ISBN 978-979-18839-0-0

Algoritma Penentuan *Slicing Interval* Pada Proses *Rapid Prototyping*

Ahmad Kholil, Gandjar Kiswanto*
Jurusan Teknik Mesin, Fakultas Teknik, Universitas Negeri Jakarta
*Departemen Teknik Mesin, Fakultas Teknik, Universitas Indonesia
E-mail : ach_cholil@yahoo.com

Abstrak

Rapid Prototyping merupakan proses fabrikasi suatu part dengan layer by layer, dimana material ditambahkan ke layer secara berturut-turut sesuai dengan lintasan pergerakan laser. Lintasan pergerakan laser atau laser trajectory merupakan representasi dari cutter contact-point (cc-point) yang terbentuk dari perpotongan bidang-z terhadap model facet prismatik atau model facet berkontur. Setiap perpotongan bidang terhadap model facet akan menghasilkan titik potong yang dijadikan boundary dalam membuat cc-point. Untuk melakukan proses slicing maka informasi index segitiga, index vertex dan koordinat vertex perlu disimpan dahulu dalam sebuah struktur data vektor. Variabel interval slicing (layer thickness) akan mempengaruhi keakuran geometrik dari output hasil, semakin kecil variabel maka keluaran hasil akan semakin mendekati model sebenarnya dan pengaruh effek bertangga bisa dikurangi untuk model sculptured.

Keyword: rapid prototyping, slicing interval

1. Pendahuluan

Rapid Prototyping (RP) atau *Layered Manufacturing* (LM) merupakan proses fabrikasi suatu produk dengan *layer by layer*. Prosesnya melibatkan penambahan *raw material* berturut-turut pada *layer*, agar terbentuk produk yang sesuai dengan model. Beberapa perusahaan manufaktur melakukan pengembangan produk yaitu proses dimana konsep produk harus diterjemahkan dari gambar teknik menjadi produk phisik. Produk phisik model pertama atau *prototype* dinamakan *prototyping*. Prototyping menjadi 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* (RP) merupakan metode yang membantu dalam proses pengembangan produk yang mudah dan cepat, sehingga dapat mempengaruhi kepuasan customer dan keuntungan perusahaan dengan membantu perusahaan mendapatkan produk untuk pasar pertama.

Sistim *Rapid Prototyping* menerima input sebuah objek *parts* 3D berupa sebuah file dengan format STL, sehingga model CAD perlu disimpan dalam bentuk file STL. 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. Format ini adalah yang paling umum digunakan karena lebih mudah dibaca dan dimengerti, serta dapat dibuka di *text editor*.

Lintasan pergerakan laser atau *laser trajectory* merupakan representasi dari *cutter contact-point* (*cc-point*) yang terbentuk dari perpotongan bidang-z terhadap model *facet* prismatik atau model *facet* berkontur *(sculptured)*. Setiap perpotongan bidang terhadap model *facet* akan menghasilkan titik potong yang dijadikan *boundary* dalam membuat *cc-point*.

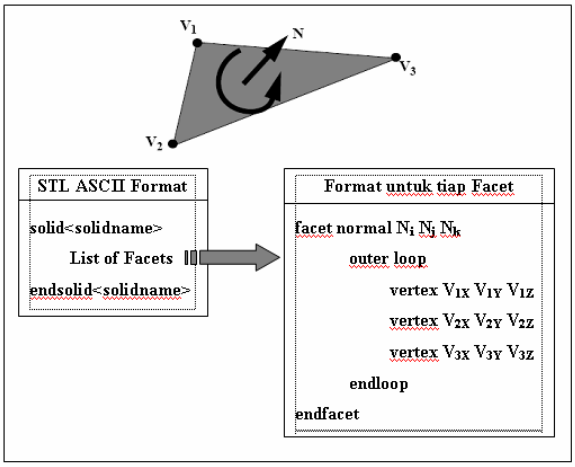
Model-model prismatik yang sederhana hanya akan memiliki sedikit *facet*, karena permukaannya lebih mudah terisi dengan segitiga *facet*. Sedangkan model berkontur bisa memiliki banyak *facet*, karena permukaannya tidak rata dan berkontur menyebabkan segitiga *facet* yanga mengisi permukaan akan lebih banyak dan ukuranya kecil-kecil. Hal ini akan menyulitkan untuk menentukan titik potong pada bidang-z tehadap segitiga *facet*. Pada bagian tersebut ada kemungkinan titik potong bertemu pada sisi-sisi *facet* atau pada titik-titi vertex dari segitiga. Untuk itu diperlukan suatu algoritma yang bisa membuat lintasan laser untuk model prismatik dan model berkontur dengan mempertimbangkan parameter *layer thickness*.

