PENGEMBANGAN METODE PENDETEKSI RUANG TERBATAS (*Bounded Volume*) UNTUK PEMESINAN MILLING AWAL (*Roughing*) MULTI-AXIS BERBASIS MODEL FASET 3D

TESIS

Himawan Hadi Sutrisno

06 06 00 296 3

TESIS INI DIAJUKAN UNTUK MELENGKAPI SEBAGIAN PERSYARATAN MENJADI MAGISTER TEKNIK

PROGRAM STUDI MANUFAKTUR

DEPARTEMEN TEKNIK MESIN

FAKULTAS TEKNIK UNIVERSITAS INDONESIA

GENAP 2007/2008

PERNYATAAN KEASLIAN THESIS

Saya menyatakan dengan sesungguhnya bahwa Thesis dengan judul :

PENGEMBANGAN METODE PENDETEKSI RUANG TERBATAS (*Bounded Volume*) UNTUK PEMESINAN MILLING AWAL (*Roughing*) MULTI-AXIS BERBASIS MODEL FASET 3D

Yang dibuat untuk melengkapi persyaratan menjadi Magister Teknik pada Program Studi Teknik Mesin Program Pascasarjana Universitas Indonesia, sejauh yang saya ketahui bukan merupakan tiruan atau duplikasi dari thesis yang pernah di publikasikan dan atau pernah dipakai untuk mendapatkan gelar mata kuliah thesis atau mendapatkan gelar Magister di lingkungan Universitas Indonesia maupun Perguruan Tinggi atau Instansi manapun, kecuali bagian yang dikutip dan sumber informasi/refrensinya dicantumkan sebagaimana mestinya.

Depok, 14 Juli 2008

HIMAWAN HADI SUTRISNO
NPM : 06 06 00 2963

PENGESAHAN

Thesis dengan judul :

PENGEMBANGAN METODE PENDETEKSI RUANG TERBATAS (*Bounded Volume*) UNTUK PEMESINAN MILLING AWAL (*Roughing*) MULTI-AXIS BERBASIS MODEL FASET 3D

dibuat untuk melengkapi persyaratan kurikulum Program Magister Teknik Bidang Ilmu Teknik Universitas Indonesia guna memperoleh gelar Magister Teknik pada Program Pascasarjana Program Studi Teknik Mesin.

Thesis ini telah disetujui untuk diajukan dalam sidang ujian Thesis.

Depok, 15 Juli 2008

Menyetujui Dosen Pembimbing,

Dr. Gandjar Kiswanto, M.Eng.

NIP. 132 137 846

KATA PENGANTAR

Segala puji dan syukur penulis panjatkan ke hadirat Allah SWT, karena atas rahmat, pertolongan, dan karunia-Nya lah penulis mampu menyelesaikan pelaksanaan dan laporan seminar ini. Pada kesempatan ini, penulis mengucapkan terima kasih kepada semua pihak yang telah memberikan doa, bimbingan, dukungan dan membantu terlaksananya penelitian ini.

Ucapan terima kasih yang sebesar-besarnya penulis sampaikan kepada :

1. Kedua Orang tua dan keluaga yang telah memberikan dukungan dalam penyelesaian kuliah ini.
2. DR.Ir. Gandjar Kiswanto, M.Eng selaku dosen pembimbing yang banyak memberikan pengarahan dan bimbingan dalam penelitian ini.
3. Dosen Penguji diantaranya : Bp. Hengky S.N, M.Eng., Bp. Bambang purwo Priyanto, Mkomp, Bp. Zos istiyanto, MT, dan semua Staf karyawan dan karyawati Universitas Indonesia
4. Seluruh staf pengajar dan karyawan dan karyawati Departemen Teknik Mesin yang telah banyak membantu dalam proses jalannya penelitian ini
5. Rekan satu tim dalam penelitian : Eko Arif S, Agung Premono, A Kholil dan Herwindo P yang telah memberikan banyak masukan dan saran terhadap penelitian dan penulisan ini.
6. P Edi Kresnha yang telah banyak memberikan dasar-dasar ilmu dan arahan dalam penelitian
7. Ratna Setyaningtyas yang telah banyak memberi dukungan dan semangat untuk pencapaian terselesainya penelitian.

Penulis juga ingin mengucapkan maaf yang sebesar-besarnya bila dalam proses penelitian dan penulisan dalam makalah ini banyak kesalahan . Besar harapan penulisan dari kebesaran hati kawan-kawan untuk dapat melanjutkan kembali penelitian ini sampai tuntas.

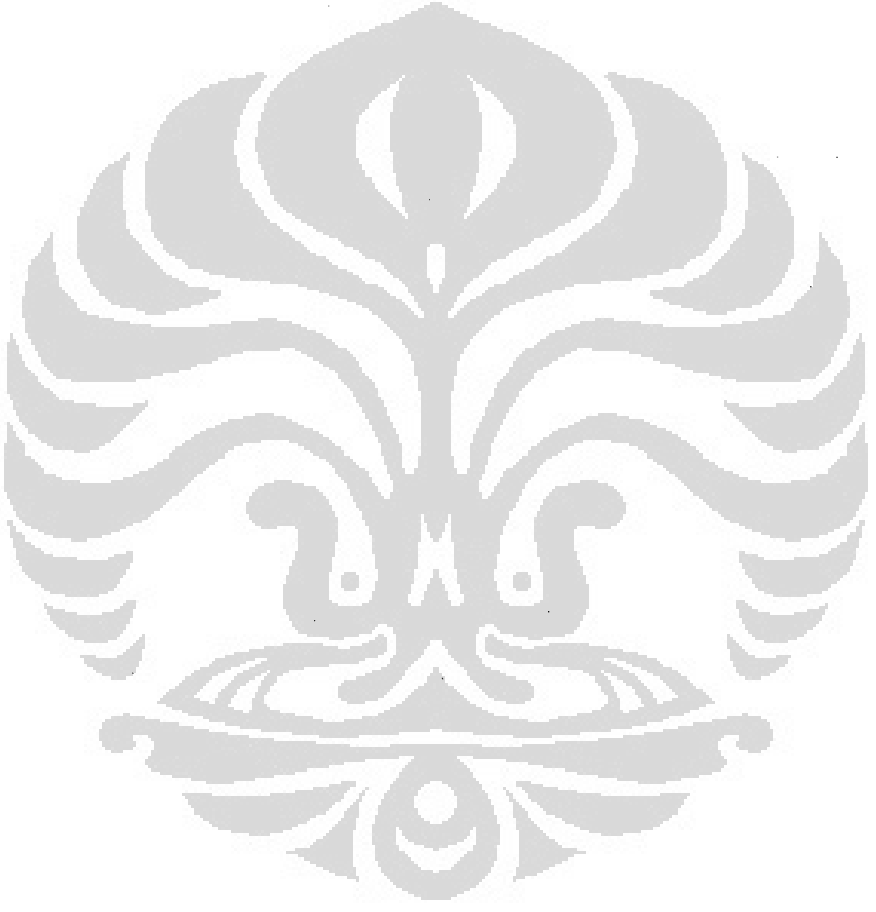
Jakarta, 14 Juli 2008

Penulis,

Himawan Hadi Sutrisno

DAFTAR ISI

DAFTAR GAMBAR



DAFTAR TABEL

ABSTRAK

Penelitian mengenai pendeteksian terhadap ruang terbatas proses pemesinan awal 5 Axis, merupakan salah satu bagian dalam pengembangan sistem CAM yang berbasis model faset 3D. Ruang terbatas dibedakan menjadi 2 bagian yaitu ruang terbatas terbuka dan ruang terbatas tertutup. Perbedaan ruang terbatas terbuka dan ruang terbatas tertutup yaitu perlunya orientasi pahat untuk proses pengerjaan ruang terbatas tertutup.

Dalam penelitian ini pendeteksian ruang terbatas dilakukan dengan mendeteksi kebertumpukan segitiga dalam bidang x dan y. untuk mendapatkan hasil deteksi pahat pada ruang terbatas, penulis mengembangkan algoritma-algoritma dalam mengolah data yang berupa *STL* file. Metode yang telah dikembangkan ini kemudian di uji terhadap beberapa model produk dan mampu memberikan informasi mengenai adanya ruang terbatas dan areanya.

BAB I

PENDAHULUAN

I.1 Latar Belakang

Dalam perkembangan teknologi yang semakin canggih, manusia dituntut dapat memenuhi segala kebutuhan dengan cepat, tepat dan akurat. Hal ini diindikasikan dengan digantikanya pemakaian mesin-mesin dalam menghasilkan suatu produk. Proses manufaktur yang handal harus mampu menghasilkan barang dengan cepat dan memiliki tingkat keakurasian yang tinggi.

Dalam proses manufaktur, sistem CAM (*Computer-Aided Manufaktur*) memberi kontribusi yang sangat besar dalam menciptakan sebuah produk. Teknologi CAM saat ini sedang banyak dikembangkan. Pengembangan sistem ini mempunyai tujuan akhir terhadap penghematan waktu yang digunakan dalam proses pemesinan, kualitas produk serta keakurasian produk yang dihasilkan.

Model benda kerja dihasilkan oleh sistem CAD (*Computer-Aided Design*). Untuk dapat diaplikasikan menjadi sebuah perintah dalam mesin CNC (*Computer Numerical Control*), memerlukan metoda penghubung. Metode penghubung ini mengadopsi data berupa G-code dan dirubah menjadi gerakan pemahatan. dalam proses pemesinan. Mesin CNC ini beroperasi berdasarkan G-code sehingga mampu menghasilkan barang yang bersifat cepat serta memiliki ketelitian yang tinggi. Hal ini berarti menuntut pula kehandalan system CAM sebagai proses penghubung antara CAD dan mesin CNC.

Dalam proses pemesinan, lintasan pahat memiliki bentuk dan arah optimum berdasarkan bentuk permukaan model *(feature)*. Hal ini dapat berbentuk cembung (*convex*), cekung (*concave*), atau gabungan cembung dan cekung *(sadle)*. Proses pemesinan benda kerja, baik untuk pemesinan awal *(roughing)* maupun pemesinan akhir *(finishing)* dilakukan berdasarkan optimasi orientasi pahat. Namun pada kenyataanya, dalam proses pemesinan sering terjadi ketidak-sesuaian dengan yang diharapkan atau juga terjadi kecacatan produk. Hal ini dapat disebabkan karena

gouging, *collision* ataupun juga disebabkan karena kurang tepatnya letak orientasi pahat terhadap bidang yang mengalami proses pemesinan.

Kekurang-tepatan orientasi pahat ini dapat menyebabkan pengulangan proses pemesinan sehingga waktu yang diperlukan jadi bertambah. Sedangkan harapan dalam proses pemesinan awal, waktu yang digunakan harus sesingkat mungkin serta dapat membuang material sebanyak-banyaknya hingga mendekati bentuk yang hampir sesuai dengan yang dikehendaki.

Untuk peningkatan efektifitas proses pemesinan awal pada pemesinan milling 5 axis, maka pada penelitian ini dikembangkan algoritma yang dapat mendeteksi daerah ruang terbatas untuk pemesinan awal mesin milling 5 axis.

I.2 Perumusan Masalah

Model yang diambil dari sebuah sistem CAD memuat titik koordinat 3 dimensi, ternyata masih belum cukup untuk bisa langsung diolah oleh sistem CAM. Untuk mendapatkan pendeteksian pada ruang terbatas maka penulis mengembangkan algoritma yang mengolah data dari CAD model. Data yang diolah berformat *STL (stereolithography)*, File *STL* ini menyimpan informasi objek *surface* dalam bentuk model faset 3D yang tersusun dari satu atau lebih segitiga.

Salah satu bagian penting dalam sistem CAM adalah pembuatan lintasan pahat (*pahat path*) pada proses pemesinan. Lintasan pahat ini akan menjadi petunjuk bagi mesin milling untuk menggerakkan pahat pada proses pemesinan sebuah material. Setalah pendeteksian pada daerah terbatas dapat terdeteksi maka titik-titik yang terdeteksi menjadi sebuah acuan untuk proses pemesinan.

I.3 Tujuan Penelitian

Tujuan penelitian ini adalah mengembangkan metode pendeteksi ruang terbatas sebagai bagian dari pengembangan sistem CAM *(Computer Aided Manufacturing)* multiaxis berbasis model faset 3D. Dengan mengetahui daerah ruang terbatas, orientasi dan kebebasan gerak pahat untuk pemesinan awal dapat di ketahui.

I.4 Pembatasan Masalah

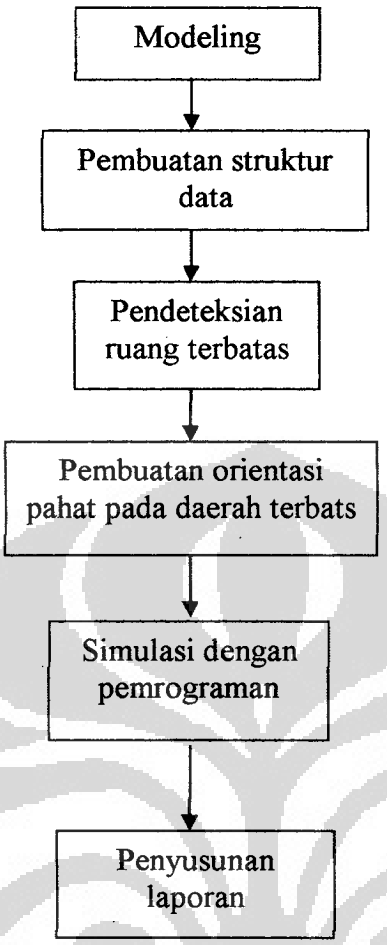
Pada penelitian kali ini penulis hanya mengembangkan teori-teori yang ada guna mendapat hasil deteksi ruang terbatas. Hal ini diambil dari data *STL* file model yang didapat dari dengan penggunaan *software* CAD, kemudian penulis akan mensimulasikan dengan pemrograman. Dalam proses pendeteksian serta penentuan daerah ruang terbatas penulis menggunakan algoritma algoritma berdasarkan persamaan garis serta nilai vector normal yang terkandung pada struktur data

I.5 Metodologi Penelitian

Dalam menentukan orientasi pahat pada daerah terbatas untuk pemesinan awal, penulis melakukan langkah-langkah sebagi berikut:

1. Penerjemahan struktur data dari CAD sistem dengan model faset 3D
2. Pembuatan struktur data dari *STL* file.
3. Pendeteksian adanya ruang terbatas.
4. Perhitungan tinggi hasil deteksi segitiga.
5. Pendeteksian orientasi pahat.
6. Simulasi dengan pemrograman

Secara umum hal ini dapat digambarkan dengan prosedur dibawah ini :



I.6 Sistematika Penulisan

Sistematika penulisan penelitian ini terdiri dari 5 bab, dengan garis besar penjelasan masing-masing bab adalah sebagai berikut,

BAB I PENDAHULUAN

Bab ini berisi penjelasan mengenai latar belakang permasalahan, tujuan penelitian, pembatasan masalah, metodologi dan sistematika penulisan laporan.

BAB II PROSES PEMESINAN

Bab ini mendeskripsikan teori, konsep, dan notasi yang menjadi landasan dalam penelitian.

BAB III ANALISIS DATA

Bab ini berisi penjelasan mengenai proses penelitian. Diawali dengan pembuatan benda dengan software uni graphic untuk menghasilkan

data *STL*. Termasuk di dalamnya penjelasan mengenai algoritma dalam penyusunan struktur data.

BAB IV PENGEMBANGAN ALGORITMA PENENTUAN ORIENTASI PAHAT PADA RUANG TERBATAS

Bab ini berisi langkah-langkah yang dilalui dalam mengimplementasikan algoritma untuk pendeteksian ruang terbatas terbuka, dan ruang terbatas tertutup, beserta formula-formula dan persamaan-persamaan yang mendukung implementasi algoritma yang digunakan dalam penelitian.

BAB V KESIMPULAN DAN SARAN PENELITIAN LEBIH LANJUT

Bab ini berisi kesimpulan dari penelitian yang telah dilakukan serta saran-saran terhadap perbaikan yang mungkin dilakukan untuk penelitian selanjutnya dan saran-saran terhadap implementasi metode dalam sistem CAM yang sedang dikembangkan.

BAB II

PROSES PEMESINAN

II.1 Pemesinan Milling

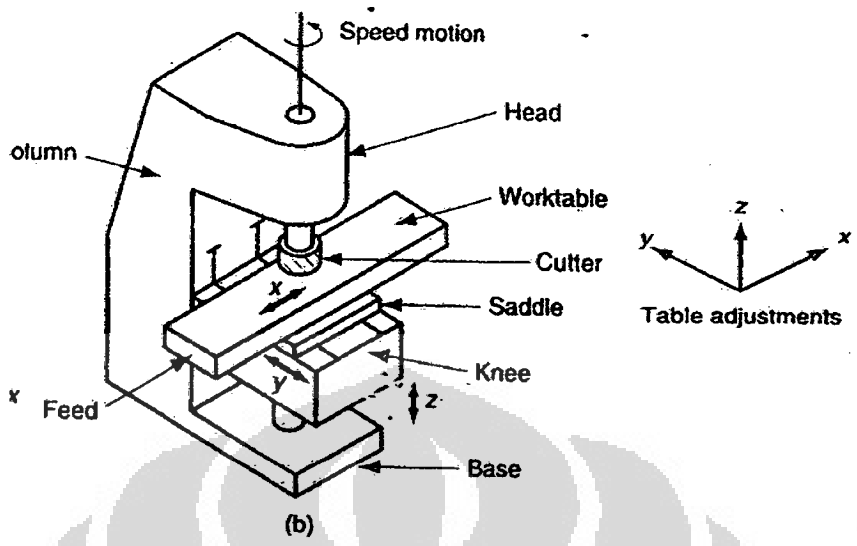
Proses manufaktur adalah pembuatan suatu bahan mentah (*raw material*) menjadi produk jadi. Umumnya manufaktur ditujukan untuk membuat peralatan, seperti peralatan industri, peralatan berat, pembuatan barang-barang elektronik dan mesin, dan sebagainya. Salah satu contoh proses manufaktur adalah perancangan dan pembuatan *mould sparepart* mobil. Pembuatan *moulding* ini biasa dilakukan dengan menggunakan mesin CNC (*Computer Numerically Controlled*). Menurut jenisnya pemesinan milling dibagi menjadi 2 kelompok besar. Yaitu pemesinan milling 3 axis dan pemesinan milling 5 axis atau biasa disebut dengan pemesinan multi axis.

II.1.1 Pemesinan Milling 3 Axis

Pemesinan milling di decade terakir banyak mengalami perkembangan. Pada awalnya pemesinan milling terdiri dari 3 derajat kebebasan [9] Secara sistematis pemesinan milling jenis ini berorientasi terhadap perubahan koordinat sumbu X, sumbu Y serta sumbu Z. Pemesinan ini biasa disebut dengan pemesinan 3 axis.

