

Pengembangan metode simulasi pergerakan pahat menggunakan VTK pada sistem-CAM berbasis model faset 3D

Gandjar Kiswanto, A. Mujahid
Laboratorium Teknologi Manufaktur
Departemen Teknik Mesin, Universitas Indonesia
Kampus Baru UI Depok 16424
E-mail : gandjar_kiswanto@eng.ui.ac.id

Abstrak

Dengan meningkatnya kompleksitas produk akhir yang diinginkan, kehandalan sistem-CAM untuk pemesinan multi-axis menjadi hal yang utama untuk menjamin keakurasian produk akhir. Laboratorium Teknologi Manufaktur, Departemen Teknik Mesin – Universitas Indonesia, mengembangkan suatu sistem-CAM untuk pemesinan milling multi-axis (3-5 axis) berbasis model faset 3D yang handal dan memiliki kecerdasan dalam otomasi perencanaan strategi pembuatan lintasan pahat. Salah satu hal penting lainnya yang harus hadir didalam sistem-CAM adalah kemampuan melakukan simulasi pergerakan pahat terhadap benda-kerja (workpiece). Pada pemesinan milling khususnya multi-axis, simulasi merupakan kebutuhan mutlak untuk memberikan gambaran perubahan posisi dan orientasi pahat (tool-posture) terhadap benda-kerja selama proses pemotongan. Penelitian ini dilakukan untuk mengembangkan metode simulasi pergerakan pahat kontinyu dengan menggunakan VTK (Visualisation Tool Kit) dan berbasis J2SDK. Proses rendering dilakukan untuk memperhalus penampakan pahat sehingga terlihat lebih realistis. Data simulasi sendiri diambil dari nilai posisi dan orientasi pahat untuk setiap titik kontak pahat (cc-point). Titik tengah ujung pahat bagian bawah untuk pahat flat-end dan titik tengah sphere untuk pahat ball-end/nose ditetapkan sebagai posisi pahat, sedangkan orientasi pahat ditetapkan mengikuti normal vektor dari cc-point sebagai tool-axis sebagai mana mestinya proses pemesinan milling multi-axis. Proses simulasi dilakukan dengan menggerakkan pahat dari satu cc-point ke cc-point berikutnya. Hasil proses simulasi ini sangat membantu user dalam melihat dan menentukan apakah lintasan pahat yang dirancang berikut perubahan orientasi pahat telah sesuai dengan yang diharapkan baik berdasarkan pengalaman ataupun pengetahuan (knowledge-based) sehingga dapat direalisasi hingga proses pemesinan aktual di mesin NC.

Kata kunci: model faset 3D, simulasi pergerakan pahat, VTK

Pendahuluan

Sistem-CAM (*Computer-Aided Manufacturing*) merupakan suatu 'tool'/software yang biasa digunakan dalam pembuatan lintasan pahat. Umumnya digunakan pada pembuatan lintasan pahat pemesinan *milling*. Departemen Teknik Mesin – Universitas Indonesia sedang mengembangkan sistem-CAM yang handal berbasis model faset 3D untuk pemesinan *milling multi-axis* (3-5-axis). Salah satu komponen utama dalam suatu sistem-CAM yang penting untuk menghindari terjadinya kesalahan dalam pembuatan lintasan pahat adalah sistem verifikasi/validasi dari pergerakan pahat.

Pada pemesinan *milling 5-axis* kemungkinan terjadinya *collision* (tabrakan) antara pahat dengan benda-kerja dan/atau dengan komponen mesin lainnya lebih tinggi dibandingkan dengan pemesinan *milling 3-axis*. Hal ini dikarenakan pada *milling 5-axis*, dapat terjadi perubahan orientasi pahat yang signifikan dari satu *cc-point* ke *cc-point* berikutnya. Bila hal ini tidak diketahui dari awal, resiko terjadinya *collision* menjadi sangat besar saat NC-code dijalankan pada mesin CNC. Oleh karena itulah sistem simulasi pergerakan pahat didalam sistem-CAM untuk mengantisipasi hal tersebut adalah suatu hal yang mutlak diperlukan.

