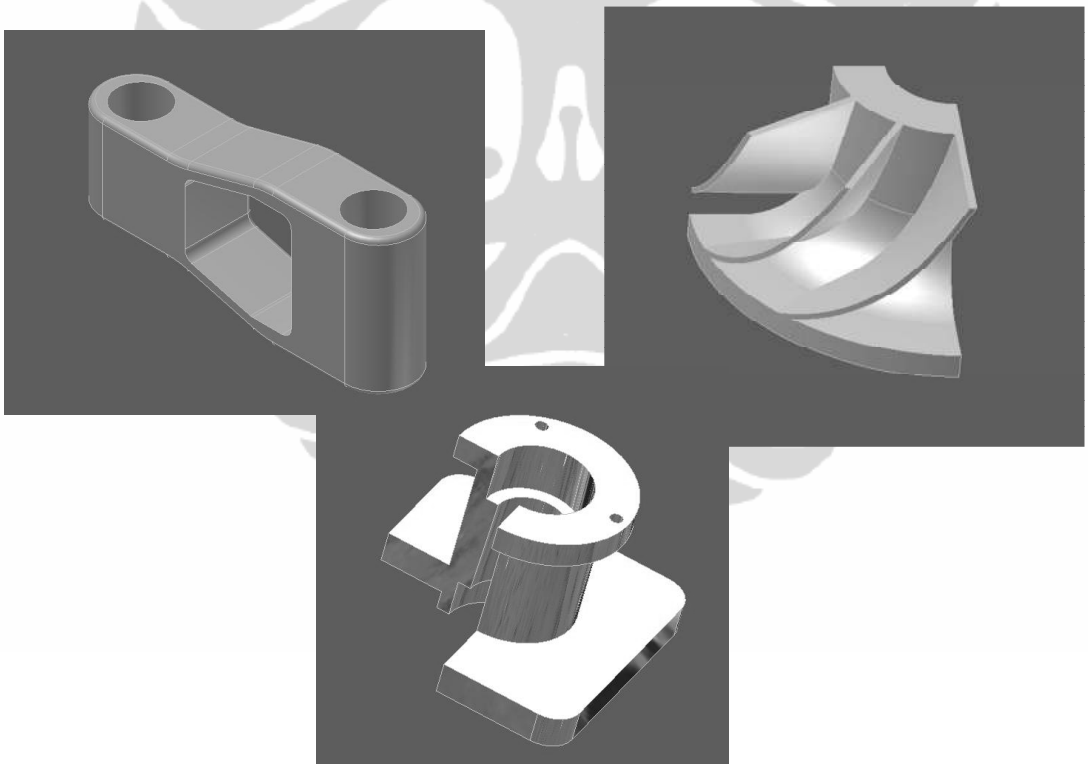
BAB 4
HASIL DAN ANALISA

Banyak komponen mesin yang memiliki bentuk yang cukup kompleks. Setiap komponen tersebut bisa jadi memiliki CBV, permukaan yang berkontur dan fitur-fitur lainnya. Untuk bagian implementasi dari penelitian ini, diambil sebuah contoh kasus penerapan algoritma terhadap impeller sebagai model yang berbentuk kompleks. Impeller dikenal sebagai salah satu part yang memiliki bentuk geometri yang rumit dan sulit untuk dibuat dengan proses pemesinan. Pada impeller terdapat CBV yang juga sulit untuk dimesin menggunakan mesin NC biasa, terlebih lagi untuk membuat G-code-nya.

Untuk keperluan ini, maka dibuat sebuah model seperempat bagian impeller dengan software Autodesk Inventor, yang nantinya akan diubah ke dalam model faset yang disimpan dalam file STL.

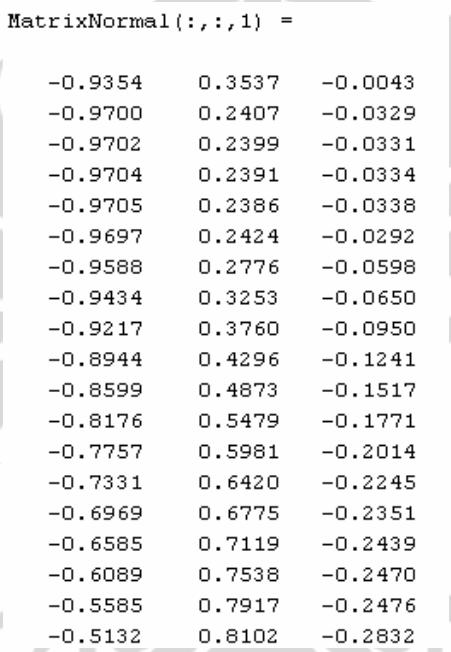


Gambar 4.1 Macam-macam Komponen dengan Bentuk Kompleks

4.1 Pengambilan Data, Validasi dan Pembuatan Struktur Data

Program ini telah berhasil melakukan pengambilan data dari file STL sekaligus melakukan validasi apakah file tersebut benar-benar STL atau bukan. Setelah itu data yang sudah diambil tadi disusun sesuai dengan struktur data seperti yang sudah dijelaskan pada bab 3.

