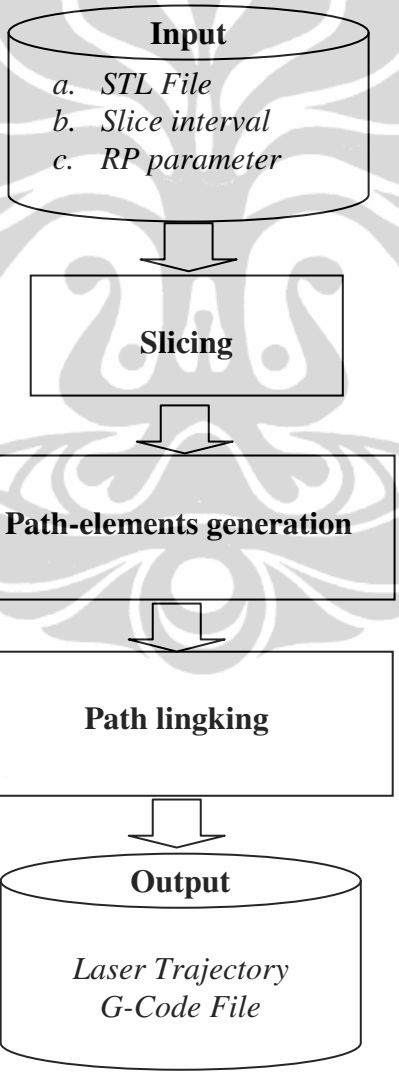
BAB III
ALGORITMA PEMBUATAN *LASER TRAJECTORY*

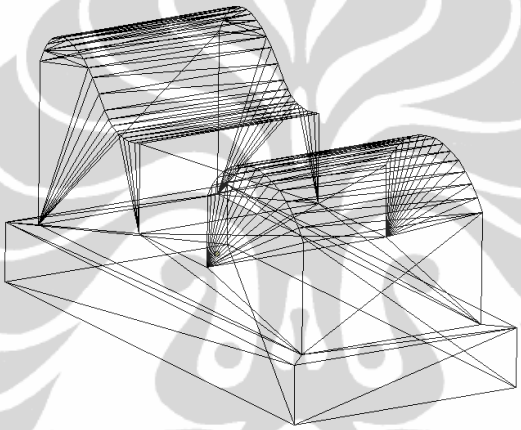
3.1 Algoritma *Generate Laser Trajectory*

Algoritma perencanaan untuk membuat *laser trajectory* model prismatik dan berkontur ini dilakukan dengan tahapan algoritma pada gambar III.1.



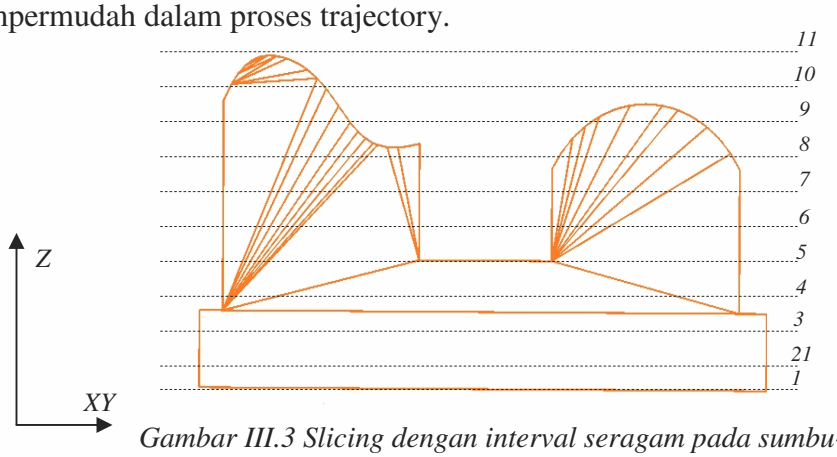
Gambar 3.1 : Algoritma laser trajectory

Tahapan algoritma pembuatan *laser trajectory* dimulai dengan mengambil File STL yang berisi informasi menganai data *facet* normal dan tiga vertex pembentuk masing-masing segitiga. File STL dapat diperoleh dari software-software CAD yang ada, seperti Unigraphic, SolidWork, Catia, atau AutoCad. Model yang dibuat harus tegak kea rah sumbu-Z untuk mempermudah pembuatan. Selain File STL, informasi lain yang berkaitan dengan parameter RP seperti *layer thickness* dan *hatch space* perlu disiapkan. Besar kecilnya nilai parameter tersebut akan mempengaruhi proses pembuatan, karena akan banyak terbentuk titik potong pada segitiga *facet*