Gambar II.1: Mesin Miliing 3 Axis. [8]

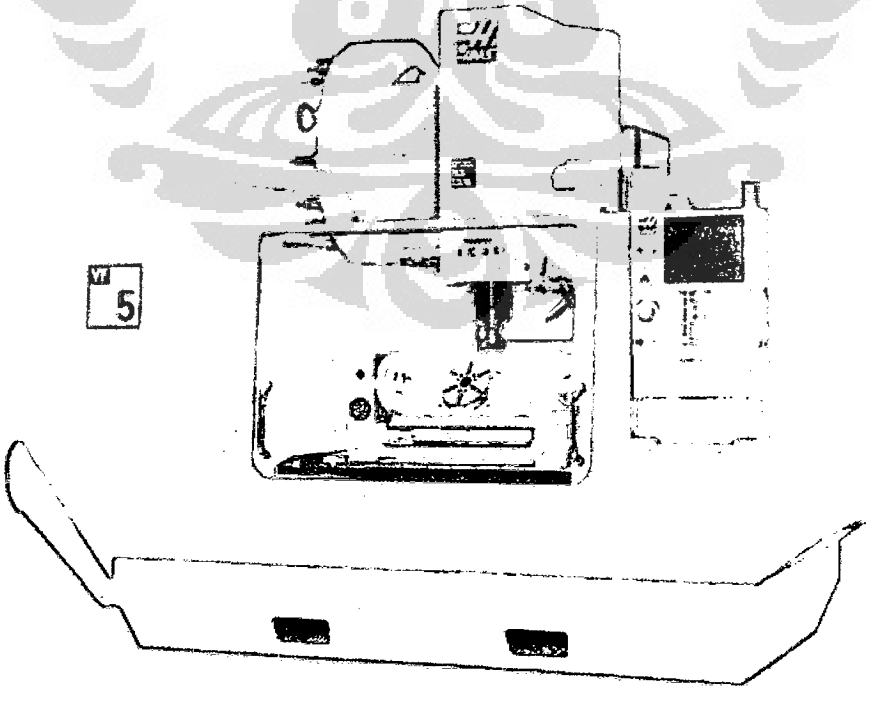
Secara konvensional proses pemesinan 3 Axis dapat di jelaskan dengan gambar berikut:



Gambar II.2: Pergerakan Sumbu Mesin Milling 3 Axis [9]

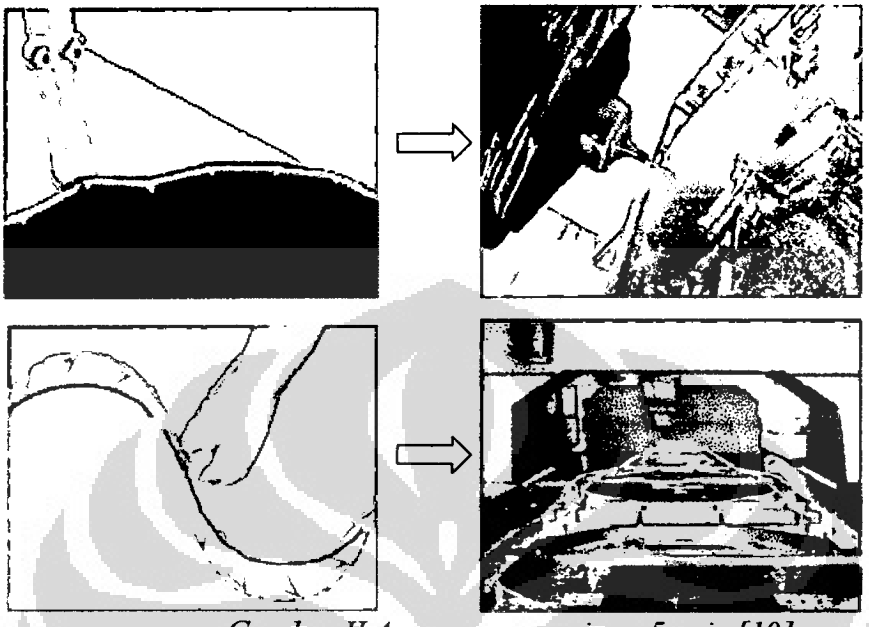
II.1.2 Pemesinan Milling 5 Axis

Proses pemesinan milling 5 axis pada dasarnya hampir sama dengan proses pemesinan 3 axis, yaitu memiliki 3 derajat kebebasan namun di tambah dengan 2 rotasi kebebasan dari meja dudukan benda kerja atau spindle [9].



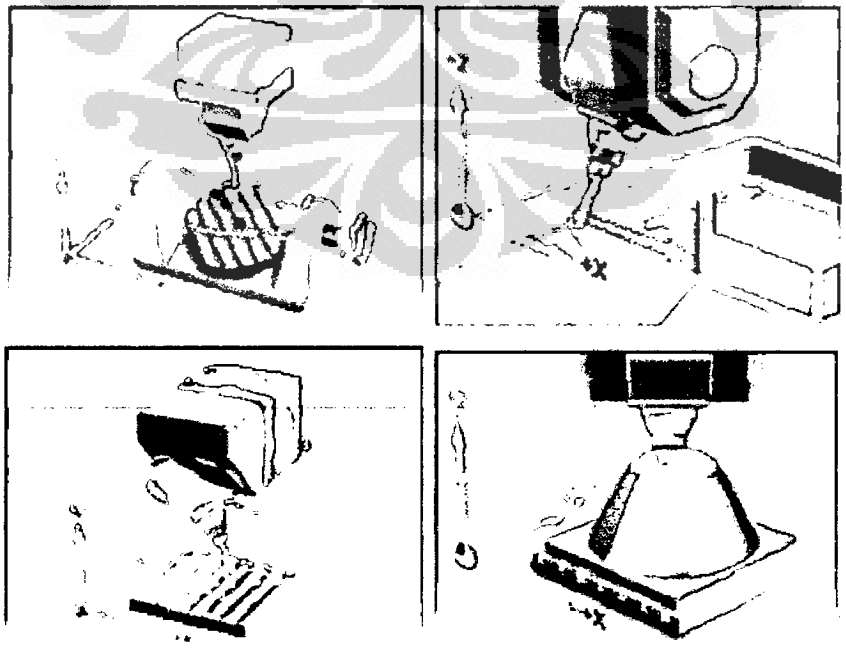
Gambar II.3: Mesin Milling 5 Axis[9]

Proses pemesinan yang menggunakan 5 derajat kebebasan memiliki banyak keuntungan. Diantaranya adalah memungkinkanya benda di atur dengan berbagai posisi dan juga dapat diletakan dalam berbagai sudut.



Gambar II.4: proses pemesinan 5 axis [10]

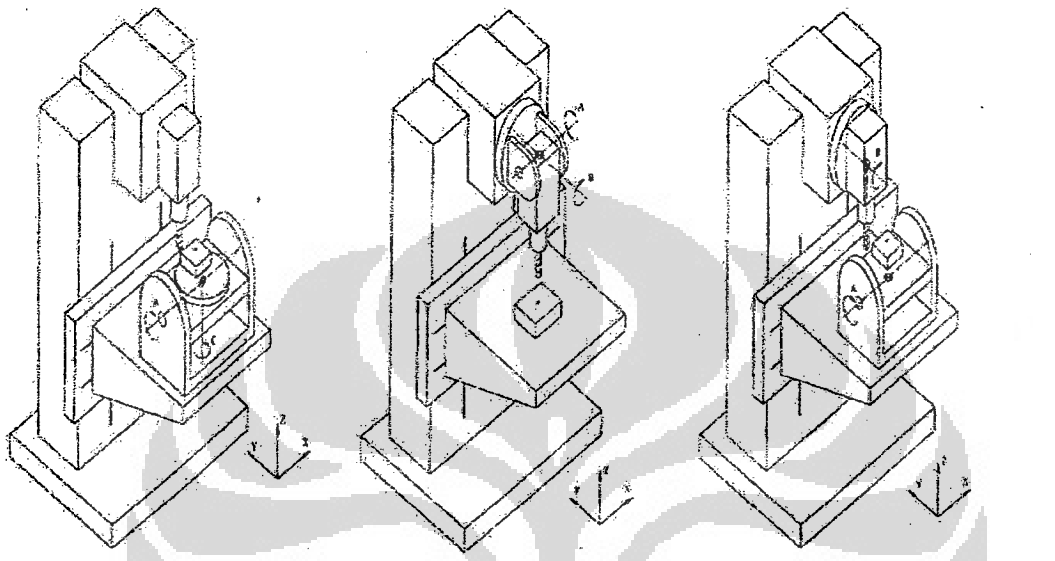
Keuntungan yang lain dari penggunaan proses pemesinan milling 5 axis adalah memudahkan operator dalam melakukan proses pemahatan tanpa harus merubah posisi pahat maupun merubah posisi pencekaman dari benda kerja. Hal ini dapat di jelaskan pada gambar II.5:



Gambar II.5: proses pemesinan 5 axis [10]

Selain memiliki banyak keunggulan dalam proses pemesinan, proses pemesinan ini juga memiliki kekurangan diantaranya adalah system control yang lebih rumit, serta proses kalibrasi juga rumit dibandingkan dengan mesin milling 3 axis.

Jenis jenis proses pemesinan milling 5 Axis dapat digambarkan sebagai berikut:



a) Table-tilting type **b) Spindle-tilting type** **c) Table-Spindle-tilting type**

Gambar II.6: Jenis Mesin Milling 5 Axis [9]

II.2 Proses Pemesinan Benda Kerja

II.2.1 Proses Pemesinan Awal *(Roughing)*

Proses pemesinan awal (*roughing)* adalah proses pemotongan material oleh pahat (*cutting pahats*) yang bertujuan untuk menghilangkan material yang tidak diperlukan dengan secepat mungkin hingga didapat bentuk yang mendekati bentuk akhir. Lintasan pahat untuk proses ini dibentuk dari serangkaian titik yang disebut *roughing points*, yang didapat melalui perpotongan antara bidang yang tegak lurus sumbu z, atau sejajar dengan bidang xy, dengan model faset[4]

II.2.2 Proses Pemesinan Akhir *(Finishing)*

Proses pemesinan akhir (*finishing)* adalah proses pemotongan material yang bertujuan untuk mendapatkan bentuk akhir produk sesuai dengan akurasi yang telah ditentukan dan dilakukan setelah proses ***roughing***. Dalam proses *finishing* ini, hal

yang dilakukan adalah pembentukan *cc-point* yang merupakan lintasan pahat dengan alur tertentu[4]

Pada proses pemesinan akhir, pembentukan cc-point dilakukan dengan mencari perpotongan antara garis yang merupakan proyeksi dari lintasan pahat yang akan dibuat dengan edge atau vertex segitiga.

II.3 Pemodelan Faset 3D

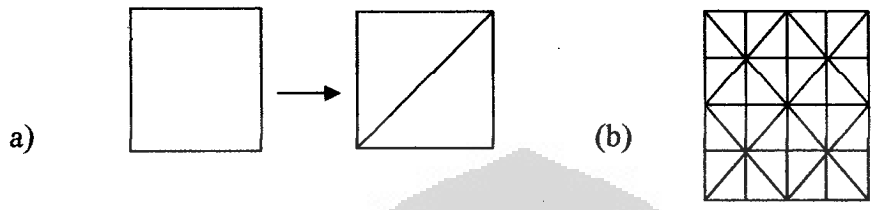
Model faset 3D adalah salah satu representasi bentuk atau model pada bidang 3 dimensi. Representasi lain yang dapat digunakan untuk model 3d adalah *parametric surface* dan *solid model,* dimana hal ini digambarkan oleh persamaan bidang sehingga mendapat tampilan model pada ruang 3 dimensi. Model faset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

Model faset 3D dapat dibuat menggunakan CAD (*computer aided design*), ataupun dilakukan *scanning* terhadap objek material nyata. Model yang dibuat dari sistem CAD, dilakukan dengan merancang sebuah bentuk benda, dan hasilnya di-*export* menjadi sebuah file yang berekstensi *STL*. Hasil dari proses *exporting* adalah sebuah pendekatan diskritisasi dari objek tersebut. Proses *exporting* 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 tiga dimensi. Hal ini direpresentasikan dengan segitiga-segitiga yang menghubungkan titik-titik tersebut. Proses pembentukan segitiga disebut triangulasi[2]

Selanjutnya model faset 3D disimpan 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 posisi-posisi verteks segitiga pada bidang 3D sebagai propertinya. *STL* merupakan kependekan dari *stereolithography*.

II.4 Struktur Data Model Faset 3D

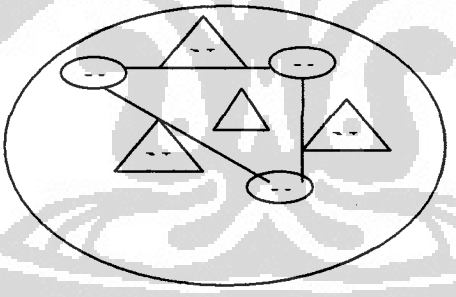
Model-Faset 3D merupakan model berbasis triangulasi segitiga[3]. Triangular mesh merupakan salah satu proses pemodelan berbasis elemen hingga dimana sebuah model akan digenerasi menjadi elemen kecil-kecil yang berbentuk segitiga[8]. Adapun visualisainya dapat dilihat dalam gambar ini:



GambarII. 7: Model Triangular Mesh 2D

Dimana apabila terdapat bidang persegi yang datar maka dalam triangulasi segitiga akan direpresentasikan oleh 2 bidang seitiga yang berhimpit.

Data struktur yang dihasilkan oleh model benda kerja pada sistem CAD memiliki dua data utama, *list index vertex* dan *list index segitiga*. Proses pembuatan list ini didasarkan pada urutan segitiga pada file *STL* dari model faset 3D. Adapun gambaran dari model struktur data adalah sebagai gambar berikut.

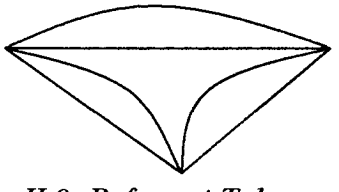


Gambar II 8: Model Struktur Data

Dari model facet tersebut di harapkan keakuarasian dapat di jaga[3]. Perhitungan keakurasian di dapat berdasar:

1. mencari jarak terjauh dari permukaan model solid parametric terhadap segitiga
2. jarak permukaan solid parametric ke sisi segitiga

hal ini dapat di gambarkan sebagai:



Gambar II.9: Referensi Toleransi Triangulasi[3]

Data yang diperoleh dari sistem CAD hanya berupa *STL* file yang kemudian akan diolah guna menjadi perintah pada sistem CAM. Format ini adalah yang paling umum digunakan karena lebih mudah dibaca dan dimengerti, serta dapat dibuka di text editor. Gambar berikut adalah contoh isi dari file berekstensi *STL*.

```
SOLID
  FACET NORMAL +2.39026E-01 -3.06303E-01 +9.21436E-01
    OUTER LOOP
      VERTEX   +8.29472E+01 +9.10050E+01 -7.60275E+00
      VERTEX   +8.33587E+01 +9.01260E+01 -8.00171E+00
      VERTEX   +8.41082E+01 +8.94832E+01 -8.40981E+00
    ENDLOOP
  ENDFACET
  FACET NORMAL +2.23838E-01 -3.03267E-01 +9.26242E-01
    OUTER LOOP
      VERTEX   +8.16371E+01 +9.02924E+01 -7.51124E+00
      VERTEX   +8.12968E+01 +9.00546E+01 -7.50687E+00
      VERTEX   +8.20742E+01 +8.93402E+01 -7.92865E+00
    ENDLOOP
  ENDFACET
  FACET NORMAL +2.26582E-01 -3.01909E-01 +9.26018E-01
    OUTER LOOP
      VERTEX   +8.16371E+01 +9.02924E+01 -7.51124E+00
      VERTEX   +8.20742E+01 +8.93402E+01 -7.92865E+00
      VERTEX   +8.28502E+01 +8.86244E+01 -8.35188E+00
    ENDLOOP
  ENDFACET
ENDSOLID
```

Gambar II.10: Contoh Isi File STL

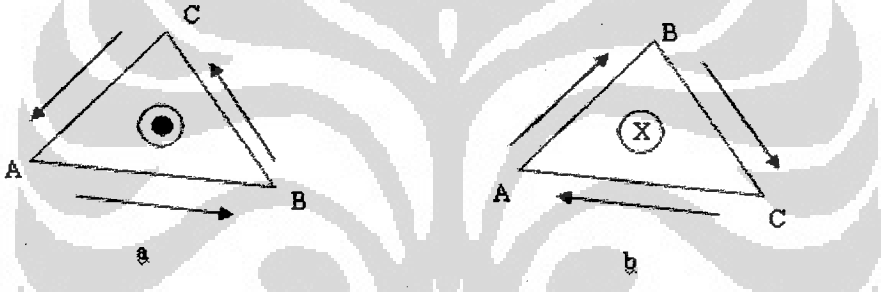
Dapat dilihat bahwa file berformat *STL* menyimpan objek surface 3D dalam bentuk segitiga yang tersusun tiga buah verteks yang berada pada lokasi tertentu dalam sistem koordinat *3D*. Berikut adalah penjelasan mengenai isi dari file *STL* di atas,

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 verteks, serta vektor normal dari segitiga tersebut.

3. Kata *OUTER LOOP* menandakan dimulainya loop dari koordinat verteks-verteks yang membangun segitiga hingga bertemu dengan kata *ENDLOOP*.
4. kata *VERTEX* merupakan verteks penyusun sebuah segitiga yang sebelumnya telah didefinisikan dengan *OUTER LOOP*. Informasi yang berada setelah kata *VERTEX* adalah posisi verteks pada sistem koordinat 3D.

II.5 Vektor Normal Pada Segitiga

Arah vektor normal segitiga akan sesuai dengan urutan pengambilan verteks 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:



Gambar II.11: Arah Vektor Normal Pada Segitiga

Arah vektor normal bergantung pada urutan verteks membentuk sebuah segitiga. Pada Gambar diatas, urutan pembentukan verteks berlawanan arah jarum jam (*counter clockwise*), maka perkalian silang (*cross product*) antara vektor-vektor yang menghubungkan verteks tersebut akan menghasilkan sebuah vektor normal yang arahnya keluar dari bidang, sedangkan pada Gambar II.11b, urutan pembentukan verteks searah jarum jam (*clockwise*), maka perkalian silang (*cross product*) antara vektor-vektor yang menghubungkan verteks akan menghasilkan sebuah vektor normal yang arahnya masuk ke bidang.

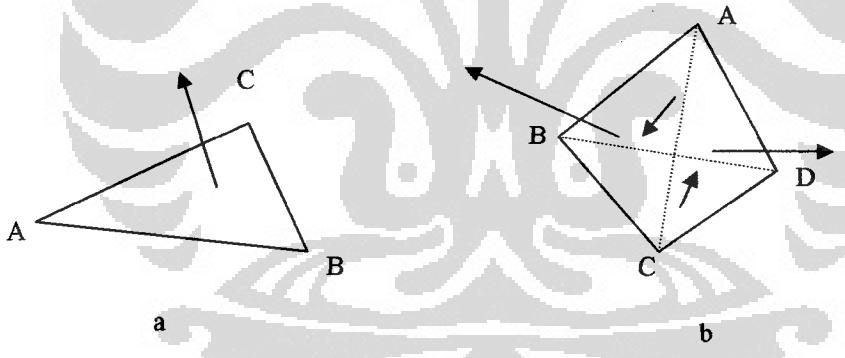
II.6 Arah Vektor Normal

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 tiga 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 ketidakkonsistenan 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.

Sebagai contoh, misalkan pada sebuah ruang 3 dimensi terdapat 4 buah titik yang akan disusun menjadi sebuah bidang datar segiempat beraturan. Letak keempat titik tersebut secara berurutan adalah A (0,4,4), B (-4,0,0), C (0,-4,4), dan D (4,0,0). Maka paling sedikit akan didapat 4 buah vektor normal bergantung pada pemilihan vertex yang akan dijadikan dasar untuk penentuan vektor sisi segiempat yang bersangkutan. Jika vertex-vertex yang terpilih adalah vertex A, B, dan C, maka vektor normal yang dihasilkan adalah (-0.71,0 0.71), kemudian jika vertex-vertex yang terpilih adalah B, C, dan D, maka vektor normal yang dihasilkan adalah (0.0, 0.71, 0.71), dan seterusnya.

Hal ini tidak terjadi jika dua buah segitiga digunakan sebagai penyusun keempat vertex tersebut. Untuk lebih jelasnya, dapat dilihat pada gambar dibawah:



Gambar II.12: Arah Vektor Normal

Arah vektor normal pada sebuah segitiga dalam ruang 3 dimensi selalu konsisten. Vektor normal pada sebuah bidang yang bukan segitiga (memiliki lebih dari empat vertex), dapat berjumlah lebih dari satu bergantung pada titik-titik yang dipilih sebagai acuan dalam membuat vektor normal.

II.7 Persamaan Garis

Persamaan garis yang digunakan dalam perhitungan vektor [2]:

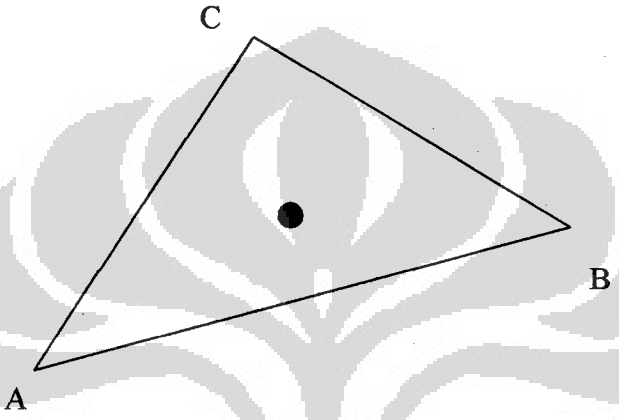
$$fx + gy + h = 0 \qquad \text{(II,1)}$$

dimana:

1. F, g , h merupakan konstanta perhitungan
2. X , y merupakan titik koordinat yang akan dilakukan perhitungan.

Persamaan garis di atas dapat di jadikan acuan pendeteksian perpotongan garis terhadap bidang segitiga.

1.



