BAB 3
PENGEMBANGAN METODE DAN ALGORITMA PEMESINAN MULTI AXIS

File STL hanya memuat informasi mengenai arah vektor normal dan koordinat vertex pada setiap segitiga / faset. Untuk mengolah data ini menjadi output yang diinginkan, maka perlu dibuat suatu struktur data untuk menyimpan informasi yang terdapat pada file STL tadi. Struktur data tadi dapat digambarkan sebagai berikut:

Tabel 3.1 struktur data faset

Indeks Faset	Indeks Vertex 1	Indeks Vertex 2	Indeks Vertex 3
1	1	2	3
2	3	2	4
3	2	4	5

Tabel 3.2 struktur data vertex

Indeks Vertex	Koord x	Koord y	Koord z
1	4	2,3	10
2	3	3	0
3	1	-4	5

Secara singkat, algoritma yang dikembangkan dalam laporan ini adalah sebagai berikut:

1. Validasi jenis file STL
2. Pembacaan data dari file STL dan pembuatan struktur data
3. Proses normalisasi model faset pada sumbu kartesian 3 dimensi
4. Proses bucketing
5. Pendeteksian closed boundary volume (CBV)
6. Pengelompokan CBV
7. Analisa batas-batas CBV
8. Penentuan orientasi mata pahat terhadap model pada awal proses pemesinan
9. Pembuatan lintasan mata pahat berikut orientasi mata pahat

3.1 Validasi Jenis File STL

Validasi terhadap jenis file mutlak diperlukan untuk memastikan apakah file yang sedang dianalisis ini benar-benar merupakan file STL atau bukan. Apabila hasil dari proses validasi menyatakan bahwa benar file itu adalah file STL, maka algoritma selanjutnya dapat dijalankan. Namun apabila hasil validasi tersebut ternyata menyatakan bahwa file tersebut bukan STL, maka proses penerapan algoritma berikutnya tidak perlu dijalankan, untuk menghindari kesia-siaan.

Untuk melakukan validasi, maka kita perlu mengenali ciri-ciri khas dari sebuah file STL. Seperti yang telah dicontohkan pada bab sebelumnya, isi dari file STL yang bertipe ASCII dapat dibaca dengan baik oleh text editor, termasuk MATLAB. Kata pertama yang muncul dari sebuah file STL adalah ‘solid’ dan diakhiri oleh kata ‘endsolid’. Setelah itu pada baris berikutnya terdapat urutan kata ‘facet normal’ yang diikuti oleh tiga bilangan yang terpisah oleh spasi. Selanjutnya, dilanjutkan dengan ‘outloop’ , kemudian kata ‘vertex’ yang diikuti oleh 3 bilangan koordinat vertex. Ada tiga baris data vertex untuk setiap faset, sesuai jumlah vertex pembentuk segitiga tersebut. Kemudian diikuti oleh kata ‘endloop’, dan ‘endfacet’. Semua urutan ini berulang untuk sesuai dengan jumlah facet yang dimiliki oleh model tersebut.

Sebuah file STL harus memenuhi struktur data seperti di atas. Namun untuk memvalidasi seluruh struktur tersebut tidaklah efisien. Sehingga untuk memvalidasi apakah file tersebut adalah STL atau bukan, cukup dengan mengecek kata pertama yang muncul di awal file, apakah kata ‘solid’ atau bukan. Jika kata pertama yang ditemui bukanlah ‘solid’, maka algoritma selanjutnya batal untuk dijalankan.

3.2 Pembacaan Data dari File STL dan Pembuatan Struktur Data

Algoritma yang dibuat harus mampu mengambil data dari file STL yang memuat model yang sudah kita buat sebelumnya. Algoritma ini juga harus dapat membedakan kapan ia harus mengambil data arah vektor normal suatu faset, atau kapan ia harus mengambil data koordinat vektor suatu facet. Maka algoritma ini harus mampu membaca dan membedakan nilai input yang berjarak spasi pada file STL. Setelah itu, algoritma ini harus mampu untuk mmbangun struktur data seperti yang telah dijelaskan di atas, sehingga operasi-operasi berikutnya dapat dijalankan.

Algoritma tersebut adalah sebagai berikut:

```
Algoritma: Validasi, Pembacaan Data STL dan Pembuatan Struktur Data
Input: Data vektor normal (i, j, k) dan koordinat vertex pada file STL
Output: Struktur data faset; Struktur data vertex

If yang terbaca adalah baris pertama
   Cek apakah yang terbaca 'solid'?
   If bukan terbaca 'solid' then
      Break / stop program
   end
Else do:
  For jumlah baris dalam file STL do:
  If yang terbaca adalah 'facet' do
     For 1:3 (banyak data setelah kata 'facet') do:
         1. Ambil data vektor normal setelah spasi
         2. Ubah data ke dalam format numerik
         3. masukkan data vektor normal ke dalam MatrixNormal layer 1
     End
  Elseif yang terbaca adalah 'vertex' do:
     For repeat=1:3
         1. ambil data koordinat vertex setelah spasi
         2. ubah data ke dalam format numerik
         3. masukkan data koordinat vertex ke dalam matrix vertex mentah
     End
  End
  1.  Pindai matrix vertex mentah, cari vertex-vertex yang sama
  2.  eliminir vertex kembar, susun koordinat vertex pada matrixoke
  3.  susun indeks verteks pada MatrixNormal layer 2
End
```