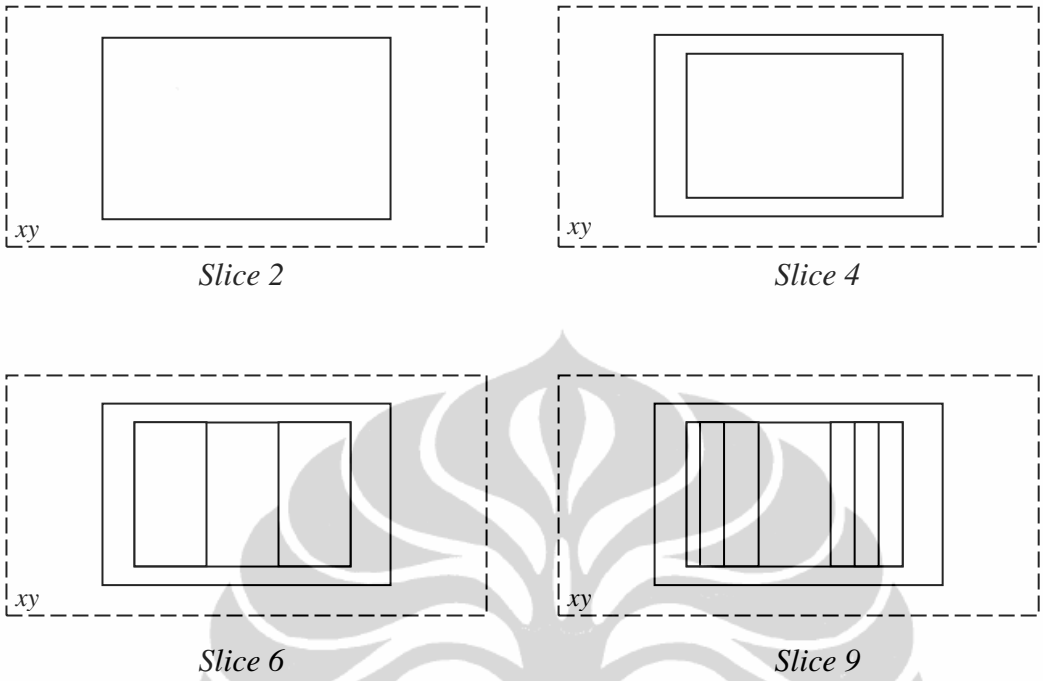


Gambar III.2 : File STL input

Tahapan selanjutnya adalah *slicing*, pada bagian ini model File STL kemudian dipotong dengan bidang pada sumbu-z. Interval *slicing* di ambil dari parameter *layer thickness* dan prosesnya dimulai dari bawah z = 0 sampai dengan z = maksimum model, untuk penelitian ini interval *slicing* dibuat seragam untuk mempermudah dalam proses trajectory.

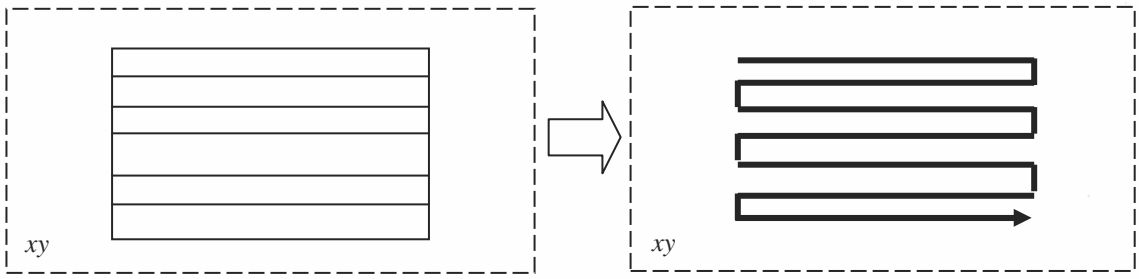


Gambar III.3 Slicing dengan interval seragam pada sumbu-z



Gambar III.4 : Permukaan layer pada tahapan slicing

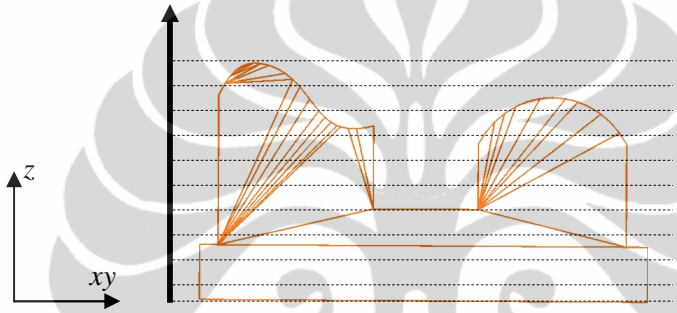
Hasil *slicing* dilakukan pada sumbu-z dari z-minimum sampai z-maksimum ini titik potongnya disimpan dalam vector data khusus dan akan digunakan untuk tahapan selanjutnya, yaitu *path elements generation*. Tahapan ini dibuat dengan membuat lintasan laser untuk setiap layer, lintasan yang dibuat dengan metode directional parallel untuk mempermudah gerakan perpindahan laser. Pada tahap ini dilakukan proses *slicing* juga terhadap sumbu-x, dengan intervalnya berdasarkan parameter hatch space. Proses *slicing* dilakukan dari x-minimum sampai x-maksimum dan dilakukan pencarian titik yang akan digunakan untuk lintasan laser tiap layer.