Berikut ini adalah hasil dari struktur data faset yang berupa matriks dengan 2 layer. Setiap baris merupakan indeks faset. Layer pertama berisi vektor normal dengan format i, j dan k, sebagaimana pada gambar berikut.



```
MatrixNormal(:,:,1) =

   -0.9354    0.3537   -0.0043
   -0.9700    0.2407   -0.0329
   -0.9702    0.2399   -0.0331
   -0.9704    0.2391   -0.0334
   -0.9705    0.2386   -0.0338
   -0.9697    0.2424   -0.0292
   -0.9588    0.2776   -0.0598
   -0.9434    0.3253   -0.0650
   -0.9217    0.3760   -0.0950
   -0.8944    0.4296   -0.1241
   -0.8599    0.4873   -0.1517
   -0.8176    0.5479   -0.1771
   -0.7757    0.5981   -0.2014
   -0.7331    0.6420   -0.2245
   -0.6969    0.6775   -0.2351
   -0.6585    0.7119   -0.2439
   -0.6089    0.7538   -0.2470
   -0.5585    0.7917   -0.2476
   -0.5132    0.8102   -0.2832
```

Gambar 4.2 MatrixNormal Layer I

Dan pada layer kedua berisi indeks vertex yang membentuk faset pada masing-masing baris.

```
MatrixNormal(:,:,2) =

     1     2     3
     1     4     2
     1     5     4
     1     6     5
     1     7     6
     1     8     7
     7     8     9
    10     9    11
    12    11    13
    14    13    15
    16    15    17
    18    17    19
    20    19    21
    22    21    23
    24    23    25
    26    25    27
```

Gambar 4.3 MatrixNormal Layer II

Sedangkan struktur data vertex dapat dilihat contohnya pada gambar 4.4 dan gambar 4.5

4.2 Proses Normalisasi

Proses normalisasi juga telah berjalan dengan baik, dan menghasilkan seperti yang dicontohkan ilustrasi berikut.

```
Matrixvertex(:,:,1) =

   13.0000   13.0523   13.0000
   13.0000   13.0523   13.0686
   13.0523   13.0686   13.1253
   13.0523   13.0000   13.0686
   13.0523   13.1253   13.0686
   13.0686   13.1253   13.1692
   13.1253   13.1692   13.2183
   13.1253   13.0686   13.1692
   13.1253   13.2183   13.1692
   13.1692   13.2183   13.3002
   13.2183   13.3002   13.3304
   13.2183   13.1692   13.3002
   13.2183   13.3304   13.3002
```

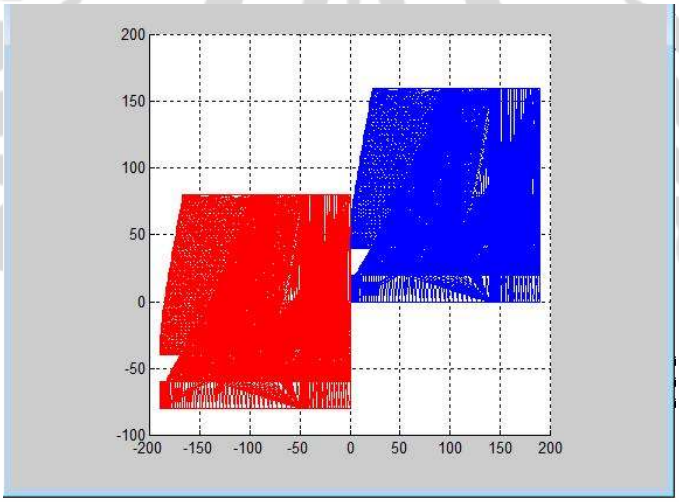
Gambar 4.4 Matriks sebelum normalisasi

Dan setelah proses normalisasi dapat kita lihat sebagai berikut:

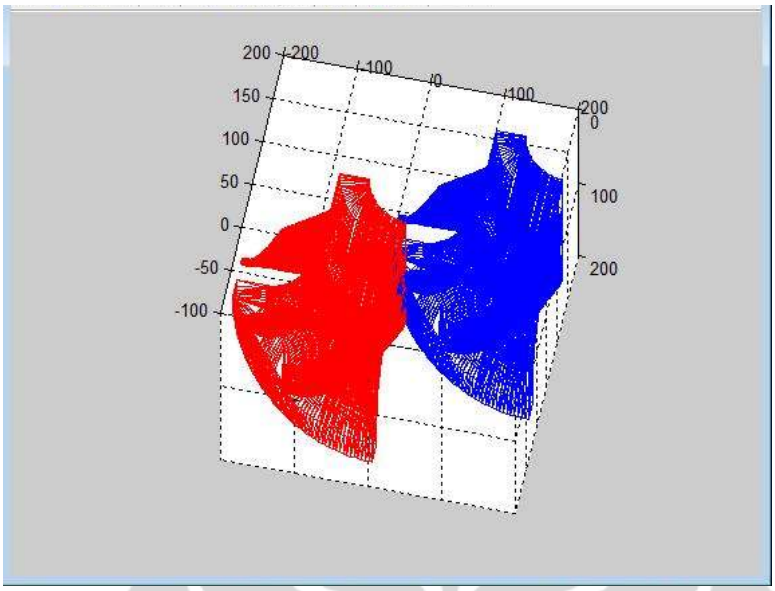
```
Matrixvertex(:,:,1) =

         0    0.0523         0
         0    0.0523    0.0686
    0.0523    0.0686    0.1253
    0.0523         0    0.0686
    0.0523    0.1253    0.0686
    0.0686    0.1253    0.1692
    0.1253    0.1692    0.2183
    0.1253    0.0686    0.1692
    0.1253    0.2183    0.1692
    0.1692    0.2183    0.3002
    0.2183    0.3002    0.3304
    0.2183    0.1692    0.3002
    0.2183    0.3304    0.3002
```

Gambar 4.5 Matriks sesudah normalisasi

Hasil dari proses penggambaran proses ini adalah sebagai berikut:



Gambar 4.6 Hasil Plot Model sebelum normalisasi (merah) dan sesudah normalisasi (biru)



Gambar 4.7 Hasil plot model pada sumbu Cartesian 3 dimensi

Dari kedua gambar di atas kita dapat melihat bahwa algoritma dan penulisan syntax sudah berjalan sebagaimana yang diharapkan. Dalam penggambaran impeller dengan resolusi tinggi seperti di atas, terdapat 2.468 faset untuk membentuk model tersebut. Dengan jumlah data yang sedemikian besar, maka jalannya program memang menjadi cukup berat dan memakan waktu.

