M1-003 PENGEMBANGAN *LASER TRAJECTORY* PROSES *RAPID PROTOTYPING* UNTUK PRODUK BERKONTUR DAN PRISMATIK

Gandjar Kiswanto, Ahmad Kholil

Laboratorium Teknologi Manufaktur dan Otomasi
Departemen Teknik Mesin – Universitas Indonesia
Kampus Baru UI – Depok 16424
gandjar_kiswanto@eng.ui.ac.id

ABSTRAK

Rapid prototyping atau layered manufacturing secara singkat merupakan proses fabrikasi produk dengan layer by layer, dimana material ditambahkan ke layer berturut-turut sesuai dengan laser trajectory. Penelitian ini bertujuan untuk mengembangkan laser trajectory proses rapid prototyping untuk produk berkontur dan prismatik dengan arah directional parallel. Pengembangan ini menggunakan parameter layer thickness dan hatch space yang menjadi variabel dari interval bidang potong pembuatan laser trajecory.

Hasil penelitian ini berupa algoritma pembuatan laser trajectory pada model STL berkontur dan prismatik yang dikembangkan oleh Laboratorium Teknologi Manufaktur dan Otomasi – Departemen Teknik Mesin – UI bersamaan dengan pengembangan mesin RP. Algoritma yang dikembangkan menampilkan graphic user interface (GUI) sehingga pemakai dapat menentukan model, memasukkan nilai parameter, dan kemudian mensimulasikan. Selain menghasilkan grafik, juga menghasilkan file G-Code yang disertai titik-titik koordinat penentu laser trajectory. Dari simulasi dengan beberapa model prismatik dan berkontur menghasilkan laser trajectory yang teratur sesuai dengan pola model dan file G-Code dalam format text.

Kata kunci : rapid prototyping, laser trajectory

ABSTRACT

Rapid Prototyping or layered manufacturing simply is parts fabrication process with layer by layer, where material added to layer successively as according to laser trajectory. This research aim to laser trajectory generation of rapid prototyping process for sculptured and prismatic parts with directional parallel. The development use parameter layer thickness, and hatch space as variable of slicing interval to generate laser trajectory.

The result is algorithm of laser trajectory generation for sculptured and prismatic of STL model. The algorithm presenting graphic user intervace (GUI) in order to user can determine model, input parameter value, and than simulate model. Beside yielding graph, also yield G-Code file accompanied coordinate points as determinant laser trajectory. From simulation of model sculptured and prismatic resulting laser trajectory consecutively as according to model pattern and G-Code file in text format.

Keywords : rapid prototyping, laser trajectory

1. Latar Belakang

Pengembangan produk oleh perusahaan manufaktur merupakan sebuah keharusan untuk memenuhi kebutuhan konsumen. Beberapa perusahaan manufaktur melakukan pengembangan produk, yaitu proses dimana konsep produk harus diterjemahkan dari gambar teknik menjadi produk phisik. Pembuatan produk phisik model pertama atau *prototype* dinamakan *prototyping*. Prototyping sangat penting karena merupakan makna terakhir dalam verifikasi bentuk, kesesuaian, dan fungsi produk. *Prototype* dibuat dalam volume sedikit dengan biaya tinggi karena semua biaya *tool* digabungkan pada *prototype* yang jumlahnya sedikit. Fabrikasi dengan *tool* khusus (seperti: *pattern* atau *molds* untuk *casting, dies* untuk *forming*, *fixture* untuk *machining*) memerlukan waktu yang lama untuk membuat dan menguji *prototype*. *Rapid prototyping* merupakan metode yang membantu dalam proses pengembangan produk yang mudah dan cepat, sehingga dapat mempengaruhi kepuasan *customer* dan keuntungan perusahaan adalah terbantu dalam mendapatkan produk untuk pasar pertama. Wholers [2] melakukan survey, dan menemukan bahwa sekitar 23,4% produk RP digunakan sebagai alat peraga, sedangkan 27,5% digunakan sebagai master pola pada proses kedua manufaktur dan untuk *direct tooling*. Industri menggunakan 15,6% untuk *fit* dan *assembly test*, 16,1% untuk test fungsional dan sisanya untuk *quoting*, proposal, dan *evaluasi ergonomi*.

Rapid Prototyping atau *Layered Manufacturing* adalah proses fabrikasi suatu produk dengan *layer by layer*, atau penambahan *raw material* berturut-turut pada *layer* hingga terbentuk produk yang sesuai dengan model. Prosesnya diawali dari model *facet* (file STL) berisi banyak segitiga yang membentuk model solid. Kemudian dilakukan proses *slicing* dan *hatching* pada model *facet* untuk membuat *laser trajectory* (lintasan laser). Setelah *laser trajectory* terbentuk, proses fabrikasi dapat dilakukan dengan mengacu pada *laser trajectory* hasil *generate* model.

Model *facet* berisi sekumpulan segitiga-segitiga pembentuk, sedangkan *laser trajectory* merupakan representasi lintasan berisi koordinat acuan mesin. Bagaimana *laser trajectory* ini terbentuk merupakan suatu pertanyaan yang mendasar dalam pengembangan mesin *prototyping*. Perlu algoritma yang dapat memadukan proses *slicing* dan *hatching* dalam pengembangan *software* pembuat *laser trajectory* pada berbagai macam produk prismatik maupun berkontur.

Prinsip *rapid prototyping* berbeda dengan prinsip *machining*, dimana *rapid prototyping* merupakan proses penambahan material, sedangkan *machining* merupakan proses melepas material. Walaupun demikian keduanya memiliki kesamaan dalam penentuan titik kontak pahat (*cc-point*) pada model *facet*. Dengan menggunakan paramater kontrol *rapid prototyping*, seperti *layer thickness*, *hatch space*, dan orientasi produk, pengembangan *laser trajectory* dapat menggunakan pengontrolan dari parameter tersebut.

Parameter kontrol merupakan hal yang penting yang perlu diperhatikan untuk mendapatkan keakuratan produk yang diinginkan. Parameter tersebut disesuaikan dengan batasan minimum dan maksimum dari kondisi mesin. Pengguna dapat memberikan input parameter tersebut sesuai dengan kebutuhaan dan kondisi yang dipersyaratkan.