GambarII.13: Pengecekan Titik Berada Di Dalam Atau Di Luar Segitiga

Titik perpotongan berada di luar atau didalam dapat direpresentasikan dengan nilai :

$$fx + gy + h > 0 \qquad \text{(II, 2)}$$

dimana:

$$f = -(y_2 - y_1), g = (x_2 - x_1) \text{ dan } h = (x_1 y_2 - x_2 y_1) \qquad \text{(II, 3)}$$

Pengecekan dilakukan dalam sistem koordinat x, y dan dilakukan pada garis AB, BC, dan CA secara berurutan. Persamaan garis di atas menghubungkan dua titik sehingga apbila menghasilkan suatu nilai maka dapat menentukan dimana letak titik yang dimaksud.

II.8 Normalisasi Sumbu

Normalisasi sumbu adalah penyesuaian letak model faset dalam ruang 3D sehingga nilai minimum dan maximum koordinat x, y, dan z di tempatkan pada titik yang kita inginkan. Agar normalisasi sumbu ini dapat dilakukan, maka nilai-nilai koordinat 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[7].

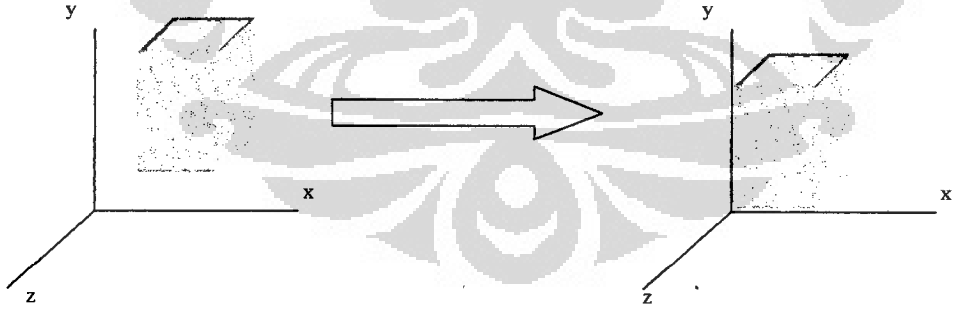
Normalisasi sumbu dilakukan dengan memindahkan nilai nilai x, y, z, sehingga salah satu sudutnya berada pada titik pusat (0,0,0) seperti pada Gambar II.14. Hal Ini memiliki arti bahwa vertex-vertex penyusun model faset dipindahkan sehingga nilai dari x minimum, y minimum, dan z maximum berada pada titik (0,0,0). Pemindahan ini dilakukan dengan melakukan perhitungan berikut:

$$\begin{aligned} jarak\_perpindahan\_sumbu\_x(dx) &= x\min - x0 \\ jarak\_perpindahan\_sumbu\_y(dy) &= y\min - y0 \\ jarak\_perpindahan\_sumbu\_z(dz) &= z\max - z0 \end{aligned} \qquad \text{(II, 4)}$$

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

$$\begin{aligned} x' &= x - dx \\ y' &= y - dy \\ z' &= z - dz \end{aligned} \qquad \text{(II, 5)}$$

Kegunaan normalisasi sumbu ini antara lain mempermudah pencarian vertex segitiga, hal ini dapat dijelaskan dengan gambar dibawah ini:



Gambar II.14: Normalisasi Sumbu Pada Sebuah Model Faset

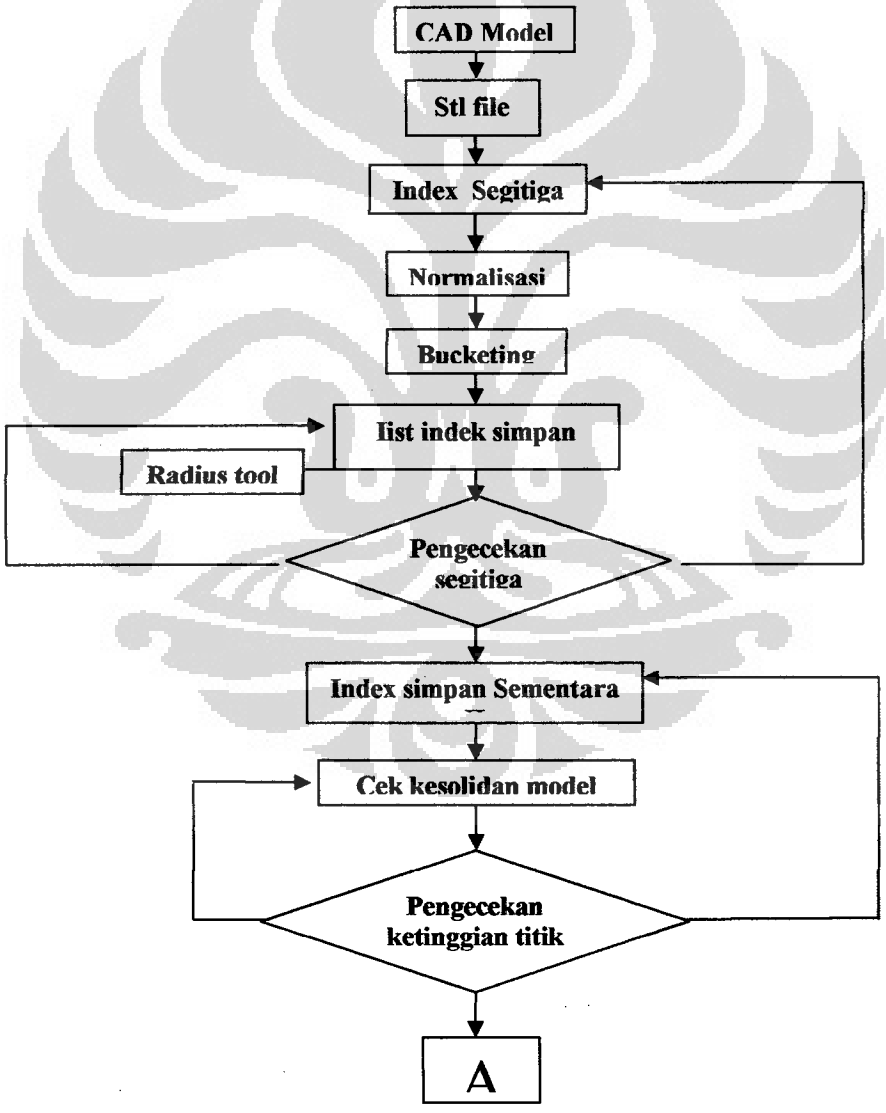
BAB III

PENANGANAN MODEL FASET

III.1 Pembentukan Model Dan Fasetisasi 3D

Pembuatan metode penghubung antara Computer Aided Design (CAD) dan Computer Aided Manufaktur merupkan suatu kesulitan tersendiri. Hal ini disebabkan adanya hubungan yang terintregrasi mulai dari bentuk benda yang di buat, efisiensi metode penghubung dari CAD terhadap CAM serta optimasi proses pemesinan.

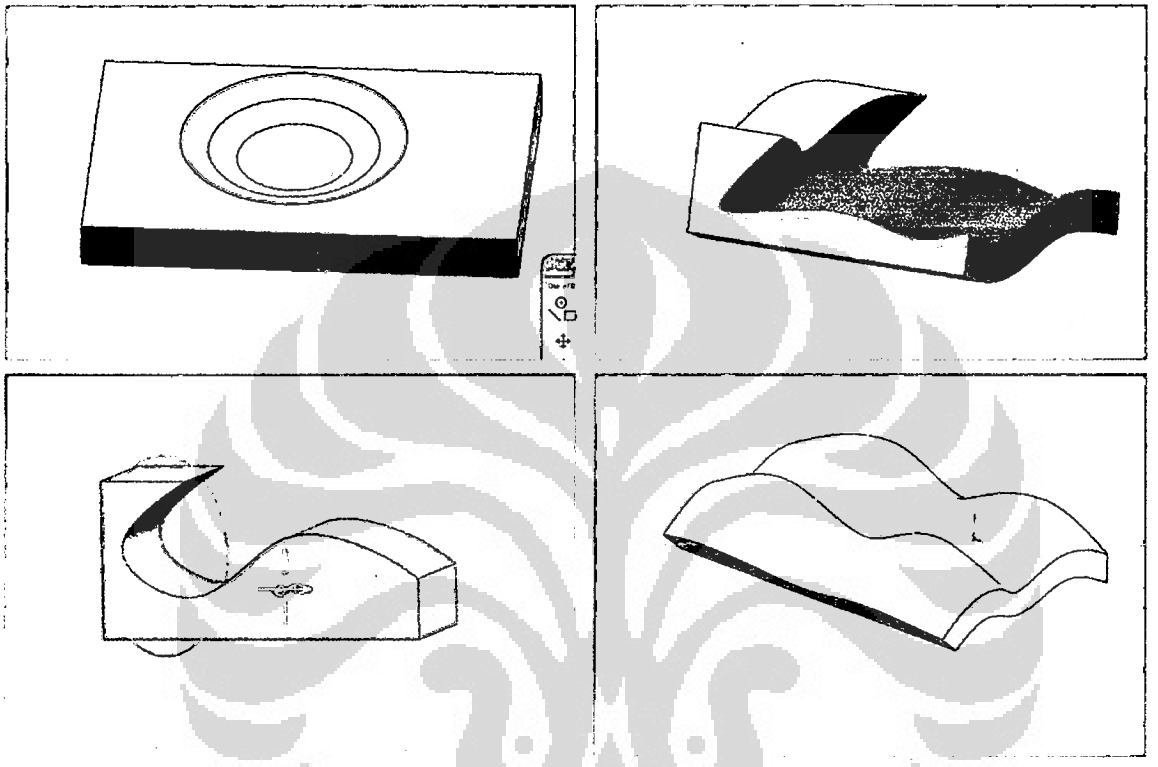
Flow cart yang digunakan dalam pembentukan model dan fasetisasi:



Pendeteksian terhadap bentuk model benda kerja menjadi elemen yang sangat penting dalam proses ini. Sebab feature benda akan memberi konsep informasi

terhadap karakteristik komponen serta metode pengerjaanya. Dan untuk desainer sendiri feature dari benda dapat merepresentasikan fungsi, rencana assembly serta merepresentasikan bagian dari benda lain terhadap satu kesatuan benda.

Desain dan bentuk benda kerja disusun dalam CAD sistem. Penulis menggunakan software Uni graphic dalam mendapatkan model benda. Bentuk benda kerja yang dianalisa adalah bentuk benda yang memiliki daerah boundary volume.:



Gambar III.1: Model 3D

Gambar diatas merupakan bentuk benda yang akan dilakukan proses pemesinan. Dalam sistem CAM yang sedang dikembangkan, informasi yang dipakai pengembangan sistem ini adalah berformat *STL*. File *STL* dapat disimpan dalam dua buah vektor, yaitu vektor segitiga dan vektor vertex. Setiap objek segitiga menyimpan informasi berupa nilai vektor normal segitiga tersebut, dan index-index vertex penyusunnya.

Di bawah ini adalah contoh file *STL* yang diberikan oleh sistem CAD dari model benda kerja diatas:

```
solid
  facet normal +0.0000000E+00 -3.9469062E-01 +9.1881407E-01
    outer loop
      vertex   +0.0000000E+00 -1.4660504E+01 +1.8926470E+01
      vertex   +2.5000000E+01 -1.7170671E+01 +1.7848189E+01
      vertex   +2.5000000E+01 -1.4660504E+01 +1.8926470E+01
    endloop
  endfacet
  facet normal +0.0000000E+00 -3.9469062E-01 +9.1881407E-01
    outer loop
      vertex   +0.0000000E+00 -1.4660504E+01 +1.8926470E+01
      vertex   +0.0000000E+00 -1.7170671E+01 +1.7848189E+01
      vertex   +2.5000000E+01 -1.7170671E+01 +1.7848189E+01
    endloop
  endfacet
  facet normal +0.0000000E+00 -4.0751357E-01 +9.1319914E-01
    outer loop
      vertex   +0.0000000E+00 -1.7170671E+01 +1.7848189E+01
      vertex   +2.5000000E+01 -1.9494341E+01 +1.6811256E+01
      vertex   +2.5000000E+01 -1.7170671E+01 +1.7848189E+01
    endloop
  endfacet
  facet normal +0.0000000E+00 -4.0751357E-01 +9.1319914E-01
    outer loop
      vertex   +0.0000000E+00 -1.7170671E+01 +1.7848189E+01
      vertex   +0.0000000E+00 -1.9494341E+01 +1.6811256E+01
      vertex   +2.5000000E+01 -1.9494341E+01 +1.6811256E+01
    endloop
  endfacet
  facet normal +0.0000000E+00 -4.2486434E-01 +9.0525702E-01
    outer loop
      vertex   +0.0000000E+00 -1.9494341E+01 +1.6811256E+01
      vertex   +2.5000000E+01 -2.1623094E+01 +1.5812168E+01
      vertex   +2.5000000E+01 -1.9494341E+01 +1.6811256E+01
    endloop
  endfacet
  facet normal +0.0000000E+00 -4.2486434E-01 +9.0525702E-01
    outer loop
      vertex   +0.0000000E+00 -1.9494341E+01 +1.6811256E+01
      vertex   +0.0000000E+00 -2.1623094E+01 +1.5812168E+01
      vertex   +2.5000000E+01 -2.1623094E+01 +1.5812168E+01
    endloop
  endfacet
  facet normal +0.0000000E+00 -4.4796913E-01 +8.9404902E-01
    outer loop
      vertex   +0.0000000E+00 -2.1623094E+01 +1.5812168E+01
      vertex   +2.5000000E+01 -2.3548513E+01 +1.4847424E+01
      vertex   +2.5000000E+01 -2.1623094E+01 +1.5812168E+01
    endloop
  endfacet
  facet normal +0.0000000E+00 -4.4796913E-01 +8.9404902E-01
    outer loop
      vertex   +0.0000000E+00 -2.1623094E+01 +1.5812168E+01
```

Gambar III.2: Data Vertex Segitiga Dari Representasi Model Benda Kerja

III.2 Penyusunan Struktur Data

Dari model yang telah dibuat, data akan dikelompokan dan diberikan index supaya mempermudah dalam proses pendeteksian. Pemberian index ini dilakukan terhadap segitiga maupun terhadap vertex yang terkandung dalam setiap segitiga.

Pemberian index ini bertujuan untuk menghindari adanya hal penyimpanan vertex segitiga yang sama dengan lebih dari satu kali. Berikut adalah tabel yang menggambarkan struktur data dalam penyimpanan objek vertex pada setiap segitiga,

Tabel III.1 Struktur Tabel Index Segitiga

Index segitiga	Index vertex 1	Index vertex 2	Index vertex 3
1	1	2	3
2	3	2	5
3	5	3	6
4	5	6	7
5	7	6	8
...	...	...	...

Tabel III.2 Struktur Tabel Index Vertex

Index vertex	Koordinat x	Koordinat y	Koordinat z
1	0.0000000E+00	-1.4660504E+01	+1.8926470E+01
2	0.0000000E+00	-1.7170671E+01	+1.8926470E+01
3	0.0000000E+00	-1.4660504E+01	-1.29780E+00
4	2.5000000E+01	-1.4660504E+01	-1.85569E+01
5	2.5000000E+01	-1.7170671E+01	-1.87777E+01
...	...	...	...

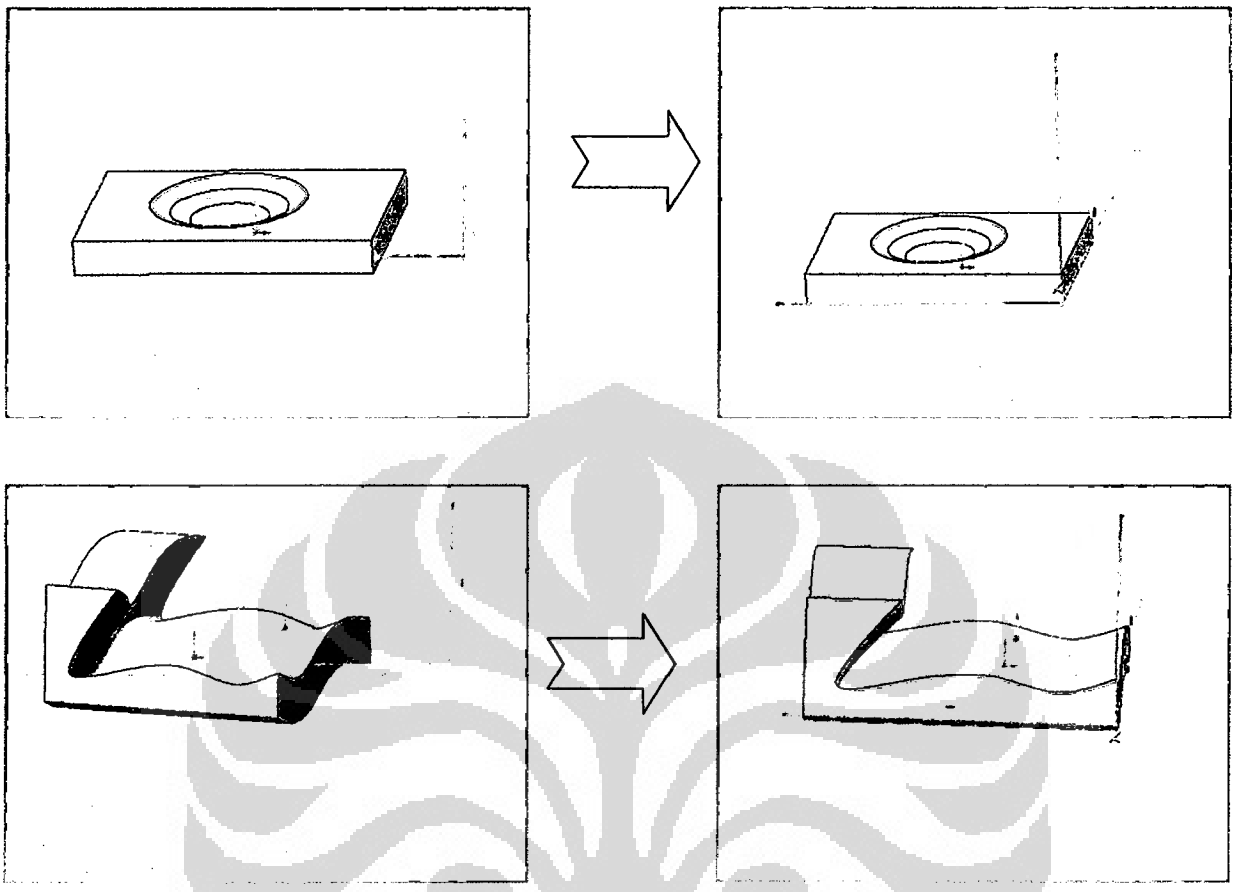
III.3 Normalisasi Sumbu

Setelah data yang di dapat diberikan nilai index, maka lagkah berikutnya yang dilakukan adalah normalisasi sumbu.

Normalisasi dilakukan dengan memindahkan nilai nilai x, y ,z dari nilai nilai terendah menuju ketitik 0,0,0. Data yang di proses dalam normalisasi sumbu dapat dilakukan secara cepat karena index segitga serta index vertex telah tersedia diatas. Proses menempatkan nilai vertex pada nilai 0,0,0 dilakukan dengan :

1. Cek seluruh segitiga dalam *STL* file
2. Menentukan nilai terendah dari sumbu x, y
3. Menentukan nilai z tertinggi
4. Pindahkan hasil yang terdeteksi ke sumbu koordinat yang diinginkan
5. Simpan hasil yang baru untuk proses berikutnya

Hal ini dapat dijelaskan dalam gambar dibawah ini.



Gambar III.3: Normalisai Sumbu Ke Titik 0, 0,0

Dalam proses pergeseran nilai minimal baik untuk nilai x maupun nilai y, maka nilai nilai x minimum dilambangkan dengan x min, nilai y minimum dilambangkan dengan ymin, dan z maximum dilambangkan dengan z max, Selanjutnya jarak antara titik pusat dengan ketiga koordinat tersebut masing-masing dilambangkan:

$$dx = xmin - x0$$
$$dy = ymin - y0 \qquad \text{(III. 1)}$$
$$dz = zmin - z0$$

Setelah jarak dari nilai yang ada terhadap sumbu 0,0,0 di dapatkan, maka pada setiap verteks penyusun model faset dilakukan penyesuaian nilai x, y, dan z dengan index vertex yang tidak berubah, Pada akhirnya mendapat nilai baru dengan rumusan sebagai berikut,

$$x' = x - dx$$
$$y' = y - dy \qquad \text{(III, 2)}$$
$$z' = z - dz$$

Dalam array yang menyimpan nilai normal maka nilai nilai yang terkandung adalah:

Tabel III.3 Struktur Tabel Index Vertex Setelah Normalisasi

Index vertex	Koordinat x'	Koordinat y'	Koordinat z'
1	0.00E+00 – *dx*	-1.466E+01 - *dy*	+1.892E+01- *dz*
2	0.00E+00 – *dx*	-1.717E+01- *dy*	+1.892E+01- *dz*
3	0.00E+00- *dx*	-1.466E+01- *dy*	-1.297E+00- *dz*
4	2.500E+01- *dx*	-1.466E+01- *dy*	-1.855E+01- *dz*
5	2.500E+01- *dx*	-1.717E+01- *dy*	-1.877E+01- *dz*
...	...	...	...