4.3 Proses Bucketing

Hasil dari algoritma proses bucketing adalah matriks bucket dengan ukuran a x b, di mana a adalah jumlah bucket dilihat dari sumbu x, dan b adalah jumlah bucket dari sumbu z. Contoh dari matriks tersebut adalah sebagai berikut.

```
>> bucket(:,:,1)

ans =

  Columns 1 through 8

       964       963       963       963       963       963       963
       963       963       963       963       963       963       963
       975       975       975       975       975       975       975
       975       976       976       976       976       976       976
       976       976       976       977       977       977       977
       955       955       955       955       955       955       955
       955       951       951       951       951       951       931
       951       951       931       931       931       931       586
       931       931       503       503       503       560       586
         0       503       503       503       503       560       573
         0       504       503       503       503       559       559
         0       504       502       502       532       532       559
         0       638       502       502       502       531       531
         0         0       501       501       501       531       531
         0         0       501       501       501       533       531
         0         0         0       500       500       535       535
```

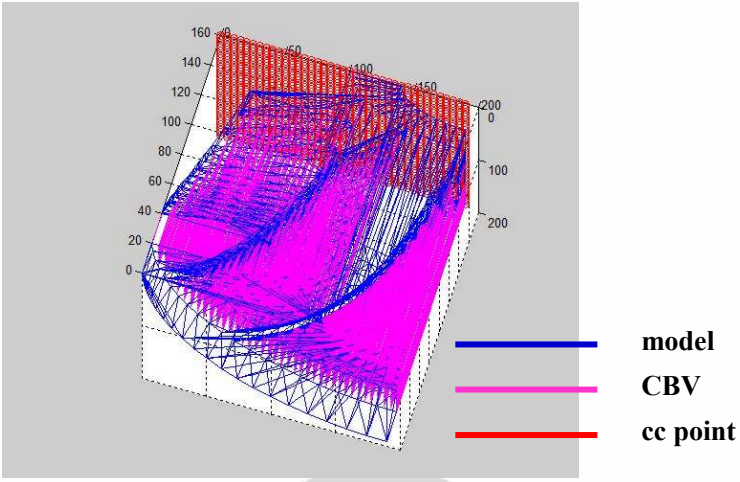
Gambar 4.8 Matriks Bucket

Matriks ini terdiri dari beberapa layer. Setiap elemen matriks menginformasikan isi dari setiap bucket yang ada, yaitu indeks faset, sedangkan elemen pada layer-layer berikutnya (pada posisi elemen yang sama) menginformasikan indeks faset berikutnya yang berada pada bucket yang sama. Namun susunan indeks faset ini belum terurut berdasarkan posisi secara geometris, sehingga pada proses selanjutnya indeks-indeks faset ini perlu diurutkan.

4.4 Pendeteksian CBV

Pendeteksian CBV merupakan bagian yang krusial dari keseluruhan penelitian ini. Algoritma yang dibuat diharapkan dapat mendeteksi CBV secara otomatis dan mendapatkan informasi yang cukup dari CBV tersebut untuk dilaksanakan proses pengerjaan pemesinan.

Dengan cara seperti yang dijelaskan di atas, kita dapat mendeteksi seluruh cc point yang mengandung CBV dalam sebuah model. Hal ini dapat kita lihat dalam gambar 4.9 sebagai visualisasi CBV (yang ditandai dengan warna magenta).



Gambar 4.9 Pendeteksian Seluruh CBV pada Model

Pendeteksian dengan metode PNVB seperti yang telah dijelaskan pada bab 3 terbukti dapat mendeteksi keberadaan CBV pada sebuah *faceted model*. Semua cc point dalam model yang memiliki pasangan vektor dengan tipe *head to head* akan diklaim sebagai cc point yang mengandung CBV. Kemudian setiap cc point ini akan disimpan dalam struktur data tersendiri yang memuat informasi-informasi yang dibutuhkan untuk proses selanjutnya, antara lain:

1. Koordinat x, y, dan z dari cc point. Perlu diingat bahwa pada setiap cc point yang memiliki CBV akan memiliki 2 nilai koordinat y (tinggi) yaitu koordinat tinggi bagian atas CBV dan bagian bawahnya, seperti terlihat pada gambar 4.9.
2. Kode yang merujuk kepada data vektor normal facet yang berpotongan dengan CBV, termasuk kode yang menunjukkan posisi titik perpotongan pada faset, apakah berada di dalam faset, berpotongan pada sisi, atau berpotongan tepat pada vertex. Hal ini diperlukan untuk perhitungan orientasi pahat pada cc point yang dimaksud.
3. Kode yang merujuk posisi cc point pada matriks 'peta' yang akan digunakan pada proses selanjutnya.

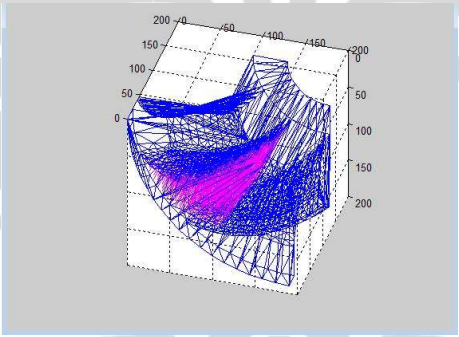
Penggunaan metode PNVB membuat proses komputasi pendeteksian CBV menjadi sederhana. Adapun kesalahan-kesalahan dapat terjadi karena adanya proses triangulasi yang kurang sempurna pada tahap pemodelan awal. Selain itu, hasil dari metode PNVB ini baru pada tahap mengumpulkan semua cc point yang mengand

ung CBV di dalamnya. Ia belum dapat secara langsung mengelompokkan setiap cc point tadi ke dalam CBV-nya masing-masing, apalagi untuk menganalisa bentuk dari CBV tersebut. Untuk mengelompokkan dan menganalisa cc point yang mengandung CBV tadi diperlukan algoritma yang berbeda.