Output dari algoritma ini adalah 2 buah matriks, yaitu MatrixNormal (struktur data faset) dan matrixoke (struktur data vertex). MatrixNormal terdiri dari 2 layer. Layer pertama berisi data arah vektor normal faset, sedangkan layer kedua berisi data indeks vertex yang membentuk faset tersebut. Indeks faset adalah urutan baris dari MatrixNormal, sehingga untuk memanggil faset yang bersangkutan, kita cukup mengetik baris ke berapa dari MatrixNormal, sesuai dengan indeks yang diinginkan. Sedangkan matrixoke hanya terdiri dari 1 layer dengan tiga kolom. Kolom pertama adalah koordinat x vertex, kolom kedua adalah koordinat y dan kolom ketiga adalah koordinat z. Indeks vertex juga merupakan baris dari matriks ini. Karena itu, ketika program menemukan kata 'vertex' pada file STL yang sedang dibaca, maka selain program ini akan menyimpan nilai ketiga koordinat, ia juga akan menyimpan nilai baris dari matrixoke dan dimasukkan ke dalam MatrixNormal pada layer kedua.

3.3 Proses Normalisasi

Model faset yang dibaca, memiliki geometri tertentu yang terletak pada sumbu kartesian sesuai seperti pada saat model itu dibuat dengan software CAD. Model tersebut bisa jadi memiliki nilai koordinat positif untuk semua titiknya, atau mungkin nilai negatif untuk semua titiknya, atau bisa jadi memuat nilai positif dan negatif. Hal ini akan menyulitkan pada proses analisis berikutnya. Akan lebih mudah

jika semua koordinat model tersebut bernilai positif, dan lebih mudah lagi jika nilai terendah dari koordinat model tersebut bernilai nol. Hal ini akan sangat membantu dalam proses bucketing.

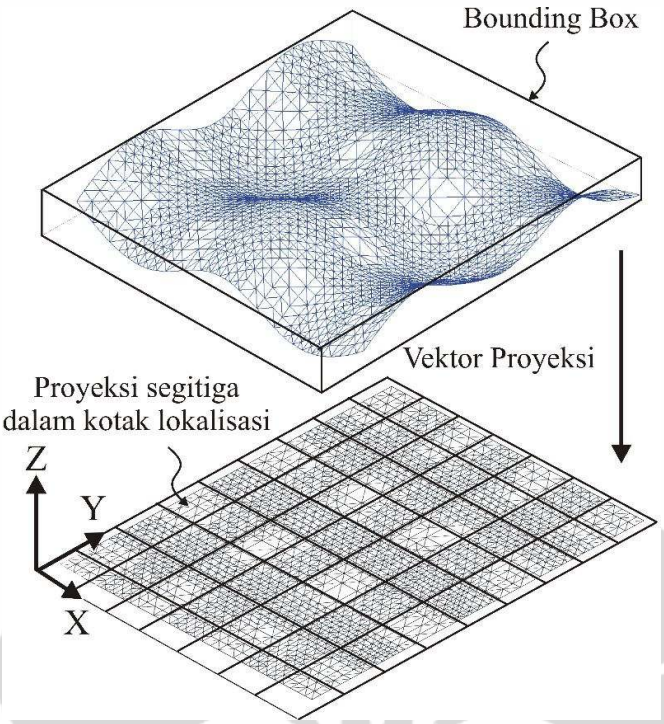
Algoritma yang dapat kita pakai untuk hal ini adalah sebagai berikut:

```
Algoritma: Normalisasi
Input: data vertex
Output: Normalisasi struktur data vertex

tentukan nilai pertama x, y, z sebagai nilai xmin, xmax, ymin, ymax, zmin dan zmax
for jumlah baris dalam matrixoke do:
   if nilai x, y atau z yang terbaca < xmin, ymin atau zmin
      ganti nilai xmin, ymin atau zmin dengan nilai x, y, atau z yang baru
   elseif nilai x, y, atau z yang terbaca > xmax, ymax atau zmax
      ganti nilai xmax, ymax atau zmaxn dengan nilai x, y, atau z yang baru
   end
end
for jumlah baris dalam matrixoke do:
   1. kurangi semua nilai koordinat xmin, ymin atau zmin
   2. tetapkan nilai xmin, ymin, zmin, xmax, ymax, dan zmax yang baru
End
```

3.4 Proses *Bucketing*

Proses *bucketing* adalah proses proyeksi dari seluruh faset atau segitiga yang terdapat pada model ke sebuah bidang proyeksi (misalkan bidang xz, y sebagai tinggi), kemudian bidang proyeksi tersebut dibagi menjadi kamar-kamar atau *bucket* yang akan memuat beberapa segitiga sesuai dengan letak koordinat bucket tersebut. Hal ini akan membantu pada proses analisis gouging, di mana deteksi terjadinya gouging tidak perlu dilakukan pada seluruh faset yang ada, namun hanya dilakukan pada bucket yang bersinggungan dengan pahat. Dengan demikian waktu yang dibutuhkan untuk melakukan pendeteksian tersebut menjadi jauh lebih singkat, terlebih lagi untuk model-model dengan ukuran yang besar.

Penentuan geometri dan ukuran bucket dapat disesuaikan dengan kebutuhan. Namun akan lebih mudah jika bucket ini berupa bujur sangkar. Semakin besar panjang sisi bucket, maka akan semakin banyak faset yang termuat di dalamnya. Ini akan menyebabkan analisis yang dilakukan pada sebuah bucket menjadi lebih lama dibandingkan jika bucket memuat faset lebih sedikit.