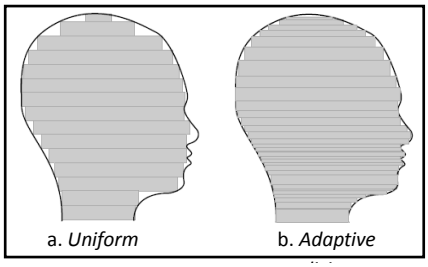
2. Reviuw *Slicing* dan Pembuatan Lintasan

Setiap teknologi *rapid prototyping* memiliki spesifikasi sendiri-sendiri [13]. Ada yang berbentuk formasi dinding terluar, metode pengisian, dan pemisahan produk dari material pelingkup

yang menentukan pola lintasan mesin. Formasi dinding terluar merupakan pola yang saat ini banyak dipakai pada mesin *prototyping*, formasi ini menghasilkan produk yang cepat dari segi proses pembuatannya dan sedikit membutuhkan material. *Prototype* yang dihasilkan sangat lemah dari segi kekuatan sehingga tidak cocok untuk diberi pembebanan. Formasi dengan metode pengisian seluruhnya akan menghasilkan produk yang sesungguhnya, formasi ini prosesnya akan semakin lama bila dibandingkan dengan formasi dinding terluar. Bila tujuannya untuk membuat *protoype* yang kuat seperti produk yang diinginkan formasi ini sangat diperlukan.

Pola lintasan mesin rapid *prototyping* dapat digerakkan secara robotik pada bidang-XY untuk Mesin FDM, pola laser untuk solidifikasi dan *sintering* material pada Mesin SLA dan SLS, atau pola pisau laser untuk Mesin LOM. Proses-proses ini membutuhkan strategi pembuatan lintasan mesin (*machine path*) yang berbeda. Beberapa pendekatan pembuatan proses *slicing* dan *NC-path* sudah diusulkan dan diimplementasikan ke karakteristik khusus dan kebutuhan-kebutuhan berbagai proses RP.

Pendekatan-pendekatan proses *slicing* dikategorikan kedalam empat kelompok [9], yaitu:



Gambar 1 : Metode slicing pada RP [7]

a. Metode *slicing* model file STL dengan ketebalan layer seragam (*uniform*).
b. Metode *slicing* model file STL dengan ketebalan layer *adaptive*.
c. Metode *slicing* model CAD dengan ketebalan layer *adaptive*.
d. Metode *slicing* dengan perhitungan kontur yang tepat.

Metode *slicing* ketebalan layer seragam, semua layer memiliki ketebalan yang sama, sedangkan metode *slicing* ketebalan layer *adaptive*, ketebalan layer akan bervariasi menurut kompleksitas geometri. Ada empat konfigurasi yang mungkin ketika melakukan *slicing* terhadap model telah dikembangkan oleh Kulkarni dan Dutta [11] untuk prosedur *slicing* yang lebih akurat pada daerah *convex dan concave curvature*. Pendekatan bertujuan menentukan ketebalan layer yang dihitung berdasarkan kurva geometri, sehingga pendekatan ini kurang cocok dilakukan untuk bentuk STL karena perhitungan didasarkan pada radius kurvatur.

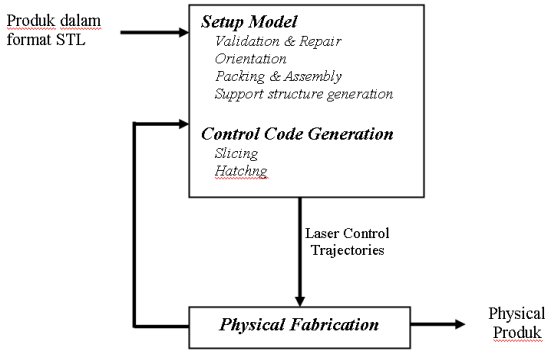
Proses pembuatan lintasan pahat (*tool path*) dapat mempengaruhi kualitas permukaan, kekuatan, kekakuan, dan waktu pembuatan produk dalam proses *rapid prototyping*. Perencanaan lintasan termasuk perencaan lintasan bagian dalam dan lintasan bagian luar [11]. Lintasan bagian dalam merupakan proses pengisian material pada bagian dalam layer. Sedangkan untuk bagian luar hanya dilakukan oleh mesin LOM, karena lintasan luar dilakukan untuk memotong lembaran *raw material*. Kock B [7] menjelaskan mengenai analisa geometri proses RP dengan formasi dinding terluar menggunakan *offset object* model STL secara tidak seragam. Metodenya menghasilkan *boundary layer* yang *smooth* menggunakan *biarch fitting*. Untuk membuat kontur yang lebih efisien

dan kompleks, Choi dan Kwok [12] mengembangkan algoritma secara hirarki pada kontur dengan *slicing* yang toleran. Hasilnya dapat digunakan untuk model komplek dan besar. Sedangkan Asiabanpur [13] mengembangkan proses pembuatan lintasan mesin untuk proses *rapid prototyping* baru berbasis serbuk, *Selective Inhibition Sintering* (SIS). Hasilnya berupa algoritma *slicing* dan *hatching* hanya untuk proses tersebut. Karena proses SIS berbeda dengan RP berbasis laser, algoritma yang dikembangkan belum bisa digunakan untuk laser trajectory.

3. Proses Rapid Prototyping

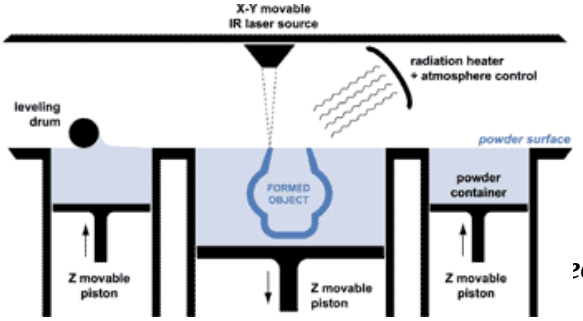
Proses *rapid prototyping* diawali dengan validasi model *CAD* tiga dimensi suatu produk, langkah ini dilakukan untuk memastikan bentuknya *solid*. Model yang sudah *valid* kemudian diorientasikan terhadap ruang pembuatan (*parts orientation*), dengan mempertimbangkan waktu pembuatan dan kualitas permukaan. Beberapa model dapat digabung menjadi satu bangunan *asembly* untuk efisiensi penggunaan mesin dan material. Berdasarkan pada persyaratan prosesnya, dukungan struktur dapat ditambahkan ke model jika diperlukan. Setelah validasi, kemudian model dipotong dengan bidang horisontal. Tiap bidang horisontal menghasilkan bidang potong sebagai penentu *laser trajectory* untuk mengontrol proses *sintering* atau solidifikasi.