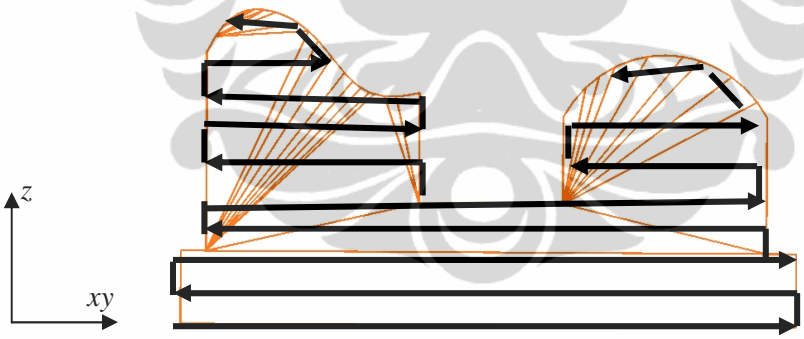
Gambar III.5 : Path elements generation

Tahap terakhir adalah *path lingking*, pada tahap ini setiap *path element generation* tiap layer kemudian dihubungkan menjadi satu lintasan laser dari z = 0 sampai z maksimum model. Dan output dari tahapan algoritma ini adalah lintasan laser dan File G-Code yang berisi kode L00, G00, G01 dan koordinat yang mengiringinya.

- L00 = Pergerakan laser menuju layer ke...dengan kondisi *laser off*.
- G00 = Pergerakan laser menuju koordinat yang dituju dengan *laser off*.
- G01 = Pergerakan laser menuju koordinat yang dituju dengan *laser on*.



Gambar III.6 : Path lingking



(a)

```
L00 1
G00 -5.256450:-5.481444:0.750000
G01 -5.256450:128.518550:0.750000
G01 -3.756450:128.518550:0.750000
G01 -3.756450:-5.481444:0.750000
G01 -2.256450:-5.481444:0.750000
G01 -2.256450:128.518550:0.750000
...
...
dst.
```

(b)

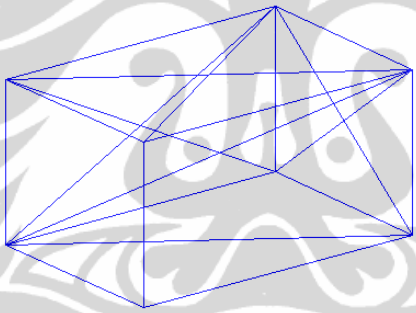
Gambar III.7 : Hasil output: a. laser trajectory, b. File G-Code

3.2 Penentuan Index Segitiga, Index Vertex, dan Koordinat Vertex

Untuk mendapatkan informasi mengenai index segitiga, index vertex, dan koordinat vertex dari file STL maka perlu dipahami terlebih dahulu mengenai pengertian kata "*solid*", "*facet normal*", "*endfacet*", "*outerloop*", "*vertex*", "*endvertex*", dan "*endsolid*" yang sudah dijelaskan pada Bab.2.

Untuk membuka dan membaca file STL maka perlu *script fid=fopen(namaFile,'r').* *Fopen* merupakan fungsi untuk membuka file, sedangkan 'r' merupakan string nama file untuk membaca file. Fungsi ini di ikuti dengan *aa=fgetl(fid),* sebagai teks file dalam satu garis.

