ;
jmlhvertex = j ;

angkatan = 0;
sisi=0;
for plane=1:jmlnvplane
    lindexvertex=iindexvertex(:,:,plane);
    lindexfacet=iindexfacet(:,:,plane);
    nzmimax=zmimax(:,:,plane);

%ambil segitiga-segitiga di setiap angkatan
ztertinggi = -1000;          % asumsi awal..
zterendah = 1000;
koordinat=lindexvertex;
faccet=lindexfacet;
counterangkat = 0;
groupneighbour=[];
matrixneighbor=[];
matrixfacet=[];
groupmatrix=[];
ambilsemuafacet=[];
vekn=[];
facetslicingOK=[];
facetygtdksama=[];
groupmatrix=[];
matrixx=[];

j=0;
ABC=[];
DEF=[];
MNO=[];
PQR=[];
    for i = nzmimax(1,1)+(tinggiangkat/2):tinggiangkat:nzmimax(1,2), % looping
sebanyak layer yang tercipta berdasarkan parameter
        counterangkat=counterangkat+1;       % Layer Thickness ---> tinggi angkat
         jumlahangkatan(counterangkat)= 0 ;   % counterangkat adalah counter
yang menghitung jumlah layer
```

```
    for j=1:jmlhsegitiga,              % nilai counterangkat yang terakhir adalah
jumlah layer
       for k = 1:3
          indexv= k;
          index1=lindexfacet(j,indexv+1);
          indexv = k+1;
          if indexv>3,
             indexv=indexv-3;        % setelah indexv mencapai 4
mengembalikkannya lagi ke 1
          end
          index2=lindexfacet(j,indexv+1);
          letakz_1=lindexvertex(index1,4);
          letakz_2=lindexvertex(index2,4);

           if (letakz_1<i && i<letakz_2)||(letakz_1>i && i>letakz_2)

            % catat index segitiga
            jumlahangkatan(counterangkat)=jumlahangkatan(counterangkat)+1;
% jumlah angkatan = jumlah layer (counting sampai dengan angkatan/
% layer terakhir yang tercipta)
            angkatan(counterangkat,jumlahangkatan(counterangkat))=j;        %
mendata index segitiga disetiap angkatan /layer yang berpotongan
% dengan z= c

            vekn(j,:,counterangkat)=[lindexfacet(j,1) lindexfacet(j,2)
lindexfacet(j,3) lindexfacet(j,4) lindexfacet(j,5) lindexfacet(j,6) lindexfacet(j,7)];

            % cari z tertinggi dan terendah
            if ztertinggi<i
               ztertinggi=i;
            end
            if zterendah>i,
               zterendah=i;    % sampai disini, berhasil mengidentifikasi z
terendah dan tertinggi
            end
            break;             % break untuk keluar dari looping k = 1 : 3
          end
       end
    end
end

facetyangdislicing=vekn;
```

Penelitian	Automatic Part Primitive Feature Identification Based on Faceted Models
Kegiatan	Pengelompokkan Data Hasil Slicing menjadi beberapa Grup suatu Fitur
Tahapan proses	Grouping Feature

```
% "PROSES GROUPING & SORTING"

 facetyangdislicing=vekn;
 pjgbrs=length(facetyangdislicing(:,1));
 pjglyr=length(facetyangdislicing(1,1,:));
 count=1;
 for ii = 1:pjglyr
    for jj = 1:pjgbrs
       ambilsemuafacet(count,:) =facetyangdislicing(jj,:,ii);   % menyimpan
"facetygdislicing" pada matriks "ambilsemuafacet"
       count=count+1;
    end
 end
matrixfacet=ambilsemuafacet;

for xj = 1:pjgbrs*pjglyr              % membuang facet yg sama
  data1=ambilsemuafacet(xj,:);
  for ij = 1:pjgbrs*pjglyr
     data2=ambilsemuafacet(ij,:);
     if data1 == data2
       facetygtdksama(ij,:)=data1;
       break
     end
  end
end

pjgbrsfct=length(facetygtdksama(:,1));
iy=1;
for sj=1:pjgbrsfct                  % membuang yang memiliki nilai 0 0 0 0
   facetpersatu=facetygtdksama(sj,:);
   if facetpersatu(1,1)~=0
      facetslicingOK(iy,:)=facetpersatu;
      iy=iy+1;
   end
end

groupmatrix=facetslicingOK;      % matriks "facetslicingOK" disubstitusikan ke
matriks "groupmatrix"