Gambar 3.1 Visualisasi Proses Bucketing
Diambil dari referensi [4]

Secara umum urutan algoritma yang dapat dipakai:

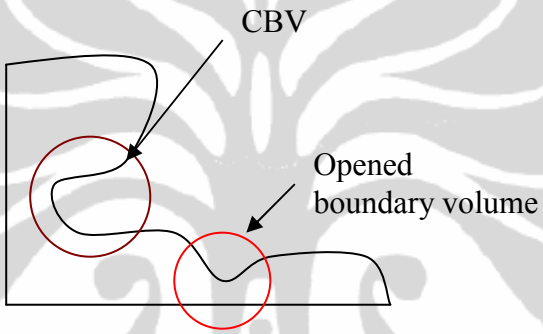
```
Algoritma: Bucketing
Input:
  Sruktur data faset
  Struktur data vertex
Output
  Bucket

1. tentukan ukuran bucket
2. tentukan jumlah bucket pada sisi x
3. tentukan jumlah bucket pada sisi z
4. buat template untuk bucket
For setiap jumlah bucket
   buat 25 titik sampel dalam setiap bucket
end
for seluruh jumlah faset
    for ke-25 titik
       cek apakah titik masuk di dalam faset
       if ya
          1. masukkan indeks faset ke dalam bucket yang aktif
          2. tambah layer pada bucket
          3. break
       end
    end
end
```

Algoritma ini akan membuat 25 titik sampel pada setiap bucket, kemudian dengan function inpolygon, dapat dicek apakah ke-25 titik tersebut masuk ke dalam setiap faset. Jika ada titik yang masuk di dalam faset, maka index faset tersebut akan dicatat

ke dalam bucket di mana titik sampel tadi berada. Semakin banyak titik sampel yang dibuat, maka akan semakin akurat hasil yang diperoleh, namun akan memakan memory dan waktu yang cukup lama untuk melakukan proses komputasinya.

Input dari langkah ini adalah struktur data faset dan struktur data vertex, dan output-nya adalah struktur data bucket. Struktur data bucket sendiri adalah sebuah matriks *m* x *n* x *l*, di mana *m* adalah banyaknya bucket ditinjau pada sumbu z, *n* adalah banyaknya bucket ditinjau pada sumbu x, sedangkan *l* adalah layer matriks yang menginformasikan banyaknya faset yang berada pada sebuah bucket. Elemen-elemen dari struktur data bucket ini adalah indeks faset dari model.

3.5 Pendeteksian *Closed Boundary Volume* (CBV) dengan PNVB



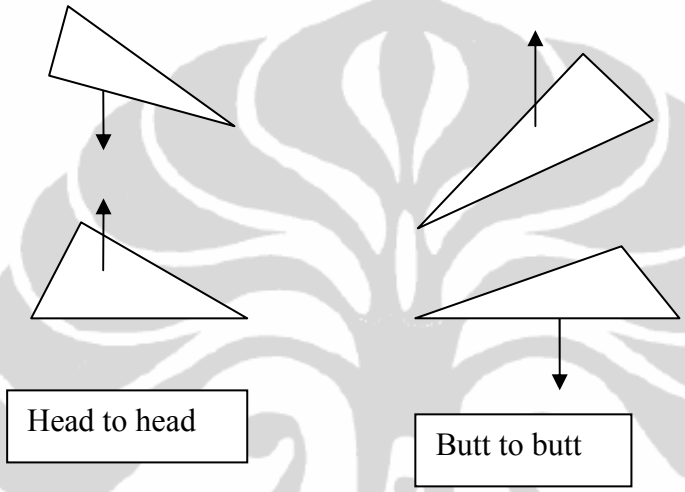
Gambar 3.2 contoh closed boundary volume dan opened boundary volume

Prinsip dari proses pendeteksian CBV ini menggunakan analisa arah vektor normal dari setiap faset. Pada sebuah *cutting-contact point* (cc point), atau titik kontak antara pahat dengan model, yang terletak pada sebuah koordinat di bidang proyeksi hasil proses bucketing sebelumnya, kita akan mendapatkan faset-faset yang berpotongan dengan cc point tersebut. Kemudian tumpukan faset-faset yang berpotongan dengan cc point tersebut kita urutkan berdasar tinggi koordinat titik perpotongan pada masing-masing faset, sehingga kita dapat mengetahui mana faset yang terletak paling atas, nomor dua dari atas, dan seterusnya sampai faset yang paling bawah. Setelah semua faset terurut dengan benar, kita dapat menganalisa masing-masing faset, apakah faset tersebut menghadap ke atas atau ke bawah? Hal ini dapat dilihat dari vektor normalnya. Inilah yang disebut dengan teknik *Paired Normal Vectors Bucketing* (PNVB). Untuk lebih mudahnya semua vektor normal dapat kita normalisasi, kita bulatkan menjadi dua tipe saja, yaitu vektor normal ke atas (nilai 1)

dan vektor normal ke bawah (nilai -1). Berikut adalah algoritma dari normalisasi vektor:

```
if vj < 0
  vj = -1
elseif vj > 0
   vj = 1
end
```

Ada dua kemungkinan urutan vektor normal pada faset-faset yang bertumpuk, yaitu posisi *butt to butt* atau posisi *head to head*, seperti terlihat pada gambar 3.3.