Langkah utama untuk proses planning termasuk orientasi, *generate* struktur pendukung jika diperlukan, *slicing* dan pemilihan parameter proses.



Gambar 2 : Diagram proses rapid prototyping [6]

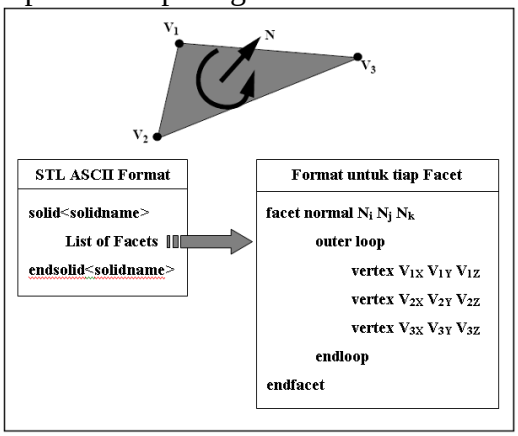
Perencanaan proses dilakukan untuk memilih parameter proses dan pembuatan instruksi kontrol untuk fabrikasi produk. Umumnya desainer menyelesaikan perencanaan proses dengan mempelajari produk dan persyaratan kualitas, yang tentunya sangat memakan waktu.



ective Laser Sintering (SLS)[16]

4. Model Facet

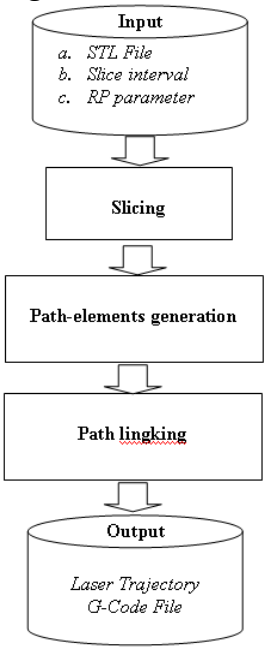
File STL merupakan kependekan dari *stereolithography*. File yang berekstensi STL terdiri dari dua jenis format. Format yang pertama adalah *binary*. Pada format *binary*, model surface yang tersusun atas segitiga-segitiga disimpan dalam bentuk biner yang tidak terbaca dalam *text editor*. Format lainnya adalah ASCII (*American Standard Code for Information Interchange*). Format ini adalah yang paling umum digunakan karena lebih mudah dibaca dan dimengerti, serta dapat dibuka di *text editor*. Format ASCII dapat dilihat pada gambar berikut.



Gambar 4 : STL ASCII Format

5. Algoritma Pengembangan *Laser Trajectory*

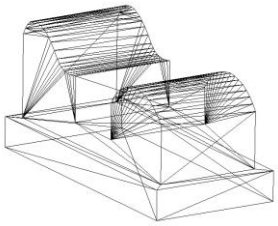
Algoritma perencanaan untuk membuat *laser trajectory* model prismatik dan berkontur ini dilakukan dengan tahapan algoritma pada gambar 5.



Gambar 5 : Algoritma pengembangan laser trajectory

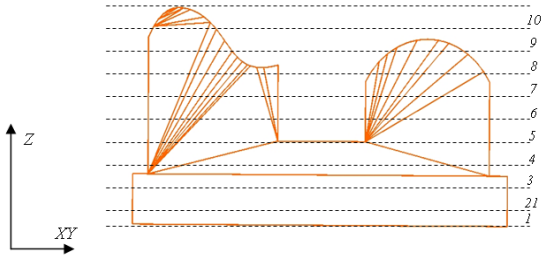
Tahapan algoritma pembuatan *laser trajectory* dimulai dengan mengambil File STL yang berisi informasi menganai data *facet* normal dan tiga vertex pembentuk masing-masing segitiga. File STL dapat diperoleh dari software-software CAD yang ada, seperti Unigraphic, SolidWork,

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*.

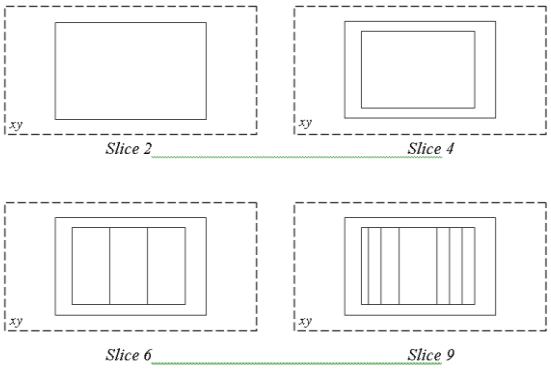


Gambar 6 : Model STL

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.

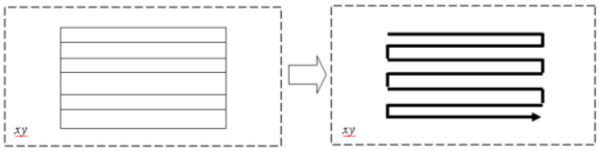


Gambar 7 : Slicing dengan interval seragam pada sumbu-z



Gambar 8 : Permukaan layer pada tahapan slicing

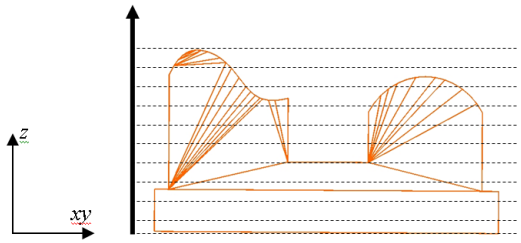
Hasil *slicing* dilakukan pada sumbu-z dari z-minimum sampai z-maksimum ini titik potongnya disimpan dalam vector data khusus dan akan digunakan untuk tahapan selanjutnya, yaitu *path elements generation*. Tahapan ini dibuat dengan membuat lintasan laser untuk setiap layer, lintasan yang dibuat dengan 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.