Pengembangan Sistem Simulasi Pergerakan Pahat

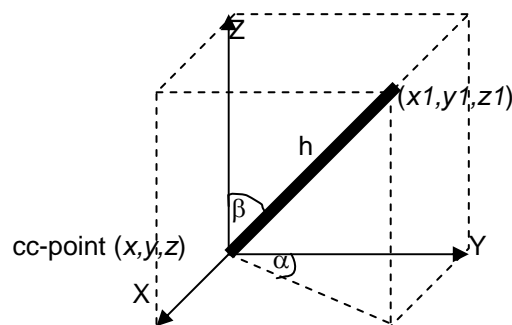
Simulasi yang dibuat dalam penelitian ini merupakan bagian dari proses *finishing* (pemesinan akhir) pada *milling* menggunakan mesin 5-axis dan merupakan rangkaian penelitian

dalam pengembangan sistem-CAM berbasis model faset 3D yang dikembangkan oleh Departemen Teknik Mesin, Universitas Indonesia. Selain mempertimbangkan koordinat x , y , dan z , simulasi juga mempertimbangkan vektor normal dari masing-masing *cc-point* pada lintasan pahat yang telah dibuat. Perhitungan untuk mendapatkan nilai vektor normal pada model faset tidak akan dijelaskan dalam makalah ini.

Untuk menampilkan simulasi pergerakan pahat diperlukan paling tidak tiga komponen, yaitu visualisasi model pahat, visualisasi garis potong vertikal, dan visualisasi pergerakan pahalanya sendiri. Selain ketiga komponen tersebut, tentunya diperlukan pula data berupa titik-titik potong pahat dengan model, atau yang sering disebut *cutter-contact points (CC-points)*. CC-point ini akan menjadi acuan atau landasan bagi pergerakan pahat. Berikut ini adalah penjelasan masing-masing komponen di atas.

Pertama, visualisasi model pahat dibuat menggunakan bangun silinder atau tabung dengan besar jari-jari dan tinggi bangun disesuaikan dengan data yang dimasukkan oleh *user*. Untuk simulasi ini, akan ditampilkan model pahat bertipe *ball-end*, sehingga agar mirip dengan model yang semestinya, pada ujung silinder 'ditempelkan' bangun setengah bola. Jadi, dua bangun yang digunakan sebagai model pahat adalah silinder dan setengah bola.

Untuk membuat visualisasi bangun silinder ini, digunakan class dari VTK (*Visualization Toolkit*) yaitu *vtkLineSource* dan *vtkTubeFilter*. *vtkLineSource* berfungsi membuat objek garis (*source object*) yang ditentukan oleh dua buah titik sebagai titik-titik ujungnya. Titik pertama adalah CC-point, karena yang dikehendaki dalam simulasi ini adalah persentuhan model pahat dengan CC-point. Titik kedua ditentukan dengan menghitung koordinat CC-point ditambah dengan tinggi model pahat atau silinder. Titik pertama dapat didapatkan dengan mudah dari data-data CC-point yang sudah disediakan. Berbeda halnya untuk mendapatkan titik kedua. Tinggi model pahat atau silinder terlebih dahulu harus diproyeksikan pada masing-masing sumbu X, Y, dan Z. Hasil proyeksi ini kemudian baru bisa ditambahkan koordinat CC-point. Untuk lebih jelasnya perhatikan ilustrasi berikut.