2. Tujuan Penelitian

Tujuan dari penelitian ini adalah menentukan algoritma yang tepat pada proses *slicing* model STL untuk produk prismatik dan produk berkontur (*sculptured*). Penelitian ini sebagai dasar tahapan penelitian selanjutnya untuk mengembangkan lintasan laser (*laser trajectory*) dengan metode lintasan *directional-parallel*.

3. Metode Penelitian

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 1. Format STL ASCII

File STL ini menyimpan informasi objek produk dalam bentuk model *facet* 3D. Model *facet* sendiri adalah suatu model atau bentuk permukaan luar bidang yang tersusun dari satu atau lebih segitiga. Asal mula segitiga-segitiga ini adalah titik-titik (*vertex*) yang menyusun model, dan dihubungkan dengan garis-garis yang akan menjadi sisi (*edge*) segitiga, sehingga terbentuklah segitiga-segitiga yang saling berhubungan dan membentuk sebuah permukaan bidang yang lebih dikenal sebagai model *facet*. Ada dua fungsi utama dari pembentukan segitiga ini. Yang pertama adalah sebagai penghubung antar vertex untuk membuat sebuah permukaan, dalam hal ini yang menjadi permukaan adalah bidang segitiga (*face*). Fungsi yang kedua adalah untuk menentukan vektor normal bidang pada wilayah tertentu. Vektor normal tersebut merupakan vektor normal segitiga, yang didapat melalui *cross product* antara 2 vektor pembentuk sisi segitiga. Arah dari vektor normal bergantung pada arah putaran vektor dari ketiga vertex yang digunakan. Arah vektor normal terhadap putaran vektor pembentuk segitiga dapat ditentukan mengikuti kaidah tangan kanan. Jika putaran searah dengan jarum jam (*clockwise*), maka vektor normal akan menuju bidang. Sebaliknya, jika putaran berlawanan arah jarum jam (*counter clockwise*), maka vektor normal keluar dari bidang.

3.1. 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 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	……	……..	……

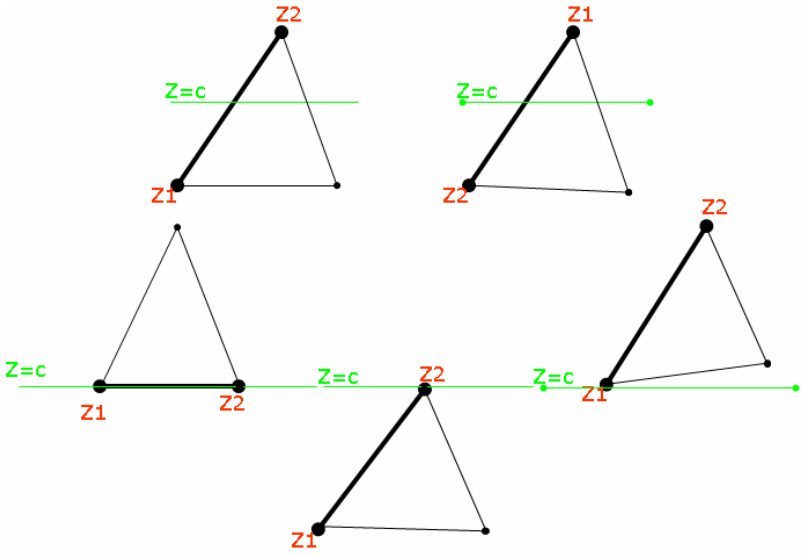
Index vertex	Koordinat x	Koordinat y	Koordinat z
1	0.12	0.14	0.32
2	0.29	0.33	0.45
dst	……	……..	……

