RANCANG BANGUN EDITOR KURVA *POLYLINE* DENGAN METODE *CURVE ANALOGIES*

Nanik Suciati – Chastine Fatichah – Faida Royani
Jurusan Jurusan Teknik Informatika,
Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember
Kampus ITS, Jl. Raya ITS, Sukolilo – Surabaya 60111, Tel. + 62 31 5939214, Fax. + 62 31 5913804
Email: nanik@its-sby.edu, chf@its-sby.edu , yani@inf.its-sby.edu

ABSTRAK

Model-model kurva banyak digunakan untuk pembuatan sketsa. Untuk merancang model kurva untuk sketsa tersebut dapat dilakukan dengan berbagai macam cara. Salah satunya adalah menyusun program secara manual dalam membentuk model-model kurva untuk menggambar sketsa yang diinginkan. Pendekatan ini memberi kontrol yang besar ke programmer. Pendekatan lain yaitu mengambil detail kurva yang dimasukkan pengguna. Salah satu metode dalam pendekatan ini adalah dengan mempelajari style-style garis dari contoh - contoh. Pendekatan ini disebut dengan pendekatan curve analogies. Penelitian ini bertujuan untuk menerapkan metode curve analogies dalam membuat editor kurva polyline. Curve Analogies bertujuan membentuk kurva baru dari kurva contoh. Inputan untuk kurva analogies ada 3 macam yaitu dua kurva garis dan satu kurva detil (kurva contoh). Dua kurva garis tersebut adalah kurva inputan dari pengguna dan kurva yang mengikuti kurva detil. Dua kurva garis ini digunakan untuk mencari nilai transformasi. Sedangkan untuk membuat kurva baru dilakukan proses synthesis dengan algoritma synthesis. Algoritma synthesis membentuk kurva baru berdasarkan style dari kurva contoh. Kurva hasil proses synthesis di transformasi sesuai dengan nilai transformasinya. Kurva baru yang dihasilkan harus selalu melalui titik kontrol yang pertama dan terakhir dari kurva garis yang diinputkan penguna. Uji coba perangkat lunak ini dilakukan dengan menjalankan beberapa skenario. Skenario pertama dengan memasukkan satu obyek gambar, kedua memasukkan lebih dari satu obyek gambar, ketiga membuat kurva contoh baru dan yang keempat melakukan sintesa kurva. Dari hasil beberapa skenario tersebut dapat disimpulkan metode curve analogies dapat digunakan untuk membuat editor kurva polyline.

Kata kunci: kurva *polyline, curve analogies,* transformasi

1. PENDAHULUAN

Terdapat beberapa cara yang bisa digunakan untuk mendesain model-model kurva untuk sketsa. Salah satu strategi yang umum digunakan adalah menyusun program secara manual dalam membentuk model-model kurva untuk menggambar sketsa yang diinginkan.

Cara lain yang bisa digunakan dalam membuat sketsa adalah meng-capture detail kurva yang dimasukkan oleh pengguna[HER-02]. Pengguna bisa menggambar bentuk global yang lebih sederhana dari kurva yang diinginkan dan menerapkan detail yang di-capture, sesuai dengan bentuk global tersebut. Sebagai contoh untuk membuat sketsa rumput, bisa diterapkan detail lengkung kecil pada kurva berbentuk garis memanjang. Interface ini akan mempermudah pengguna dalam membuat sketsa. Salah satu metode dalam pendekatan kedua adalah dengan mempelajari style-style garis dari contoh. Pendekatan ini diambil dari terminologi Hertzmann dkk, yang disebut dengan pendekatan *curve analogies*.

Analogi adalah perbandingan secara sistematis antar struktur dengan menggunakan properti dan relasi antar onyek dalam sebuah source structure untuk memperoleh properti dan relasi antar obyek dari target structure. Secara umum masalah curve analogies digambarkan sebagai berikut. Misalkan terdapat kurva A dan A'. Dari kedua kurva tersebut dipelajari transformasinya kemudian transformasi tersebut digunakan untuk menghasilkan kurva baru B' dari kurva B (gambar 2.9). Dalam penelitian ini dibuat suatu program editor kurva polyline sederhana yang digunakan untuk mendesain model-model kurva untuk sketsa dengan metode *curve analogies*.