Misalkan sebuah balok berikut memiliki enam sisi, maka akan terbentuk 12 segitiga *facet* dan 8 titik vertex pembentuk solid.



```
solid
 facet normal +0.0000000E+00 +0.0000000E+00 +1.0000000E-00
  outer loop
    vertex   +2.5000000E+01 +2.0000000E+00 +2.2000000E+01
    vertex   +2.5000000E+01 +4.2000000E+01 +2.2000000E+01
    vertex   +0.0000000E+00 +4.2000000E+01 +2.2000000E+01
  endloop
 endfacet
 facet normal +0.0000000E+00 +0.0000000E+00 +1.0000000E-00
  outer loop
    vertex   +0.0000000E+00 +4.2000000E+01 +2.2000000E+01
    vertex   +0.0000000E+00 +2.0000000E+00 +2.2000000E+01
    vertex   +2.5000000E+01 +2.0000000E+00 +2.2000000E+01
  endloop
 endfacet
 facet normal +0.0000000E+00 +1.0000000E-00 +0.0000000E+00
  outer loop
    vertex   +2.5000000E+01 +4.2000000E+01 +2.2000000E+01
    vertex   +2.5000000E+01 +4.2000000E+01 +2.0000000E+00
    vertex   +0.0000000E+00 +4.2000000E+01 +2.0000000E+00
  endloop
 endfacet
 facet normal +0.0000000E+00 +1.0000000E-00 +0.0000000E+00
  outer loop
    vertex   +0.0000000E+00 +4.2000000E+01 +2.0000000E+00
    vertex   +0.0000000E+00 +4.2000000E+01 +2.2000000E+01
    vertex   +2.5000000E+01 +4.2000000E+01 +2.2000000E+01
  endloop
```

```
 endfacet
 facet normal +0.0000000E+00 +4.4408920E-17 -1.0000000E-00
  outer loop
    vertex   +2.5000000E+01 +4.2000000E+01 +2.0000000E+00
    vertex   +2.5000000E+01 +2.0000000E+00 +2.0000000E+00
    vertex   +0.0000000E+00 +2.0000000E+00 +2.0000000E+00
  endloop
 endfacet
 facet normal +0.0000000E+00 +4.4408920E-17 -1.0000000E-00
  outer loop
    vertex   +0.0000000E+00 +2.0000000E+00 +2.0000000E+00
    vertex   +0.0000000E+00 +4.2000000E+01 +2.0000000E+00
    vertex   +2.5000000E+01 +4.2000000E+01 +2.0000000E+00
  endloop
 endfacet
 facet normal +0.0000000E+00 -1.0000000E-00 -8.8817841E-17
  outer loop
    vertex   +2.5000000E+01 +2.0000000E+00 +2.0000000E+00
    vertex   +2.5000000E+01 +2.0000000E+00 +2.2000000E+01
    vertex   +0.0000000E+00 +2.0000000E+00 +2.2000000E+01
  endloop
 endfacet
 facet normal +0.0000000E+00 -1.0000000E-00 -8.8817841E-17
  outer loop
    vertex   +0.0000000E+00 +2.0000000E+00 +2.2000000E+01
    vertex   +0.0000000E+00 +2.0000000E+00 +2.0000000E+00
    vertex   +2.5000000E+01 +2.0000000E+00 +2.0000000E+00
  endloop
 endfacet
 facet normal +1.0000000E+00 +0.0000000E+00 +0.0000000E+00
  outer loop
    vertex   +2.5000000E+01 +2.0000000E+00 +2.0000000E+00
    vertex   +2.5000000E+01 +4.2000000E+01 +2.0000000E+00
    vertex   +2.5000000E+01 +4.2000000E+01 +2.2000000E+01
  endloop
 endfacet
 facet normal +1.0000000E+00 +0.0000000E+00 +0.0000000E+00
  outer loop
    vertex   +2.5000000E+01 +4.2000000E+01 +2.2000000E+01
    vertex   +2.5000000E+01 +2.0000000E+00 +2.2000000E+01
    vertex   +2.5000000E+01 +2.0000000E+00 +2.0000000E+00
  endloop
 endfacet
 facet normal -1.0000000E+00 +0.0000000E+00 +0.0000000E+00
  outer loop
    vertex   +0.0000000E+00 +4.2000000E+01 +2.0000000E+00
    vertex   +0.0000000E+00 +2.0000000E+00 +2.0000000E+00
    vertex   +0.0000000E+00 +2.0000000E+00 +2.2000000E+01
  endloop
 endfacet
 facet normal -1.0000000E+00 +0.0000000E+00 +0.0000000E+00
  outer loop
    vertex   +0.0000000E+00 +2.0000000E+00 +2.2000000E+01
    vertex   +0.0000000E+00 +4.2000000E+01 +2.2000000E+01
    vertex   +0.0000000E+00 +4.2000000E+01 +2.0000000E+00
  endloop
 endfacet
endsolid
```