4.5 Pengelompokan CBV

Pada gambar 4.9 di atas kita dapat melihat tiga CBV dalam model potongan impeller. Langkah berikutnya adalah mengelompokkan setiap cc point ke dalam CBV yang berlainan, agar setiap CBV dapat dianalisa masing-masing.

Dengan demikian, masing-masing cc point dapat dipisahkan menurut CBV masing-masing. Proses pengelompokkan CBV ini telah dapat berjalan dengan baik. Visualisasi dari salah satu CBV tersebut adalah seperti yang terlihat pada gambar 4.10 berikut.

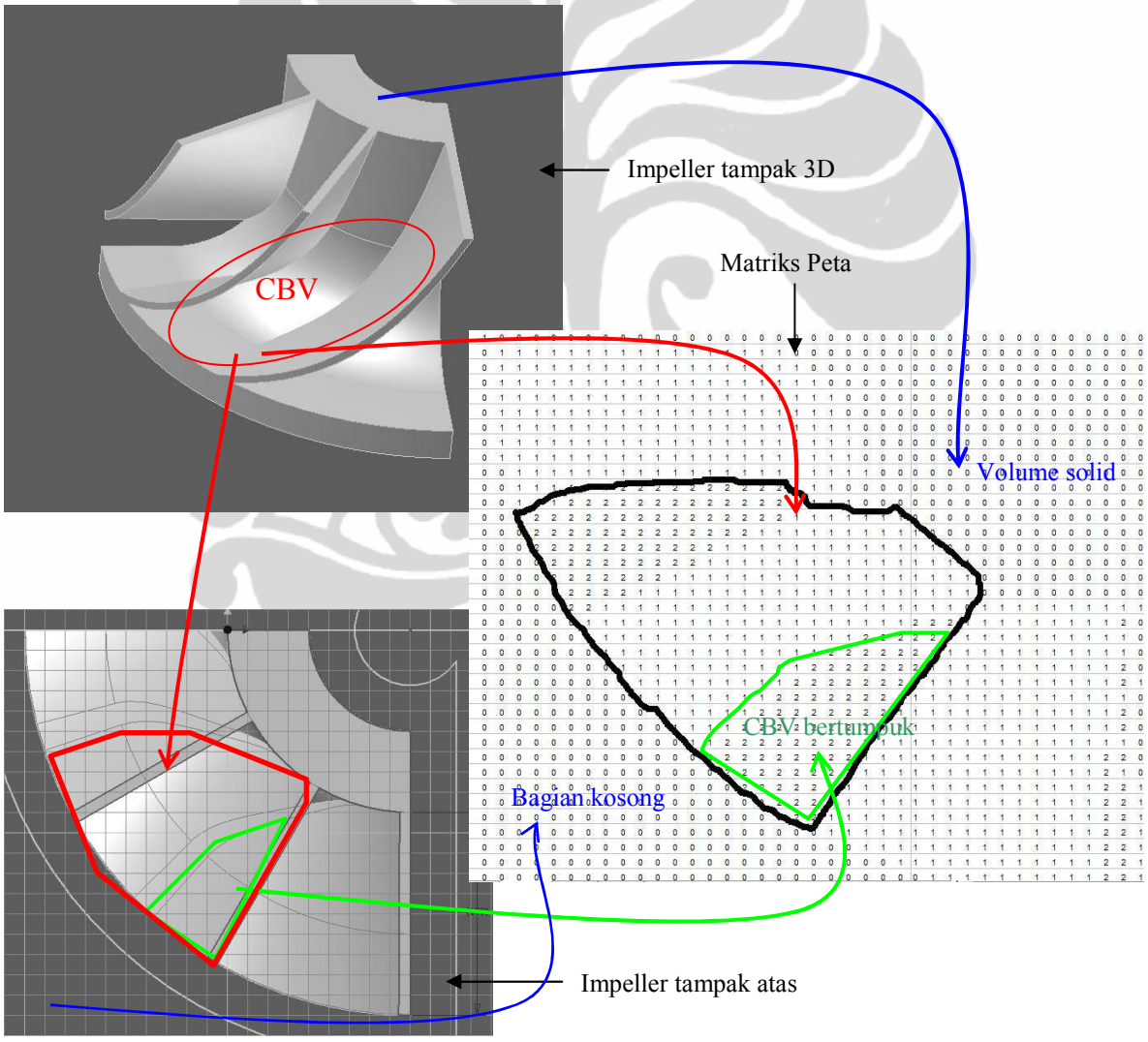


Gambar 4.10 Visualisasi Salah Satu CBV Hasil Pengelompokan CBV

Pengelompokan ini didapat dari algoritma sebagaimana yang sudah dijelaskan pada bab 3 yang lalu, bahwa setiap CBV dikelompokkan berdasarkan kecenderungan tinggi cc point, dan berdasarkan adanya volume solid yang membatasi antara satu CBV dengan CBV lainnya, baik secara horizontal maupun vertikal.

Cc point – cc point yang ada dalam CBV dapat kita petakan ke dalam sebuah matriks ‘peta’, yang mana setiap elemen matriks melambangkan satu cc point yang ada pada model. Dan hasil dari tahap pengelompokan ini adalah struktur data yang baru, yaitu sebuah matriks 3 dimensi di mana setiap layernya berisi data setiap CBV, sehingga jumlah layer pada matriks ini juga menginformasikan jumlah CBV yang terdeteksi pada model.

Pada laporan ini, dengan studi kasus potongan impeller, penulis memilih salah satu CBV pada model untuk dianalisis lebih lanjut. Dalam gambar 4.10 dapat kita lihat CBV bagian tengah yang diambil sebagai contoh, karena memiliki bentuk yang paling representatif dari CBV yang dimiliki oleh sebuah impeller. Luasan CBV yang terpetakan pada matriks 'peta' dapat kita lihat pada gambar 4.11. Bagian yang diberi garis hitam (ditunjuk dengan panah merah) adalah bentuk CBV tampak atas yang akan kita analisa lebih lanjut untuk dibuat lintasan pahatnya. Pada gambar 4.11 tersebut terlihat bahwa pada impeller, antara CBV yang satu dengan CBV yang lainnya terdapat bagian-bagian yang bertumpuk, ada pula bagian yang berhimpit.

