Makalah Nomor: KNSI-171

PAINTERLY RENDERING DENGAN MEDIA CAT , PENSIL, KRAYON, DAN TINTA

Kartika Gunadi[1], Rudy Adipranata[2], Amanda[3]

[1,2] Jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra,
[3] Siwalankerto 121, Surabaya 60236
[1] kgunadi@petra.ac.id, [2] rudya@petra.ac.id

Abstrak

Non-photorealistic rendering (NPR) adalah teknik untuk memanipulasi gambar foto realis menjadi gambar tidak realis. Salah satu bentuk *NPR* adalah *artistic rendering* atau disebut juga *painterly rendering*. Media lukisan yang ditirukan adalah cat, pensil, krayon, dan tinta.
Proses untuk menirukan efek media lukisan dilakukan dengan mengimplementasikan berbagai algoritma dasar pengolahan citra digital, dan berbagai algoritma yang memberikan efek dari media lukisan. Aplikasidibuat denganC# dengan Microsoft Visual Studio 2005 sebagai IDE-nya.
Hasil pengujian menunjukkan bahwa proses *painterly rendering* telah mampu menirukan karakteristik dari media lukisan dengan baik.

Kata kunci : *Non-Photorealistic Rendering*, *Painterly Rendering*

1. Pendahuluan

Non-photorealistic rendering (NPR) adalah teknik memanipulasi gambar realis menjadi gambar tidak realis. Salah satu bentuk *NPR* adalah *artistic rendering* atau disebut juga *painterly rendering* yaitu teknik untuk memanipulasi gambar foto menjadi tampak seperti lukisan.

Tujuan penelitian ini untuk memanipulasi gambar realis menjadi gambar tidak realis akibat dari efek lukisan cat pensil, crayon, dan tinta.

Berbagai masalah dalam ada diantaranya, bagaimana melakukan segmentasi, mengatur arah goresan kuas ataupun pensil dan menghasilkan tekstur untuk membuat gambar semirip mungkin dengan lukisan.

Metode penelitian dilakukan dengan studi algoritma dasar pengolahan citra digital, dan algoritma-algoritma yang berhubungan manipulasi gambar untuk menghasilkan efek lukisan.

2. Algoritma Rendering

2.1 *Hertzmann Painterly Rendering*

Varying The Brush Size

Sub-Algoritma ini mengatur ukuran kuas dinyatakan dalam radius $R_1 \ldots R_n$. Untuk setiap layer, akan digunakan sebuah *reference image* yaitu hasil dari *source image* yang telah melalui proses *blur,* yang umumnya menggunakan algoritma *Gaussian,* dengan stardar deviasi fs ***Ri***, di mana fs adalah sebuah konstanta dan ***Ri*** adalah ukuran radius kuas.
Persamaan yang digunakan untuk menghitung *image difference* [3] adalah persamaan 1.

$$D = \|(r_1,g_1,b_1) - (r_2,g_2,b_2)\| = \left((r_1-r_2)^2 + (g_1-g_2)^2 + (b_1-b_2)^2\right)^{1/2} \quad (1)$$

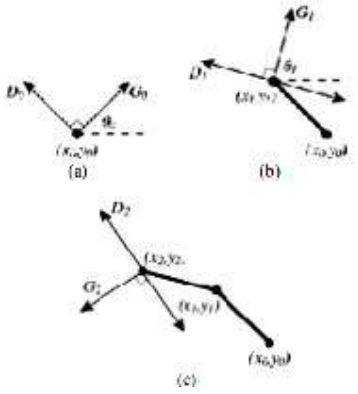
Dimana:

- *D* = nilai *difference* dari RGB *color model.*
- *R1* = nilai *red color component* dari RGB *color model* milik *pixel1.*
- *G1* = nilai *bluecolor component* dari RGB *color model* milik *pixel1.*
- *B1* = nilai *greencolor component* dari RGB *color model* milik *pixel1.*
- *R2* = nilai *red color component* dari RGB *color model* milik *pixel2.*
- *G2* = nilai *bluecolor component* dari RGB *color model* milik *pixel2.*
- *B2* = nilai *greencolor component* dari RGB *color model* milik *pixel2.*

Creating Curved Brush Stroke

Sub-algoritma ini membatasi penggunaan goresan kuas dengan warna konstan dengan bantuan *image gradient* untuk peletakkan kuas. Metode ini meletakkan titik kontrol dengan mengikuti normal dari gradien. Saat perbedaan warna pada suatu titik telah melebihi *threshold* yang ditentukan, goresan kuas akan berakhir pada titik kontrol tersebut.