GroupMatrix=groupmatrix;
```

```
% "PROSES ADJACENT TRIANGLES"

aa=length(groupmatrix(:,1));

pjgbrs=length(groupmatrix(:,1));            % panjang baris matriks  "groupmatrix"
matrixneighbor=zeros(aa,7);
matrixadjacent=zeros(aa,7);

e=0;
f=0;
g=1;
h=0;
i=1;
j=0;

matrixx=[];
d=0;

while i <= pjgbrs
    ABC=groupmatrix(i,:);
    pjgmatrixneighbor=length(matrixneighbor);

 if g==1
    i=i+1;
    h=h+1;
    j=j+1;
    matrixneighbor(g,:,j)=ABC;
    groupmatrix(h,:)=[0 0 0 0 0 0 0];
    g=g+1;

 elseif g > 1
    k=0;
    DEF=matrixneighbor(g-1,:,j);

    if   (ABC(1,2)==DEF(1,2) & ABC(1,3)== DEF(1,3) & ABC(1,4)~= DEF(1,4)) |
(ABC(1,2)==DEF(1,2) & ABC(1,3)== DEF(1,4) & ABC(1,4)~= DEF(1,3))...
        |(ABC(1,2)==DEF(1,3) & ABC(1,3)== DEF(1,2) & ABC(1,4)~= DEF(1,4)) |
(ABC(1,2)==DEF(1,3) & ABC(1,3)== DEF(1,4) & ABC(1,4)~= DEF(1,2))...
        |(ABC(1,2)==DEF(1,4) & ABC(1,3)== DEF(1,2) & ABC(1,4)~= DEF(1,3)) |
(ABC(1,2)==DEF(1,4) & ABC(1,3)== DEF(1,3) & ABC(1,4)~= DEF(1,2))...
        |(ABC(1,3)==DEF(1,2) & ABC(1,4)== DEF(1,3) & ABC(1,2)~= DEF(1,4)) |
(ABC(1,3)==DEF(1,2) & ABC(1,4)== DEF(1,4) & ABC(1,2)~= DEF(1,3))...
        |(ABC(1,3)==DEF(1,3) & ABC(1,4)== DEF(1,2) & ABC(1,2)~= DEF(1,4)) |
(ABC(1,3)==DEF(1,3) & ABC(1,4)== DEF(1,4) & ABC(1,2)~= DEF(1,2))...
        |(ABC(1,3)==DEF(1,4) & ABC(1,4)== DEF(1,2) & ABC(1,2)~= DEF(1,3)) |
(ABC(1,3)==DEF(1,4) & ABC(1,4)== DEF(1,3) & ABC(1,2)~= DEF(1,2))...
        |(ABC(1,4)==DEF(1,4) & ABC(1,2)== DEF(1,3) & ABC(1,3)~= DEF(1,2)) |
(ABC(1,4)==DEF(1,4) & ABC(1,2)== DEF(1,2) & ABC(1,3)~= DEF(1,3))...
        |(ABC(1,4)==DEF(1,3) & ABC(1,2)== DEF(1,2) & ABC(1,3)~= DEF(1,4)) |
(ABC(1,4)==DEF(1,3) & ABC(1,2)== DEF(1,4) & ABC(1,3)~= DEF(1,2))...
        |(ABC(1,4)==DEF(1,2) & ABC(1,2)== DEF(1,3) & ABC(1,3)~= DEF(1,4)) |
(ABC(1,4)==DEF(1,2) & ABC(1,2)== DEF(1,4) & ABC(1,3)~= DEF(1,3))
```