Jadi, titik $(x1, y1, z1)$ diperoleh melalui

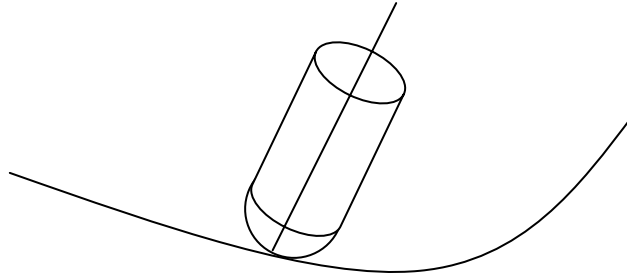
$$\begin{aligned}x1 &= x + h \cdot \sin\beta \cdot \sin\alpha \\y1 &= y + h \cdot \sin\beta \cdot \cos\alpha \\z1 &= z + h \cdot \cos\beta.\end{aligned}$$

vtkLineSource merupakan turunan dari class *vtkSource*. Pada setiap proses *rendering* grafik pada VTK, masukan-masukannya (*inputs*) selalu diawali dari class *vtkSource* atau turunannya. Di sini sudah didapatkan satu *source* menggunakan *vtkLineSource* untuk menjadi masukan bagi proses *rendering* berikutnya.

vtkTubeFilter digunakan sebagai *filter* yang membuat objek silinder dengan objek garis sebagai inputnya. Garis yang menjadi input adalah objek *vtkLineSource* yang sudah dibuat di atas. Garis tersebut kemudian menjadi sumbu vertikal bagi silinder dari alas hingga tutupnya. Silinder tersebut sebenarnya tersusun atas potongan-potongan segitiga (*triangle strips*) yang disusun mengelilingi sumbu. Bagaimana arah mengelilinginya, secara *default* ditentukan oleh normal garis tersebut. *vtkTubeFilter* memiliki properti yang bisa mengatur jari-jari dan tutup

silinder. Besar jari-jari didapat dari masukan yang diberikan *user*, sementara tutup silinder digunakan agar silinder mirip dengan model pahat. Untuk mengatur resolusi, dapat diatur berapa jumlah sisi yang akan ditampilkan oleh *vtkTubeFilter*. Semakin besar jumlah sisi semakin halus sisi silinder, artinya resolusinya semakin bagus.

Bangun kedua adalah bangun setengah bola. Untuk membuat bangun ini diperlukan class *vtkSphereSource*. Class ini memiliki properti seperti radius dan titik pusat yang bisa diatur sesuai kebutuhan. Besar radius mengikuti ukuran jari-jari yang dimasukkan oleh *user*, sementara titik pusat diatur mengikuti CC-point. Dengan demikian pergerakan bola ini akan bersamaan dengan pergerakan silinder.



Gambar 1 : model pahat di atas model

Segment code di bawah ini menunjukkan penjelasan di atas yang ditulis dalam bahasa pemrograman Java.

```
152 ..... vtkLineSource line= new vtkLineSource();
153 ..... line.SetResolution(3);
154 ..... vtkSphereSource sphere= new vtkSphereSource();
155 ..... sphere.SetRadius(radius);
156 ..... sphere.SetThetaResolution(20);
157 ..... sphere.SetPhiResolution(20);
165 ..... line.SetPoint1(v.getX(),v.getY(),v.getZ()+radius);
166 ..... line.SetPoint2(v.getX() + height*v.getI(),
167 ..... v.getY() + height*v.getJ(),
168 ..... v.getZ() + height*v.getK());
169 ..... sphere.SetCenter(v.getX(),v.getY(),v.getZ()+radius);
170 .....
171 ..... vtkTubeFilter tubes= new vtkTubeFilter();
172 » » » tubes.SetInput(line.GetOutput());
173 » » » tubes.SetRadius(radius);
174 » » » tubes.CappingOn();
175 » »
176 » » » tubes.SetNumberOfSides(60);
```

Supaya kelihatan seperti setengah bola, bangun bola ditampilkan beririsan dengan bangun silinder, sehingga sebenarnya titik pertama dari *vtkLineSource* tidak tepat pada CC-point melainkan berjarak sebesar jari-jari bola dari CC-point. Demikian pula dengan titik pusat *vtkSphereSource*.

Komponen **kedua adalah garis potong vertikal**. Yang dimaksud dengan visualisasi garis potong ini adalah visualisasi titik-titik di mana terdapat perpotongan antara pahat dengan model. Titik-titik ini sebenarnya adalah CC-point. Visualisasi CC-point dapat dilakukan melalui dua cara. Pertama digambarkan sebagai deretan titik CC-points, atau kedua digambarkan sebagai deretan garis. Untuk menampilkan CC-points sebagai deretan garis diperlukan class *vtkPoints* dan *vtkLine*. Sementara untuk deretan CC-point diperlukan class *vtkPoints* dan *vtkPolyVertex*.