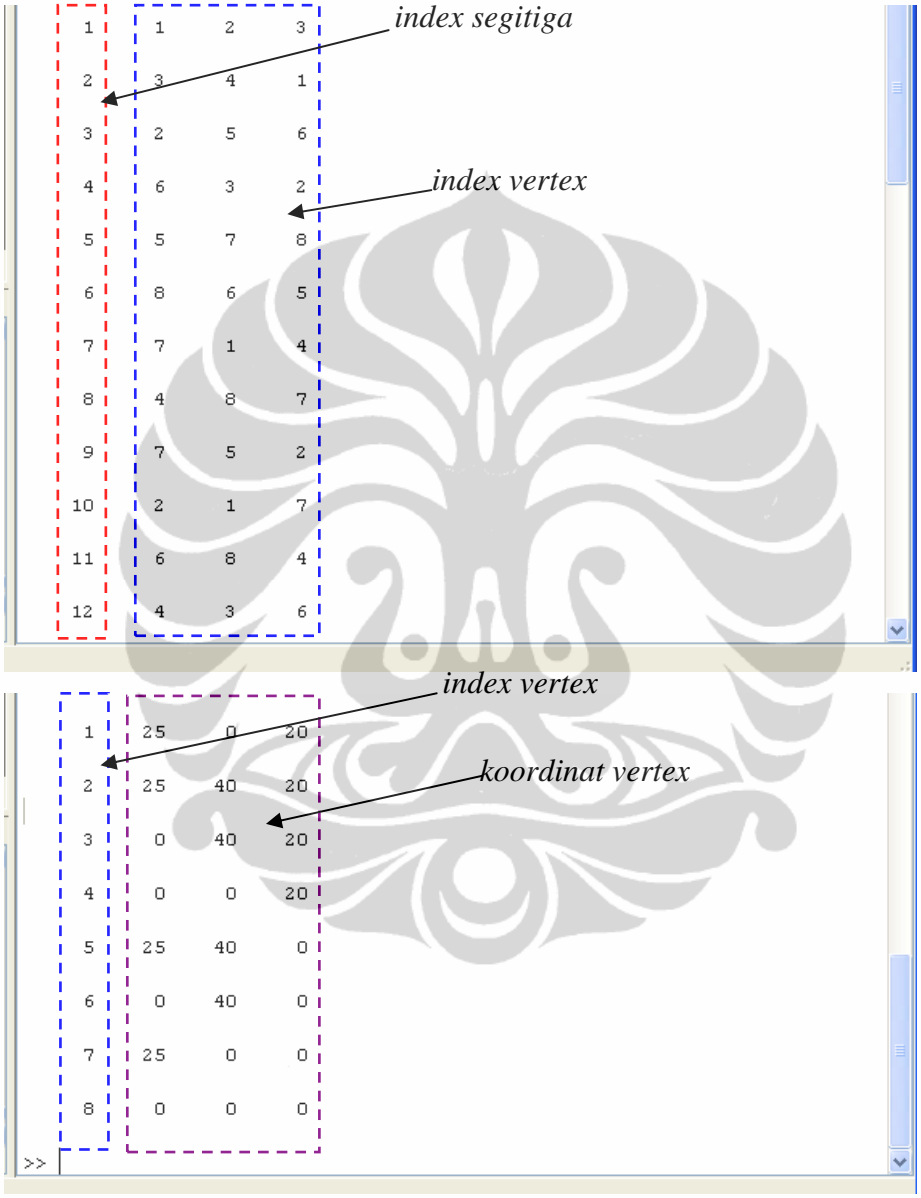
Gambar III.8 : Model dan File STL Balok

di dalam file STL tersebut tidak ada kata mengenai index segitiga dan index vertex. sehingga perlu dibuat program m-file untuk membaca informasi index segitiga, index vertex, dan koordinat vertex seperti berikut ini,

```
cla reset;
namaFile='Balok.txt';
fid=fopen(namaFile,'r');
%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
        ii = 0;
        count=1;

    elseif strcmp(a,'vertex')
        ii=ii+1;
        [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;

        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;

    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;

    end
end

deltaX=xMinMax(1)-0;
deltaY=yMinMax(1)-0;
deltaZ=zMinMax(1)-0;

% normalisasi
xMinMax(1)=xMinMax(1)-deltaX;
xMinMax(2)=xMinMax(2)-deltaX;
yMinMax(1)=yMinMax(1)-deltaY;
yMinMax(2)=yMinMax(2)-deltaY;
zMinMax(1)=zMinMax(1)-deltaZ;
zMinMax(2)=zMinMax(2)-deltaZ;

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

% menentukan index vertex dan koordinat vertex
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
```

Setelah program tersebut dijalankan maka didalam *command window* akan muncul informasi mengenai index segitiga, index vertex, dan koordinat vertex.



Gambar III.9 : Index segitiga, index vertex, dan koordinat vertex

3.3 Penentuan Index Segitiga Yang Berpotongan

Algoritma untuk mendapatkan perpotongan antara sebuah model 3D dengan bidang datar, bidang yang dipilih adalah bidang z, yaitu bidang yang tegak lurus dengan sumbu koordinat z (atau bidang yang sejajar dengan bidang yang dibentuk oleh sumbu koordinat xy). Bidang-bidang yang digunakan sebagai pemotong model bergantung kepada interval antar dua bidang berdekatan, serta koordinat z minimum dan maksimum. Misalnya jika z_{min}=0 dan z_{max}=10 sedangkan interval antar dua bidang *slicing interval* = 2, maka ada enam bidang, masing-masing z={0, 2, 4, 6, 8, 10}.

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

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

Ketiga informasi ini menjadi *database* utama, bisa digambarkan sebagai berikut:

Index segitiga	Index vertex 1	Index vertex 2	Index vertex 3
1	1	2	3
2	1	2	4
dst	……	……..	……