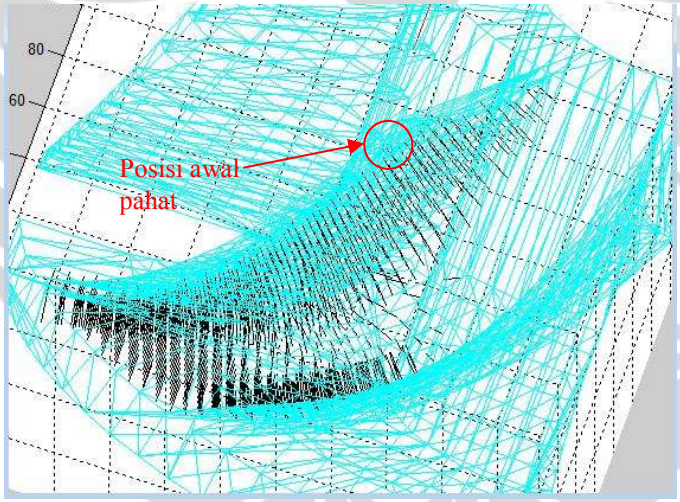


Gambar 4.11 Pemetaan cc point CBV dalam Matriks

Pada bagian-bagian yang hanya memiliki 1 CBV, maka angka yang terbaca pada matriks 'peta' adalah angka 1, sedangkan untuk bagian-bagian yang memiliki CBV lebih dari satu (bertumpuk), maka angka dalam matriks 'peta' akan menunjukkan jumlah CBV yang ada pada cc point tersebut, sebagaimana yang ditunjukkan area hijau. Sedangkan untuk bagian yang ditunjukkan oleh panah biru adalah adalah bagian yang berisi volume solid saja (tidak ada CBV) atau bagian yang kosong. Pada matriks 'peta' bagian ini akan terbaca dengan angka nol. Syntax penulisan bagian pengelompokan ini dapat dilihat pada bagian lampiran.

4.6 Penghitungan Vektor Normal CC Point pada CBV

Berikutnya adalah penghitungan vektor normal pada setiap cc point. Implementasi algoritma sebagaimana yang telah dijelaskan pada bab 3 memberikan hasil visualisasi seperti pada gambar 4.12.



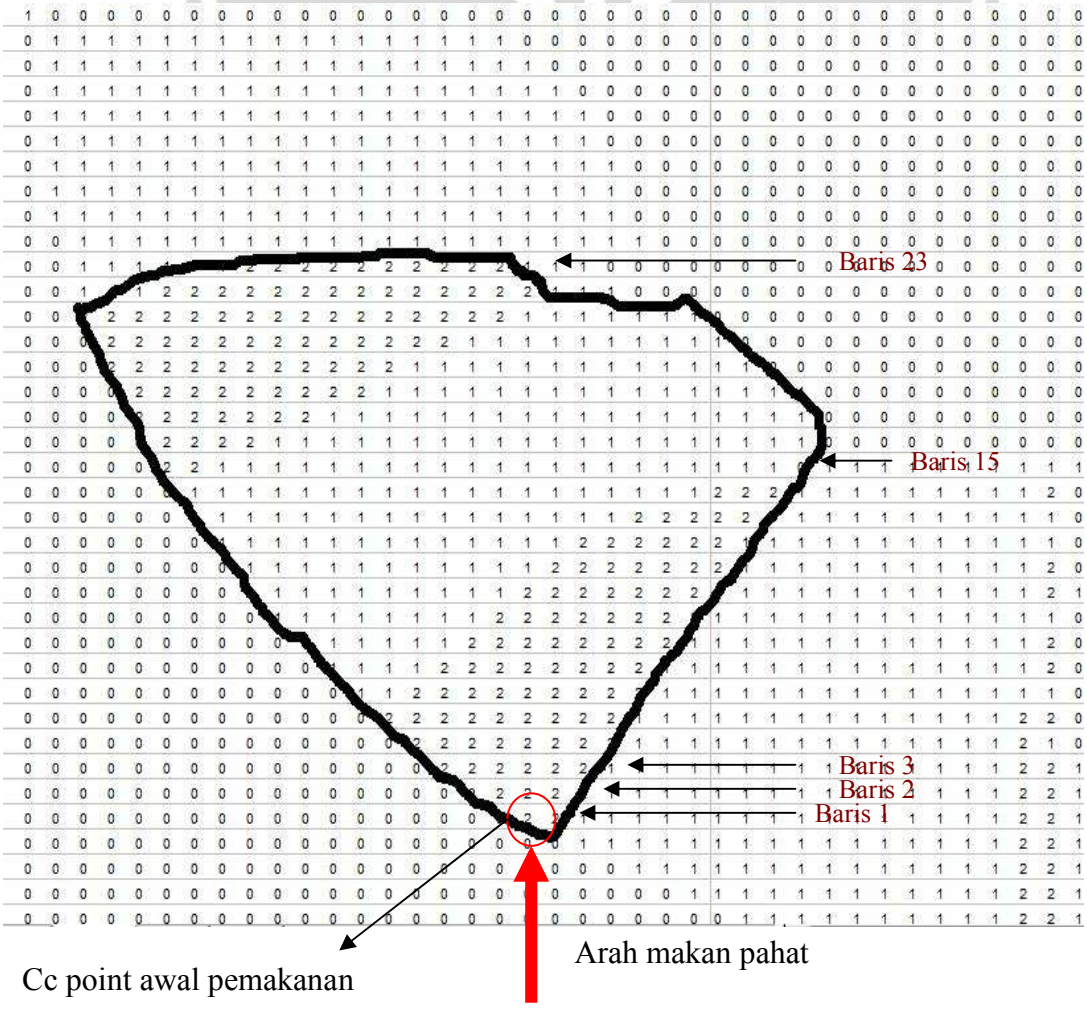
Gambar 4.12 Visualisasi Vektor Normal pada cc point CBV