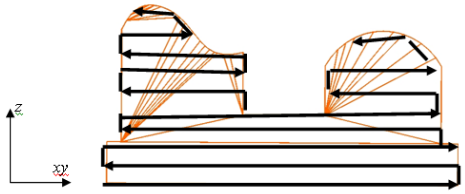
Gambar 9 : Path elements generation

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 10 : 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 11 : Hasil output: a. laser trajectory, b. File G-Code

5.1 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 z-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	……	……..	……

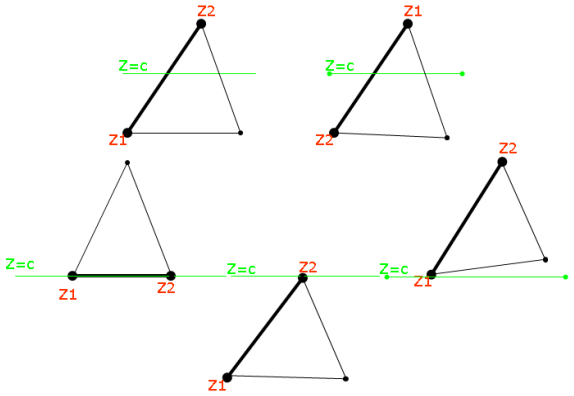
Index vertex	Koordinat-x	Koordinat-y	Koordinat-z
1	0.12	0.14	0.32
2	0.29	0.33	0.45
dst	……	……..	……

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

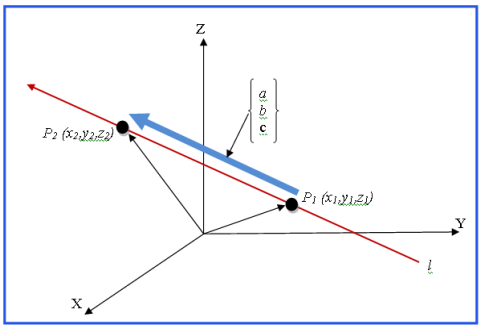


Gambar 12 : Titik perpotongan segitiga yang mungkin.

5.2 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 13 : Persamaan garis pada ruang R3[8].

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

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

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

Dengan demikian persamaan (a) dapat disubstitusi menjadi

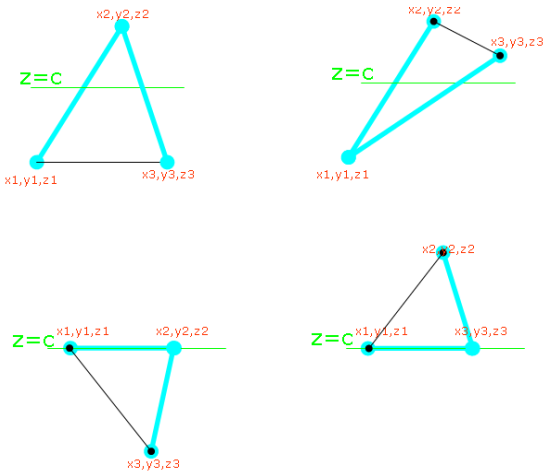
$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \left(\begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} - \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \right) t + \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \quad \text{(c)}$$

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

$$\begin{pmatrix} x \\ y \end{pmatrix} = \left(\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} - \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \right) t + \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \quad \text{(d)}$$

Maka dari persamaan (d) 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 14 : Penentuan titik selanjutnya.

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

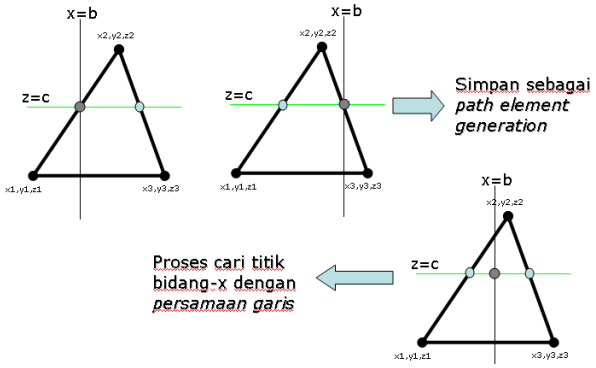
4. Algoritma ini berjalan terus sampai diketemukannya seluruh titik potong dengan sumbu-z. dan akan berakhir sampai pada z-maksimum.

5.3 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 15 : Penentuan titik-titik path elements.

Seperti terlihat pada gambar garis pada ruang R3 (gambar 13) proses pencarian titik pada sumbu-x, dilakukan dengan persamaan (c) 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 $t = \dfrac{b - x_1}{x_2 - x_1}$

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

Maka dari persamaan (e) 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.

5.4 Algoritma Keseluruhan

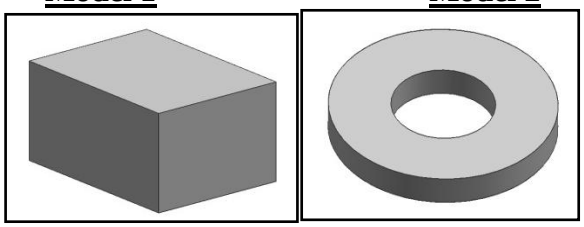
Secara umum proses pembuatan *laser trajectory* ini terdiri dari beberapa tahapan proses berikut,

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.

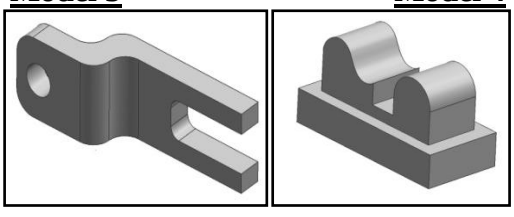
6. Simulasi Program dan Analisa Hasil

Simulasi program dilakukan dengan menggunakan komputer yang memiliki spesifikasi Processor Intel Pentium Dual Core 3.20 GHz dan Memory 512 MB. Skenario simulasi dilakukan dengan tiga model prismatik dan tiga model berkontur dengan nilai parameter *slice interval* dan *hatch space berbeda*.