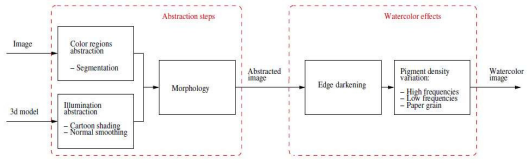
Algoritma peletakkan kuas diawali pada suatu titik (x,y) dengan radius R. Goresan kuas direpresentasikan dengan suatu daftar titik kontrol, warna, dan ukuran kuas. Titik kontrol (x,y) akan ditambahkan pada kuas dan warna yang diambil dari gambar referensi pada titik (x,y) dijadikan sebagai warna kuas. Selanjutnya dilakukan perhitungan untuk menentukan titik selanjutnya pada kurva. Titik selanjutnya (x_1,y_1) diletakkan pada arah $(\theta+\pi/2)$ dengan jarak R dari (x,y). Radius R digunakan sebagai jarak antara titik kontrol karena R mewakili level dari *detail* yang akan ditangkap dengan ukuran kuas tersebut. Gambar 1 merupakan ilustrasi dari pengambilan arah goresan kuas: [3]



Gambar 1. (a) Goresan Kuas Berawal Dari Titik (x_0,y_0) dan Dilanjutkan ke Arah D_0 Dengan Normal G_0. (b) Dari Titik Kedua (x_1,y_1) Ada Dua Arah yang Dapat Dipilih, D_1 Dipilih Untuk Mengurangi Kelengkungan Pada Goresan. (c) Proses Ini Berulang Untuk Menggambar Keseluruhan Goresan Kuas. Sumber: Hertzmann (1998, p. 5)

2.2 *Interactive Watercolor Rendering*

Algoritma ini bertujuan untuk menghasilkan efek gambar lukisan cat air. Secara garis besar, algoritma ini digambarkan pada Gambar 2. [1]



Gambar 2. Algoritma *Watercolor Rendering*

Proses dimulai proses segmentasi, proses *morphological smoothing* untuk lebih mengurangi *detail* gambar. Proses simulasi efek cat air. Efek yang akan diberikan antara lain *edge darkening* dan *pigment density variation*. Persamaan untuk menghitung *density* dapat dilihat pada Persamaan 2.

$$d = 1 + \beta(T - 0.5) \quad (2)$$

Dimana:

- d = nilai *density*.
- β = nilai *scaling factor*.
- T = nilai *texture* yang digunakan, $T \in [0,1]$

Nilai *density* akan digunakan untuk memodifikasi warna pada gambar berdasarkan persamaan 3:

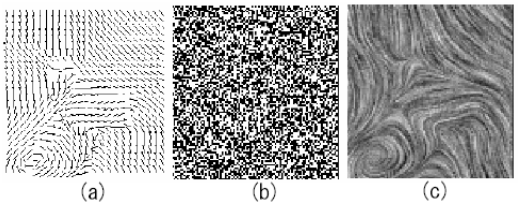
$$C' = C - (C - C^2)(d - 1) \quad (3)$$

Dimana:

- C' = nilai *color* yang baru.
- C = nilai *color* sekarang.
- d = nilai *density*.