Formula penghitungan vektor normal pada setiap titik perpotongan cc point dengan faset dipilih berdasarkan kode yang ada pada matriks CBV, apakah titik perpotongan tersebut berada di dalam faset, berpotongan pada sisi, ataukah berpotongan pada vertex? Berdasarkan informasi tersebut kemudian dilakukan pendataan nilai-nilai vektor normal yang berada di sekeliling titik perpotongan, untuk kemudian dihitung resultan normalnya. Seluruh vektor normal telah dapat dihitung pada setiap cc point dalam CBV. Data nilai vektor normal ini nantinya akan

digunakan untuk menghitung orientasi vektor pada cc point yang bersangkutan. Adapun orientasi pahat yang berada di antara dua buah cc point, maka orientasinya akan diperoleh dari hasil interpolasi antara dua vektor normal pada cc point .

4.7 Penentuan Arah Makan Pahat pada CBV dan Posisi Awal Pahat

Pada gambar 4.12 dapat dilihat lingkaran biru (yang dilingkari dengan lingkaran merah). Titik tersebut adalah titik awal pahat untuk memulai pemakanan pada CBV. Berdasarkan analisa terhadap CBV, maka pada studi kasus CBV impeller ini algoritma memutuskan bahwa arah makan pahat adalah dari arah bawah, karena sisi batas bagian bawah CBV merupakan sisi yang paling banyak memuat cc point bebas, yaitu tidak terhalang oleh volume solid. Hal ini akan memudahkan pahat untuk mencapai semua bagian dalam CBV.

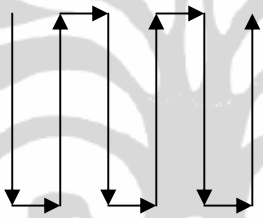


Gambar 4.13 Arah Makan dan Posisi Awal Pahat

Pada gambar 4.13 terlihat elemen matriks 'peta' yang dilingkari lingkaran merah, yang menunjukkan cc point yang menjadi posisi awal pahat. Menurut default pada algoritma ini, posisi awal pahat akan berada pada bagian atas sebuah CBV. Orientasi awal pahat langsung didapatkan dari hasil *cross product* antara vektor normal cc point dengan normal bidang yang sejajar dengan arah makan pahat.

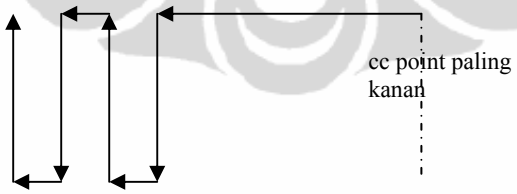
4.8 Pembuatan Lintasan dan Orientasi Pahat

Dengan adanya posisi inisiasi untuk proses pemakanan seperti gambar 4.13 di atas, maka proses pembuatan lintasan dan orientasi pahat dapat dimulai. Dengan menggunakan algoritma pada bab 3 yang lalu (penulisan syntax dapat dilihat pada bagian lampiran), maka secara umum gerakan pahat dalam baris yang sama adalah sebagai berikut:



Gambar 4.14 Alur Lintasan Pahat

Jika pahat telah mencapai cc point paling kanan, sedangkan masih ada cc point yang belum terkena pahat pada baris yang sama, maka pahat akan bergerak dengan pola sebagai berikut:

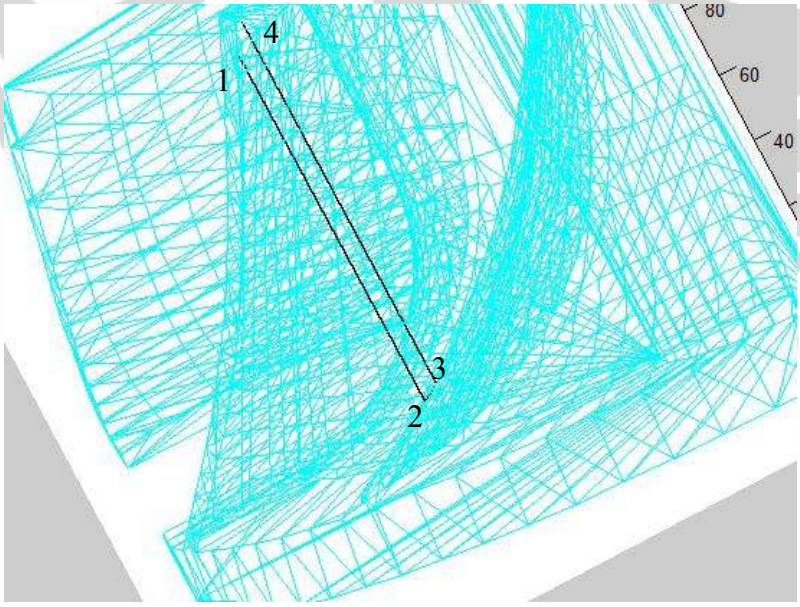


Gambar 4.15 Alur Pahat setelah Mencapai Posisi Paling Kanan

Pada bab sebelumnya dijelaskan bahwa pola yang dipilih untuk pemesinan CBV pada penelitian adalah pola pemesinan lurus, dengan gerak makan mengikuti pola cc point yang ada. Setelah pahat berada pada posisi inisiasi, langkah berikutnya adalah melakukan gerak makan ke bawah sampai pada bagian bawah cc point. Setelah

itu, pahat akan bergerak ke kanan berpindah ke cc point berikutnya di baris yang sama (baris pada matriks 'peta'), kemudian melakukan gerak makan ke atas sampai bagian atas cc point. Sampai pada langkah ini, karena algoritma telah mendeteksi bahwa seluruh cc point pada baris ini telah habis, maka pahat akan bergerak maju ke baris berikutnya. Secara default, pahat akan berpindah ke kanan, dan setelah mencapai cc point paling kanan pada baris ini, jika masih ada cc point yang belum termakan, pahat akan bergerak ke kiri. Setelah seluruh cc point pada baris termakan, pahat akan kembali bergerak maju ke baris berikutnya dan demikian seterusnya, sampai seluruh cc point dalam CBV sudah termakan oleh mata pahat, maka iterasi algoritma pembentukan lintasan dan orientasi pahat ini akan berhenti. Keluaran dari tahap ini adalah sebuah daftar koordinat dan vektor orientasi pahat. Matriks data lintasan pahat inilah nantinya yang diharapkan dapat menjadi input untuk pembentukan G-code pada pemesinan NC.