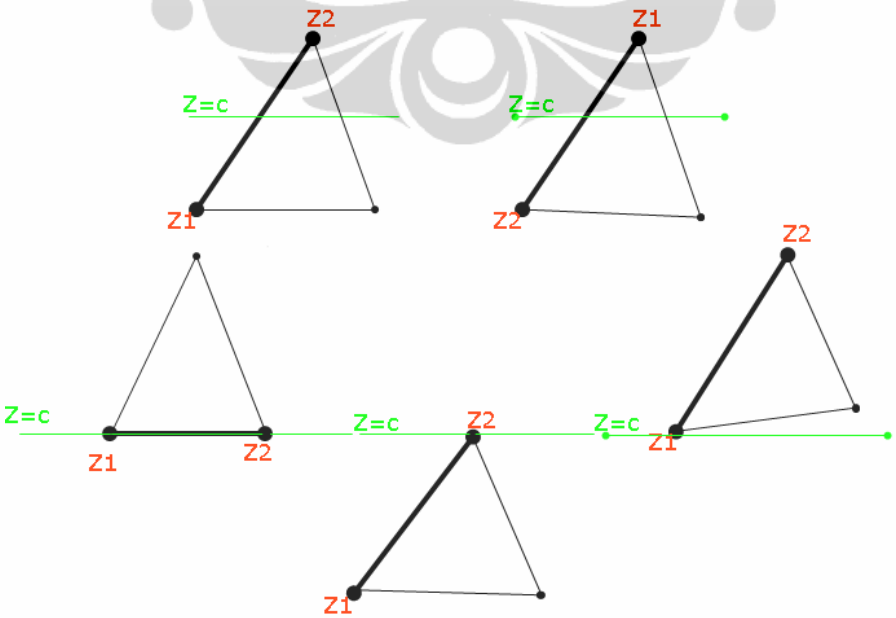
Index vertex	Koordinat x	Koordinat y	Koordinat z
1	0.12	0.14	0.32
2	0.29	0.33	0.45
dst	……	……..	……

Jadi, untuk menentukan lokasi sebuah segitiga terhadap bidang, cari indexnya, kemudian tentukan tiga index verteknya, dan dapatkan koordinat (x,y,z) dari ketiga verteknya.

Setelah informasi mengenai index segitiga, index vertex, dan koordinat vertex sudah teridentifikasi dan tersimpan dalam array data, selanjutnya adalah mencari index segitiga yang berpotongan dengan bidang potong z. Metode pencarian ini dilakukan dengan metode *brute searching*, metode ini dilakukan dengan mengecek setiap segitiga yang ada apakah berpotongan dengan bidang z atau tidak. Jika berpotongan, indeks segitiga tersebut disimpan dalam sebuah struktur data array. Pengecekan ini dilakukan untuk setiap bidang z yang akan diiriskan dengan model *facet*.

Pendekatan untuk pengecekan setiap bidang potong dilakukan dengan kondisi berikut:

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



Gambar III.10 : Titik perpotongan segitiga yang mungkin.

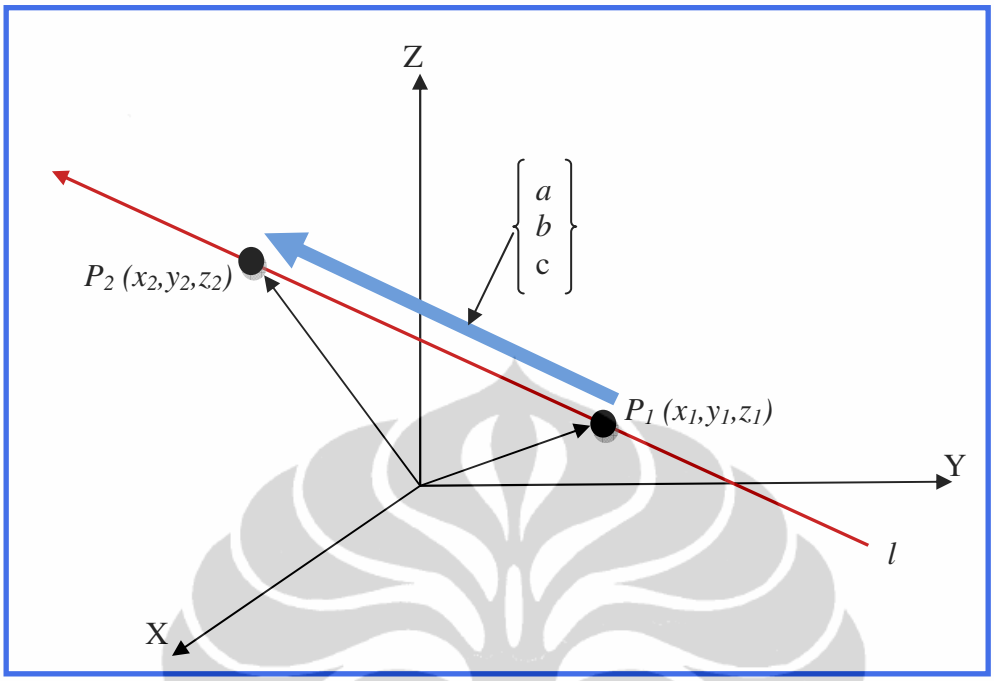
3.4 Penentuan *CC-Point* Tiap *Slicing*

Penentuan CC-Point tiap *slicing* dilakukan dengan metode *adjacent serching*, metode ini diawali dengan mencari secara acak sebuah segitiga yang berpotongan dengan bidang z (sebut saja segitiga u), begitu didapatkan satu segitiga tersebut, maka algoritma berjalan sebagai berikut:

1. Ambil satu sisi segitiga, misal sisi s yang berpotongan dengan bidang z, kemudian tentukan titik perpotongannya. Simpan titik ini pada sebuah struktur data vektor. Segitiga yang tersimpan dalam struktur data *angkatan* kemudian diambil salah satu. Lalu salah satu dari sisinya di ambil. Sisi yang diambil memiliki dua titik dan dua titik tersebut di ambil nilai Z –nya, *letakZ_1 = V(index1,3)* dan *letakZ_2 = V(index2,3).*

2. Kemudian posisi tersebut di periksa sisi tersebut dan dilakukan perhitungan dengan metode yang sudah pernah dilakukan oleh Ganjar K. [8], yaitu menetukan formulasi untuk mencari titik potong bidang $z = c$ dengan sisi dari sebuah segitiga. Sisi segitiga dibentuk oleh dua buah titik $p_1(x_1,y_1,z_1)$ dan $p_2(x_2,y_2,z_2)$. Langkah pertama adalah mengecek apakah c berada di dalam rentang z_1 dan z_2. Jika iya, maka berarti sisi segitiga tersebut berpotongan dengan bidang $z = c$. Langkah selanjutnya adalah mencari titik dimana perpotongan terjadi.

   Secara matematis, persamaan garis dalam ruang R3 dirumuskan sebagai berikut.



Gambar III.11 : Persamaan garis pada ruang R3[8].

Berdasarkan gambar di atas, persamaan garis-*l* yang dibentuk oleh 2 titik adalah

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix} t + \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \qquad \text{(III.1)}$$

Karena $P_2 = P_1 + \begin{pmatrix} a \\ b \\ c \end{pmatrix}$, maka $\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} - \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}$ (III.2)

Dengan demikian persamaan (III.1) dapat disubstitusi menjadi

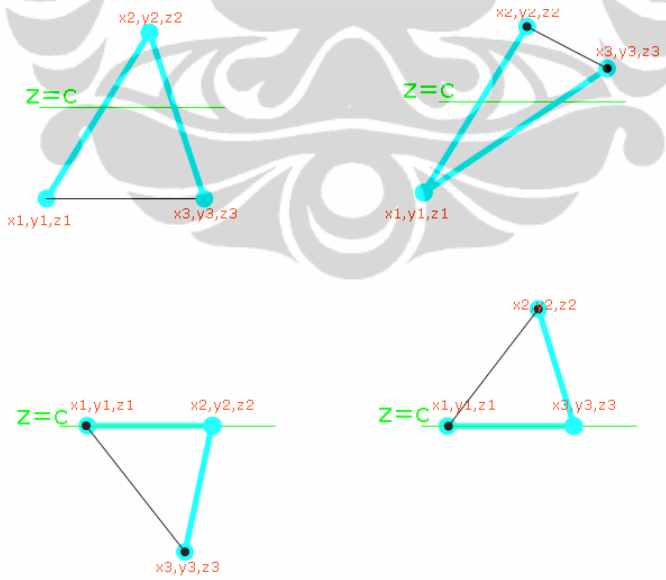
$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \left(\begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} - \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \right) t + \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \qquad \text{(III.3)}$$

Karena $z = c$, maka nilai t dapat dihitung, yaitu $t = \dfrac{z - z_1}{z_2 - z_1}$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \left(\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} - \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \right) t + \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \qquad \text{(III.4)}$$

Maka dari persamaan (III.4) dapat diperoleh titik (x,y,z) sebagai titik perpotongan antara sisi segitiga yang dibentuk oleh titik $p_1(x_1,y_1,z_1)$ dan $p_2(x_2,y_2,z_2)$ dengan bidang z = c. Dan titik potong yang telah ditentukan tersebut kemudian disimpan dalam variabel data *V2* (penulisan variabel di Software pemrograman).

3. Setelah titik potong sisi pertama ditemukan, selanjutnya mencari sisi yang indek segitiganya sama kemudian simpan sisi segitiga tersebut. Letak Z dari sisi yang ditemukan tersebut kemudian disimpan sebagai titik ketiga dari indek segitiga pertama. Sehingga proses selanjutnya adalah melakukan pencarian titik potong untuk setiap kondisi:
   - Jika titik potong jatuh diantara vertex 1 dan vertex 3
   - Jika titik potong jatuh diantara vertex 2 dan vertex 3
   - Jika titik potong jatuh pada vertex 2
   - Jika titik potong jatuh pada vertex 3



Gambar III.12 : Penentuan titik selanjutnya.

Kemudian dilakukan perhitungan untuk mencari titik potong (x,y,z) dengan persamaan garis lurus (III.3) pada bagian sebelumnya. Dan hasil perhitungan titik potong disimpan dalam variabel data *V2*.