2.3 *Line Integral Convolution*

Line Integral Convolution adalah visualisasi *vector field* berdasarkan tekstur. Metode ini mengambil 2D *vector field* dan *white noise image* sebagai *input* dan menghasilkan gambar dengan arah dari *vector field* melalui belokan dari *white noise* dan *low pass filter* yang diterapkan pada *local streamline* dari *vector field*. Gambar 3. berupa ilustrasi proses *Line Integral Convolution*: [2]

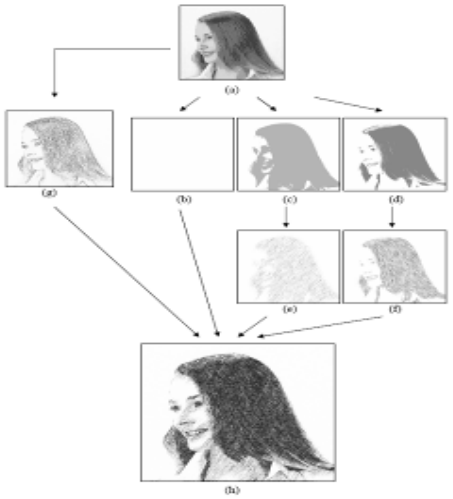


Gambar 3. *Line Integral Convolution*

Sumber: Imamiya et al (2004, p. 3)

2.4 *Enhanced LIC Pencil Filter*

Gambar *input* dipisah kedalam beberapa intensitas *layer* yang berbeda. Untuk setiap *layer* dihasilkan *stroke image* yang akan digabung menjadi satu.



Gambar 4. Algoritma *Enhanced LIC Pencil Filter*

Sumber: Imamiya et al (2004, p. 4)

2.5 *Simulating Wax Crayon*

Hal yang perlu diperhatikan adalah volume *wax* yang diterima dan energi yang diberikan pada *crayon* saat menggambar. *Wax* yang telah terdapat pada kertas dilumurkan saat *crayon* bergerak melaluinya ke atas dan ke bawah. Saat menggambarkan *wax crayon,* digunakan garis dan kurva sebagai dasar penggambaran.

Pertama *height* dari *wax* yang sudah menempel pada kertas disesuaikan dengan *force* yang diberikan. Pengaturan dari *height* ini merupakan penerapan dari persamaan 4:

$$F = \lambda A \Delta L \quad (4)$$

Dimana:

- F = nilai *force* yang diberikan.
- λ = suatu nilai konstanta.
- A = nilai *area* yang dihitung.
- ΔL = nilai *compression.*

Untuk menghitung jumlah *wax* yang menempel pada kertas agar proporsional dengan *force* yang diberikan, digunakan persamaan seperti pada persamaan 5:

$$\vec{F}f = \mu \vec{N} \frac{\vec{N} \cdot \vec{F}c}{\left\| \vec{N} \right\| \left\| \vec{F}c \right\|} \quad (5)$$

Dimana:

- $\vec{F}f$ = nilai *force* dari gesekan.
- μ = suatu nilai koefisien dari gesekan.
- $\vec{N}$ = nilai *normal* dari permukaan kertas.
- $\vec{F}c$ = nilai *force* dari krayon.

Proses *smearing wax* adalah proses untuk mengangkat sejumlah *wax* dari kertas dan menyebarkan sejumlah *wax* ke area di sekelilingnya. Proses ini menggunakan *smearing mask* sebagai *mask* untuk proporsi *wax* yang akan dipindahkan. Persamaan untuk menghitung *smearing mask* dapat dilihat pada persamaan 6:

$$S_{xy} = \frac{1}{\left\| (x, y) \right\|} \left(\alpha \Delta z + \beta (x, y) \cdot \vec{V} \right) (6)$$

Dimana:

- S_{xy} = nilai *mask* untuk posisi (x,y)
- Δz = nilai *crayon height* pada posisi (x,y)
- $\vec{V}$ = *vector* arah gerakan krayon.
- α dan β = nilai konstanta.