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



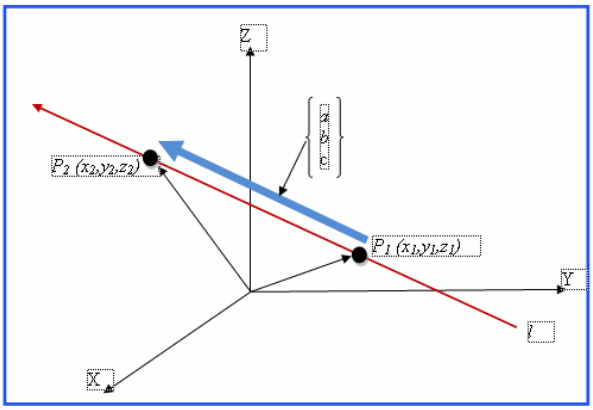
Gambar 2. Titik perpotongan segitiga yang mungkin.

3.2. 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 3. Persamaan garis pada ruang R3[1].

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

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

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

Dengan demikian persamaan (1) dapat disubstitusi menjadi

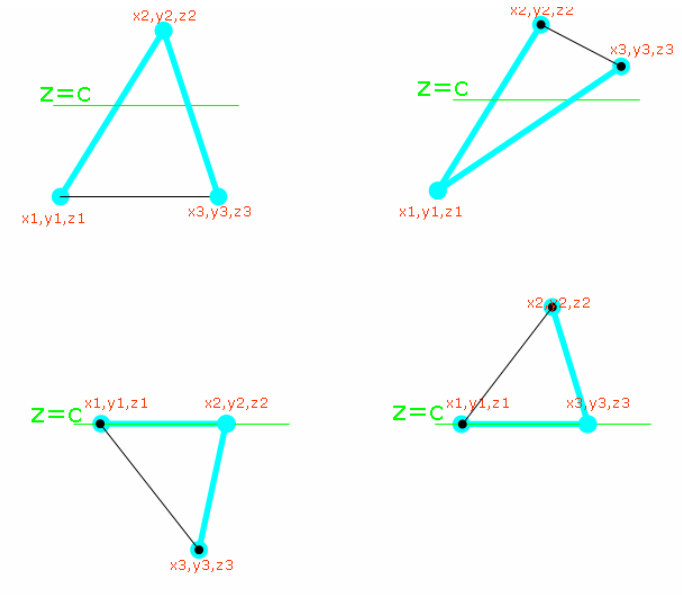
$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \left(\begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} - \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \right) t + \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \quad (3)$$

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

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

Maka dari persamaan (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

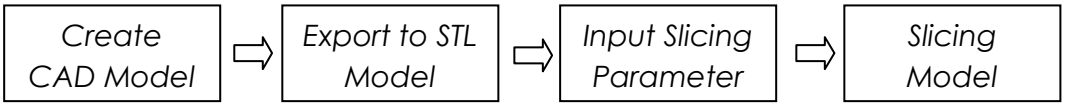


Gambar 4. Penentuan titik selanjutnya.

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*.