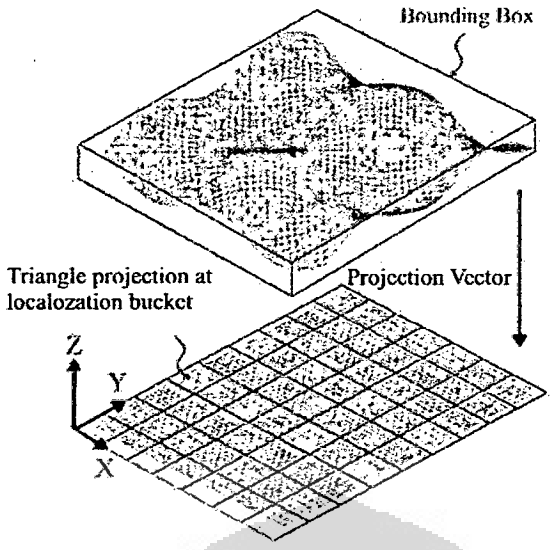
III.4 Bucketing

Pengembangan fungsi *bucketing* adalah untuk melokalisasi segitiga-segitiga berdasarkan letaknya, Dalam hal ini pendeteksian dan pencarian segitiga menjadi terlokalisir. Sebelum proses *bucket*ing dilakukan, model faset harus diproyeksikan terlebih dahulu dalam bidang xy. Dengan cara menghilangkan atau mengabaikan sumbu z. Proyeksi model ini akan dibagi-bagi menjadi serangkaian segi empat, dimana setiap segi empat menyimpan informasi segitiga-segitiga yang berpotongan pada wilayah proyeksi tersebut Gambar III.4.

Ukuran lebar tiap *bucket* dapat ditentukan sendiri, semakin besar ukuran *bucket*, semakin banyak segitiga yang dicakupnya. Namun jika ukuran yang terlalu besar dapat mengurangi kecepatan pencarian segitiga sebab makin banyak segitiga yang harus diperiksa pada satu wilayah tertentu.

Bucket merupakan array dua dimensi dari vektor. Index *bucket* pada titik tertentu dapat dicari menggunakan rumus

$$i = x / lebar\_bucket$$
$$j = y / tinggi\_bucket \qquad \text{(III, 3)}$$



Gambar III.4: Metode Bucketing[6]

Dimana i dan j adalah index *bucket* yang berada pada proyeksi model. Dari lebar dan tinggi ini, jumlah *bucket* yang memanjang terhadap sumbu x dan sumbu y dapat ditentukan juga dengan rumus:

$$n\_x = x\_\max / lebar\_bucket$$
$$n\_y = y\_\max / panjang\_bucket \qquad \text{(III,4)}$$

Dimana n_x dan n_y adalah jumlah *bucket* melebar dan memanjang pada sumbu x dan sumbu y, kemudian x_max dan y_max merupakan koordinat x maximum dan koordinat y maximum.

Proses pembentukan dan pencarian *bucket* akan lebih mudah jika sebelumnya telah dilakukan normalisasi sumbu. Hal ini disebabkan karena normalisasi sumbu menempatkan titik x minimum dan y minimum pada titik pusat (0,0,0). Tanpa adanya normalisasi sumbu, maka penghitungan jumlah *bucket* dan index *bucket* untuk sebuah titik pada posisi tertentu harus disesuaikan dengan nilai x minimum dan maximum, serta nilai y minimum dan maximum. Dan untuk perhitunganya menjadi

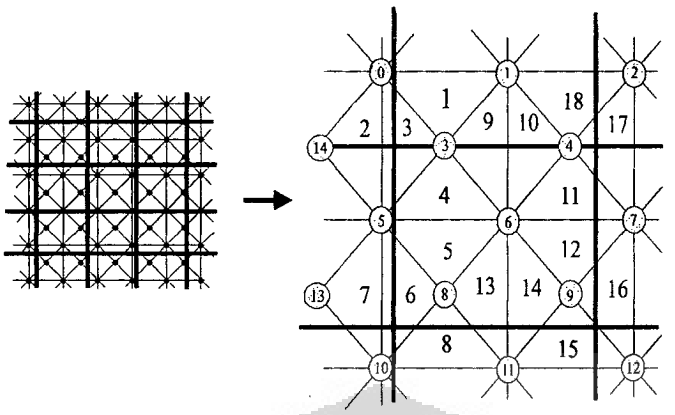
$$i = (x - x\_\min) / lebar\_bucket$$
$$j = (y - y\_\min) / tinggi\_bucket \qquad \text{(III, 5)}$$

dan rumusan tersebut di presentasikan menjadi:

$$n\_x = (x\_\max - x\_\min) / lebar\_bucket$$
$$n\_y = (y\_\max - y\_\min) / panjang\_bucket \qquad \text{(III, 6)}$$

Dengan adanya fungsi *bucket* ini, proses pencarian / identifikasi segitiga pada wilayah tertentu menjadi lebih cepat, sebab tidak perlu dilakukan pengecekan pada seluruh

wilayah model faset, cukup dengan memeriksa *bucket* yang berada di wilayah yang bersangkutan.



Gambar III.4: Segitiga Dalam Bucket

Setiap index *bucket* menyimpan informasi berupa index-index segitiga yang masuk ke dalam wilayah proyeksi *bucket* tersebut.

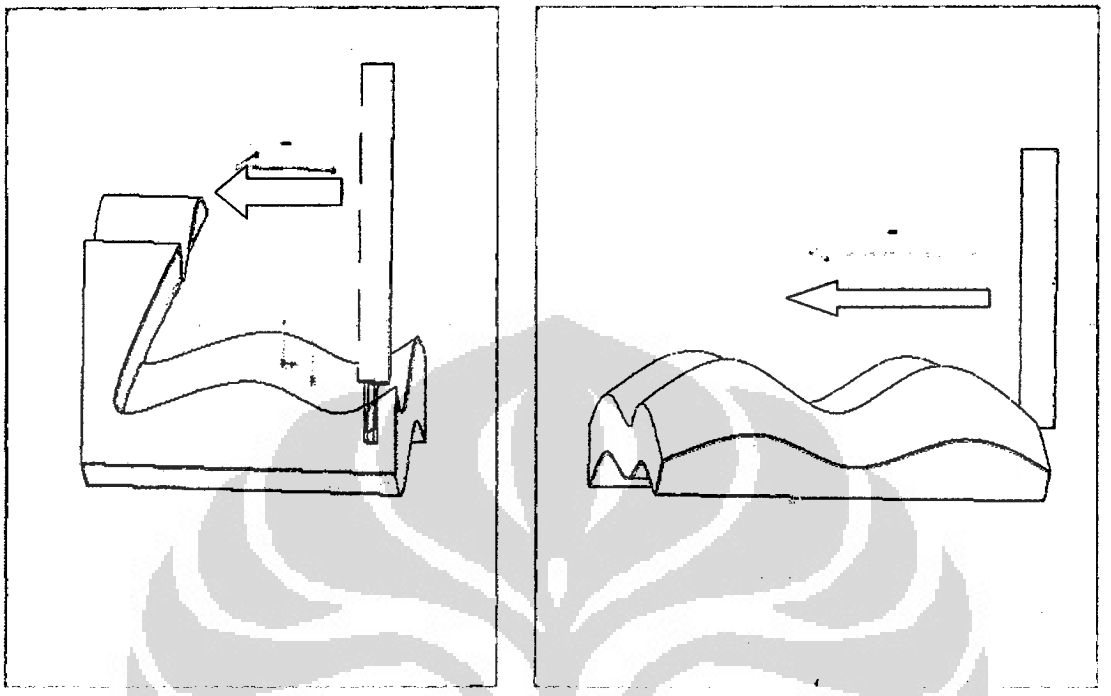
Tabel III.4: Index Bucket

Index Bucket	Segitiga-Segitiga yang terkandung
(1,1)	1,3,4,6,8
(2,1)	2,15,17,19,21
(3,1)	34,39,43,47,56
(4,1)	78,97,86,56,87
(5,1)	79,82.85,96,98
.....	.....

III.5 Penentuan Lintasan Pahat

Lintasan pahat dan nilai step over pada proses pemesinan sangat berperan dalam pembentukan benda kerja serta waktu yang diperlukan selama proses pemesinan. Semakin banyak lintasan pahat serta nilai step over yang kecil dalam proses pemesian maka akan menghasilkan benda yang halus. Namun hal ini membutuhkan waktu yang panjang. Untuk proses pemesinan awal *(roughing)* pada umumnya menggunakan lintasan pahat yang terpendek agar waktu yang digunakan dalam proses ini semakin cepat.

Dalam penelitian ini, lintasan pahat berdasar radius pahat. Dimana pusat sumbu pahat akan menjadi dasar acuan dalam perhitungan,



Gambar III.5: Arah Lintasan Pahat

Berdasar nilai radius pahat yang ada dan bidang telah dinormalisasi pada sumbu 0, 0, 0 maka pahat akan bergeser sesuai arah feature dengan besaran sebesar radius pahat yang telah ditentukan. Pergeseran akan dilakukan searah sumbu x dan selanjutnya akan bergeser ke sumbu y.

BAB IV
PENGEMBANGAN ALGORITMA PENDETEKSIAN RUANG TERBATAS

IV.1 Pengidentifikasian Ruang Terbatas

Dalam proses pengidetifikasian ruang terbatas, lintasan pahat ini dibentuk dengan alur tertentu berdasarkan sumbu pahat. Besaran nilai radius pahat dapat di tentukan oleh operator. Pada prosesnya pahat yang digunakan akan bergerak searah lintasan sumbu x dan akan begeser sesuai dengan lintasan sumbu y.

Ruang Terbatas dapat didefinisikan dengan adanya keterbatasan gerak pahat pada model yang terdeteksi. Pahat tidak dapat penetrasi sehingga proses pemesinan pada daerah tersebut tidak bebas.

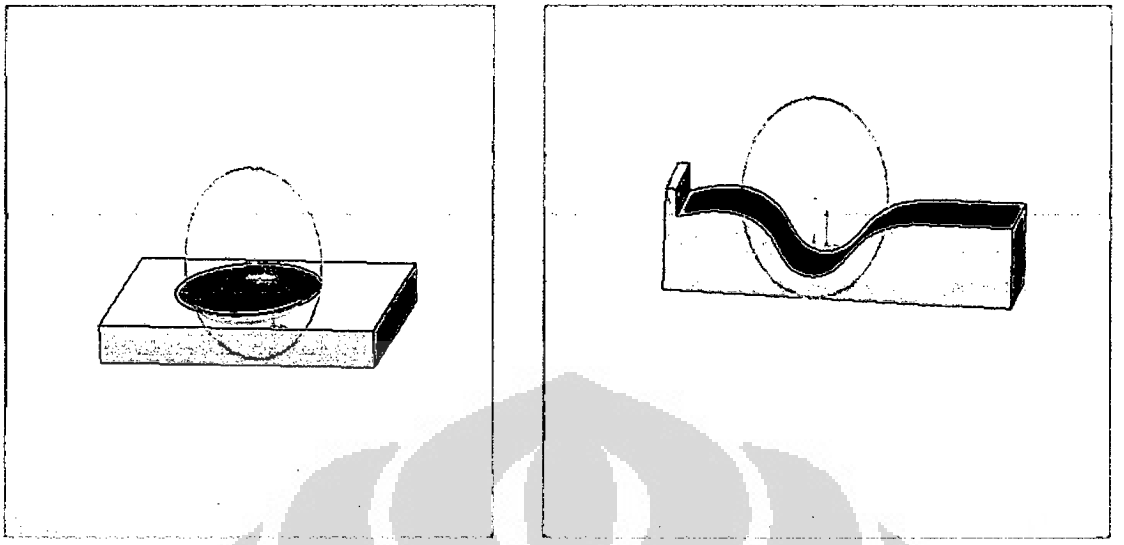
Gambar IV 1 Pembagian ruang terbatas

Penentuan daerah ruang terbatas di bagi menjadi dua kelompok yaitu : ruang terbatas terbuka *(open boundary volume)* dan daerah ruang terbatas tertutup *(close boundary volume)*.

IV.1.1 Daerah Ruang Terbatas Terbuka

Daerah ruang terbatas terbuka atau *Open boundary volume (OBV)* adalah suatu daerah dimana proses pemesinan dapat dilakukan dengan tanpa adanya deteksi awal untuk mendapat orientasi pahat. Proses pemesinan ini dapat dilakukan dengan menggunakan milling 3 Axis. Daerah ini penulis deteksi dengan adanya perbedaan kemiringan antara sumbu koordinat pahat dengan arah vektor normal minimal 30 derajat pada suatu feature benda. Dengan batasan bahwa titik pendeteksian akan

menemukan kembali vector normal yang memiliki arah berlawanan pada waktu proses pendeteksian berlangsung.

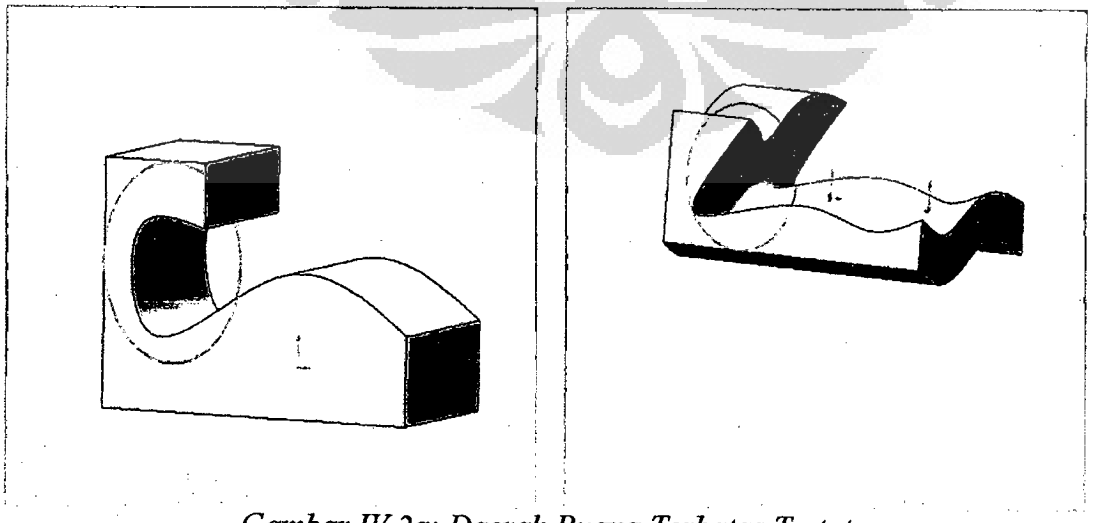


Gambar IV.1: Daerah Ruang Terbatas Terbuka

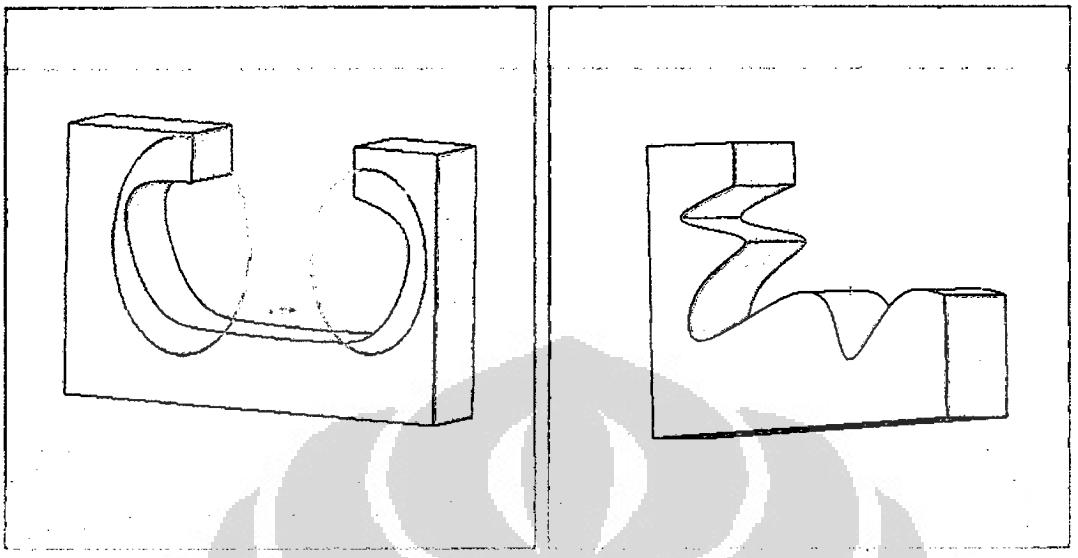
Pendeteksian daerah ruang terbatas terbuka berdasar perbedaan arah vektor normal pada segitiga. Setiap vertex dari *STL* file telah memiliki kandungan informasi vektor normal.

IV.1.2 Daerah Ruang Terbatas Tertutup

Daerah ruang terbatas tertutup atau *Close Boundary Volume (CBV)* adalah suatu daerah pada model produk yang memiliki keterbatasan gerak pahat pada saat dilakukan proses pemesinan. Daerah ruang terbatas tertutup ditandai dengan adanya kebertumpukan segitiga yang telah mengalami pengecekan. Proses pemesinan pada daerah ini menggunakan pemesinan 5 Axis, karena pahat memungkinkan untuk berganti posisi atau mengalami perubahan orientasi.



Gambar IV.2a: Daerah Ruang Terbatas Tertutup.

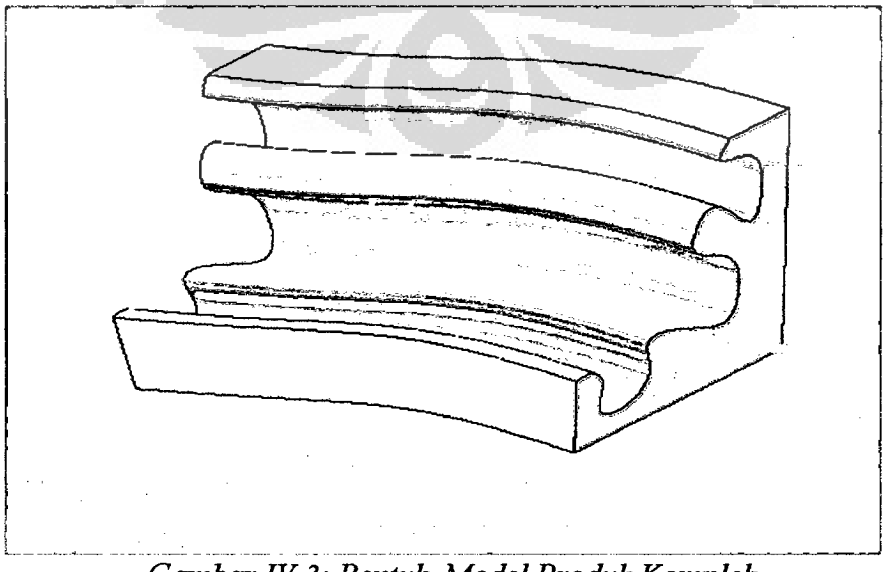


Gambar IV.2b: Daerah Ruang Terbatas Tertutup.

IV.2 Penentuan Radius Pahat

Sumbu radius pahat menjadi dasar penentuan nilai x, y. Nilai x dan y yang didapat dari nilai radius pahat menjadi awal perhitungan untuk pendeteksian segitiga segitiga yang bertumpuk.