Adapun tujuan dari penelitian ini adalah membuat program editor kurva *polyline* dengan metode *curve analogies.*

Permasalahan yang dihadapi dalam penelitian ini adalah :

1. Bagaimana mempelajari transformasi kurva dengan metode *curve analogies.*
2. Bagaimana merancang struktur data dan algoritma yang sesuai untuk program editor sketsa dengan metode *curve analogies.*

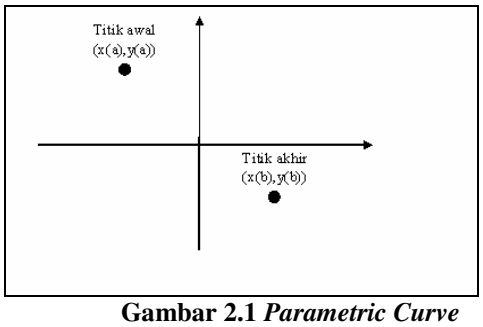
2. KURVA POLYLINE

Kurva memegang peranan penting dalam kehidupan. Dalam bidang Teknik perancangan dan manufaktur kurva diperlukan untuk berbagai macam produk. Ada beberapa macam bentuk kurva diantaranya *Polyline* , *Bezier*, *Bspline* dan sebagainya. Kurva adalah geometri satu dimensi, sering digambarkan sebagai rangkaian titik-titik Kurva dapat ditulis dalam bentuk parametrik yang merupakan cara untuk menggambarkan kurva di bidang *Cartesian*. Kurva dapat dianggap sebagai kumpulan dari titik-titik pada sebuah bidang**[STW--__]**. Nilai y tidak dirumuskan sebagai fungsi dari nilai x, melainkan keduanya merupakan fungsi t dimana t itu sendiri merupakan suatu parameter. Bentuk persamaannya dapat dilihat pada persamaan(1):

$$x = x(t), y = y(t) \quad (1)$$

Dengan fungsi persamaan 1 dan domain variabel t, a ≤t ≤b maka (x(a),y(a)) merupakan titik awal dan (x(b),y(b)) merupakan titik akhir (gambar 2.1). Untuk penelitian ini yang digunakan adalah kurva *polyline* (gambar 2.2).

Kurva *polyline* adalah vektor dari vertek-vertek yang dihubungkan oleh garis**[OTT-04]**. Definisi lain dari kurva *polyline* adalah sekumpulan titik yang dihubungkan oleh garis lurus dan secara eksplisit tidak tertutup**[TAL-95]**.



Gambar 2.1 ***Parametric Curve***

Kurve *polyline* merupakan kurva linier. Setiap kurva *polyline* direpresentasikan dalam suatu daftar yang terurut dari titik kontrol (t_i,p_i) dimana masing-masing nilai t_i dalam daftar tersebut adalah unik dan nilai parameter maksimum dan minimum dalam list mendefinisikan *range* [tmin,tmax]. Bentuk persamaan kurva *polyline* sama dengan bentuk persamaan kurva parametrik.

Kurva *polyline* dievaluasi pada nilai t tertentu dengan fungsi interpolasi linier dan dibuat dengan cara menggambar garis antar titik kontrol. Dengan fungsi interpolasi linier maka kurva harus melalui semua titik kontrol. Fungsi interpolasi linier tersebut dapat dirumuskan seperti pada persamaan 2.

$$f(t) = \begin{cases} p \longrightarrow & \text{Jika (t,p) adalah titik kontrol} \\ \dfrac{(t-t_{lower})P_{upper}+(t_{upper}-t)P_{lower}}{t_{upper}-t_{lower}} \longrightarrow & \text{Selain titik kontrol} \end{cases} \quad (2)$$

Dengan :

p = y pada koordinat cartesian
t = x pada koordinat Cartesian
t_{lower} = nilai t sebelumnya
t_{upper} = nilai t sesudahnya
p_{lower} = nilai p sebelumnya
p_{upper} = nilai p sesudahnya

Sebagai contoh : titik kontrol dari sebuah kurva adalah P_0 = (3,4), P_1=(6,2), P_2 = (8,3), P_3 = (10,6) jika kurva tersebut dievaluasi pada titik kontrol yang pertama dan kedua dengan nilai 3< t <6 maka diperoleh fungsi seperti pada persamaan berikut :

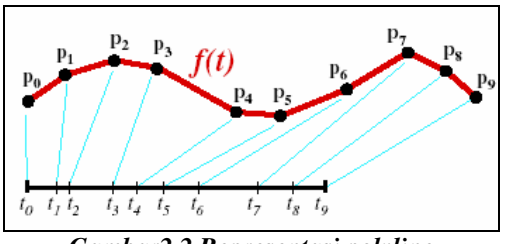
$$f(t) = \frac{(t-3)2+(6-t)4}{6-3} = \frac{-2t+18}{3}$$

dengan nilai t = 3.5:

$$f(3.5) = \frac{-2t+18}{3} = \frac{-2(3.5)+18}{3} = 3.67$$

Dengan demikian kurva tersebut akan melalui koordinat (3.5,3.67). Fungsi Interpolasi Linier ini digunakan untuk mencari titik sampel (neighborhood) di sekitar titik kontrol kurva.