```
    i=i+1;
    h=h+1;
    matrixneighbor(g,:,j)=ABC;
    e=e+1;
    groupmatrix(h,:)=[0 0 0 0 0 0 0];
    g=g+1;
    k=k+1;
    end

  if k==0
    h=h+1;
    i=i+1;
  end
end

  c=0;
  idu=length(groupmatrix(:,1));
  if h==pjgbrs
     count=0;
    b=length(matrixneighbor(:,1));
    for a = 1:b
       GHI=matrixneighbor(a,:,j);
       if GHI(1,1)~=0 & GHI(1,2)~=0 & GHI(1,3)~=0 & GHI(1,4)~=0
          c=c+1;
       end
    end
       d=d+1;
       JKL(d,:)=c;
         for acd=1:idu
            groupadjacent=groupmatrix(acd,:);
            if groupadjacent(1,1)~=0
               count=count+1;
            end
          end
          if count==0
            n=j;
            matrixadjacent(:,:,n)=matrixneighbor(:,:,j);
            break
          end
        if d==2
           if JKL(1,1)~=JKL(2,1)
             JKL(1,1)=JKL(2,1);
             JKL(2,1)=0;
             d=1;
             niox=2;
           else
              jjj=j;
              matrikneighbor=matrixneighbor(:,:,jjj);
              bos=1;
              boss=2;
              f=f+1;
```

```
            while bos~=boss
            aa=matrikneighbor;
            pjggrup=length(groupmatrix(:,1));
            pjgmtrx=length(matrixneighbor(:,1,:));
            aaa=0;
            pjggroup=length(groupmatrix(:,1));
            pjgmtrix=length(matrixneighbor(:,1,j));
            aaa=0;
            for bbb=1:pjgmtrix
               XXX=matrixneighbor(bbb,:,j);
               if XXX(1,1) ~=0 & XXX(1,2)~=0 & XXX(1,3)~=0 & XXX(1,4)~=0
                  aaa=aaa+1;
               end
            end
            for indxgrup=1:pjggrup
               gmats=groupmatrix(indxgrup,:);
               if gmats(1,1) ~=0 & gmats(1,2)~=0 & gmats(1,3)~=0 &
gmats(1,4)~=0
                  bos=bos+1;
               end
            end

            bos=bos-1;
            bro=bos;
            ccc=aaa;
            ddd=0;
            pjgaa=length(aa(:,1));
            abcd=0;
            for idxaa=1:pjgaa
               ACD=aa(idxaa,:);
               if ACD(1,1)~=0 & ACD(1,2)~=0 & ACD(1,3)~=0 & ACD(1,4)~=0
                  abcd=abcd+1;
               end
            end
            ccc=abcd;
            for eee = 1 : pjggrup
               MMM=groupmatrix(eee,:);
                if MMM(1,1)~=0 & MMM(1,2)~=0 & MMM(1,3)~=0 & MMM(1,4)~=0
                  AAA=MMM;
               for fff = 1 :pjgmtrx
                  BBB=aa(fff,:,:);
                  if (AAA(1,2)==BBB(1,2) | AAA(1,2)==BBB(1,3) |
AAA(1,2)==BBB(1,4) | AAA(1,3)==BBB(1,2)|
AAA(1,3)==BBB(1,3)|AAA(1,3)==BBB(1,4)|...
                     AAA(1,4)==BBB(1,2)| AAA(1,4)==BBB(1,3)|
AAA(1,4)==BBB(1,4))
                     ccc = ccc +1 ;
                     aa(ccc,:,:)=AAA;
                     groupmatrix(eee,:)=[0 0 0 0 0 0 0];
                     break
                  end
               end
            end
            end
```

```
            for indxgrup=1:pjggrup
                gmats=groupmatrix(indxgrup,:);
                if gmats(1,1) ~=0 & gmats(1,2)~=0 & gmats(1,3)~=0 &
gmats(1,4)~=0
                    boss=boss+1;
                end
            end
            boss=boss-2;
            bross=boss;

            if  bro==bross
                valueaa=aa;
                pjgaaa=length(valueaa(:,1));
                cont=0;
                niox=1;
                JKL=[];
                for indxaaa=1:pjgaaa
                    BDE=valueaa(indxaaa,:);
                    if BDE(1,1) ~=0 & BDE(1,2)~=0 & BDE(1,3)~=0 & BDE(1,4)~=0
                        cont=cont+1;
                        adjacent=BDE;
                        matrixadjacent(cont,:,f)=adjacent;
                    end
                end
                break
            end
            bos=1;
            boss=2;
            matrikneighbor=aa;
            end
        end
          count=0;
         for acd=1:idu
          groupadjacent=groupmatrix(acd,:);
          if groupadjacent(1,1)~=0
            count=count+1;
          end
        end
        if count==0
          break
        end
      else
         niox=0;
      end
      bc=0;
    for cd=1:pjgbrs
      MNO=groupmatrix(cd,:);
      if MNO(1,1)~=0 & MNO(1,2)~=0 & MNO(1,3)~=0 & MNO(1,4)~=0
      bc=bc+1;
      end
    end