Dalam kenyataanya, bentuk feature dapat memiliki bentuk yang komplek. Dimana terdapat daerah ruang terbatas terbuka serta ruang terbatas tertutup lebih dari 1 seperti gambar di bawah ini



Gambar IV.3: Bentuk Model Produk Komplek

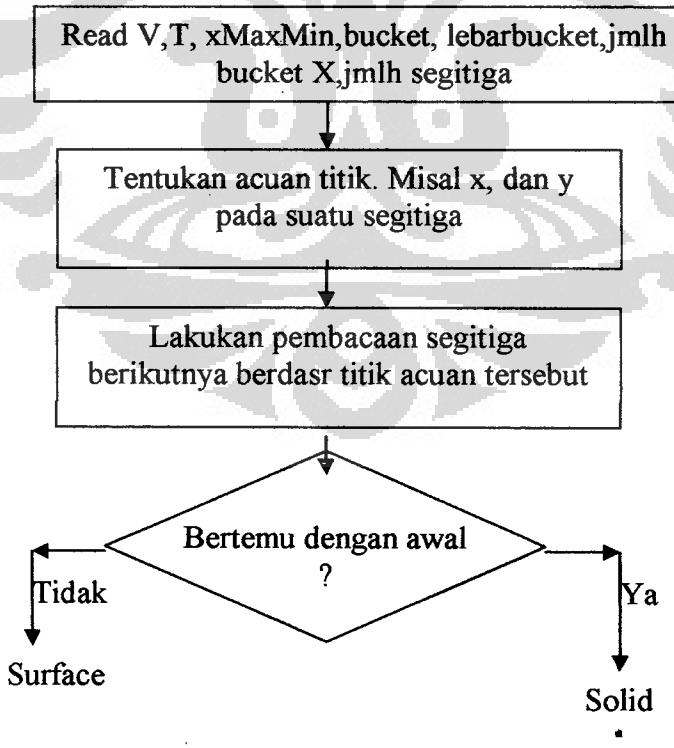
Pengecekan daerah boundary volume baik terbuka maupun tertutup di awali dari titik 0,0,0 setiap segitiga yang terdeteksi dengan titik sumbu radius pahat disimpan sementara dalam array tiga dimensi. Penyimpanan ini berdasar pada index vertex dan juga index bucket yang telah disimpan pada saat pembacaan *STL* file.

Semakin kecil nilai dari radius pahat ini, memungkinkan segitiga yang terdeteksi dan tersimpan dalam array tiga dimensi semakin banyak.

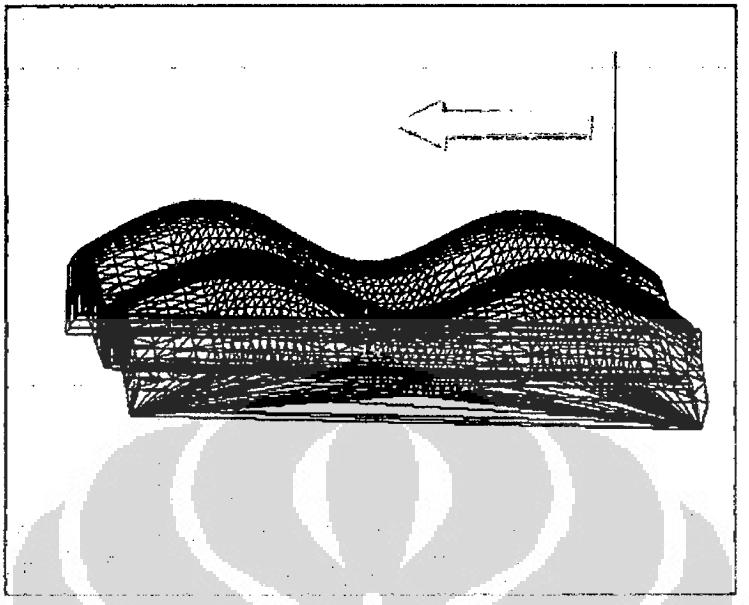
IV.3 Pengecekan Ke-*solid*-an Model

Format *STL* yang menjadi input pendeteksian ruang terbatas dapat berasal dari surface atau yang berasal dari bentuk solid. Format data *STL* yang berasal dari bentuk surface dapat langsung di aplikasikan ke dalam program. Namun apabila input data *STL* berasal dari benda solid maka, penulis harus mendefinisikan terlebih dahulu segitiga-segitiga yang akan mengalami pendeteksian ruang terbatas. Karena segitiga yang digunakan dalam proses ini hanya segitiga yang merepresentasikan feature model.

Dalam proses pendeteksian ke-solid-an model flow cart yang digunakan:



Pengembangan Algoritma pendeteksian model berbentuk solid atau berbentuk surface berdasarkan pada keterkaitan atau keterhubunganya segitiga segitiga pada *STL* file.



Gambar IV.4: Pendeteksian Kesolidan Model

Dalam pendeteksian kesolidan model, parameter parameter yang digunakan adalah:

a. Data index segitiga. Hal ini di dapat dari data yang tersimpan dalam array index segitiga pada waktu pembacaan *STL* file.
b. Data indek vertex segitiga
c. Koordinat x maximal
d. Data index bucket
e. Data jumlah keseluruhan segitiga

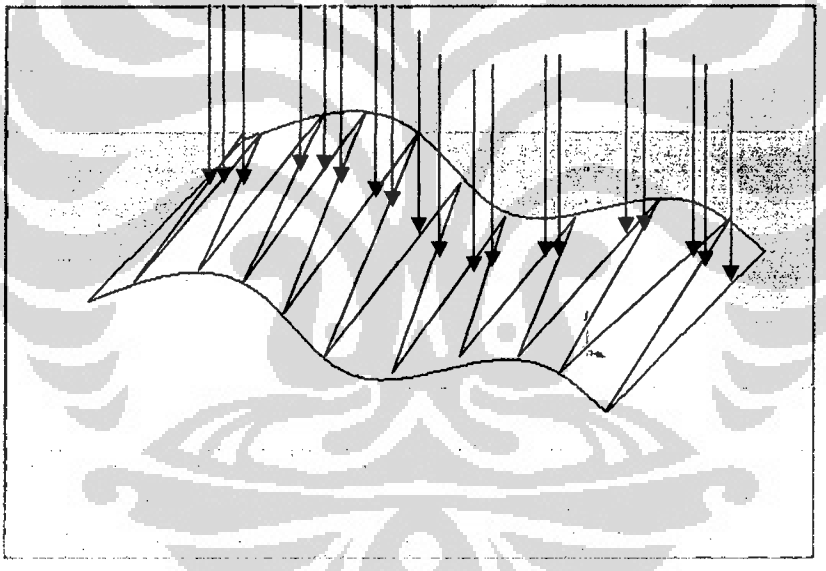
Sedangkan metode yang dikembangkan dalam menentukan kesolidan benda yang terkandung dalam data *STL* adalah:

1. Tentukan titik awal dalam memulai pendeteksian solid.
2. Titik acuan di dapatkan secara acak pada sumbu x. Hal ini didefinikan dengan x random dalam pemrograman.
3. Tentukan pula index bucket yang mewakili segitiga yang terdeteksi sebagai x random
4. Setelah titik x random dapat ditentukan, maka langkah berikutnya adalah mengurutkan pembacaan segitiga berdasar index bucket berikutnya.

5. Data yang tersimpan dalam metode ini berupa nilai x tetap dan y yang berubah-ubah sesuai dengan index bucket.

Tabel IV.1: Hasil Deteksi Kesolidan Model.

NO	Index Bucket	Index Vertex	Koord. X	Koord. Y	Koord.Z
1	(1.1	1	2.5000000E+01	-1.4660504E+01	+1.8926470E+01
2	(1,2)	3	2.5000000E+01	-1.7170671E+01	+1.8926470E+01
3	(1,3)	5	2.5000000E+01	-1.4660504E+01	-1.29780E+00
4	(6,2)	41	2.5000000E+01	-1.4660504E+01	-1.85569E+01
5	(6,3)	44	2.5000000E+01	-1.7170671E+01	-1.87777E+01
....	.....	.....	.....	.....	.....

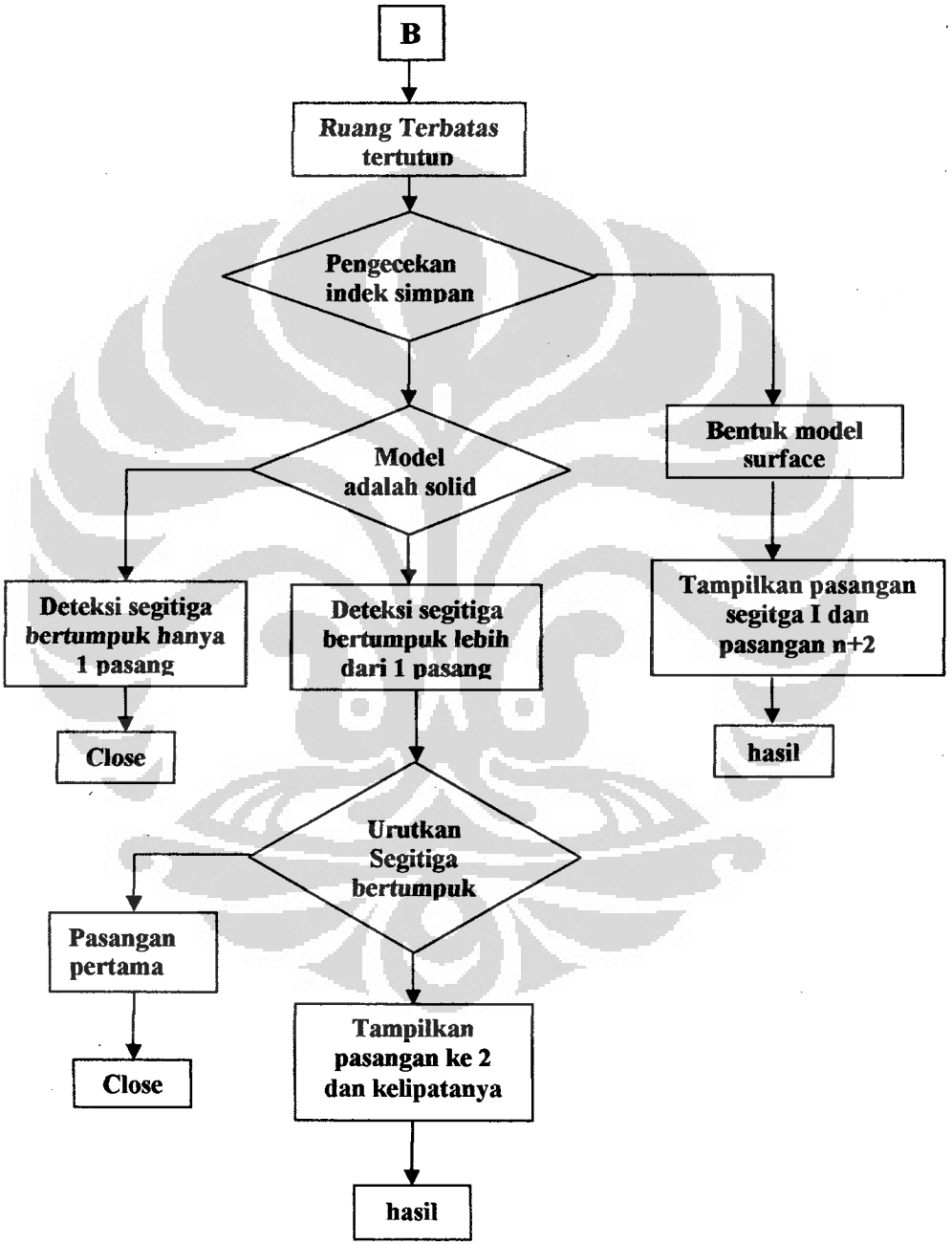


Gambar IV.5: Pendeteksian Kesolidan Untuk Surface

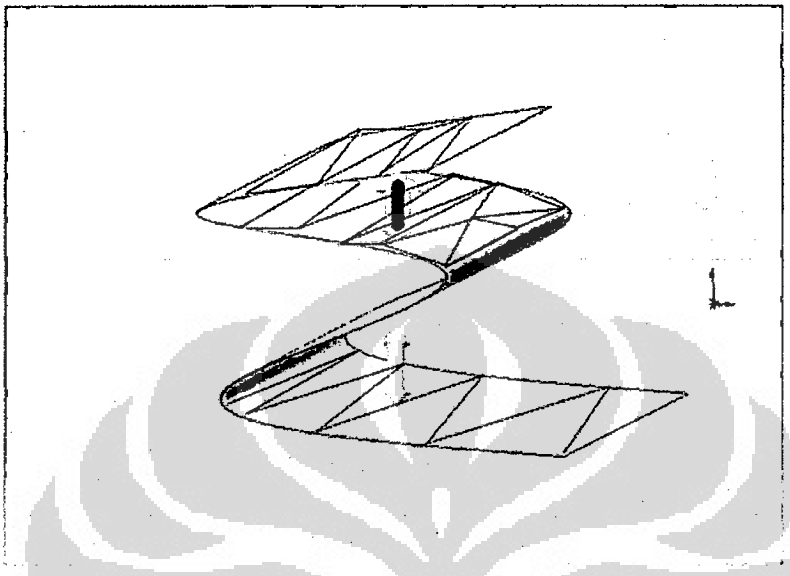
Apabila pembacaan segitiga dapat menemukan kembali titik yang dijadikan acuan, maka *STL* file yang di deteksi berasal dari bentuk model solid. Dan sebaliknya apabila titik tersebut tidak kembali menemukan titik x acuan berarti data *STL* tersebut berasal dari surface.

IV.4 Pendeteksian Segitiga Yang Bertumpuk Pada Ruang Terbatas Tertutup

Dengan nilai radius pahat yang telah ditentukan, langkah berikutnya adalah pendeteksian terhadap segitiga-segitiga yang memiliki arah vektor normal yang berbeda. Proses pendeteksian ruang terbatas tertutup sama dilakukan dengan Flow Cart dibawah:



Proses pencarian atau identifikasi segitiga bertumpuk pada wilayah tertentu menjadi lebih cepat karena telah diberikan index terhadap setiap segitiga maupun terhadap vertex yang mewakili bentuk model benda kerja. Hal ini dapat dijelaskan dalam gambar di bawah ini.



Gambar IV.6: Segitiga-Segitiga Yang Dideteksi.

Proses pendeteksian dapat dilakukan dengan langkah-langkah sebagai berikut :

1. Tentukan letak titik-titik yang ingin dideteksi kebertumpukannya berdasarkan letaknya pada bidang x, y (tanpa memperhitungkan z). Hal ini dilakukan berdasar pada arah pergerakan lintasan pahat terhadap sumbu x dan terhadap sumbu y. Hal ini dapat ditentukan dengan nilai radius pahat terhadap sumbu x, dan y.
2. Setiap segitiga yang ada dalam *STL* file, dideteksi menggunakan rumus persamaan garis

$$\frac{y - y1}{y2 - y1} = \frac{x - x1}{x2 - x1} \qquad \text{(IV, 1)}$$

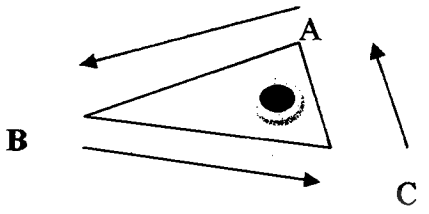
Setelah dilakukan modifikasi rumus, didapatkan:

$$Fx + gy + h = 0, \qquad \text{(IV, 2)}$$

dimana

$$f = -(y2-y1)$$
$$g = x2-x1 \qquad \text{(IV, 3)}$$
$$h = x1y2-x2y1$$

3. x dan y adalah lokasi titik yang berpotongan dengan segitiga. x1, y1, x2, dan y2 adalah lokasi vertex 1 dan 2 di segitiga (dari sisi segitiga).



Gambar IV.7: Deteksi Titik Terhadap Segitiga

Penentuan vertex 1 dan 2 dilakukan bergantian di antara vertex-vertex segitiga sesuai dengan counter clock wise.

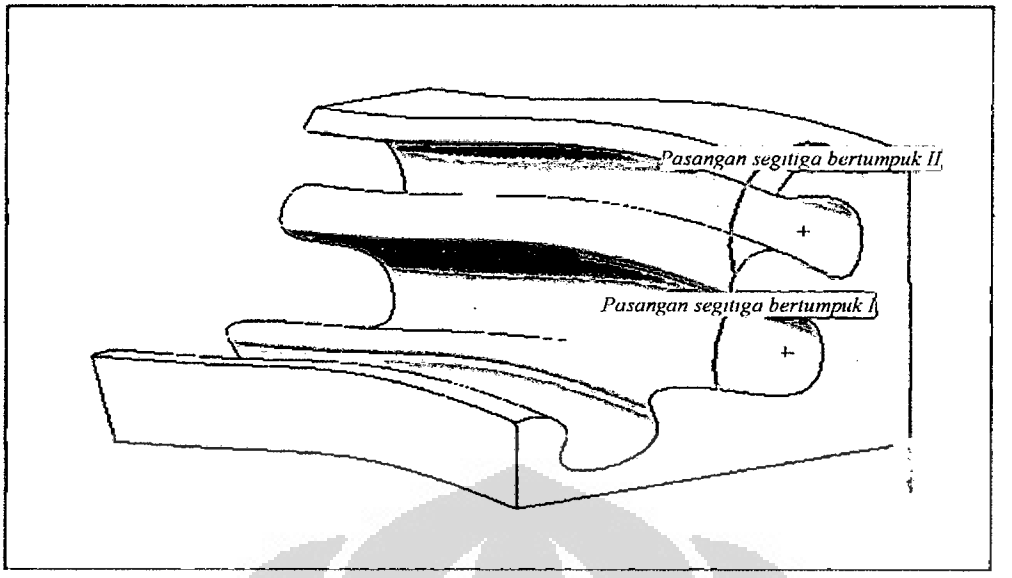
4. Dari hasil deteksi di atas simpan hasil dalam array segitiga.
5. Proses berikutnya setelah array terbentuk maka kita definisikan hal tersebut sebagai indikasi adanya daerah ruang terbatas tertutup.
6. Selanjutnya akan kita catat index dari segitiga-segitiga tersebut.

Jika garis sumbu radius pahat berpotongan dengan bidang segitiga maka garis sumbu tersebut berada di kiri dari setiap garis yang membentuk satu segitiga. maka segitiga itulah yang menjadi pusat perhitungan.

Tabel IV.2: Daftar Index Segitiga Yang Bertumpuk

No	Index segitiga I	Index segitiga II	Index segitga Ke-n
1.	17	23	....
2.	21	22	.....
3.	20	19	.....
....		.....	....

Dalam proses pendeteksian segitiga bertumpuk, segitiga-segitiga yang terdeteksi berdasar index bucket memungkinkan untuk ketidak-beraturan letaknya atau posisi segitiga. Hal ini disebabkan karena penyimpanan segitiga yang bertumpuk dilakukan secara acak. Sedangkan model produk dapat memiliki lebih dari 1 pasang segitiga yang bertumpuk. Seperti yang terjadi pada model dibawah ini:



Gambar IV.9: Pasangan Segitiga Bertumpuk Lebih Dari 1

Ketika proses pendeteksian menemukan hal seperti gambar diatas, maka hasil deteksi menjadi acak, karena dalam 1 indux bucket memungkinkan adanya 2 pasang segitiga yang bertumpuk atau lebih. Seperti yang terjadi pada tabel di bawah ini:

Tabel IV.3: Daftar Index Segitiga Yang Bertumpuk Pada Model Komplek

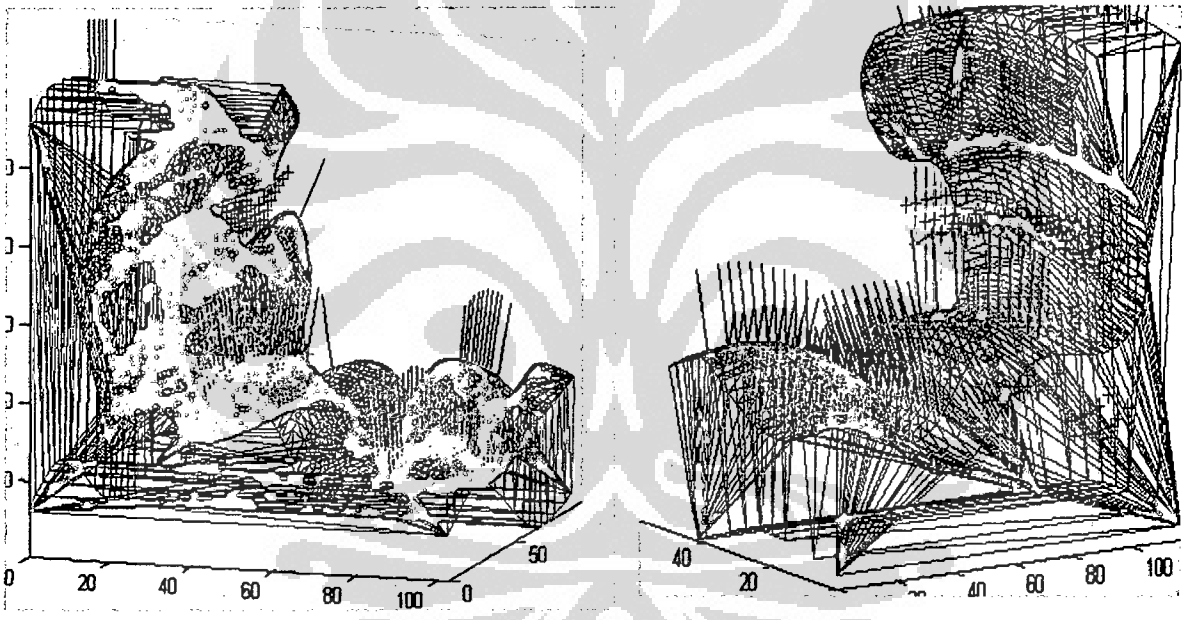
No	Index Bucket	Index segitiga	Koordinat x, y, z segitiga
1.	(17,5)	23, 64, 78, 90, 112, 134, 125, 145, ...	.....
2.	(17,6)	42, 67, 88, 94, 99, 132, 167, 133, 144, ...	.....
3.	(17,7)	19, 56, 58, 89, 78, 94, 69, 198, 188, 177,.....	.....
....		.....	....