Gambar 3.3 Posisi *Head to Head* dan *Butt to Butt*

Jika posisi yang terjadi adalah *butt to butt*, maka kita dapat menyimpulkan bahwa ada volume solid di antara kedua faset tersebut. Sedangkan jika yang terjadi adalah posisi head to head, maka kita dapat menyimpulkan keberadaan sebuah CBV di sana. Jika posisi yang terjadi bukan butt to butt atau bukan pula head to head, maka kemungkinan besar terjadi kesalahan dalam proses triangulasi model, atau terjadi kesalahan dalam analisis perpotongan antara cc point dengan faset yang bersangkutan

Pendeteksian CBV dapat dilakukan dengan algoritma sebagai berikut:

```
Algoritma: Pendeteksian CBV
Input:
  Struktur data faset
  Struktur data vertex
  Bucket
Output:
  Pendeteksian CBV
  Pengurutan faset dalam cc point (matriks fctdiurutinok)
  Matriks 'peta'

1. Tentukan geometri pahat
2. Buat titik-titik cc point pada bidang proyeksi
3. Deteksi perpotongan cc point dengan faset di tiap bucket:
For setiap cc point
   Tentukan bucket letak cc point
   For jumlah faset dalam bucket
      If faset posisi tegak
         Ignore
      End
      Cek apakah cc point berpotongan dengan faset
      If ya
         1. Masukkan data faset ke dalam facetdiurutin
         2. Cek letak perpotongan cc point dengan faset: di dalam, pada edge, atau
            pada vertex
         3. Masukkan kode letak perpotongan pada facetdiurutin
         4. cari koordinat tinggi titik perpotongan, masukkan dalam facetdiurutin
      End
   End
   Tambah layer pada facetdiurutin
End
4. Urutkan faset per cc point do:
For jumlah cc point pada facetdiurutin
   1. Buat matriks baru fctdiurutinok untuk urutan faset yang benar
   2. buat matriks peta layer 1 → indeks faset
   For jumlah baris pada facetdiurutin do:
      Asumsikan titik perpotongan tersebut paling tinggi, masukkan ke fctdiurutinok
      For jumlah baris fctdiurutinok pd cc point yang sama
         bandingkan tinggi perpotongan
         if ada yang lebih besar dari elemen facetdiurutin
            catat baris fctdiurutinok ke berapa yg lebih besar
         elseif tidak ada yang lebih besar
            asumsi dibenarkan
         end
      end
      if nilai baris paling besar
         letakkan paling atas
      elseif nilai bukan paling besar
         letakkan sesuai tempatnya
      end
   end
end
5. deteksi pasangan vektor normal dengan posisi head to head do:
for jumlah cc point pada fctdiurutinok
   for jumlah baris per cc point
  if vj < 0
     vj = -1
  elseif vj > 0
     vj = 1
      end
      if vj(baris)==-1 dan vj(baris+1)==1
         ada CBV
         catat jumlah CBV
         catat koordinat tinggi CBV
      end
      if ada CBV
         buat matriks peta layer 2 → jumlah CBV
         buat matriks peta layer 3 → tinggi CBV
      end
   end
end
```

Penjelasan dari algoritma di atas adalah sebagai berikut:

1. buat koordinat cc point sesuai dengan geometri pahat yang digunakan
2. cek per cc point: cc point tersebut terletak pada bucket yang mana?
3. cek cc point itu berpotongan dengan segitiga yang mana saja? Simpan indeks segitiga berpotongan pada matriks facetdiurutin.
4. cari tinggi titik perpotongan di masing-masing segitiga
5. urutkan segitiga di matriks facetdiurutin dari yang paling tinggi ke paling rendah
6. seluruh vektor normal pada segitiga-segitiga tersebut dibuat searah sumbu z
7. cek apakah ada vektor normal yang tegak ke bawah diikuti vektor normal yang tegak ke atas (*head to head*), tandai koordinatnya sebagai letak CBV

Matriks kelompok CBV yang merupakan hasil dari langkah ini terdiri dari layer-layer, di mana setiap layer berisi data cc point pada satu CBV. Berarti jumlah layer pada matriks ini juga menginformasikan jumlah CBV yang terdeteksi pada model.

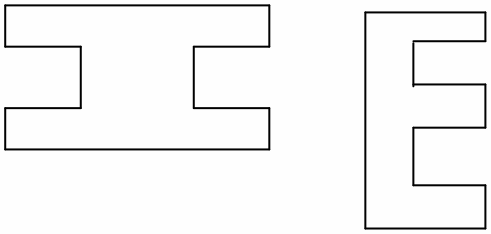
3.6 Pengelompokan CBV

CBV yang terdeteksi pada algoritma sebelumnya meliputi seluruh CBV yang ada pada sebuah model. Untuk dapat melakukan proses pemesinan pada setiap CBV, kita perlu menganalisa setiap CBV secara khusus. Karena itu data CBV yang telah diperoleh dengan algoritma sebelumnya perlu dilanjutkan dengan algoritma untuk 'memecah' data CBV tersebut dan mengelompokkannya ke dalam CBV masing-masing.

Posisi CBV yang satu terhadap posisi CBV yang lainnya dapat digolongkan menjadi tiga tipe, yaitu:

1. tipe terpisah mutlak

   Pada tipe ini, antara CBV yang satu dengan CBV yang lain jika dilihat dari atas ada dinding pemisah yang jelas. Ada volume solid yang memisahkan antara CBV-CBV tersebut, sehingga CBV dapat dikelompokkan berdasarkan terdeteksinya volume solid tersebut.