    ef=length(groupmatrix(:,1));
    fg=0;
    matrixx=[];
```

```
      for gh = 1:ef
         PQR=groupmatrix(gh,:);
         if PQR(1,1)~=0 & PQR(1,2)~=0 & PQR(1,3)~=0 & PQR(1,4)~=0
            fg=fg+1;
            matrixx(fg,:)=PQR;
         end
      end

      if niox ==0 | niox ==2
      eee=2;
      groupmatrix=matrixx;
      i=1;
      h=0;
      pjgbrs=length(groupmatrix(:,1));

      else
      groupmatrix=matrixx;
      i=1;
      h=0;
      pjgbrs=length(groupmatrix(:,1));
      g=1;
      d=0;
      end
    end
end
valueadjacent=[];
for jmlj=1:j
valueadjacent=matrixadjacent(:,:,jmlj);
jmlsgt=jmlj;
end

if jmlsgt>1
normvek=nvplanerot(plane,:);
ki=0;
sisi=sisi+1;
arah=num2str(normvek);

AV='ARAH VEKTOR :          ';
 for jmlklm = 1:26
    ki=ki+1;
    data(1,ki)=AV(1,jmlklm);
 end
 ki=0;
 for kolom = 1:7
    ki=ki+1;
    data(2,ki)=arah(1,kolom);
 end
 sisi=sisi+1;
 kosakata(sisi,:)=data(1,:);
 sisi=sisi+1;
 kosakata(sisi,:)=data(2,:);

 n=0;
 m=0;
 groupneighbor=[];
```

```
 for n=1:j
    m=m+1;
    groupneighbor(:,:,m)=matrixadjacent(:,:,n);              % hasil grouping
adjacent triangles
 end

 GroupNeighbor=groupneighbor;

% "PROSES ELIMINASI PART BOUNDARY"

 pjgindxvertex=length(indexvertex(:,1));     % panjang matriks indexvertex
(nomor vertex)
 matrixindexv=zeros(pjgindxvertex+1,4);   % membuat zero matriks
  p=2;
  for q=1:pjgindxvertex
     matrixindexv(p,:)=lindexvertex(q,:);
     p=p+1;
  end

 pjgneighbor=length(groupneighbor(:,1));
 m=1;
 matrixkoordx=[];
  for t=1:j
     matrixneighbor=groupneighbor(:,:,t);
     for u= 1:pjgneighbor
     matrixkoordx(n,:,m)=matrixindexv((matrixneighbor(u,2))+1,2);
     n=n+1;
     end
     m=m+1;
  end

 o=1;
 urutkoordx=[];
 for v=1:j
 urutkoordx(:,:,o)=sortrows(matrixkoordx(:,1,v));
 pjgurutx=length(urutkoordx(:,1));
 layer(o,1)=v;
 koordxmax(o,:)=urutkoordx(pjgurutx,1,v);
 o=o+1;
 end
lyrdgkoordmax= [];
lyrdgkoordmax=[layer koordxmax];
pjglyr=length(lyrdgkoordmax(:,1));
urutkoordmax=sortrows(lyrdgkoordmax,2);
lyrbndry=urutkoordmax(pjglyr,1);
groupneighbor(:,:,lyrbndry)=[];
w=0;
groupfeature=[];
kk=groupfeature;

for x=1:jmlsgt-1
   w=w+1;
   groupfeature(:,:,w)=groupneighbor(:,:,x);
end

GroupFeature=groupfeature;
```

Penelitian	Automatic Part Primitive Feature Identification Based on Faceted Models
Kegiatan	Identifikasi Jenis Fitur pada Grup suatu Fitur Model Faset 3D
Tahapan proses	Aturan Identifikasi Fitur