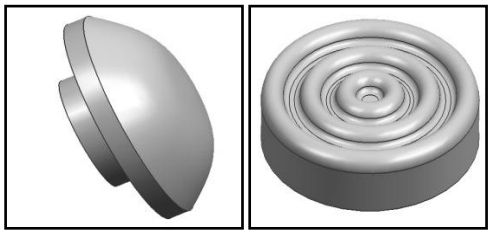
Model 1 **Model 2**



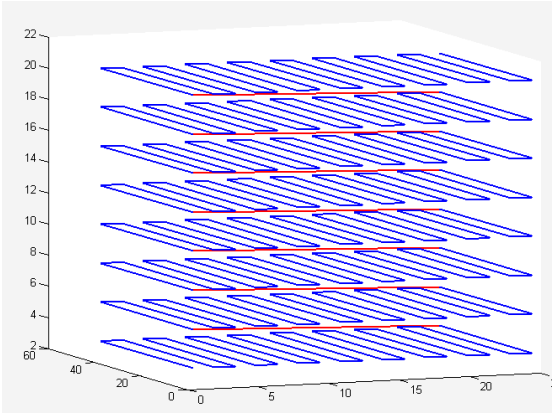
Model 3 **Model 4**



Model 5 **Model 6**



Gambar 16 : Model berkontur dan prismatik



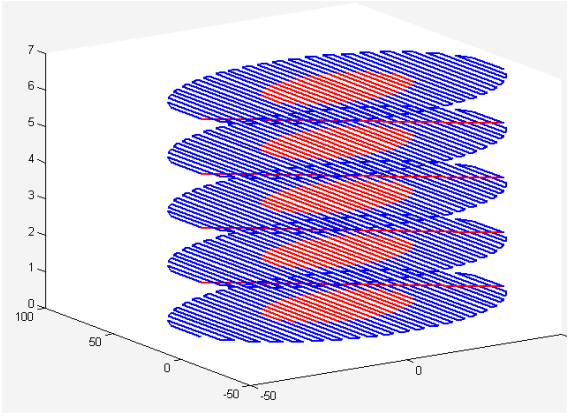
Gambar 17 : Hasil simulasi model 1

Pada simulasi Model 1, parameter *layer thickness* 2.50 mm, dan *hatch space* 1.50 mm, program bekerja membuat lintasan pada model selama 2 s dengan hasil yang baik. Hasil dari gambar 17 menunjukkan lintasan laser dengan arah *directional parallel*. Ada dua warna yang terdapat pada lintasan laser, warna biru menunjukkan lintasan laser untuk setiap layer dengan pacaran *laser on*, sedangkan warna biru merupakan gerak lintasan laser menuju ke posisi layer selanjutnya dengan kondisi *laser off*. Jumlah layer yang terbentuk berjumlah 8 layer, jumlah ini berdasarkan dari jarak pada sumbu-z dibagi dengan *layer thickness*. Semakin besar nilai *layer thickness* yang dimasukkan maka akan mengurangi jumlah layer. Dengan kata lain jumlah layer akan bergantung pada nilai *layer thickness* yang dimasukkan. Jumlah lintasan untuk masing-masing layer yang terbentuk berjumlah 17 lintasan, jumlah ini bergantung pada jarak sumbu-x dibagi dengan hatch space. Semakin kecil nilai *hatch space* maka akan menambah jumlah lintasan masing-masing layer.

```
L00 1
G00 0.750000:2.000000:3.250000
G01 0.750000:42.000000:3.250000
G01 2.250000:42.000000:3.250000
G01 2.250000:2.000000:3.250000
G01 3.750000:2.000000:3.250000
G01 3.750000:42.000000:3.250000
G01 5.250000:42.000000:3.25000
……
LOO 2
G00 0.750000:2.000000:5.750000
G01 0.750000:42.000000:5.750000
G01 2.250000:42.000000:5.750000
G01 2.250000:2.000000:5.750000
……
LOO 3
G00 0.750000:2.000000:8.250000
G01 0.750000:42.000000:8.250000
G01 2.250000:42.000000:8.250000
G01 2.250000:2.000000:8.250000
……
LOO 4
G00 0.750000:2.000000:10.750000
G01 0.750000:42.000000:10.750000
G01 2.250000:42.000000:10.750000
G01 2.250000:2.000000:10.750000
……
LOO 5
G00 0.750000:2.000000:13.250000
G01 0.750000:42.000000:13.250000
G01 2.250000:42.000000:13.250000
G01 2.250000:2.000000:13.250000
……
LOO 6
G00 0.750000:2.000000:15.750000
G01 0.750000:42.000000:15.750000
G01 2.250000:42.000000:15.750000
G01 2.250000:2.000000:15.750000
……
LOO 7
G00 0.750000:2.000000:18.250000
G01 0.750000:42.000000:18.250000
G01 2.250000:42.000000:18.250000
G01 2.250000:2.000000:18.250000
……
LOO 8
G00 0.750000:2.000000:20.750000
G01 0.750000:42.000000:20.750000
G01 2.250000:42.000000:20.750000
G01 2.250000:2.000000:20.750000
……
G01 24.750000:2.000000:20.750000
G01 24.750000:42.000000:20.75000
```

Gambar 18 : File output model 1 dengan kode mesin

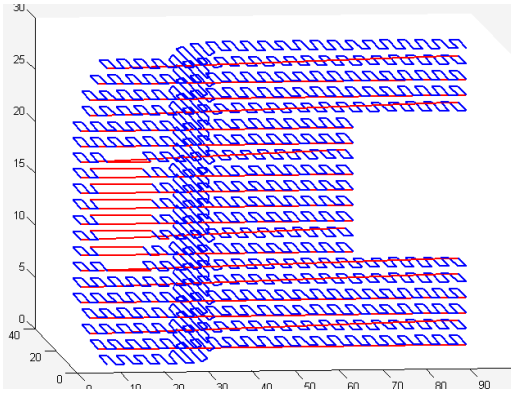
Seperti yang telah dikemukakan sebelumnya bahwa program yang dihasilkan berupa plot lintasan laser dan file output yang berisi *G-Code* untuk mesin yang diikuti nilai koordinat. Pada gambar 18 menunjukkan file output untuk model ini, didalam file tersebut berisi kode L00 sekaligus menunjukkan tanda lokasi layer ke berapa, diikuti G00 yang berarti gerak laser menuju koordinat tujuan atau layer berikutnya dengan kondisi *laser off*. Sedangkan G01 merupakan gerak laser menuju koordinat tujuan dengan kondisi *laser on*.