Gambar 3.4 CBV terpisah mutlak dan CBV bertumpuk

2. tipe bertumpuk

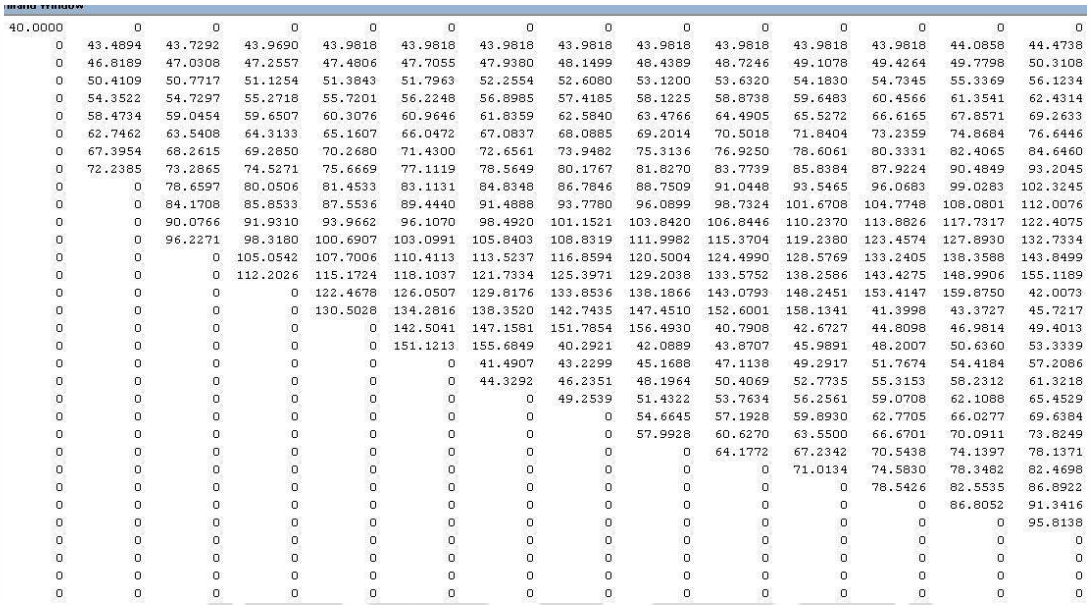
   Ada pula kasus di mana CBV-CBV pada sebuah model berada pada posisi bertumpuk, di mana ada volume solid yang memisahkan mereka secara vertikal. Kondisi ini tidak dapat terbaca secara jelas pada pemetaan dari atas yang telah kita buat sebelumnya, namun hal ini dapat dianalisa lebih lanjut pada cc point yang bersangkutan.

3. tipe berhimpit

   Tipe yang ketiga adalah di mana ada CBV yang berbeda namun tidak memiliki pembatas volume solid baik secara vertikal maupun horizontal yang terdeteksi dalam matriks 'peta' sebagaimana yang akan dijelaskan pada bagian berikutnya. Namun yang menjadi pembedanya adalah posisi tinggi dari masing-masing CBV. Kasus semacam ini terjadi misalnya pada impeller.

Metode yang digunakan adalah dengan teknik pengelompokan berdasarkan pola tinggi cc point dan keberadaan volume solid di antara volume CBV. Untuk menerapkan teknik ini, dapat digunakan pemetaan model dari atas untuk memetakan kondisi dari setiap CBV.

Algoritma pengelompokan CBV ini merupakan algoritma analisa terhadap matriks yang dihasilkan dari algoritma sebelumnya, di mana matriks berupa 'peta' kondisi model yang diproyeksikan pada bidang xz (seolah-olah dilihat dari atas). Matriks 'peta' ini terdiri dari 3 layer. Layer pertama berisi indeks-indeks cc point. Layer kedua berisi jumlah CBV yang dikandung pada setiap cc point. Dan layer ketiga berisi koordinat tinggi CBV pada setiap cc point.