```
% "PROSES FEATURE IDENTIFICATION"

b=0;
normalvectorgroup=[];
for cc=1:w
   grupfitur=groupfeature(:,:,cc);
   pjggrupfitur=length(grupfitur(:,1));
   ff=1;
   for gg=1:pjggrupfitur
      if abs(grupfitur(gg,7)) <= 10^-10       % filter komponen vektor normal
yang bernilai kecil sekali <= 10^-10 dianggap nol.
         grupfitur(gg,7)=0;
      end
      matrixnvfitur(ff,:)=[grupfitur(gg,5) grupfitur(gg,6) grupfitur(gg,7)];  %
mengambil vektor normal dari matriks "grupfitur"
      ff=ff+1;
   end
   b=b+1;
   normalvectorgroup(:,:,b)=matrixnvfitur;      % vektor normal dari matriks
"groupfeature" yang tidak berisi  matiks berisi nol
end

hh=0;
for ii=1:b
   vektornormalgrup=normalvectorgroup(:,:,ii);   % matriks "normalvectorgroup"
dimasukan matriks "vektornormalgrup"
   nvunique=unique(vektornormalgrup,'rows');    % memfilter matriks
"vektornormalgrup" agar tidak ada yang berulang dan memasukan nilainya ke
matriks "nvunique"
   nvunique(all(nvunique==0,2),:)=[];          % menghilangkan angka 0 matriks
"nvunique"
   pjgnvunique=length(nvunique(:,1));          % menghitung jumlah baris matriks
"nvunique"
   hh=hh+1;
   pjgnvgroup(hh,1)=pjgnvunique;               % jumlah baris matriks "nvunique"
dimasukan ke matriks "pjgnvgroup"
   pjgnvgroupurut=sortrows(pjgnvgroup);        % mengurutkan nilai pada baris
matriks "pjgnvgroup" dari nilai kecil ke besar
end
pjggrupnv=pjgnvgroupurut;
jmlnvgrup=length(pjggrupnv(:,1));
jmlnvfitur=pjggrupnv(jmlnvgrup,1);
matrixzeronv=[];
pjgnormvec=pjgnvgroupurut(b,1);
jj=0;
nvoke=[];
```

```
 for kk=1:x
   vektornormalgrup=normalvectorgroup(:,:,kk);
   pjgvngrup=length(vektornormalgrup(:,1));
   veknormpersatu(1,:)=vektornormalgrup(1,:);
   ll=1;
   pjgveknormpersatu=length(veknormpersatu(:,1));
   lli=pjgnormvec;
   matrixzeronv=zeros(jmlnvfitur,3);
   for ab=2:pjgvngrup
      datanv1=vektornormalgrup(ab,:);
      count=0;
      for nn = 1:pjgveknormpersatu
      datanv2=veknormpersatu(nn,:);
      if  abs(datanv1(1,1)-datanv2(1,1)) < 10^-5 & abs(datanv1(1,2)-datanv2(1,2))
< 10^-5 & abs(datanv1(1,3)-datanv2(1,3)) < 10^-5
         count=count+1;
      else
         p=1;
      end
   end
  if count==0
      ll=ll+1;
         veknormpersatu(ll,:)=datanv1;
         pjgveknormpersatu=pjgveknormpersatu+1;
   end
   end
   pjgveknormpersatu=length(veknormpersatu(:,1));
   oo=0;
   for pp=1:pjgveknormpersatu
      venormpersatu=veknormpersatu(pp,:);
      if venormpersatu(1,1)==0 & venormpersatu(1,2)==0 &
venormpersatu(1,3)==0
      xyz=1;
      else
      oo=oo+1;
      matrixzeronv(oo,:)=venormpersatu;
      end
   end
   lim=matrixzeronv;
   jj=jj+1;
   nvoke(:,:,jj)=matrixzeronv;
   matrixzeronv=[];
   veknormpersatu=[];
 end
qq=0;
rr=jj;
ss=0;
charnv=[];
nvsipa=[];
nvsp=[];
for tt=1:rr
   nvok=nvoke(:,:,tt);
   uu=0;
   qq=qq+1;
   vv=0;
   for ww=1:jmlnvfitur
      nvokpersatu=nvok(ww,:);
```