Proses *rendering* untuk menggambarkan *wax cray*, digunakan model warna Kubelka-Monk yaitu model warna yang tidak hanya memperhitungkan *color space RGB* tetapi juga memperhitungkan *transmittance, scattering,* dan *interfence. KM-model* menghasilkan warna yang berbeda untuk tingkat ketebalan yang berbeda, karena itu model ini dianggap lebih cocok untuk dipakai dibandingkan dengan model warna *RGB.* [5]

2.6 *Outline Painting*

Algoritma ini untuk menghasilkan lukisan dari *outline* sebuah gambar. Algoritma ini meliputi proses *morphological smoothing, edge detection, intensity scanning,* dan *forming lines.*

Morphological smoothing untuk mengurangi *detail* pada gambar, umumnya digunakan *median filter* dan *bilateral filter.*

Edge detection merupakan bagian yang paling penting dari pengambilan *outline. Edge detection* yang paling umum digunakan adalah *Canny Edge Detection,* tetapi algoritma ini hanya menghasilkan garis *outline* tanpa memperhitungkan intensitas dari *outline.* Algoritma *edge detection* lain yang memberikan hasil *edge* dengan memperhitungkan intensitas dari daerah *edge* adalah *Laplacian Edge Detection.*

Setelah melakukan *edge detection,* dilakukan proses *intensity scan* untuk menghitung intensitas dari *edge* yang dihasilkan.Pertama, setiap *pixel* akan dihitung intensitasnya berdasarkan persamaan 7:

$$Intensity(R, G, B) = \frac{\sqrt{R^2 + G^2 + B^2}}{\sqrt{3}} (7)$$

Kemudian, ditentukan sebuah *range* untuk mengambil area dengan intensitas yang tinggi. Dari *range* tersebut, apabila nilai intensitas pada area dalam *range* memenuhi *threshold,* maka akan digambar lingkaran sebesar *range.* Bila nilai intensitas masih di bawah *threshold,* maka *range* akan diperkecil sampai nilai memenuhi *threshold.*

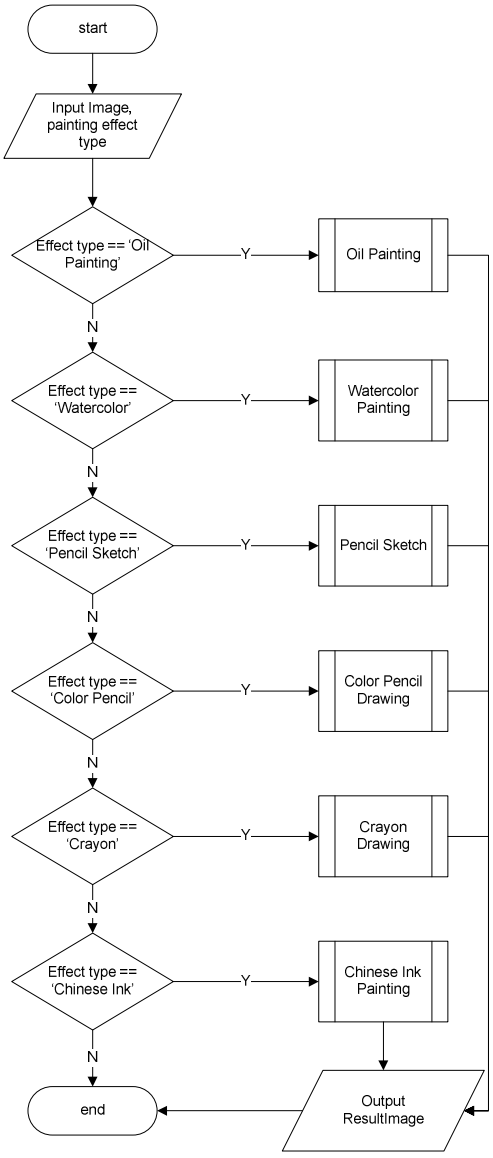
Proses *forming lines* dilakukan dengan menggabungkan lingkaran-lingkaran yang telah digambar. Lingkaran dengan jarak di bawah *threshold* akan disambungkan menjadi sebuah garis. Dalam pembentukan garis sebaiknya dicari garis yang tidak terlalu pendek. Metode untuk penggambaran garis dapat bervariasi. Salah satu metode paling mudah adalah dengan langsung mengisi lingkaran-lingkaran yang berdekatan dengan suatu warna [6].