40.0000	0	0	0	0	0	0	0	0	0	0	0	0	0
0	43.4894	43.7292	43.9690	43.9818	43.9818	43.9818	43.9818	43.9818	43.9818	43.9818	43.9818	44.0858	44.4738
0	46.8189	47.0308	47.2557	47.4806	47.7055	47.9380	48.1499	48.4389	48.7246	49.1078	49.4264	49.7798	50.3108
0	50.4109	50.7717	51.1254	51.3843	51.7963	52.2554	52.6080	53.1200	53.6320	54.1830	54.7345	55.3369	56.1234
0	54.3522	54.7297	55.2718	55.7201	56.2248	56.8985	57.4185	58.1225	58.8738	59.6483	60.4566	61.3541	62.4314
0	58.4734	59.0454	59.6507	60.3076	60.9646	61.8359	62.5840	63.4766	64.4905	65.5272	66.6165	67.8571	69.2633
0	62.7462	63.5408	64.3133	65.1607	66.0472	67.0837	68.0885	69.2014	70.5018	71.8404	73.2359	74.8684	76.6446
0	67.3954	68.2615	69.2850	70.2680	71.4300	72.6561	73.9482	75.3136	76.9250	78.6061	80.3331	82.4065	84.6460
0	72.2385	73.2865	74.5271	75.6669	77.1119	78.5649	80.1767	81.8270	83.7739	85.8384	87.9224	90.4849	93.2045
0	0	78.6597	80.0506	81.4533	83.1131	84.8348	86.7846	88.7509	91.0448	93.5465	96.0683	99.0283	102.3245
0	0	84.1708	85.8533	87.5536	89.4440	91.4888	93.7780	96.0899	98.7324	101.6708	104.7748	108.0801	112.0076
0	0	90.0766	91.9310	93.9662	96.1070	98.4920	101.1521	103.8420	106.8446	110.2370	113.8826	117.7317	122.4075
0	0	96.2271	98.3180	100.6907	103.0991	105.8403	108.8319	111.9982	115.3704	119.2380	123.4574	127.8930	132.7334
0	0	0	105.0542	107.7006	110.4113	113.5237	116.8594	120.5004	124.4990	128.5769	133.2405	138.3588	143.8499
0	0	0	112.2026	115.1724	118.1037	121.7334	125.3971	129.2038	133.5752	138.2586	143.4275	148.9906	155.1189
0	0	0	0	122.4678	126.0507	129.8176	133.8536	138.1866	143.0793	148.2451	153.4147	159.8750	42.0073
0	0	0	0	130.5028	134.2816	138.3520	142.7435	147.4510	152.6001	158.1341	41.3998	43.3727	45.7217
0	0	0	0	0	142.5041	147.1581	151.7854	156.4930	40.7908	42.6727	44.8098	46.9814	49.4013
0	0	0	0	0	151.1213	155.6849	40.2921	42.0889	43.8707	45.9891	48.2007	50.6360	53.3339
0	0	0	0	0	0	41.4907	43.2299	45.1688	47.1138	49.2917	51.7674	54.4184	57.2086
0	0	0	0	0	0	44.3292	46.2351	48.1964	50.4069	52.7735	55.3153	58.2312	61.3218
0	0	0	0	0	0	0	49.2539	51.4322	53.7634	56.2561	59.0708	62.1088	65.4529
0	0	0	0	0	0	0	0	54.6645	57.1928	59.8930	62.7705	66.0277	69.6384
0	0	0	0	0	0	0	0	57.9928	60.6270	63.5500	66.6701	70.0911	73.8249
0	0	0	0	0	0	0	0	0	64.1772	67.2342	70.5438	74.1397	78.1371
0	0	0	0	0	0	0	0	0	0	71.0134	74.5830	78.3482	82.4698
0	0	0	0	0	0	0	0	0	0	0	78.5426	82.5535	86.8922
0	0	0	0	0	0	0	0	0	0	0	0	86.8052	91.3416
0	0	0	0	0	0	0	0	0	0	0	0	0	95.8138
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

Gambar 3.5 Contoh Matriks ‘Peta’

Algoritma yang dipakai adalah sebagai berikut:

```
Algoritma: Pengelompokan CBV
Input:
  Pengurutan faset dalam cc point (fcdiurutinok)
  Matriks ‘peta’ (index cc point, jumlah CBV dalam cc point, tinggi ccy teratas)
Output: Matriks kelompok CBV (CBVoke)

Tentukan CBV yang aktif (awal→1)
Pemindaian pada matriks peta layer 3:
For banyaknya kolom matriks peta (banyaknya cc point pd sumbu x)
   For banyaknya baris matriks peta (banyaknya cc point pd sumbu z)
      if kolom 1 dan baris 1
         tetapkan sebagai pembanding kiri
      end
      if baris 1
         tetapkan sebagai pembanding atas
      end
      1. Buat perbandingan dengan pembanding kiri
      If ada perbandingan yang cukup besar
         CBV yang aktif + 1
      end
      2. Buat perbandingan dengan pembanding atas
      If ada perbandingan yang cukup besar
         CBV yang aktif + 1
      End
      If terdeteksi volume solid secara horisontal
         CBV yang aktif + 1
      end
      for banyaknya baris pada fctdiurutinok
          masukkan data CBV ke matriks CBV pada CBV yang aktif
          if terdeteksi volume solid secara vertikal
             CBV yang aktif + 1
          End
      end
   end
end
rapikan matriks CBV menjadi CBVoke
```

Penjelasan dari algoritma di atas adalah sebagai berikut:

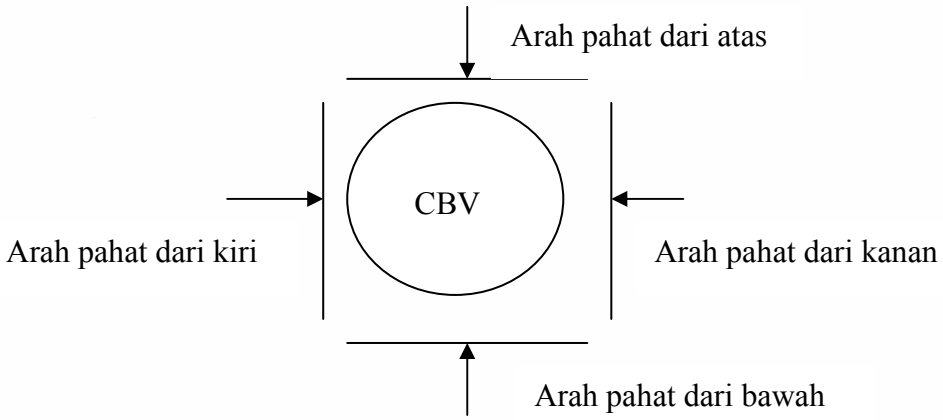
1. Pindai matriks 'peta' yang berisi koordinat tinggi cc point CBV dengan pemindaian per kolom, diikuti pemindaian elemen setiap baris pada kolom tersebut.
2. Tetapkan nilai pertama bukan nol pada kolom sebagai patokan nilai CBV awal
3. Bandingkan nilai setiap elemen dengan nilai elemen di atas atau di sampingnya. Jika terbaca ada perbedaan tinggi yang mencolok, maka akan dibaca sebagai CBV yang berbeda.
4. Jika ditemukan nilai 'nol', yang mana berarti ada volume solid baik secara horisontal maupun vertikal, maka CBV yang terbaca selanjutnya adalah CBV yang berbeda.