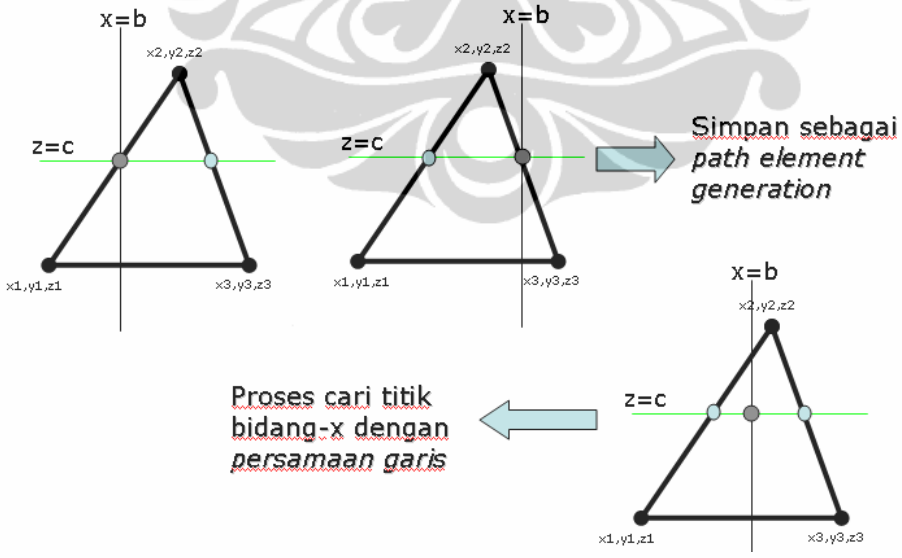
4. Algoritma ini berjalan terus sampai diketemukannya seluruh titik potong dengan sumbu-z. dan akan berakhir sampai pada z-maksimum.

3.5 Penentuan Lintasan Setiap Layer

Pada bagian ini dilakukan proses *slicing* untuk membuat lintasan pada bidang-xy. Untuk itu perlu ditentukan bidang yang akan dipotong. bidang-x dipilih sebagai bidang potong terhadap *facet*.

Algoritma dilakukan dengan mengambil salah satu titik z bagian tertentu dari variabel data V2 yang bersinggungan dengan potongan bidang-x .Kondisi ini digambarkan sbb:

- Jika *V2(k,satu,1)* sama dengan x, maka disimpan dalam array data *V_Lintasan*.
- Jika *V2(k,dua,1)* sama dengan x, maka disimpan dalam array data *V_Lintasan*.
- Jika x diantara *V2(k,satu,1)* dan *V2(k,dua,1)* maka dilakukan perhitungan titik potong pada sumbu-x dengan persamaan garis lurus.



Gambar III.13 : Penentuan titik-titik path elements.

Seperti terlihat pada gambar garis pada ruang R3 (gambar III.11) proses pencarian titik pada sumbu-x, dilakukan dengan persamaan (III.3) berikut:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \left(\begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} - \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \right) t + \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}$$

Karena bidang potong adalah bidang-x, maka $x = b$, maka nilai t dapat dihitung, yaitu $t = \dfrac{b - x_1}{x_2 - x_1}$

$$\begin{pmatrix} y \\ z \end{pmatrix} = \left(\begin{pmatrix} y_2 \\ z_2 \end{pmatrix} - \begin{pmatrix} y_1 \\ z_1 \end{pmatrix} \right) t + \begin{pmatrix} y_1 \\ z_1 \end{pmatrix} \qquad \text{(III.5)}$$

Maka dari persamaan (III.5) dapat diperoleh titik (x,y,z) sebagai titik perpotongan antara sisi segitiga yang dibentuk oleh titik $p_1(x_1,y_1,z_1)$ dan $p_2(x_2,y_2,z_2)$ dengan bidang x = b.

3.6 Algoritma Keseluruhan

Secara umum proses pembuatan *laser trajectory* ini terdiri dari beberapa tahapan proses berikut,

a. Memasukkan file yang akan dibuat *laser trajectory*.
b. Menentukan index segitiga, index vertex, dan koordinat vertex.
c. Mencari index segitiga tiap interval *slicing* bidang-z (metode *brute searching*).
d. Menyimpan index segitiga yang telah ditemukan kedalam variabel data.
e. Menentukan segitiga pertama yang telah ditemukan dan disimpan dalam variabel data, kemudian mencatat index vertexnya dan menyimpan titik potongnya kedalam variabel data baru.
f. Mencari segitiga selanjutnya dan kemudian tentukan titik potongnya apakah bertemu di titik atau diantara titik.
g. Melakukan dengan berulang sampai semua titik potong bidang-z telah ditemukan dan disimpan dalam variabel data.
h. Menghubungkan titik-titik potong bidang-z yang telah ditemukan sampai membentuk kurva kontur untuk setiap layer.

i. Membuat lintasan untuk setiap layer. Lintasan dibuat dengan terlebih dahulu menentukan titik x sesuai dengan *hatch space* yang telah ditentukan sebelumnya.
j. Mengurutkan titik potong berdasarkan arah sumbu-y.
k. Menghubungkan titik-titik lintasan yang telah diurutkan dengan fungsi plot grafik.
l. Mengeluarkan file output dalam bentuk *.txt,* yang berisi kode mesin L00, G00, dan G01 di ikuti koordinat lintasanya.