Class *vtkPoints* digunakan untuk menampung semua titik-titik yang akan ditampilkan. Ini dilakukan melalui method *insertPoint(i, x, y, z)*. Parameter *i* menunjukkan indeks titik tersebut, sedangkan *x, y, z* adalah titik-titiknya.

Berikut ini adalah *segment code* visualisasi CC-point sebagai deretan titik.

```
513 ..... vtkPoints point = new vtkPoints();
514 ..... vtkPolyVertex vertex = new vtkPolyVertex();
524 ..... Vertex v = (Vertex)iterator.next();
525 .....
526 ..... point.InsertPoint(x++, v.getX(), v.getY(), v.getZ());
531 ..... vertex.GetPointIds().SetId(q, q);
```

Untuk menghubungkan objek *vtkPolyVertex* dengan *vtkPoints* dilakukan seperti pada baris 531 di atas (variable *x* dan *q* sama-sama menunjuk pada indeks titik), kemudian dilanjutkan dengan menggunakan class *vtkUnstructuredGrid* seperti gambar berikut ini.

```
534 ..... vtkUnstructuredGrid grid = new vtkUnstructuredGrid();
535 ..... grid.Allocate(1, 1);
536 ..... grid.InsertNextCell(vertex.GetCellType(), vertex.GetPointIds());
537 ..... grid.SetPoints(point);
538 ..... grid.Squeeze();
```

Untuk menampilkan CC-point sebagai deretan garis tidak jauh berbeda dengan cara di atas. Perbedaan hanya terlihat pada *segment code* yang menggunakan class *vtkUnstructuredGrid*.

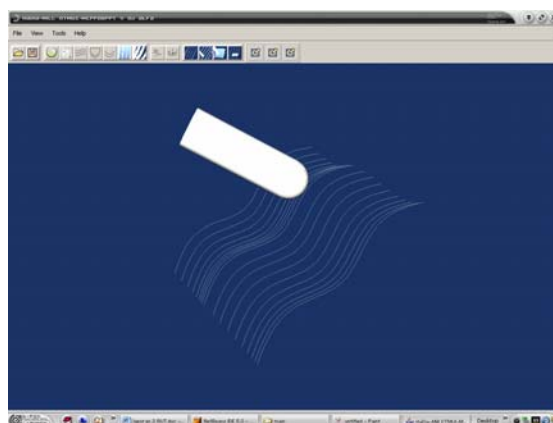
```
438 ..... vtkPoints point = new vtkPoints();
439 ..... vtkLine line = new vtkLine();
468 ..... vtkUnstructuredGrid grid = new vtkUnstructuredGrid();
469 ..... grid.Allocate(1, 1);
470 ..... grid.InsertNextCell(line.GetCellType(), line.GetPointIds());
471 ..... grid.SetPoints(point);
472 ..... grid.Squeeze();
```

Komponen **ketiga** adalah **visualisasi pergerakan pahat**. Jika kedua komponen di atas sudah siap, maka tinggal menggerakkan model pahat di atas model bidang. Hal ini dilakukan dalam sebuah *looping* (perulangan) yang senantiasa mengubah (*update*) nilai titik pertama dari objek *vtkLineSource* sesuai dengan nilai CC-point yang sedang dilewatinya. Perubahan nilai dari titik pertama ini otomatis juga mengubah nilai titik kedua dan titik pusat dari bola (objek *vtkSphereSource*). Perulangan ini dilakukan hingga semua CC-point selesai terlewati.