3. Implementasi Sistem

Secara garis besar, sitem kerja perangkat lunak adalah dengan menerima *input* gambar dari pengguna lalu gambar akan diproses melalui salah satu dari algoritma, *Output* dari perangkat lunak adalah hasil dari salah satu proses yang dijalankan, digambarkan dalam *flowchart* pada Gambar 5.

4. Pengujian

Pengujian dilakukan terhadap proses *painterly rendering*, dan perbandingan hasil *painterly rendering* dengan hasil dari perangkat lunak lainnya



Gambar 5. Algoritma Garis Besar Sistem

4.1 Pengujian *Oil PaintingRendering*

Pengujian terhadap parameter dari *oil painting rendering* meliputi pengujian terhadap parameter *stroke length* dan *color threshold.*

Dari hasil pengujian, dapat dilihat bahwa *stroke length* mempengaruhi hasil pada tingkat kemiripan dengan gambar *input.*ditampilkan pada Tabel 1.

Pada *stroke length* 1, gambar yang dihasilkan gambar yang terdiri dari titik-titik, tidak mirip dengan kuas. *Stroke* yang terlalu panjang memungkinkan terjadinya goresan kuas yang terlalu melebar dan keluar dari area warna dari *stroke.*

Color threshold yang rendah menghasilkan gambar yang semakin mirip dengan gambar *input*, namun juga menghasilkan tingkat *area* kosong yang semakin tinggi, ditampilkan pada Tabel 2.

Tabel 1. Hasil Pengujian Untuk *Parameter Stroke Length*

Stroke Length	Hasil *Oil Painting Rendering*
1	
4	
16	

.

4.2 Pengujian *Watercolor Painting Rendering*

Pada proses *watercolor painting rendering,* parameter yang digunakan adalah parameter untuk mengatur tingkat *segmentation* dan *smooth* yang digunakan untuk proses `removeDetail`. Tabel 5.3 berikut adalah beberapa contoh dari hasil *watercolor painting rendering* dengan beberapa tingkat *detail* yang berbeda

4.3 Pengujian *Pencil Sketch Drawing Rendering*

Parameteryang digunakan pada *pencil sketch drawing* dan *color pencil drawing rendering* antara lain *stroke angle, stroke length, edge detail, color source.*

Tabel 2. Hasil Pengujian Untuk *Color Threshold*

Color Threshold	Hasil *Oil Painting Rendering*
10	
20	
40	

Dari beberapa parameter tersebut, *stroke angle* hanya memberi pengaruh pada sudut arah *stroke* yang digambar. *Stroke length* memberikan efek yang lebih besar berupa panjang *stroke* dan tingkat *blur* dari gambar. Tabel 4. berikut berisikan hasil pengujian terhadap parameter *stroke length*.

4.4 **Pengujian** Parameter *Crayon Drawing Rendering*

Parameter yang digunakan pada *crayon drawing rendering* hanya berupa *edge detail*. Pada proses *crayon drawing rendering* ini, yang diproses adalah gambar yang berupa garis. *Edge detail* adalah banyaknya *edge* yang akan diproses.

Parameter ini hanya mempengaruhi jumlah garis yang akan diproses. Tabel 6. berikut berisikan hasil pengujian terhadap berbagai *edge detail*

Kesimpulan

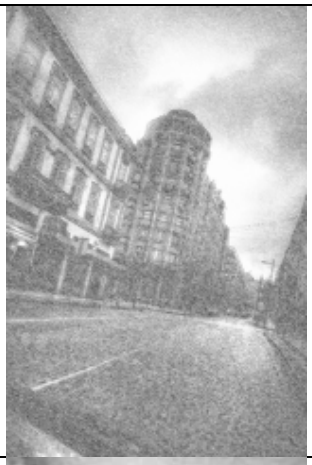
Berdasarkan hasil pengujian dapat disimpulkan beberapa hal sebagai berikut:

Tabel 4. Hasil Pengujian *Watercolor Painting Rendering* dengan Tingkat *Detail* yang Berbeda

Segmentation dan *Smooth*	Hasil *Watercolor Rendering*
Segmentation radius = 1 *Segmentation threshold* = 10 *Smooth radius* = 1	
Segmentation radius = 3 *Segmentation threshold* = 25 *Smooth radius* = 3	
Segmentation radius = 7 *Segmentation threshold* = 30 *Smooth radius* = 3	

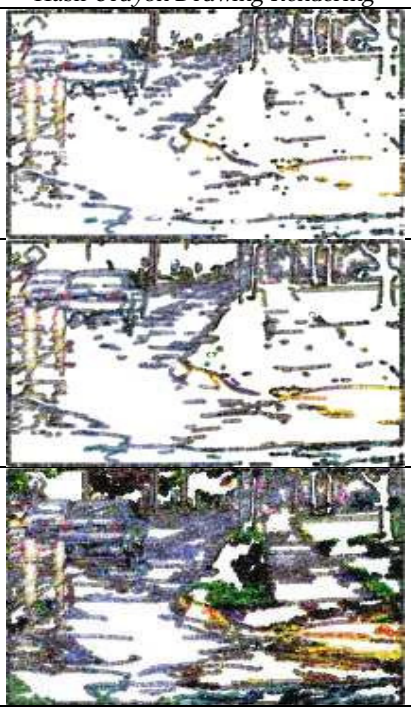
- Proses *oil painting rendering,* semakin panjang *stroke* dan semakin besar nilai *threshold,* maka semakin kecil tingkat kemiripan dengan gambar asli.
- Pada proses *watercolor painting rendering,* proses segmentasi dan *morphological smoothing* mengurangi *detail* gambar sehingga menghasilkan kesan yang lebih artistik. Proses *dispersion* warna pada gambar dinilai telah dapat menghasilkan tekstur warna basah dari cat air.
- Untuk *crayon drawing rendering,* semakin banyak *edge* dari gambar, maka semakin terlihat jelas *crayon* yang dihasilkan. Untuk *chinese ink painting,* hasil gambar akan lebih baik bila *edge* tidak rapat.

Tabel 5. Hasil Pengujian Terhadap Parameter *Stroke Length*

Stroke Length	Hasil *Pencil Sketch Rendering*
2	
4	
8	

- Proses *pencil sketch rendering* dipengaruhi oleh panjang *stroke* dalam tingkat ketajaman hasil gambar, demikian halnya dengan proses *color pencil drawing rendering.*
- Kekurangan pada *pencil sketch rendering, stroke* hanya berupa garis lurus dan tidak dapat membentuk jenis arsiran. Efek dari *crayon* yang licin kurang terlihat

Tabel 6. Hasil Pengujian Terhadap Berbagai *Edge Detail*

Hasil *Crayon Drawing Rendering*


Daftar Pustaka:

[1] Bousseau, A., Kaplan, M., Thollot, J., Sillion, F. (2006).*Interactive watercolor rendering with temporal coherence and abstraction,* NPAR '06 Proceedings of the 4th International Symposium on Non-photorealistic Animation and Rendering.

[2] Cabral, B., and Leedom, L. (1993).*Imaging vector field using line integral convolution*, SIGGRAPH93 Conference Proceeding.

[3] Hertzmann, A. (1998). *Painterly rendering with curved brush strokes of multiple sizes*, Proceedings of the 25th annual conference on Computer Graphics and interactive Techniques.

[4] Imamiya, A., Tanii, K., Yamamoto, S., and Mao, X. (2004). *Enhanced LIC pencil filter,* Computer Graphics, Imaging and Visualization Proceeding.

[5] Rudolf, D., Mould, D., Neufeld, E. (2003). *Simulating wax crayons, Proceedings* of the 11th Pacific Conference on Computer Graphics and Applications.

[6] Strothotte, T. and Schlechtweg, S. (2002).*Non-photorealistic computer graphics.*San Francisco: Morgan Kaufmann Publishers.