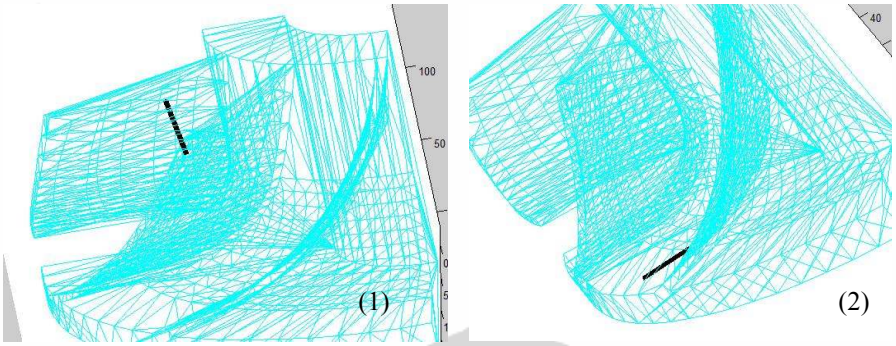
Berikut ini adalah visualisasi jalan pahat pada baris pertama proses pemakanan pada CBV di baris 1 (lihat gambar 4.13).



Gambar 4.16 Alur Pahat pada Baris 1

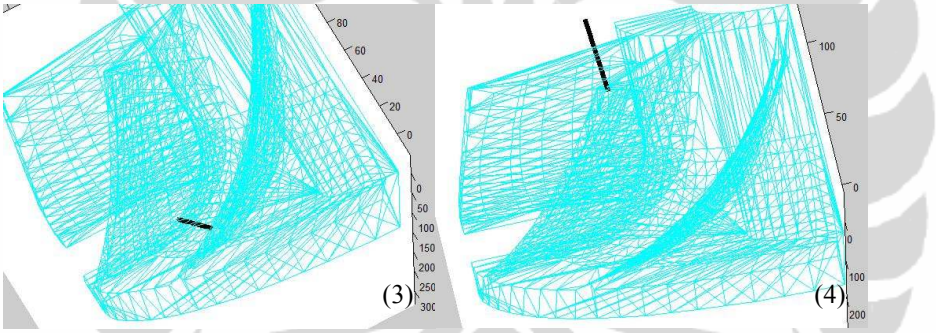
Titik 1 adalah titik atau posisi inisiasi pahat, kemudian melakukan gerak makan ke bawah (ke titik 2), berpindah ke kanan (ke titik 3) dan akhirnya melakukan gerak makan ke atas (ke titik 4). Sampai di sini seluruh cc point pada baris pertama sudah terkena pahat.

Orientasi pahat pada titik 1 dan 2 berdasarkan hasil perhitungan cross product adalah sebagai berikut.



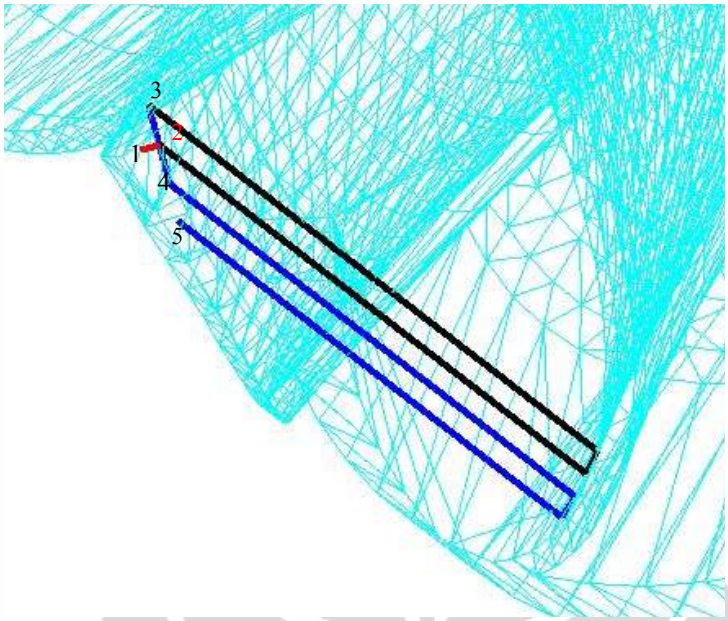
Gambar 4.17 Orientasi Pahat pada Titik 1 dan 2

Sedangkan pada titik 3 dan 4 adalah sebagai berikut



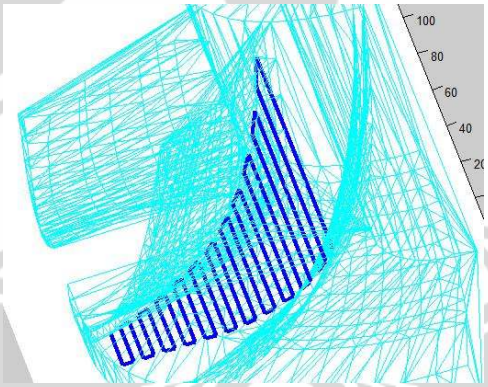
Gambar 4.18 Orientasi Pahat pada Titik 3 dan 4

Berikutnya adalah alur pahat pada baris 2 (lihat gambar 4.13). Titik 1 pada gambar di bawah adalah titik 4 pada gambar sebelumnya. Garis merah menunjukkan gerak maju pahat menuju titik 2 untuk melakukan pemakanan pada baris berikutnya. Dari titik 2 pahat menuju titik 3 mengikuti alur yang berwarna hitam. Titik 3 merupakan cc point paling kanan dari baris kedua, namun masih ada cc point lain yang belum terkena pahat, sehingga pahat kemudian bergerak ke kiri menuju titik 4, dan menyelesaikan pemakanan pada baris kedua sampai titik 5 mengikuti alur garis yang berwarna biru gelap.