Apabila model yang dideteksi memiliki lebih dari 1 pasang segitiga bertumpuk maka dilakukan proses pensortiran. Proses ini didasarkan pada pengurutan nilai yang telah di dapat berdasarkan besaran nilai z.

Nilai z akan di urutkan terlebih dahulu dari nilai z terendah hingga nilai z tertinggi dari pasangan segitiga yang bertumpuk. Sehingga nilai z yang akan menjadi perhitungan dalam deteksi ruang terbatas di dapatkan.

Tabel IV.4: Daftar Urutan Segitiga Yang Bertumpuk Pada Model Komplek

No	Index Bucket	Index segitiga	Koordn x	Koordn y	Koordn z
1.	(17,5)	28 dengan	.....	.....	2.408818e+001
		69	.....	.....	3.737574e+001
2.	(17,5)	42 dengan	.....	.....	3.712506e+001
		133	.....	.....	5.376183e+001
3.	(17,5)	58 dengan	.....	.....	2.595482e+001
		189	.....	.....	3.852628e+001
....		.....	....	.....	.....



Gambar IV.18a: Daerah Ruang Terbatas

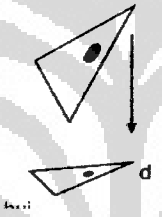
IV.5 Penentuan Tinggi Titik Dalam Bidang

Dalam model facet 3D setiap segitiga memiliki informasi yang mengandung nilai sumbu x, sumbu y, dan juga sumbu z. Dari hasil deteksi yang telah dapat dilakukan diatas maka langkah selanjutnya adalah mencari nilai z dititik yang mengalami penumpukan.

Hal ini dapat diindikasikan dengan radius pahat yang telah ditentukan. Dari nilai radius pahat yang hanya merepresentasikan nilai sumbu x dan sumbu y maka untuk dapat menghitung sumbu z dilakukan dengan persamaan:

$$t = ax + by + c \qquad \text{(IV, 7)}$$

Dimana : a, b, c merupakan konstanta dan x, y merupakan titik yang mewakili. Dalam proses pencarían, segitiga di proyeksikan ke bidang x, y dan meniadakan sumbu z. hal ini dilakukan untuk mempermudah dalam perhitungan.

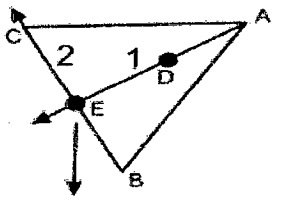


Gambar IV.10: Proyeksi Segitiga Pada Bidang X.Y

Data data proyeksi dapat dengan mudah didapatkan karena pada prose awal telah dilakukan bucketing sumbu serta telah diberikan nilai index. Dengan menggunakan persamaan garis sebagai berikut:

Pemeriksaan dilakukan pada setiap edge dengan aturan segitiga clock wise.setelah titik x,y yang di kehendaki telah di dapat maka kita harus menentukan tinggi fature benda kerja dengan rumusan:

$$\begin{pmatrix} Xa \\ Ya \end{pmatrix} + n\begin{pmatrix} Xd - Xa \\ Yd - Ya \end{pmatrix} = \begin{pmatrix} Xb \\ Yb \end{pmatrix} + m\begin{pmatrix} Xc - Xb \\ Yc - Yb \end{pmatrix} \qquad \text{(IV, 7)}$$



Gambar IV.11: Pencarian Nilai E Untuk Menentukan Nilai Z

Dari persamaan diatas, pencarían letak titik E dapat dilakukan dengan mencari nilai m dan nilai n terlebih dahulu. Nilai m dan n dapat di tentukan dengan cara eliminasi:

$$E = A + n\,(D-A) \qquad \text{(IV, 8)}$$

atau

$$E = B + m\,(C-B) \qquad \text{(IV, 9)}$$

Dengan persamaan garis yang sama, selanjutnya dilakukan pencarían terhadap nilai z dari titik yang berada pada bidang segitiga. Hal ini dilakukan dengan rumusan:

$$\begin{pmatrix} Xe \\ Ye \\ Ze \end{pmatrix} = \begin{pmatrix} Xb \\ Yb \\ Zb \end{pmatrix} + n \begin{pmatrix} Xc - Xb \\ Yc - Xb \\ Zc - Zb \end{pmatrix} \qquad \text{(IV, 10)}$$

Diantara variable variable tersebut, yang akan dicari adalah nilai dari z. hal ini dapat dibantu dengan nilai x dan nilai y yang ada. Dengan memanfaatkan nilai x atau y maka rumusan menjadi:

$$Ze = Zb + n(Zc - Zb) \qquad \text{(IV, 11)}$$

Dari perhitungan Ze didapat, maka pencarian nilai z *(Zd)* yang sebenarnya juga dapat dilakukan sehingga mendapat rumusan sebagai berikut:

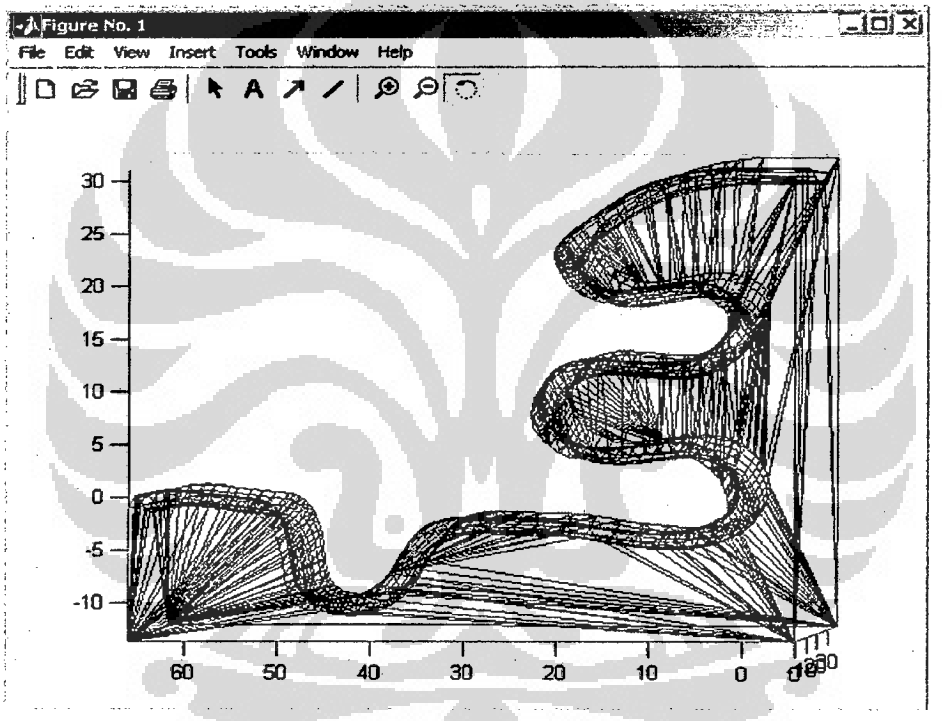
$$\begin{pmatrix} Xd \\ Yd \\ Zd \end{pmatrix} = \begin{pmatrix} Xa \\ Ya \\ Za \end{pmatrix} + n \begin{pmatrix} Xe - Xa \\ Ye - Ya \\ Ze - Za \end{pmatrix} \qquad \text{(IV, 12)}$$

Proses ini dilakukan sama dengan proses sebelumnya dengan memanfaatkan nilai x dan y guna mencari nilai n. Dari hasil perhitungan nilai tinggi yang di dapat, maka simpan semua hasil tinggi dalan array yang memuat informasi informasi seperti dalam tabel di bawah ini.

Tabel IV.5: Nilai Z Pada Segitiga Bertumpuk.

NO	Index bucket	Index segitiga	Nilai Z sgitiga bawah	Nilai Z segitiga atas
1	(31, 5)	368	1.3666077E+01	4.5246279E+01
2	(31, 5)	371	1.3666077E+01	3.5921579E+01
3	(31, 5)	373	1.3666077E+01	3.5921579E+01
.....	.....	.....	...	....

Setelah mendapat nilai nilai tinggi atau nilai koordinat Z maka tampilkan hasil yang ada sebagai berikut:

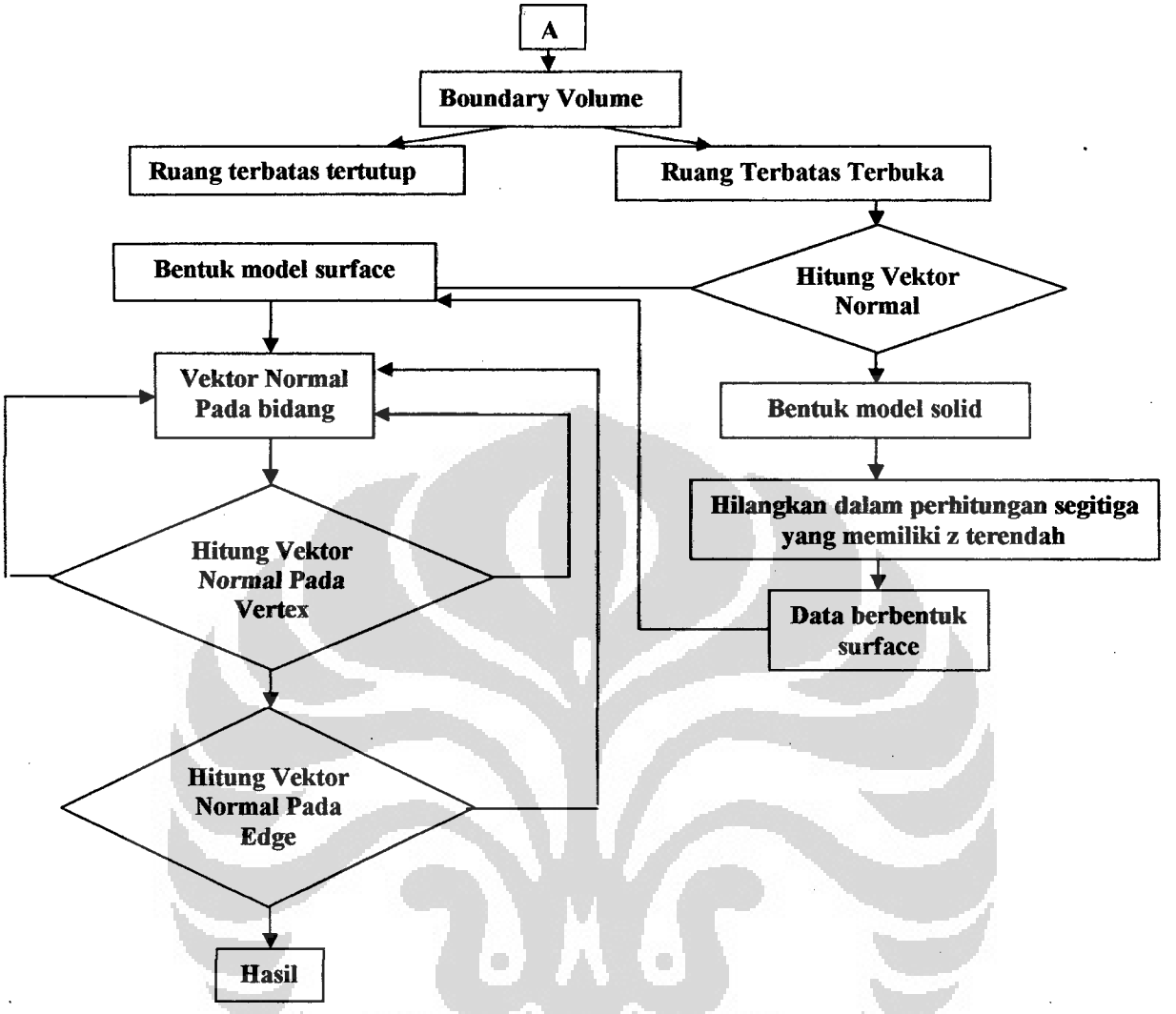


Gambar IV.12: Deteksi Tinggi Segitiga Bertumpuk.

IV.6 Pendeteksian Ruang Terbatas Terbuka

Pendeteksian daerah ruang terbatas tertutup berdasar perbedaan arah vektor normal pada segitiga. Setiap vertex dari *STL* file telah memiliki kandungan informasi vektor normal.

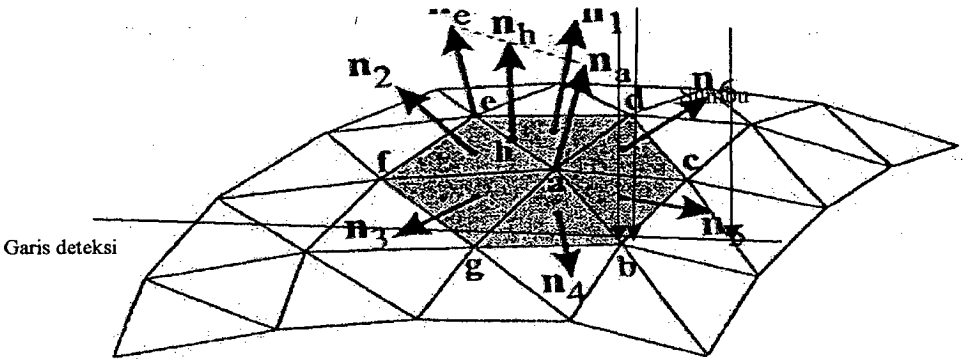
Flow cart yang dilakukan adalah:



Setiap vektor normal akan diperhitungkan untuk mendapat posisi yang tepat arah vektor normal jika dibandingkan dengan arah sumbu radius pahat yang menyentuh segitiga dan juaga berperan sebagai pembanding.

Hal hal yang diperhatikan dalam pendeteksian ruang terbatas terbuka antara lain:

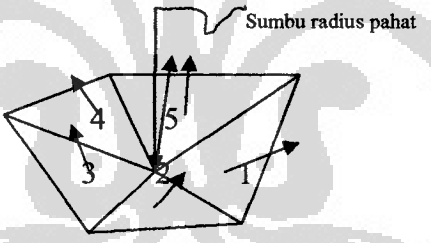
1. Pengecekan vektor normal berawal dari nilai vektor normal pada *STL* file
2. Berdasar nilai x dan y radius pahat, Sumbu radius pahat seolah olah merupakan titik normal dalam proses pendeteksianya. setiap segitiga dideteksi dengan cara membandingkan nilai vektor normal dengan sumbu pusat radius pahat.
3. Pada kenyataanya pergeseran nilai sumbu radius pahat pada saat melakukan proses deteksi tidak selalu tepat jatuh pada bidang segitiga dan atau tepat berada pada titik vektor normal pada setiap segitiga.



Gambar IV.13: Titik Deteksi Segitiga Pada Model [6]

4. Sumbu radius pahat dapat jatuh pada garis segitiga atau juga dapat jatuh pada titik pada sebuah segitiga. Dengan demikian nilai vertor normal yang didapat bukan berasal dari vektor normal yang ada dari data *STL* file yang ada.
5. Pada gambar 12 titik sumbu radius pahat jatuh tepat pada titik segitiga, nilai vektor normal dapat di hitung dengan [6]:

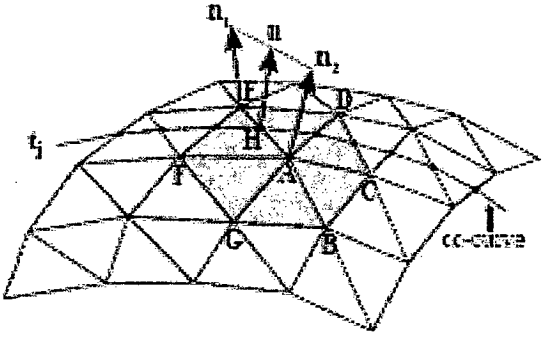
$$n = \frac{\sum_i ni}{\left|\sum_i ni\right|} \qquad \text{(IV, 4)}$$



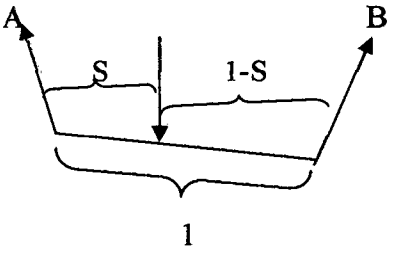
Gambar IV.14: Perhitungan Vektor Normal Pada Titik

6. Dan apabila sumbu radius pahat jatuh pada garis pada segitiga maka nilai vektor normal di tentukan dengan [6]:

$$n(s) = \frac{(1-s)nj + s\_nk}{\left|(1-s)nj + s\_nk\right|} \qquad \text{(IV, 5)}$$



Gambar IV.15a: Perhitungan Vektor Normal Pada Garis Segitiga[6].



Gambar IV.15b: Perhitungan Vektor Normal Pada Garis Segitiga.

7. Untuk proses penggambaran maka di perlukan panjang unit vektor dalam merepresentasikan normal vektor. Hal ini di dapat dengan perhitungan sebagi berikut:

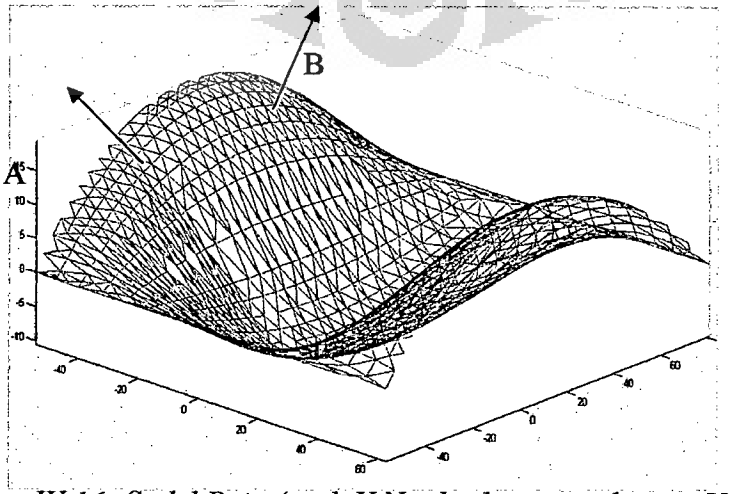
$$vektor\_unitAB = \sqrt{\frac{vektorB - A(i,j,k)}{panjangvektorB - A(\sqrt{i^2 + j^2 + k^2}}} \qquad \text{(IV, 6)}$$

Setelah pendeteksian komplit maka data yang didapat disimpan dalam array segitiga berdasarkan index segitiga, index bucket, nilai x,y,z, serta data nilai vektor normal.

Tabel IV.6: Hasil Deteksi Vektor Normal

NO	Index segitiga	Index bucket	Vektor unit i	Vektor unit j	Vektor unit k
1	79	(19, 3)	-0.2201	-0.0568	0.9738
2	84	(19, 3)	-0.2432	-0.0630	0.9875
3	85	(19, 3)	-0.2048	-0.0916	0.9798
4	91	(20, 4)	-0.2059	-0.0947	0.9887
...	....	.....	....	.....	.....