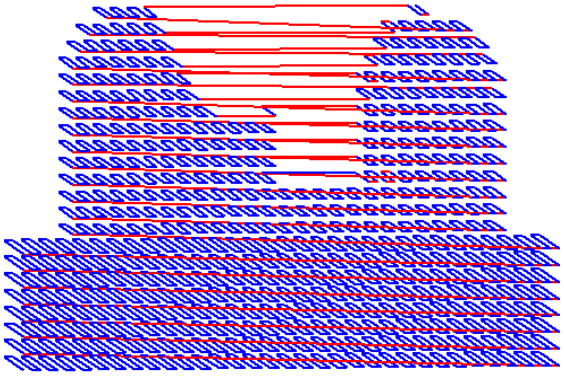
Gambar 19 : Hasil simulasi model 2

Pada gambar 19 merupakan hasil program pada model yang berbentuk seperti cincin. Proses *generate* dilakukan dengan nilai *layer thicknes* 1.50 mm, dan *hatch space* 2.50 mm. Lintasan laser terbentuk pada model setelah proses berjalan 10 s, dan hasilnya cukup baik walaupun model 2 ini terbentuk dari kurva lingkaran. Jumlah layer sebanyak 5 merupakan hasil bagi dari jarak sumbu-z terhadap nilai *layer thickness*. Sedangkan jumlah lintasan sebanyak 40 merupakan hasil dari pembagian diameter lingkaran dengan *hatch space*.

Dari plot yang dihasilkan terdapat dua warna garis. Garis biru merupakan garis lintasan dengan kondisi *laser on*, dan garis merah merupakan garis lintasan dengan *laser off*. Karena model 2 ini merupakan bentuknya cincin yang tengah-tengahnya berlubang, maka hasil *laser trajectory* pada bagian lubang ditengah akan terbentuk garis warna merah, karena bagian tersebut dinyatakan dengan kondisi *laser off*.



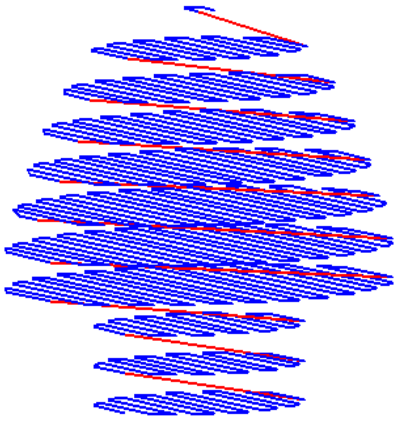
Gambar 20 : Hasil simulasi model 3



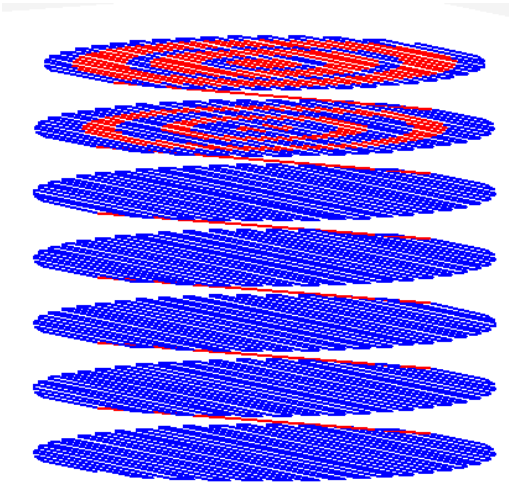
Gambar 21 : Hasil simulasi model 4

Hasil gambar 21 menunjukkan lintasan laser memiliki pola yang teratur walaupun pada benda berkontur. Seperti pada model prismatik, jumlah layer tergantung dari nilai *layer thickness* yang dimasukkan, sedangkan jumlah lintasan untuk tiap-tiap layer tergantung nilai *hatch space*. Pada bagian berkontur akan nampak seperti efek bertangga (*stair step effect*) dari setiap layer. Efek ini dapat menyebabkan berkurangnya kualitas permukaan produk yang dihasilkan. Pengaruh ini dapat dikurangi dengan memperkecil nilai *layer thickness* yang dimasukkan. Tetapi akibatnya akan memperlama proses pembuatan karena semakin banyak jumlah layer yang harus di buat lintasannya.

Waktu yang diperlukan untuk proses pembuatan model ini sekitar 6 s, dengan kondisi nilai parameter *layer thickness* 1.50 mm dan *hatch space* 1.00 mm. Nilai ini terlalu besar dan sangat-sangat tidak akurat jika dibuat dalam *prototyping*. Nilai yang dimasukkan ini sebatas untuk simulasi dan pemodelan agar mudah dilihat lintasannya.



Gambar 22 : Hasil simulasi model 5



Gambar 23 : Hasil simulasi model 6

Model 6 pada gambar 23 memperlihatkan *laser trajectory* dengan nilai *layer thickness* 2.00 mm, dan *hatch space* 2.00 mm. Waktu proses pembuatan model ini sekitar 92 s.

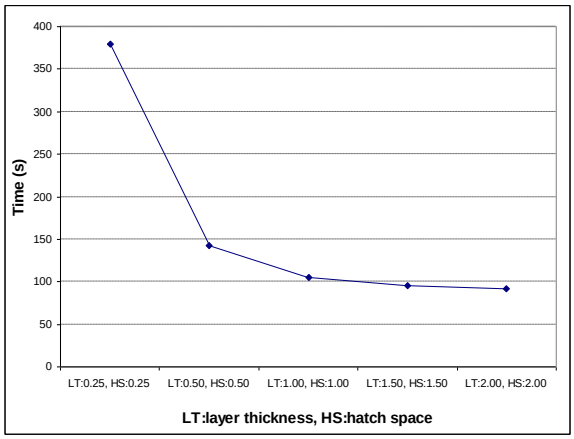
Bentuk permukaan atas dari model ini sebenarnya bergelombang, tetapi karena bentuk grafik di software pemrograman ini tidak diatur sehingga seakan-akan seperti ada celah ditengah. Namun bentuk lintasan yang ada sudah tampak jelas dan mengikuti pola yang sesuai dengan algoritma.