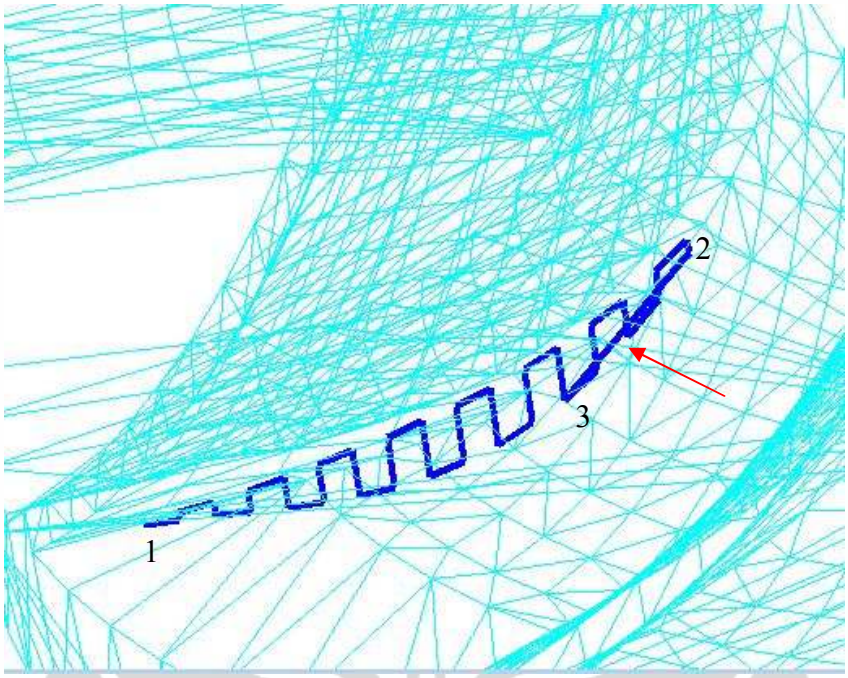
Gambar 4.19 Alur Pahat pada Baris 2

Sedangkan yang berikut adalah alur pahat pada baris ke-15.



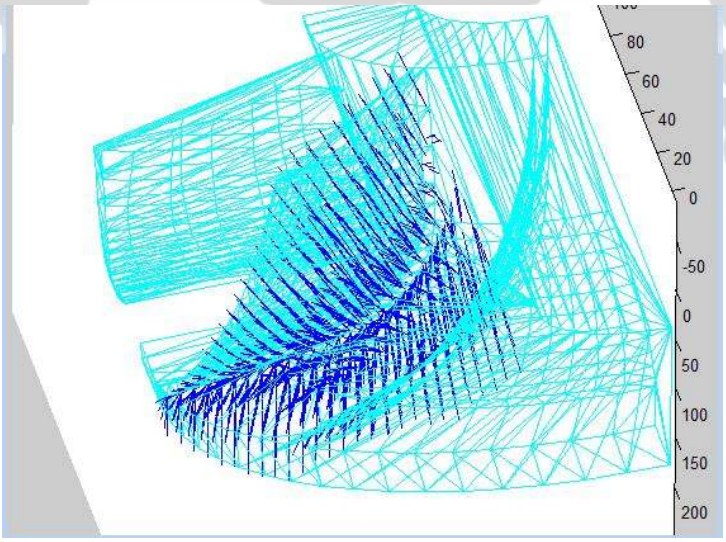
Gambar 4.20 Alur Pahat Baris ke-15

Gambar 4.21 berikutnya memperlihatkan alur pahat pada baris ke-22 (dari 23 baris yang ada pada CBV ini). Dimulai pada titik 1, kemudian mengikuti alur sampai titik 2. Garis dari titik 2 sampai titik 3 (yang ditunjuk oleh garis merah) disebabkan karena pada saat pahat hendak berpindah dari baris ke-22 ke baris ke-23, cc point di depan titik 2 tidak termasuk CBV, sehingga algoritma mencari cc point yang di depannya terdapat cc point yang termasuk CBV. Karena itu alur pahat bergerak dari titik 2 menuju titik 3. Perlu ditegaskan sekali lagi bahwa alur pahat yang dibuat pada penelitian ini belum memperhitungkan adanya gouging.



Gambar 4.21 Alur pahat pada baris ke-22

Hasil visualisasi seluruh posisi dan orientasi pahat dalam CBV seperti pada gambar 4.22 berikut.



Gambar 4.22 Visualisasi Posisi dan Orientasi Pahat dalam CBV

Demikian seterusnya hingga seluruh cc point dalam CBV terkena pahat. Algoritma yang dipakai telah dapat membuat pahat bergerak melakukan pemakanan pada semua cc point yang ada, walaupun pada tahap ini algoritma belum mempertimbangkan adanya *gouging* / *collision* antara pahat dengan model. Titik kontak antara pahat dengan cc point terletak pada titik center pada permukaan bawah pahat, sehingga toolpath yang dihasilkan nantinya juga perlu menyesuaikan hal ini (algoritma yang dibuat sekarang belum memasukkan faktor *offset*). Ada pun daftar lintasan dan orientasi pahat (*toolpath*) secara lengkap dapat dilihat pada lampiran.