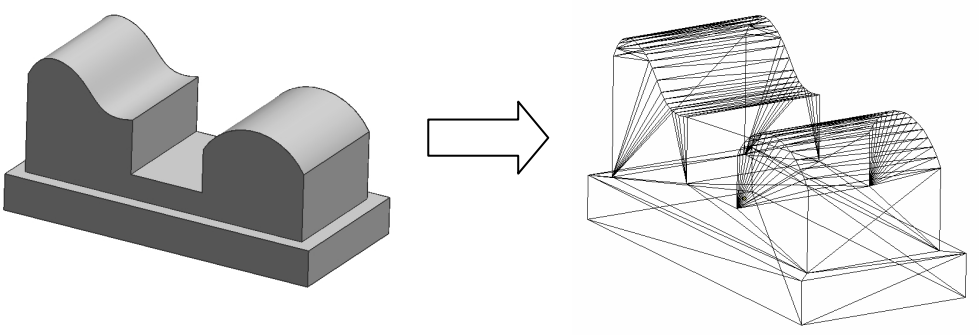
4. Algoritma ini berjalan terus sampai diketemukannya seluruh titik potong dengan sumbu-z. dan akan berakhir sampai pada z-maksimum.

Setelah algoritma dibuat dengan salah satu software, program dijalankan dengan mengikuti tahapan berikut,



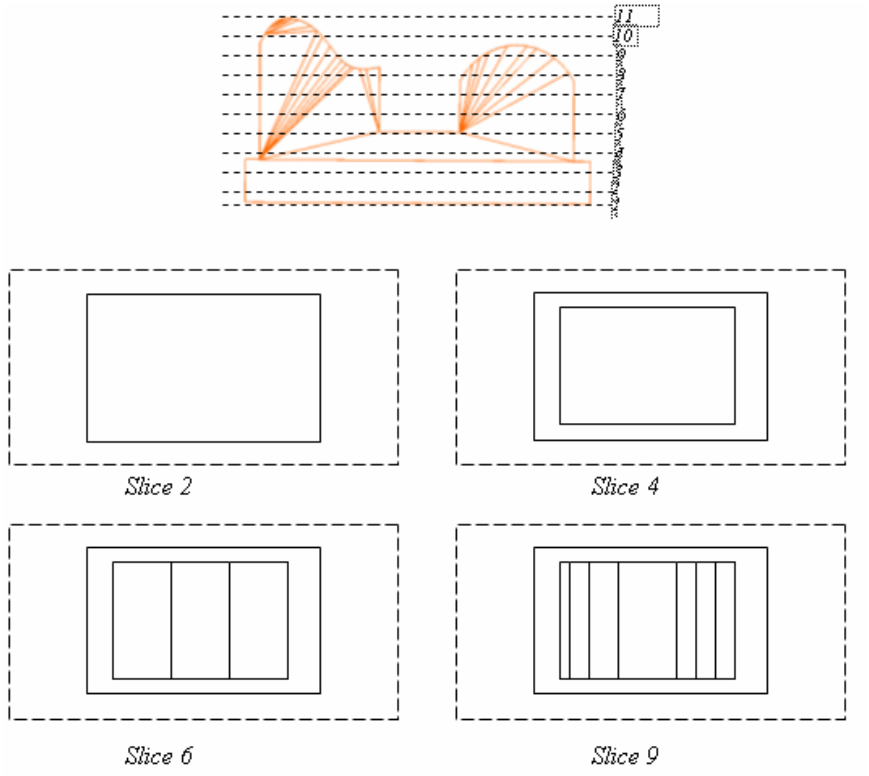
4. Hasil dan Pembahasan

Berikut ini salah satu contoh model yang dijadikan file untuk simulasi, model dibuat dengan software berbasis CAD yang memiliki fasilitas export data CAD ke model *facet*.