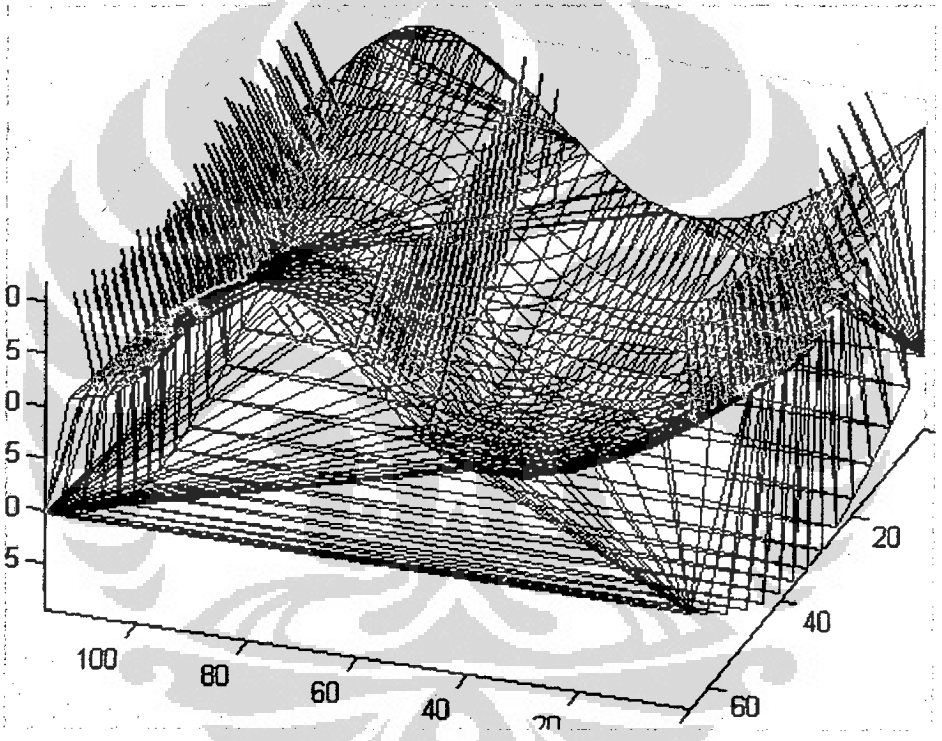
.Dalam pendeteksian ruang terbatas terbuka, tidak menutup kemungkinan data *STL* yang di deteksi merupakan representasi dari bentuk sadel point.



Gambar IV.16: Sadel Poin (arah V.N a berlawanan dengan V.N b)

Algoritma yang dikembangkan untuk mendeteksi hal tersebut adalah:

1. Deteksi titik yang berbeda arah dengan vektor normal bidang pada titik sumbu radius pahat
2. Cek vektor normal di samping titik tersebut, karena jika sadel point titik tersebut dapat mempunyai perbedaan arah yang signifikan dengan titik yang terdeteksi awal.
3. Pengecekan arah vektor normal dapat dilakukan searah dengan koordinat x atau searah dengan koordinat y.
4. Simpan dalam array segitga berdasar index segitiga dan index bucket.
5. Tampilkan hasil.



Gambar IV.17: Ruang terbatas Terbuka dengan Vektor Normal.

IV.4 Simulasi Dengan Pemrograman

Berikut hasil simulasi program untuk membaca STL file yang di dapat dari sistem CAD.

function A=Boundary_volume

```
lebarBucket = 5;
tinggiBucket = 5;
radiusPahat = 1.5;
panjangPahat = 40;
kemiringan = 30;
heightTolerance= 5;    % toleransi perbedaan tinggi untuk cekungan
zTolerance = 1; % toleransi ketinggian untuk membuat garis melintang
r=radiusPahat;
fid=fopen('gambar.txt');
%tampilkanGambar(fid);
xMinMax = [1000 -1000];
yMinMax = [1000 -1000];
zMinMax = [1000 -1000];
i=1;
j=1;
count=0;
arr = zeros(3);
while (1)
   aa=fgetl(fid);
   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
      % untuk penggambaran
      ii = 0;
      % end untuk penggambaran
      count=1;
   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;
      % untuk penggambaran
      plot3(X,Y,Z);
      hold on;
      % end untuk penggambaran
   elseif strcmp(a,'vertex')
      % untuk penggambaran
      % pembacaan X
      ii=ii+1;
      % end untuk penggambaran
```

```
      [a,b]= strtok(b,' ');
      [b,c]= strtok(b,' ');
      haha=inline(a);
      huhu(1) = haha(1);
      X(ii) = haha(1);
      haha=inline(b);
      huhu(2) = haha(2);
      Y(ii) = haha(1);
      haha=inline(c);
      huhu(3) = haha(3);
      Z(ii) = haha(1);

        V(j,1)=0;
        V(j,2)=0;
        V(j,3)=0;
        %disp([V(j,1) V(j,2) V(j,3)])

      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;
        end
      end
      if index==0, % masukkan ke vertex

          %disp('masuk');
        V(j,1)=huhu(1);
        V(j,2)=huhu(2);
        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);
        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;
      else
        arr(count)=index;
          %disp('tidak masuk');
        j=j-1;
      end
      j=j+1;
      count=count+1;
    end
end
```

```
% normalization
% calculate the distance from left-bottom-front of the model (minimum of
% the bounding box)
deltaX=xMinMax(1)-0;
deltaY=yMinMax(1)-0;
deltaZ=zMinMax(2)-0;

% calculate the bounding box to be placed in the normal position
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;
% move all the vertex to the normal position
for ii=1:j,
   V(ii,1)=V(ii,1)-deltaX;
   V(ii,2)=V(ii,2)-deltaY;
   V(ii,3)=V(ii,3)-deltaZ;
   %disp([ii V(ii,1) V(ii,2) V(ii,3)]);
end
% end of normalization

%buketing
jmlBuketX=(xMinMax(2)-xMinMax(1))/lebarBucket; % see how many bucket in x position based
on the bounding box and bucket width
jmlBuketY=(yMinMax(2)-yMinMax(1))/tinggiBucket; % see how many bucket in y position
based on the bounding box and bucket height
jmlBuketX=floor(jmlBuketX)+1;  % to make sure that the numbers of bucket can handle all the
triangles and vertices
jmlBuketY=floor(jmlBuketY)+1;  % to make sure that the numbers of bucket can handle all the
triangles and vertices
jmlSegitiga = i;   % save the number of triangles into a variable "jmlSegitiga"
jmlVertex = j;     % save the number of vertices into a variable "jmlVertex"
buket = zeros(jmlBuketX,jmlBuketY);     % menyimpan jml bucket dalam setiap bucket (karena
tidak ada kelas vector seperti di java)
isiBuket = zeros(jmlBuketX,jmlBuketY); % menyimpan index-index segitiga dalam setiap bucket

% mulai penyimpanan segitiga
% dilakukan dengan cara mengecek
for i=1:jmlSegitiga,
   xBound = [1000 -1000];
   yBound = [1000 -1000];
   xv = zeros(3);
   yv = zeros(3);
   for j=1:3,
      Ver(1)=V(T(i,j),1); % x
      Ver(2)=V(T(i,j),2); % y

      posX = floor(Ver(1)/lebarBucket)+1;
      posY = floor(Ver(2)/tinggiBucket)+1;
      xv(j) = posX;
      yv(j) = posY;
      if xBound(1)>xv(j),
         xBound(1)=xv(j);
      end
      if xBound(2)<xv(j),
         xBound(2)=xv(j);
      end
```

```
      if yBound(1)>yv(j),
        yBound(1)=yv(j);
      end
      if yBound(2)<yv(j),
        yBound(2)=yv(j);
      end

      if isiBuket(posX,posY)==0,
        isiBuket(posX,posY)=isiBuket(posX,posY)+1;
        buket(posX,posY,isiBuket(posX,posY))=i;
      elseif buket(posX,posY,isiBuket(posX,posY))~=i,
        isiBuket(posX,posY)=isiBuket(posX,posY)+1;
        buket(posX,posY,isiBuket(posX,posY))=i;
      end
      %disp([xBound(1) xBound(2)]);
    end
    %if abs(xBound(2)-xBound(1))>1 || abs(yBound(2)-yBound(1))>1,
    % isi bucket yang berlebih ditolerir, namun bucket tidak boleh
    % kekurangan isi
      for k=xBound(1):xBound(2),
        for l=yBound(1):yBound(2),
          if inpolygon(k,l,xv,yv)==1,
            if isiBuket(k,l)==0,
              isiBuket(k,l)=isiBuket(k,l)+1;
              buket(k,l,isiBuket(k,l))=i;
              %disp('ok 1 : ');
              %disp([isiBuket(k,l) buket(posX,posY,isiBuket(posX,posY))]);
            elseif buket(k,l,isiBuket(k,l))~=i,
              isiBuket(k,l)=isiBuket(k,l)+1;
              buket(k,l,isiBuket(k,l))=i;
              %disp('ok 2 : ');
              %disp([isiBuket(k,l) buket(posX,posY,isiBuket(posX,posY))]);
            end
          end
        end
      end
      %end
end
%end buketing


% cek apakah bentuk yang bersangkutan adalah solid??
disp('apakah solid??');
%huu = isSolidModel(V,T,xMinMax,buket,isiBuket,lebarBucket,tinggiBucket,jmlBuketY);
kesolidan = isSolidModel2(V,T,xMinMax,buket,isiBuket,lebarBucket,jmlBuketY,jmlSegitiga);
disp(kesolidan);

% operasi inti
% menyimpan titik dan vektor normal setiap bagian yang dilewati pahat
vertexChecking = 0; % parameter adalah bagianXsurface,bagianYsurface,vertexCheckKe-,x,y,z
nVertexCheck = 0;   % parameter adalah bagianXsurface,bagianYsurface
vectorChecking = 0; % parameter adalah bagianXsurface,bagianYsurface,vertexCheckKe-,i,j,k
bagX = 0;   % index x untuk setiap bagian yang dilewati garis pahat
bagY = 0;   % index y untuk setiap bagian yang dilewati garis pahat
% pertanyaan :
% 1. bagaimana cara membedakan bagian bawah (untuk solid) dengan
% surfacenya, sebab keduanya masuk perhitungan
% 2. apakah yang diambil VN hanya bagian atas saja? Tidak, karena kalau
% bertumpuk vektor yang menghadap ke bawah tidak akan terdeteksi
```

```
% mencari ruang terbatas dan menampilkannya
doubleR = radiusPahat*2;
indexTemp=0;   % banyaknya bagian yang bertumpuk
jarak = xMinMax(2)-xMinMax(1);
tinggi = 0;
pertengahanTinggi = 0;
counterSimpan = 0;
%disp([xMinMax(2)-xMinMax(1) radiusPahat]);
%disp(jarak/radiusPahat);
simpanTitikZDanVNormBertumpuk = 0;  % z1-z2-z3-zn.., dan menyimpan vektor normal nya
jumlahTitikZBertumpuk = 0;  % menyimpan jumlah titik z yang bertumpuk
vertexOperation = 0;    % menyimpan vertex-vertex yang diperiksa, argumen terdiri dari index
x,index y, x/y/z/i/j/k, 0 semua jika bertumpuk
jmlTitikPadaBaris = 0;  % menyimpan banyaknya vertex pada baris tertentu
x = xMinMax(1)-radiusPahat;
%for x=xMinMax(1)+radiusPahat:doubleR:xMinMax(2),
while 1,
   x = x+doubleR;
   if x>=xMinMax(2),
      break;
   end
   % untuk bagian surface yang dilewati garis pahat
   bagX = bagX+1;
   bagY = 0;
   % end of untuk bagian surface yang dilewati garis pahat
   indexSimpan = 0;   % array menyimpan segitiga
   for y=yMinMax(1)+radiusPahat:doubleR:yMinMax(2),
      % untuk bagian surface yang dilewati garis pahat
      bagY = bagY+1;
      nVertexCheck(bagX,bagY) = 0;   % pada bagian ke (bagX,bagY)
      % end of untuk bagian surface yang dilewati garis pahat
      posX = floor(x/lebarBucket)+1;
      posY = floor(y/tinggiBucket)+1;
      Vpotong.x=x;
      Vpotong.y=y;
      jmlSgtBerpot = 0;  % lakukan sesuatu jika ada segitiga yang bertumpuk
      banyak = isiBuket(posX,posY);
      for i=1:banyak,
         indexTri = buket(posX,posY,i);
         dd=0;
         aa=3;
         if indexTri>0,
            for j=1:3,
               index1 = j;
               index2 = j+1;
               if index2>3,
                  index2=index2-3;
               end

               % vertex pertama garis
               Vertex1.x = V(T(indexTri,index1),1);
               Vertex1.y = V(T(indexTri,index1),2);
               % vertex kedua garis
               Vertex2.x = V(T(indexTri,index2),1);
               Vertex2.y = V(T(indexTri,index2),2);
               %hitung
               t=-1*(Vertex2.y-Vertex1.y)*Vpotong.x+(Vertex2.x-
Vertex1.x)*Vpotong.y+Vertex1.x*Vertex2.y-Vertex2.x*Vertex1.y;
               %disp([t i]);
               if t>0,
```

```
            dd=dd+1;
          elseif t<0,
            aa=aa-1;
          else
            aa=aa-1;
            dd=dd+1;
          end
        end
      end
      if dd==3 || aa==0,
        jmlSgtBerpot=jmlSgtBerpot+1;
        %disp(jmlSgtBerpot);
        indexSimpan(jmlSgtBerpot)=indexTri;

        % untuk bagian surface yang dilewati garis pahat
        % cari titik tengah segitiga
        D = [x y];
        Ver1 = [V(T(indexTri,1),1) V(T(indexTri,1),2) V(T(indexTri,1),3)];
        Ver2 = [V(T(indexTri,2),1) V(T(indexTri,2),2) V(T(indexTri,2),3)];
        Ver3 = [V(T(indexTri,3),1) V(T(indexTri,3),2) V(T(indexTri,3),3)];
        hasilZ = hitung_z2(Ver1,Ver2,Ver3,D);

        nVertexCheck(bagX,bagY) = nVertexCheck(bagX,bagY)+1;
        % simpan xyz titik potong
        vertexChecking(bagX,bagY,nVertexCheck(bagX,bagY),1) = Vpotong.x;   % x
        vertexChecking(bagX,bagY,nVertexCheck(bagX,bagY),2) = Vpotong.y;   % y
        vertexChecking(bagX,bagY,nVertexCheck(bagX,bagY),3) = hasilZ(1);   % z
        % simpan ijk titik potong
        if hasilZ(2) == 0,
          % titik potong jatuh pada vertex segitiga
          if hasilZ(3) == 1,
            % titik potong jatuh pada vertex pertama segitiga
            vektorNormal1                                                  =
hitungVekNorBerbobot2(V,T,buket,isiBuket,T(indexTri,1),lebarBucket,tinggiBucket);
          elseif hasilZ(3) == 2,
            % titik potong jatuh pada vertex kedua segitiga
            vektorNormal1                                                  =
hitungVekNorBerbobot2(V,T,buket,isiBuket,T(indexTri,2),lebarBucket,tinggiBucket);
          elseif hasilZ(3) == 3,
            % titik potong jatuh pada vertex ketiga segitiga
            vektorNormal1                                                  =
hitungVekNorBerbobot2(V,T,buket,isiBuket,T(indexTri,3),lebarBucket,tinggiBucket);
          end
          vectorChecking(bagX,bagY,nVertexCheck(bagX,bagY),1) = vektorNormal1(1);   % i
          vectorChecking(bagX,bagY,nVertexCheck(bagX,bagY),2) = vektorNormal1(2);   % j
          vectorChecking(bagX,bagY,nVertexCheck(bagX,bagY),3) = vektorNormal1(3);   %
k
        elseif hasilZ(2) == 1,
          % titik potong jatuh pada edge segitiga
          if hasilZ(3) == 1,
            % titik potong jatuh pada edge segitiga dari vertex
            % pertama dan kedua
            index1 = T(indexTri,1);
            index2 = T(indexTri,2);
          elseif hasilZ(3) == 2,
            % titik potong jatuh pada edge segitiga dari vertex
            % kedua dan ketiga
            index1 = T(indexTri,2);
            index2 = T(indexTri,3);
          elseif hasilZ(3) == 3,
```

```
            % titik potong jatuh pada vertex ketiga segitiga
            % titik potong jatuh pada edge segitiga dari vertex
            % ketiga dan pertama
            index1 = T(indexTri,3);
            index2 = T(indexTri,1);
          end
          % hitung vektor normal masing-masing vertex
          vektorNormal1                                                    =
hitungVekNorBerbobot2(V,T,buket,isiBuket,index1,lebarBucket,tinggiBucket);
          vektorNormal2                                                    =
hitungVekNorBerbobot2(V,T,buket,isiBuket,index2,lebarBucket,tinggiBucket);
          % masukkan nilai xyz ke vertex1 dan vertex2
          vertex1(1) = V(index1,1);
          vertex1(2) = V(index1,2);
          vertex1(3) = V(index1,3);
          vertex2(1) = V(index2,1);
          vertex2(2) = V(index2,2);
          vertex2(3) = V(index2,3);
          vertexTengah(1) = Vpotong.x;
          vertexTengah(2) = Vpotong.y;
          vertexTengah(3) = hasilZ(1);
          % hitung normal vertexnya
          vektorNormal                                                     =
hitungVekNorInterNormVer(vektorNormal1,vektorNormal2,vertex1,vertex2,vertexTengah);
          % masukkan ke dalam vectorChecking
          vectorChecking(bagX,bagY,nVertexCheck(bagX,bagY),1) = vektorNormal(1);   % i
          vectorChecking(bagX,bagY,nVertexCheck(bagX,bagY),2) = vektorNormal(2);   % j
          vectorChecking(bagX,bagY,nVertexCheck(bagX,bagY),3) = vektorNormal(3);   % k
        else
          % titik potong jatuh pada face segitiga
          vectorChecking(bagX,bagY,nVertexCheck(bagX,bagY),1) = T(indexTri,4);   % i
          vectorChecking(bagX,bagY,nVertexCheck(bagX,bagY),2) = T(indexTri,5);   % j
          vectorChecking(bagX,bagY,nVertexCheck(bagX,bagY),3) = T(indexTri,6);   % k
        end
        % untuk bagian surface yang dilewati garis pahat
      end
    end
    % lihat apakah solid type lebih dari 1 atau tidak
    n = 1;
    % hilangkan yang boundary tertutup
    if jmlSgtBerpot==2 && kesolidan==1,
      % bagian yang diambil adalah atas atau bawah?
      if vertexChecking(bagX,bagY,1,3)<vertexChecking(bagX,bagY,2,3),
        n = 2;
      end
      vertexOperation(bagX,bagY,1) = vertexChecking(bagX,bagY,n,1); % x
      vertexOperation(bagX,bagY,2) = vertexChecking(bagX,bagY,n,2); % y
      vertexOperation(bagX,bagY,3) = vertexChecking(bagX,bagY,n,3); % z
      vertexOperation(bagX,bagY,4) = vectorChecking(bagX,bagY,n,1); % i
      vertexOperation(bagX,bagY,5) = vectorChecking(bagX,bagY,n,2); % j
      vertexOperation(bagX,bagY,6) = vectorChecking(bagX,bagY,n,3); % k
      continue;
    elseif jmlSgtBerpot == 1,
      % hanya 1 layer
      vertexOperation(bagX,bagY,1) = vertexChecking(bagX,bagY,n,1); % x
      vertexOperation(bagX,bagY,2) = vertexChecking(bagX,bagY,n,2); % y
      vertexOperation(bagX,bagY,3) = vertexChecking(bagX,bagY,n,3); % z
      vertexOperation(bagX,bagY,4) = vectorChecking(bagX,bagY,n,1); % i
      vertexOperation(bagX,bagY,5) = vectorChecking(bagX,bagY,n,2); % j
      vertexOperation(bagX,bagY,6) = vectorChecking(bagX,bagY,n,3); % k
```

```
        continue;
    elseif jmlSgtBerpot > 0,
        sejajar = 1;
        tinggiZ = vertexChecking(bagX,bagY,1,3); % z
        for i=2:nVertexCheck(bagX,bagY),
            if tinggiZ~=vertexChecking(bagX,bagY,i,3),
                sejajar=0;
                break;
            end
        end
        if sejajar==1,
            vertexOperation(bagX,bagY,1) = vertexChecking(bagX,bagY,n,1); % x
            vertexOperation(bagX,bagY,2) = vertexChecking(bagX,bagY,n,2); % y
            vertexOperation(bagX,bagY,3) = vertexChecking(bagX,bagY,n,3); % z
            vertexOperation(bagX,bagY,4) = vectorChecking(bagX,bagY,n,1); % i
            vertexOperation(bagX,bagY,5) = vectorChecking(bagX,bagY,n,2); % j
            vertexOperation(bagX,bagY,6) = vectorChecking(bagX,bagY,n,3); % k
            continue;
        end
    end
    % jika terdapat segitiga yang bertumpuk, cari tingginya
    if jmlSgtBerpot>1,

        % cek apakah perpotongan tersebut berada pada titik yang
        % berbeda (jika berpotongan pada garis / titik)

        % urutkan letak titik perpotongan dari titik terkecil ke titik
        % terbesar. Jangan lupa pastikan titik yang sama persis harus
        % dihapus

        % buat garis berselang-seling antara titik terkecil dengan
        % titik terbesar

        indexTemp=indexTemp+1;
        % simpan titik yang berpotongan
        %D = [x y];
        huhu=0;
        arrayTemp = 0; % untuk menyimpan array Z temporary, for pengurutan only
        aa = 1;
        for i=1:jmlSgtBerpot,
            % cari titik tengah segitiga
            %Ver1        =       [V(T(indexSimpan(i),1),1)       V(T(indexSimpan(i),1),2)
V(T(indexSimpan(i),1),3)];
            %Ver2        =       [V(T(indexSimpan(i),2),1)       V(T(indexSimpan(i),2),2)
V(T(indexSimpan(i),2),3)];
            %Ver3        =       [V(T(indexSimpan(i),3),1)       V(T(indexSimpan(i),3),2)
V(T(indexSimpan(i),3),3)];
            %D(3) = hitung_z(Ver1,Ver2,Ver3,D);
            % simpan titik tersebut
            ashoyy(indexTemp,i,1) = vertexChecking(bagX,bagY,i,1);
            ashoyy(indexTemp,i,2) = vertexChecking(bagX,bagY,i,2);
            ashoyy(indexTemp,i,3) = vertexChecking(bagX,bagY,i,3);

            %ashoyy(indexTemp,i,1) = D(1);   % ashoyy(urutan tiang, vertex atas/bawah, elemen
titik x/y/z)
            %ashoyy(indexTemp,i,2) = D(2);
            %ashoyy(indexTemp,i,3) = D(3);
            % tambahkan z untuk menghitung pertengahan
            huhu=huhu+ashoyy(indexTemp,i,3);
```

```
            % untuk menghitung segitiga bertumpuk yang lebih dari 1
            %arrayTemp(i) = D(3);
            arrayTemp(aa) = vertexChecking(bagX,bagY,i,3);
            aa=aa+1;
            arrayTemp(aa) = vectorChecking(bagX,bagY,i,1);
            aa=aa+1;
            arrayTemp(aa) = vectorChecking(bagX,bagY,i,2);
            aa=aa+1;
            arrayTemp(aa) = vectorChecking(bagX,bagY,i,3);
            aa=aa+1;
          end
          tambahan=4;
          % urutkan z bertumpuk dari yang terendah hingga yang tertinggi
          [m n] = size(arrayTemp);
          arrayTemp = lakukanInsertionsort2(arrayTemp,1,n,1,tambahan);
          % untuk menghitung segitiga bertumpuk yang lebih dari 1
          % masukkan segitiga bertumpuk tersebut ke tempat yang telah
          % disediakan
          simpanTitikZDanVNormBertumpuk(indexTemp,1,1) = arrayTemp(1);
          simpanTitikZDanVNormBertumpuk(indexTemp,1,2) = arrayTemp(2);
          simpanTitikZDanVNormBertumpuk(indexTemp,1,3) = arrayTemp(3);
          simpanTitikZDanVNormBertumpuk(indexTemp,1,4) = arrayTemp(4);
          counter = 1;   % membantu mengecek penumpukan
          akhir = jmlSgtBerpot*tambahan;
          i = 1+tambahan;
          %for i=1+tambahan:tambahan:akhir,
          while i<=akhir,
            if abs(arrayTemp(i)-arrayTemp(i-tambahan))==0.0001,      % memastikan bahwa titik
yang bersangkutan benar-benar bertumpuk
              % do nothing
              %continue;
            else
              counter=counter+1;
              simpanTitikZDanVNormBertumpuk(indexTemp,counter,1) = arrayTemp(i);
              simpanTitikZDanVNormBertumpuk(indexTemp,counter,2) = arrayTemp(i+1);
              simpanTitikZDanVNormBertumpuk(indexTemp,counter,3) = arrayTemp(i+2);
              simpanTitikZDanVNormBertumpuk(indexTemp,counter,4) = arrayTemp(i+3);
            end
            i = i+tambahan;
          end
          jumlahTitikZBertumpuk(indexTemp) = counter;
          if counter>1,
            % set 0 semua
            vertexOperation(bagX,bagY,1) = 0;
            vertexOperation(bagX,bagY,2) = 0;
            vertexOperation(bagX,bagY,3) = 0;
            vertexOperation(bagX,bagY,4) = 0;
            vertexOperation(bagX,bagY,5) = 0;
            vertexOperation(bagX,bagY,6) = 0;
          end
        end
      end
      jmlTitikPadaBaris(bagX)=bagY;
    end
    fclose(fid);

    % strategi :
    % 1. cek dengan cara yang sudah ada
    % 2. cek kesadelan dengan mengecek kiri dan kanan dari titik periksa
    % 3. cek perbedaan ketinggian antar titik periksa
```

```
% cek searah sumbu y
% kode vektor normal
% 0 : bertumpuk
% 1 : miring ke depan
% 2 : datar
% 3 : miring ke belakang
areaCheck = 0;
vektor = [0 0 0];
for i=1:bagX,
   for j=1:jmlTitikPadaBaris(i),
      areaCheck(i,j) = 2;  % nilai default
      if vertexOperation(i,j,4)~=0 || vertexOperation(i,j,5)~=0 || vertexOperation(i,j,6)~=0,
         % cegah bukan pada bagian bertumpuk
         vektor(1) = vertexOperation(i,j,4);
         vektor(2) = vertexOperation(i,j,5);
         vektor(3) = vertexOperation(i,j,6);
         if isMoreThanNDegree(vektor,kemiringan)==1,
            if vektor(2)>0, % cek y-nya
               areaCheck(i,j) = 1;
            elseif vektor(2)<0,
               areaCheck(i,j) = 3;
            end
         end
      else
         % bertumpuk
         areaCheck(i,j) = 0;
      end
   end
end

% jika 0-2-3 / 0-3 : identifikasi
% jika 1-2-0 / 1-0 : identifikasi
% jika 1-2-3 / 1-3 : identifikasi
% jika 2-2-2-.... : cek tinggi diantaranya
% mulai proses : 1 / 0
% stop proses : 3
% jika perbedaan tinggi > lebar pahat : identifikasi
areaCheck2 = 0;
for i=1:bagX,
   inProses=0;
   lastVekCheck = 2;   % nilai default
   currentVekCheck = 2;
   % jika awalan bernilai 1 (cekungan pertama, proses sedang in)
   lastVekCheck = areaCheck(i,1);
   if lastVekCheck == 1,
      inProses=1;
      %disp('pertama sudah ok');
      %disp(i);
   end
   simpan(1)=0;
   simpan(2)=-1;
   for j=2:jmlTitikPadaBaris(i),
      areaCheck2(i,j) = 0;  % nilai default
      isSadel(i,j) = 0; % nilai default
      kiri = 0;
      kanan=0;
      currentVekCheck = areaCheck(i,j);
      if inProses==0,
         if currentVekCheck==1 && lastVekCheck==2,
```

```
            inProses=1;
          elseif currentVekCheck==1 && lastVekCheck==3,
            inProses=1;
          elseif currentVekCheck==2 && lastVekCheck==0,
            inProses=1;
            simpan(1)=1;   % menandakan keluar dari CBV, dan bertemu datar
            simpan(2)=j;   % menyimpan letak keluar dari CBV
          elseif currentVekCheck==3 && lastVekCheck==0,
            inProses=1;
          end
        else
          if currentVekCheck==2 && lastVekCheck==3,
            inProses=0;
          elseif currentVekCheck==1 && lastVekCheck==2,   % jika baru keluar dari CBV, dan
bertemu datar, kemudian ketemu cekung
            inProses=1;
            if simpan(1)==1,
              % nolkan sebelumnya
              for k=j:-1:simpan(2),
                areaCheck2(i,k) = 0;  % nilai default
              end
              simpan(1) = 0;  % reset nilainya
              simpan(2) = -1; % reset nilainya
            end
          elseif currentVekCheck==1 && lastVekCheck==3,
            % berhenti dan mulai lagi secara bersamaan
            inProses=0;
            inProses=1;
          elseif currentVekCheck==0 && lastVekCheck==1,
            inProses=0;
          elseif currentVekCheck==0 && lastVekCheck==2,
            inProses=0;
          end
          % sebelum memberi nilai, periksa dulu kesadelan titik yang
          % bersangkutan
          if i>1,
            % cek sebelah kiri
            kiri = cekSamping(i-1,jmlTitikPadaBaris,vertexOperation,2,vertexOperation(i,j,2));
          end
          if i<bagX,
            % cek sebelah kanan
            kanan = cekSamping(i+1,jmlTitikPadaBaris,vertexOperation,2,vertexOperation(i,j,2));
          end

          if kiri==-1 && kanan==1,
            % keadaan sadel
            isSadel(i,j) = 1;
          else
            % tidak sadel
            isSadel(i,j) = 0;
          end
          areaCheck2(i,j)=1;
        end
        lastVekCheck = areaCheck(i,j);
      end
      % jika terakhirnya 1, dan bukan berada pada code 3 (cekungan terakhir),
      % maka harus mundur, dan nol-kan hingga bertemu 0, atau hingga bertemu
      % perubahan code 1-2/1-3 dari belakang
      j = jmlTitikPadaBaris(i);
      if areaCheck(i,j)~=3,   % jika terakhir bukan code 3
```

```
%       disp('ada mundur...');
%       disp(j);
      currentVekCheck = areaCheck(i,j);
      lastVekCheck = areaCheck(i,j);
      while areaCheck2(i,j)==1,  % selama masih dianggap boundary volume
        areaCheck2(i,j)=0;  % nolkan
        currentVekCheck = areaCheck(i,j);
        if j==1,    % jika sudah titik awal, keluar
          break;
        elseif currentVekCheck==2 && lastVekCheck==1,  % jika ada perubahan 1-2, keluar
          break;
        elseif currentVekCheck==3 && lastVekCheck==1,  % jika ada perubahan 1-3, keluar
          break;
        end
        lastVekCheck = areaCheck(i,j);
        j = j-1;
      end
    end
end
% end of cek searah sumbu y

% mengecek berdasarkan ketinggian
% tinggi diperiksa berdasarkan letak antar 2 titik periksa
% kode 0 = sama tinggi
% kode 1 = tinggi ke rendah
% kode 2 = rendah ke tinggi
% kode -1 = daerah bertumpuk, tidak dicek
cekTinggi=0;
heightCheck=0;
for i=1:bagX,
   heightCheck(i,1)=0;
   for j=2:jmlTitikPadaBaris(i),
      heightCheck(i,j)=0; % default
      if vertexOperation(i,j-1,4)==0 && vertexOperation(i,j-1,5)==0 && vertexOperation(i,j-
1,6)==0,
        % daerah bertumpuk
        heightCheck(i,j)=-1;
      elseif vertexOperation(i,j-1,3)-heightTolerance>vertexOperation(i,j,3),
        % turun
        heightCheck(i,j)=1;
      elseif vertexOperation(i,j-1,3)+heightTolerance<vertexOperation(i,j,3),
        % naik
        heightCheck(i,j)=2;
      else
        % tingginya sama
        heightCheck(i,j)=0;
      end
   end
end
% jika 1-0-2 / 1-2 : identifikasi
heightCheck2=0;
for i=1:bagX,
   inProses=0;
   for j=1:jmlTitikPadaBaris(i),
      heightCheck2(i,j)=0;   % default
      if inProses==0,
        if heightCheck(i,j)==1,
          inProses=1;
        end
```

```
      else
        heightCheck2(i,j)=1;
        if heightCheck(i,j)==-1,
          % daerah bertumpuk
          heightCheck2(i,j)=0;
          inProses=0;
        end
        if heightCheck(i,j)==2, % jika rendah ke tinggi, lihat langkah selanjutnya
          if j<jmlTitikPadaBaris(i),
            if heightCheck(i,j+1)==0 || heightCheck(i,j+1)==1,  % jika selanjutnya datar atau
turun
              heightCheck2(i,j)=0;
              inProses=0;
            end
          end
        end
      end
    end
    % jika terakhir bukan pada code 2(naik), mundur
    j = jmlTitikPadaBaris(i);
    if heightCheck(i,j)~=2,
      while heightCheck2(i,j)==1, % selama masih dianggap boundary volume
        if j==0,   % sudah berada di titik awal
          break;
        elseif heightCheck(i,j)==2, % jika sudah berada pada posisi naik, berhenti
          break;
        end
        heightCheck2(i,j)=0;
        j=j-1;
      end
    end
end
% end of mengecek berdasarkan ketinggian

% tampilkan setiap vertex dan vektor normalnya, kec yang bertumpuk
for i=1:bagX,
  for j=1:jmlTitikPadaBaris(i),
    if (areaCheck2(i,j)==1 || heightCheck2(i,j)==1) && isSadel(i,j)==0,
      if (vertexOperation(i,j,4)~=0 || vertexOperation(i,j,5)~=0 || vertexOperation(i,j,6)~=0) &&
vertexOperation(i,j,6)>0,
        % titik pertama
        A(1) = vertexOperation(i,j,1)+deltaX;
        B(1) = vertexOperation(i,j,2)+deltaY;
        C(1) = vertexOperation(i,j,3)+deltaZ;
        % titik kedua
        A(2) = vertexOperation(i,j,1) + deltaX + vertexOperation(i,j,4)*10;  % x+i
        B(2) = vertexOperation(i,j,2) + deltaY + vertexOperation(i,j,5)*10;  % y+j
        C(2) = vertexOperation(i,j,3) + deltaZ + vertexOperation(i,j,6)*10;  % z+k
        % gambarkan vektor normal
        % plot3(A,B,C,'k');
        hold on;
      end
    end
  end
end

if indexTemp>0,
  % persiapan penggambaran garis tengah
  patokan = 0;       % sebagai patokan y dan z, array 2d dengan el 1 adalah... dan el2 adalah yz
  jmlPatokan = 0;    % jumlah patokan y dan z
```

```
  garisMelintang = 0; % garis melintang yang akan digambar, array 2d dengan el1 adalah ... dan
el2 adalah xyz
  jmlTtkGaris = 0;   % jml titik pada satu garis melintang
  % end of persiapan penggambaran garis tengah
  % tampilkan garis tinggi yang kedua
  for i=1:indexTemp,
    awalan=1;
    if kesolidan==1,
      if mod(jumlahTitikZBertumpuk(i),2)==1,
        awalan=1;
      else
        awalan=2;
      end
    end
    for j=awalan:2:jumlahTitikZBertumpuk(i)-1,
      % titik pertama
      A(1) = ashoyy(i,1,1)+deltaX;
      B(1) = ashoyy(i,1,2)+deltaY;
      C(1) = simpanTitikZDanVNormBertumpuk(i,j,1)+deltaZ;
      % titik kedua
      A(2) = ashoyy(i,2,1)+deltaX;
      B(2) = ashoyy(i,2,2)+deltaY;
      C(2) = simpanTitikZDanVNormBertumpuk(i,j+1,1)+deltaZ;
      % gambarkan
      %plot3(A,B,C,'y');
      hold on;
      % gambarkan titik tengah
      tengah = (C(1)+C(2))/2;
      %
      %plot3(A(1),B(1),tengah,'g+');
      hold on;

      % analisis patokan
      tempPat = searchIndexBy2Index(jmlPatokan,patokan,B(1),tengah,0.1,zTolerance);
      if tempPat>jmlPatokan,
        jmlPatokan=jmlPatokan+1;
        jmlTtkGaris(tempPat) = 0;
      end
      patokan(tempPat,1) = B(1);
      patokan(tempPat,2) = tengah;
      % simpan garis melintang
      jmlTtkGaris(tempPat) = jmlTtkGaris(tempPat)+1;
      garisMelintang(tempPat,jmlTtkGaris(tempPat),1) = A(1);
      garisMelintang(tempPat,jmlTtkGaris(tempPat),2) = B(1);
      garisMelintang(tempPat,jmlTtkGaris(tempPat),3) = tengah;
    end
  end
  % gambarkan garis melintang
  for i=1:jmlPatokan,
    A=0; B=0; C=0;
    for j=1:jmlTtkGaris(i),
      A(j) = garisMelintang(i,j,1);
      B(j) = garisMelintang(i,j,2);
      C(j) = garisMelintang(i,j,3);
    end

    %plot3(A,B,C,'r');
    hold on;
  end
end
```

```
%disp([xMinMax(1) xMinMax(2) yMinMax(1) yMinMax(2) zMinMax(1) zMinMax(2)]);
axis([xMinMax(1)+deltaX xMinMax(2)+deltaX yMinMax(1)+deltaY yMinMax(2)+deltaY
zMinMax(1)+deltaZ zMinMax(2)+deltaZ]);
```