Matriks kelompok CBV yang merupakan hasil dari langkah ini terdiri dari layer-layer, di mana setiap layer berisi data cc point pada satu CBV. Berarti jumlah layer pada matriks ini juga menginformasikan jumlah CBV yang terdeteksi pada model.

3.7 Penghitungan Normal CC Point dan Analisa Batas-batas CBV

Setelah kita dapat mengelompokkan data-data CBV ke dalam CBV-CBV yang terpisah, maka data-data pada masing-masing CBV perlu dianalisa batas-batasnya. Hal ini diperlukan untuk menentukan arah masuk pahat ke dalam CBV, karena proses pengerjaan CBV tidak dapat dilakukan dengan arah masuk pahat dari atas seperti yang biasa dilakukan pada proses pemesinan biasa.

Untuk melakukan proses pemesinan pada CBV, kita perlu mengetahui dari arah mana pahat dapat masuk ke CBV tersebut. Seperti telah dijelaskan sebelumnya, bahwa CBV tak dapat dicapai oleh pahat dari posisi atas, namun harus dicapai dengan sudut tertentu dan dari arah tertentu pula. Pada pembuatan algoritma pada penelitian ini, ditentukan 4 arah kemungkinan masuknya pahat ke sebuah CBV, dilihat pada bidang proyeksi. Hal tersebut dapat dilihat pada gambar 3.6 berikut.



Gambar 3.6 Kemungkinan Arah Masuk Pahat ke dalam CBV

Urutan algoritma yang dapat dipakai untuk melakukan hal ini adalah sebagai berikut:

```
Algoritma: Penghitungan Normal cc point dan Analisa Batas-batas CBV
Input:
  Matriks 'peta'
  Matriks kelompok CBV
Output:
  Batas-batas CBV
  Arah masuk pahat

For banyaknya baris pada CBVoke
   1. Cek posisi titik perpotongan cc point pada faset
   2. Hitung vektor normal cc point sesuai posisinya
End

Pilih salah satu CBV
For banyaknya baris pada CBVoke
   If terdeteksi telah pindah kolom pada matriks peta
      Masukkan data koordinat pada batas atas CBV
   else
      Masukkan data koordinat pada batas bawah CBV
   end
end
for banyaknya baris pada CBVoke
   cari elemen kolom pertama pada setiap baris
   masukkan data koordinat pada batas kiri CBV
end
for banyaknya baris pada CBVoke
   cari elemen kolom terakhir pada setiap baris
   masukkan data koordinat pada batas kanan CBV
end
for batas atas, bawah, kiri dan kanan
   for banyaknya elemen pada setiap batas CBV
      cek pada matriks peta, apakah ada volume solid di atas / bawah / kiri/
      kanannya?
      Hitung cc point bebas
   end
end
Hitung persentase cc point bebas berbanding seluruh cc point pada batas
Tentukan batas dengan persentase cc point terbesar
Tentukan arah makan pahat
```

Gambaran singkat dari algoritma di atas adalah sebagai berikut:

1. penentuan cc point – cc point yang berlaku sebagai batas atas, batas bawah, batas kanan dan batas kiri dari sebuah CBV
2. menghitung cc point bebas atau cc point yang dapat dimasuki (tidak terhalang oleh volume solid) pada setiap batas
3. menentukan batas yang memiliki cc point bebas paling banyak berbanding jumlah cc point pada setiap batas sebagai 'gerbang' masuknya pahat ke dalam CBV.
4. menentukan arah masuk pahat pada sebuah CBV

Setelah implementasi dari algoritma ini akan diketahui dari mana pahat seharusnya datang menuju CBV.

3.8 Posisi dan Orientasi Awal Pahat

Sampai pada tahap ini, maka perlu dihitung vektor normal pada setiap cc point dengan rumus pada bab 2.5.3, kemudian kita dapat menghitung orientasi pahat (v_{op}) pada cc point tersebut dengan membuat *cross product* antara vektor normal cc point (n) dengan normal bidang (n_b) yang sejajar dengan arah masuk pahat. Jika arah masuk pahat dari atas atau bawah, maka pengalian dilakukan dengan normal bidang yz, sedangkan jika arah masuk pahat dari arah kanan atau kiri, pengalian dilakukan dengan normal bidang xy.

$$v_{op} = n \text{ x } n_{op} \tag{3.1}$$

Setelah itu dapat ditentukan posisi awal / inisiasi pahat pada proses pengerjaan CBV. Hal tersebut dapat dilakukan dengan algoritma:

```
Algoritma: Posisi dan Orientasi Awal Pahat
Input:
  Batas-batas CBV
  Arah masuk pahat
Output:
  Posisi dan Orientasi Awal Pahat

For banyaknya elemen pada batas sebagai arah masuk pahat
   Tentukan baris terluar
   Hitung elemen pada baris terluar
   For banyaknya elemen pada baris terluar batas
      Tentukan titik tengah
      Tentukan posisi inisiasi pahat pada bagian atas CBV
   End
end
```

Penjelasannya:

1. menentukan cc point – cc point dalam batas yang ditentukan sebagai arah makan pahat (bisa batas atas, batas bawah, batas kanan maupun batas kiri) yang berada pada baris terluar
2. menentukan titik tengah dari cc point terluar tadi sebagai koordinat awal mata pahat
3. menentukan orientasi pahat dengan cara yang dijelaskan pada paragraf sebelumnya.