Lintasan setiap layer terdiri dari garis-garis lurus dengan jarak antar garis disesuaikan dengan nilai hatch space. Lintasan tersebut dibatasi oleh lingkaran yang terbentuk dari titik-titik hasil perpotongan model *facet* terhadap bidang-z. Dengan merubah nilai layer *thickness* dan *hatch space* dari model 6 tersebut dapat diperoleh hubungan antara nilai parameter dengan waktu generate trajectory model.

Tabel 1 Hubungan nilai parameter terhadap waktu generate trajectory model 6 (40 x 20 mm)

Pengujian	Parameter *Prototyping*		Jumlah *Layer*	Waktu *generate*
	Layer Thickness	*Hatch Space*		
Simulasi 1	2.00 mm	2.00 mm	7	92 s
Simulasi 2	1.50 mm	1.50 mm	9	95 s
Simulasi 3	1.00 mm	1.00 mm	14	105 s
Simulasi 4	0.50 mm	0.50 mm	28	142 s
Simulasi 5	0.25 mm	0.25 mm	56	380 s

Dari tabel 1 menunjukkan adanya hubungan antara nilai parameter *prototyping* dengan jumlah layer dan waktu *generate* model berkontur. Semakin kecil nilai parameter yang dimasukkan maka semakin banyak jumlah layer yang dihasilkan dan semakin besar waktu yang dibutuhkan untuk membuat model tersebut. Pada gambar 24 terlihat grafik hubungan antara parameter prototyping terhadap waktu generate. Hubungan ini tidek linier terutama pada nilai layer thickness dan hatch space yang kecil terlihat waktu yang semakin lama.

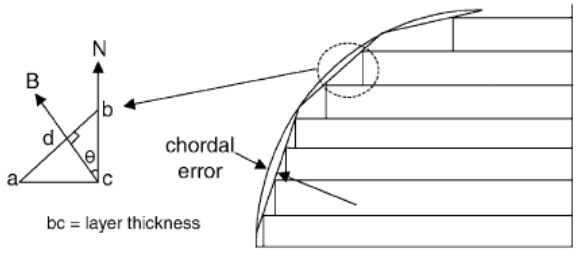


Gambar 24 : Grafik hubungan antara parameter prototyping terhadap waktu.

Keakuratan permukaan dapat dijelaskan sebagai deviasi geometri dari model CAD sebelumnya terhadap produk. Kerugian keakuratan merupakan *error* yang diperoleh dari tiga tahapan proses, yaitu : (1) *pre-process errors*, (2) *process planning errors*; dan (3) *post-process errors*.

Pre-process errors merupakan *errror* yang tidak bisa dipisahkan dari representasi model produk sistim CAD untuk tujuan pertukaran data. Permukaan model produk biasanya direpresentasikan dalam format STL, yang berisi sekumpulan *facet* segitiga. *Errror* ini akibat dari nilai *chordal error* pada saat *exporting* file ke format STL. Biasanya nilai ini diset sebelum triangulasi model CAD dilakukan. Semakin kecil nilai *chordal* maka semakin baik permukaan yang terfasetisasi. Penelitian mengenai pengaruh *error* ini belum dilakukan pada saat ini, tetapi dari penelitian ini bisa didapatkan gambaran mengenai pengaruh *error* awal untuk penelitian lebih lanjut.

Process planning error, *error* ini terbentuk pada bagian yang mengalami "*stair-step effect*", seperti ditunjukkan dalam gambar 25, dan pengaruhnya akan menjadi lebih jelas dan nyata pada permukaan berkontur. *Error* ini dapat dihitung dengan tinggi *cusp*, yaitu jarak antara yang diharapkan dengan pendekatan permukaan pada tiap *facet*.



Gambar 25 : Stairs-step effect terhadap ketebalan layer [6]

Dengan gambar 25, *process planning error* bisa ditentukan dengan menghitung nilai *Cusp height* sebagai berikut [6],

Cusp height (h_c) = $bc \cos \theta$ *(f)*
Maximum cusp height (MCH) = max (h_c) *(g)*

$$\text{Average cusp height (ACH)} = \frac{\sum_{i=1}^{N_f} h_{ci}}{N_f} \qquad (h)$$

Jadi error pada saat perencanaan proses dapat dihitung dari nilai rata-rata *cusp height* yang mencerminkan pertengahan deviasi linier dari semua *facet*, dan nilai error maksimum terjadi saat *cusp height* maksimum. Untuk itu penelitian mengenai nilai *error* ini bisa dilakukan pada penelitian lebih lanjut, karena titik potong untuk setiap z dari algoritma ini telah diketahui.

Post-process error termasuk *error* proses, *shrinkage* dan *warpage*. *Error* proses terutama pada mekanisme penggerakan (*deliver*) laser dan sudut dorongan (*induced angle*) dengan permukaan produk yang dibuat. Error *shrinkage* terutama pada solidifikasi produk yang dibuat. Untuk proses *rapid prototyping* yang menggunakan energi panas untuk solidifikasi/sinter material, seperti SLS, *prototype* cenderung untuk menyusut setelah mendingin. *Warpage* adalah bentuk lain ketidakakuratan yang disebabkan oleh distribusi yang tidak seimbang dari energi panas dan resultan gaya pengikatan. *Error* ini jelas belum bisa dihitung jika belum diimplementasikan ke mesin. Diharapkan dengan adanya pengembangan ini berlanjut ke pembuatan mesin *rapid prototyping* untuk diperoleh hasil yang sempurna dari rangkaian penelitian.