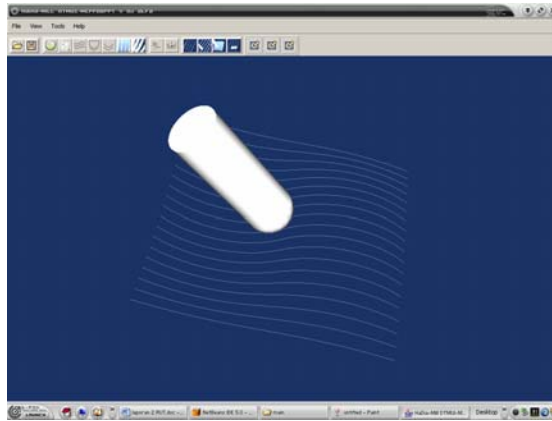
Implementasi dan Hasil

Setelah melalui beberapa tahapan perbaikan maka algoritma simulasi diimplementasi didalam sistem-CAM yang sedang dikembangkan. Dua model faset digunakan sebagai model pengujian pembuatan simulasi pergerakan pahat.

Berikut ini pada Gambar 2 dan 3 adalah hasil visualisasi pergerakan model pahat.



Gambar 2 : Simulasi pergerakan pahat diatas lintasan pahat model A



Gambar 3 : Simulasi pergerakan pahat diatas lintasan pahat model B

Kesimpulan

Sistem simulasi pergerakan pahat didalam sistem-CAM mutlak diperlukan untuk menghilangkan (mengurangi) resiko terjadinya *collision* (tabrakan) antara pahat dengan benda-kerja atau dengan komponen pemesinan lainnya. Pada penelitian ini pengembangan metode simulasi pergerakan pahat kontinyu menggunakan VTK (*Visualisation Tool Kit*) dan berbasis J2SDK. Proses rendering dilakukan untuk memperhalus penampakan pahat sehingga terlihat lebih realistis. Data simulasi sendiri diambil dari nilai posisi dan orientasi pahat untuk setiap titik kontak pahat (cc-point).

Ucapan Terima Kasih

Penulis mengucapkan terima kasih kepada Kementerian Negara Riset dan Teknologi (KNRT) yang telah membiaya sebagian dari penelitian ini melalui Riset Unggulan Terpadu XII 2005-2006.

Daftar Acuan

- [1]. Byoung K.C, Robert B.J, *Sculptured Surface Machining*. Kluwer Academic Publishers, 1998.
- [2]. Dejonghe P., An integrated approach for tool path planning and generation for multi-axis milling, ISBN 90-5682-315-9, PhD-thesis, K.U. Leuven, Leuven 2001.
- [3]. Kiswanto G., Tool path generation for multi-axis milling based on faceted models, ISBN : 90-5682-449-X, K. U. Leuven, Leuven 2003.
- [4]. Lauwers B., Kiswanto G., Kruth J. -P., Development of five-axis milling tool path generation algorithm based on faceted models, *Annals of the CIRP*, vol. 52, no. 1, pp. 85-88, 2003.
- [5]. Hwang J.S., *Interference-free tool-path generation in the NC machining of parametric compound surfaces*, *Computer Aided Design*, 1992, vol. 24, no. 12, pp. 675-676.
- [6]. Hwang J.S., Chang T.-C., *Three-axis machining of compound surfaces using flat and filleted endmills*, *Computer Aided Design*, 1998, vol. 30, no. 8, pp. 641-647.
- [7]. Jensen C.G., Mullins S.H., Anderson D.C., *Scallop elimination based on precise 5-axis tool placement, orientation and step-over calculations*, *ASME-Advances in Design Automation*, 1993, vol. 65-2, pp. 535-544.
- [8]. Kiswanto G., Kruth J.-P., Lauwers B., *Tool path generation for 5-axis milling based on faceted models*, *Journal of Engineering*, Vol.1., 2002.
- [9]. Kruth J.-P., Klewais P., *Optimization and dynamic adaptation of the cutter inclination during 5-axis milling of sculptured surfaces*, *Annals of CIRP*, 1994, vol. 43/1, pp. 443-448.
- [10]. Lai J.-Y., Wang D.-J., *A strategy for finish cutting path generation of compound surfaces*, *Computer in Industry*, 1994, no. 25, pp. 189-209.
- [11]. <http://www.vtk.org>