```
if nvokpersatu(1,1)~=0 |nvokpersatu(1,2) ~=0 | nvokpersatu(1,3)~=0
   uu=uu+1;
   nvsp(uu,:)=nvokpersatu;
else
  break
end
end
nvsipa=nvsp;
jmlnv=length(nvsp(:,1));
pjgnvsipa=length(nvsipa(:,1));
xyz=0;
mtrxbedasdt=[];
count=0;
for xx = 1:jmlnv
      nvspsatu=nvsp(xx,:);
      if xx==pjgnvsipa;
        angle= acos(nvspsatu(1,1)*nvsp(1,1)+ nvspsatu(1,2)*nvsp(1,2) +
nvspsatu(1,3)*nvsp(1,3)) *180 /pi;
      vv=vv+1;
      allangle(vv,:)=angle;
      pjgallangle=length (allangle(:,1));
        de=0;
        for zz=1:pjgallangle
          anglepersatu=allangle(zz,:);
          if anglepersatu ==90
            de=de+1;
          end
          if de==4 & jmlnv ==4
           ss=ss+1;
             abc=1;
           charnv(ss,:)=abc;
           break
          end
          if zz == pjgallangle;
           hhh=length(mtrxbedasdt(:,1));
           for fff = 1 : hhh
             if fff == hhh
                ggg = abs(mtrxbedasdt(hhh,1)-mtrxbedasdt(1,1));
                if ggg > 0.1
                  count=count+1;
                end
              else
                ggg = abs(mtrxbedasdt(fff,1)-mtrxbedasdt(fff+1,1));
                if ggg > 0.1
                   count=count+1;
                end
              end
            end
            counter=count;
            if sudut ~= 90 & bedasudut <= 0.1 & counter == 0;
               ss=ss+1;
               abc=2;
               charnv(ss,:)=abc;
            else
             ss=ss+1;
```

```
                    abc=3;
                    charnv(ss,:)=abc;
                  end
                 break
               end
               sudut=allangle(zz+1,:);
               bedasudut=abs(sudut-anglepersatu);
               xyz=xyz+1;
               mtrxbedasdt(xyz,:)=bedasudut;
             end
          break
          end
          nvspsatudua=nvsp(xx+1,:);
          angle= acos(nvspsatu(1,1)*nvspsatudua(1,1)+
nvspsatu(1,2)*nvspsatudua(1,2) + nvspsatu(1,3)*nvspsatudua(1,3)) *180 /pi;
          vv=vv+1;
          allangle(vv,:)=angle;
          end
    nvsp=[];
    allangle=[];
end
pjgfiturOK=length(charnv(:,1));
for idxnv=1:pjgfiturOK
    NVgrup=charnv(idxnv,:);
    if NVgrup == 1
       sisi=sisi+1;
       PM='PROSES : POCKET MILLING   ';
       FM='FEATURE : POCKET          ';

       ki=0;
       for jmlklm= 1:26
          ki=ki+1;
          data(3,ki)=FM(1,jmlklm);
       end
       ki=0;
       for kolom=1:26
          ki=ki+1;
          data(4,ki)=PM(1,kolom);
       end
       kosakata(sisi,:)=data(3,:);
       sisi=sisi+1;
       kosakata(sisi,:)=data(4,:);

    elseif NVgrup == 2
       sisi=sisi+1;
       PM='PROSES : DRILLING         ';
       FM='FEATURE : CYLINDRICAL     ';

       ki=0;
       for jmlklm= 1:26
          ki=ki+1;
          data(3,ki)=FM(1,jmlklm);
       end
       ki=0;
       for kolom=1:26
          ki=ki+1;
          data(4,ki)=PM(1,kolom);
       end
```

```
        kosakata(sisi,:)=data(3,:);
        sisi=sisi+1;
        kosakata(sisi,:)=data(4,:);
    else
        sisi=sisi+1;
        PM='PROSES : POCKET MILLING   ';
        FM='FEATURE : PROFILE         ';
        ki=0;
        for jmlklm= 1:26
            ki=ki+1;
            data(3,ki)=FM(1,jmlklm);
        end
        ki=0;
        for kolom=1:26
            ki=ki+1;
            data(4,ki)=PM(1,kolom);
        end
        kosakata(sisi,:)=data(3,:);
        sisi=sisi+1;
        kosakata(sisi,:)=data(4,:);
    end
end
else
    ki=0;
end
end
HASIL=kosakata
```