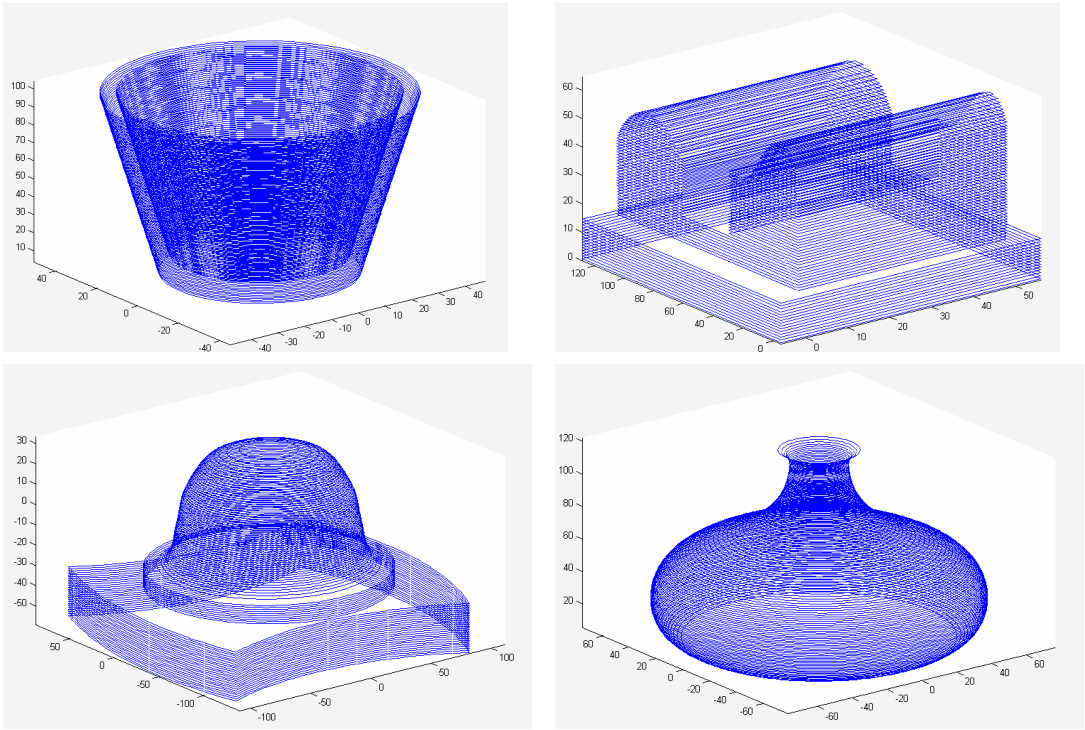
Gambar 5. Transfer model CAD ke model STL

Kemudian model *facet* dipotong dengan bidang-z. Interval *slicing* dimulai dari bawah z = 0 sampai z = maksimum model, dan pada penelitian ini interval *slicing* dibuat seragam.



Gambar 6. Proses *slicing* dengan interval seragam

Hasil dari software yang dikembangkan dengan beberapa contoh model dapat terlihat pada gambar berikut. Hasilnya setiap model memiliki layer yang seragam sesuai dengan interval *slicing* yang dimasukkan pada algoritma. Waktu proses *slicing* untuk setiap model tergantung besar kecilnya ukuran file. Semakin besar ukuran file maka semakin lama waktu prosesnya, hal ini karena jumlah segitiganya yang banyak menyebabkan jumlah titik potong setiap layer menjadi banyak.



Gambar 7. Hasil simulasi dengan interval *slicing* seragam

Dari hasil simulasi, penulis dapat menyimpulkan bahwa algoritma yang dibuat dapat digunakan pada model-model *facet*. Proses *slicing* terhadap model berjalan sesuai dengan tujuannya untuk menghasilkan layer seragam. Untuk melakukan proses *slicing* maka informasi index segitiga, index vertex dan koordinat vertex perlu disimpan dahulu dalam sebuah struktur data vektor. Variabel interval *slicing* akan mempengaruhi keakuran geometrik dari output hasil, semakin kecil variabel maka keluaran hasil akan semakin mendekati model sebenarnya dan pengaruh effek bertangga bisa dikurangi pada model *sculptured*. Hal ini merupakan dasar pengembangan untuk membuat software yang mampu membuat lintasan laser (*laser trajectory*) proses rapid prototyping dengan kontur tiap layer terisi penuh.

Daftar Pustaka

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.

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.

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

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

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

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

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.

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