3.9 Pembentukan Lintasan dan Orientasi Pahat

Terdapat beberapa pilihan pola lintasan pahat yang dapat digunakan. Pada laporan ini penulis menggunakan pola lurus, karena pola ini dapat dibuat dengan algoritma yang cukup sederhana. Selain itu dengan pola ini kita dapat dengan mudah melakukan pengecekan terhadap cc point yang sudah dilalui oleh pahat sehingga kita dapat mengecek secara manual jika ada cc point yang terlewatkan, atau jika ada error dalam perjalanan pahat.

Algoritma yang digunakan adalah sebagai berikut:

```
Algoritma : Pembentukan Lintasan dan Orientasi Pahat
Input:
  Matriks kelompok CBV
  Matriks 'peta'
Output:
  Daftar lintasan dan orientasi pahat

Buat toolpath dengan dua koordinat awal
While masih ada cc point yang belum terkena pahat
   Cek apakah masih ada cc point yang belum termakan pada baris yang aktif?
   If ya
      Lakukan perpindahan pahat ke kanan atau ke kiri:
      If cc point bukan yg paling kanan dan belum pernah mentok kanan
         1. Gerak makan ke kanan, masukkan data koordinat ke toolpath
         2. Hitung dan masukkan orientasi pahat ke toolpath
         3. Catat cc point yang sdh termakan pada baris tersebut
         4. Catat cc point yang sudah termakan pada CBV
      elseif cc point bukan yg paling kanan dan sudah pernah mentok kanan
         1. Gerak makan ke kiri, masukkan data koordinat ke toolpath
         2. Hitung dan masukkan orientasi pahat ke toolpath
         3. Catat cc point yang sdh termakan pada baris tersebut
         4. Catat cc point yang sudah termakan pada CBV
      elseif cc point adalah cc point paling kanan
         1. cari cc point yg belum termakan pahat di sebelah kiri
         2. Gerak makan ke kiri, masukkan data koordinat ke toolpath
         3. Hitung dan masukkan orientasi pahat ke toolpath
         4. Catat cc point yang sdh termakan pada baris tersebut
         5. Catat cc point yang sudah termakan pada CBV
      end
  elseif cc point pada baris sudah habis
      cek apakah cc point di atas masuk dalam CBV?
      If ya
         1. Gerak makan maju / ke atas, masukkan data koordinat ke toolpath
         2. Hitung dan masukkan orientasi pahat ke toolpath
         3. Catat cc point yang sdh termakan pada baris tersebut
         4. Catat cc point yang sudah termakan pada CBV
      elseif tidak
         1. cari cc point yg termasuk CBV pada baris atas
```

```
            2. Gerak makan ke atas, masukkan data koordinat ke toolpath
            3. Hitung dan masukkan orientasi pahat ke toolpath
            4. Catat cc point yang sdh termakan pada baris tersebut
            5. Catat cc point yang sudah termakan pada CBV
        end
    end
    lakukan gerak makan atas/bawah:
    If pahat ada di atas
        1. Gerak makan ke bawah, masukkan data koordinat ke toolpath
        2. Hitung dan masukkan orientasi pahat ke toolpath
        3. Catat cc point yang sdh termakan pada baris tersebut
        4. Catat cc point yang sudah termakan pada CBV
     elseif pahat di bawah
       1. gerak makan ke atas, masukkan data koordinat ke toolpath
       2. hitung dan masukkan orientasi pahat ke toolpath
       3. catat cc point yang sudah termakan pada baris tersebut
       4. catat cc point yang sudah termakan pada CBV
     end
end
```

Penjelasan dari algoritma di atas:

1. Apakah seluruh cc point dalam CBV sudah termakan pahat?
2. Jika belum, apakah seluruh cc point dalam baris yang sedang dikerjakan sudah terkena pahat?
3. Jika nomor 2 jawabannya belum, maka apakah cc point yang sedang dikerjakan adalah cc point yang paling kanan dalam baris tersebut?
4. Jika nomor 3 tidak, dan pahat belum pernah mentok kanan dalam baris tersebut, maka pahat melakukan perpindahan ke kanan.
5. Jika nomor 3 tidak, namun pahat sudah pernah mentok kanan, maka pahat melakukan perpindahan ke kiri.
6. Jika nomor 3 ya, maka pahat mencari cc point yang belum dimakan ke sebelah kiri, dan melakukan perpindahan ke sana.
7. Jika nomor 2 ya, maka apakah cc point di atas cc point yang sedang dikerjakan sekarang merupakan bagian dari CBV?
8. Jika nomor 7 ya, maka pahat melakukan perpindahan gerak makan ke cc point di atasnya.
9. Jika nomor 7 tidak, maka dicari cc point yang merupakan bagian dari CBV pada baris di atas, kemudian pahat berpindah ke sana.
10. Jika nomor 1 ya, stop iterasi.