Dari pemrograman ini dapat terdeteksi jumlah segitiga yang bertumpuk yang menjadi dasar penetuan bidang terbatas *(boundary volume)*

BAB V
KESIMPULAN DAN SARAN PENELITIAN LEBIH LANJUT

V. 1 Kesimpulan

Dalam penelitian yang dilakukan kali ini penulis menyimpulkan bahwa:

1. Pendeteksian ruang terbatas pada model produk didasarkan pada vektor normal segitiga dan perhitungan nilai z setiap titik pada segitiga.
2. Metode pendeteksian ruang terbatas berdasar beberapa metode diantaranya:
   a. Penyusunan struktuk data dari model CAD dan STL file
   b. Pengecekan setiap segitiga pada STL file yang menjadi pusat perhitungan
   c. Pengecekan ke-solid-an model untuk mendefinisikan bentuk STL dari model solid atau model surface
   d. Pengecekan ketinggian titik potong dari radius pahat dan segitiga yang menjadi pusat perhitungan dan dasar pendeteksian.
   e. Pengecekan segitiga bertumpuk dari hasil deteksi segitga dan hasil dteksi tinggi titik potong antara segitiga dan radius pahat.
   f. Pengecekan vektor normal pada setiap segitga terhadap perbedaan arah dari radius pahat yang terdeteksi.
   g. Tampila hasil dari vector normal yang memiliki arah berbeda dengan arah normal radius pahat.

V. 2 Saran Penelitian

Berikut adalah saran yang dapat diberikan penulis berkaitan dengan kelanjutan penelitian terhadap pengembangan sistem CAM .

1. Perbaikan dan evaluasi implementasi metode pendeteksian ruang terbatas pada proses pemesinan masih dapat dikembangkan lebih lanjut. Karena dalam prosesnya penulis hanya memanfaatkan nilai sumbu radius pahat untuk nilai awal pendeteksian.
2. Perlu adanya pengembangan lebih lanjut untuk otomatisasi sistem CAM pada proses pemesinan CNC.

DAFTAR ACUAN

[1] B.K.Choi, RB Jerard, "Sculptured Surface Machining, Theory and Apllication", Kluwer Academic Publisher, 1998

[2] H. Anton. Dasar-dasar Aljabar Linier, Jilid I. Jakarta : Interaksara. 2000

[3] Kiswanto G, "Optimasi Jumlah Cutter-Contact Point Pada Pembuatan Lintasan Pahat Proses Pemesinan Milling Berbasis Model Faset 3D" Kumpulan Abstrak, SNTTM V, 2006

[4] Kiswanto G, A Mujahid"Pengembangan Algoritma Cepat Penentuan Itik-Kontak Pahat (*Cutter Contact Point*) Pada System-Cam Berbasis Model Faset 3d Untuk Pemesinan Awal *(Roughing)* Dan Akhir *(Finishing)* SNTTM V, 2006

[5] Kiswanto G, "Pengembangan Metode Penentu Jarak Antar Lintasan Pahat (Step-Over) Yang Akurat Pada Pembuatan Lintasan Pahat Proses Pemesinan *Milling* Berbasis Model Faset 3d, Proseding SNTTM V, 2006

[6] Kiswanto G, B Lauwers, JP Kruth, "Gouging Elimination Through Pahat Lifting in Pahatpath Generation for Five-Axis Milling Based on Faceted Models", IJAMT 2007, 32 : 293 – 309

[7] Kresnha. P.E. Pengembangan Sistem CAM Berbasis Model Faset 3D untuk Pemesinan 5 Axis. *Laporan Kerja Praktik*, 2007

[8] Shimada, Kenji, David C.Gossard"Automatic triangular mesh generation of trimmed parametric surface for finite element analisys, Computer Aided geometric design 1998, 15 : 199 – 222

[9] 3 Axis And 5 Axis Machining.
http://www.fanuc.com/3_Axis_and_5_Axis_Machining.htm. Tanggal akses : 10 january 2008, Jam 14.00

[10] 5 Axis Milling
http:// www.siemens.com/5_Axis_milling. Tanggal akses: 5 maret 2008, Jam 09.00