7. Kesimpulan

Dari hasil simulasi, penulis dapat menyimpulkan sebagai berikut:

1. Algoritma yang dikembangkan sudah dapat membuat laser trajectory untuk produk berkontur dan prismatik dengan arah lintasan *directional parallel*, dan seluruh bagian dalam dari kontur tiap layer terisi penuh dengan lintasan.
2. Untuk melakukan proses *slicing* maka informasi *index segitiga, index vertex* dan *koordinat vertex* perlu disimpan dahulu dalam sebuah struktur data vektor.
3. Parameter prototyping seperti: ketebalan *layer* (*layer thickness*), *hatch space* dan orientasi model produk sangat penting dalam *rapid prototyping* karena mempengaruhi keakuratan permukaan dan waktu proses pembuatan.
4. Keakuratan permukaan dapat dijelaskan sebagai deviasi geometri dari model CAD sebelumnya terhadap produk yang menyebabkan kerugian keakuratan. Kerugian keakuratan, pertama tergantung pada proses awal karena pertukaran data pada sistim CAD ke format STL. Kerugian kedua pada tahap proses perencanaan dimana pada bagian produk berkontur akan terbentuk effek tangga bertingkat (*stair-step*) yang tampak jelas akibat *layer thickness* yang besar. Dan kerugian ketiga pada saat proses, kerugian pada saat proses terutama pada mekanisme penggerakan (*deliver*) laser dan sudut dorongan (*induced angle*) dengan permukaan produk dan kerugian akibat proses solidifikasi.
5. Parameter *layer thickness* akan mempengaruhi jumlah layer, semakin kecil parameter ini dimasukkan maka semakin banyak jumlah layer yang dihasilkan, menyebabkan semakin mendekati model sebenarnya sehingga effek bertangga bisa dikurangi untuk model berkontur. Tetapi dengan banyaknya jumlah layer perlu diperhatikan waktu pembuatan karena akan semakin lama prosesnya.
6. Parameter *hatch space* akan mempengaruhi jumlah lintasan untuk setiap layer. Nilai yang kecil menyebabkan jumlah lintasan semakin banyak sehingga jarak total dari *laser trajectory* semakin panjang.

7. Pemilihan orientasi pembuatan pada sumbu-z memudahkan proses *slicing* untuk setiap layer, karena bidang potong tegak lurus dengan sumbu-z.

8. Saran Penelitian Lebih Lanjut

Penelitian ini baru sebatas pembuatan *laser trajectory* untuk produk prismatik dan produk berkontur dalam bentuk simulasi grafik dan file *G-Code* dengan format kode mesin dan koordinatnya. Untuk itu perlu dilakukan proses pertukaran data ke perangkat ke kontroler mesin agar bisa dilakukan pengujian dan sekaligus sebagai pengembangan mesin *rapid prototyping* demgan mempertimbangkan error yang terjadi. Parameter-parameter lain seyogyanya bisa dikembangkan dengan penambahan parameter pemilihan orientasi yang diinginkan, tidak harus dengan sumbu-z, tetapi bisa fleksibel dengan orientasi produk yang sesuai. Selain itu metode *adaptive slicing* bisa dikembangkan untuk simulasi selanjutnya untuk melengkapi program pengembangan mesin prototyping yang bermutu.

DAFTAR ACUAN

[1]. DeGarmo E.P., Black J.T., Kohser R.A., Klamecki B.E., *Materials and Processes in Manufacturing*, ninth edition, John Wiley & Sons, Inc., 2003, pp. 862.

[2]. Wholers T., *Rapid Prototyping State of The Industry – 1999 World Wide Progress*, RPA-SME Publication, 1999.

[3]. Bourell D.L, Beaman J.J, Marcus H.L, Barlow J.W., *Solid freeform fabrication: an advanced manufacturing approach*, Proceedings of the SFF Symposium, 1990, pp. 1-7.

[4]. Diane A.S., Chu K.R., Montgomery D.C., *Optimising Stereolythography throughput*, Journal of Manufacturing Systems, 1997, vol.16, no.4, , pp.290 – 303.

[5]. Zhou J.G., Hersovici D., *Parameter Tuning and Optimisation for SLA Rapid Prototyping Manufacturing Process*, in: Proceeding of the international Converence on Manufacturing Automation (ICMA'97), 1997, vol.2, , pp.894 – 902.

[6]. Choi S.H., Samavedam S., *Modelling and Optimisation of Rapid Prototyping*, Computer in Industry, Elsevier, 2002, vol.47, pp.39-53.

[7]. Koc B., Lee Y.-S., *Computational Geometric Analysis and Planning for 3D Rapid Prototyping Processes*, Dissertation of Industrial Engineering, North Carolina State University, 2001, pp. 13

[8]. Gandjar K., Mujahid A., *Pengembangan Algoritma Cepat Penentuan Titik Kontak Pahat (cutter contact point) pada Sistim-CAM Berbasis Model Facet 3D Untuk Pemesinan Awal (roughing) dan Akhir (finishing)*, Prociding SNTTM V, UI Jakarta, 2006.

[9]. Kulkarni P., Marsan A., Dutta D., *A review of process planning techniques in layered manufacturing*, Rapid Prototyping Journal, 2000, vol.6, no.1, pp.18–35.

[10]. Dolenc A., Makila I., *Slicing procedures for layered manufacturing techniques*, Computer Aided Design, 1994, vol.26, no.2, pp.119–26.

[11]. Kulkarni P., Dutta D., *An Accurate Slicing Procedure for Layered Manufacturing*, Computer-Aided Design, 1996, vol. 28, no. 9, pp.683-697.

[12]. Choi S.H., Kwok K.T., *Hierarchical Slice Contours for Layered-Manufacturing*, Computer in Industry, Elsevier, 2002, vol.48, pp.219-239.

[13]. Asiabanpour B., Khoshnevisb B., *Machine path generation for the SIS process*, Robotics and Computer-Integrated Manufacturing, 2004, vol.20, pp.167–175.

[14]. Dutta D., Prinz F.B., Rosen D., Weis L., *Layered Manufacturing: Current Status and Future Trends*, Journal of Computing and Information Science in Engineering, 2001, vol. 1, pp.63-65.

[15]. Choi S.H., Samavedam S., *Visualisation of rapid prototyping*, Rapid Prototyping Journal, 2001, vol. 7, no. 2, pp. 99-114.

[16]. www.rp4baghdad.org, download 06:02:08, jam 14.3