Kurva *polyline* dapat dimodifikasi dengan menambah dan menghapus titik control. Kurva ini dapat diubah dengan mudah ke bentuk representasi lain dengan cara melakukan *resampling* atau membuat *non-uniform BSpline* dari titik control.



Gambar2.2 Representasi polyline

Transformasi

Titik beserta garis yang menghubungkannya digunakan untuk merepresentasikan obyek, sehingga kemampuan untuk mentransformasikan obyek merupakan dasar dari grafika komputer. Operasi-operasi transformasi ada beberapa macam diantaranya translasi, rotasi dan *scaling*.

a. Translasi

Translasi diaplikasikan terhadap sebuah obyek dengan memposisikan kembali obyek tersebut sepanjang lintasan garis lurus dari satu koordinat ke koordinat yang lain. Sebuah titik diubah dengan menambahkan jarak translasi t_x dan t_y dari koordinat

awal (x,y) ke posisi baru (x',y') seperti pada gambar 2.3 dan dirumuskan pada persamaan 3.

$$x' = x + t_{x'} \text{ dan } y' = y + t_{y'} \quad (3)$$

Jarak translasi ($t_{x'}$,$t_{y'}$) di sebut dengan vektor translasi atau *shift vector*.

Persamaan (3) dapat digambarkan sebagai persamaan matrik tunggal dengan menggunakan kolom vektor untuk merepresentasikan posisi koordinat dan vektor translasi. Persamaan tersebut dituliskan seperti pada persamaan (4). Dengan adanya persamaan (4), memungkinkan persamaan translasi ditulis dalam bentuk persamaan matrik (persamaan (5)).

$$P = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad P' = \begin{bmatrix} x'_1 \\ x'_2 \end{bmatrix} \qquad T = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

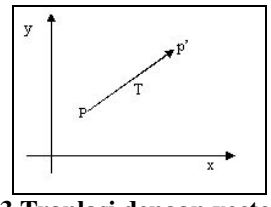
$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \quad (4)$$

$$P' = P + T \quad (5)$$

dengan P' = Titik dengan koordinat baru (yang telah ditranslasi)

P = Titik asal

T = Matrik translasi

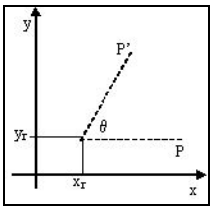


Gambar 2.3 Tranlasi dengan vector tranlasi T

Translasi merupakan sebuah *rigid-body transformation* yang memindahkan obyek tanpa merusak obyek tersebut. Ini berarti, setiap titik pada sebuah obyek ditranslasi dengan vector translasi yang sama.

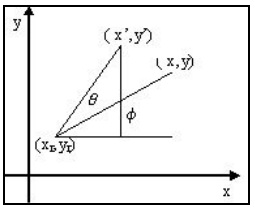
b. Rotasi

Rotasi dua dimensi diaplikasikan pada sebuah obyek dengan memposisikan kembali obyek sepanjang lintasan melingkar dalam bidang xy. Untuk merotasi sebuah obyek, yang harus dilakukan adalah menentukan sudut rotasi dan posisi titik pivotnya (x_r,y_r). Titik pivot ini digunakan sebagai titik rotasi dimana obyek akan dirotasi (Gambar 2.4). Nilai positif untuk sudut rotasi menetapkan arah rotasi berlawanan arah jarum jam dan nilai negatif merotasi obyek kearah yang searah jarum jam.



Gambar 2.4 Rotasi dengan titik pivot (x_r,y_r)

Matrik rotasi ada pada persamaan (5). Rotasi sebuah titik dengan posisi titik pivot bukan (0,0), melainkan (x_r,y_r) seperti yang terlihat pada gambar 2.5.



Gambar 2.5 Merotasi sebuah titik posisi (x,y) ke posisi(x',y') dengan sudut θ terhadap titik rotasi (x_r,y_r)

Dengan menggunakan hubungan trigonometri maka persamaan transformasi untuk merotasi sebuah titik terhadap titik pivot yang tidak tentu dapat dirumuskan seprti persamaan (6).

$$y' = y_r + (x - x_r)\sin\theta + (y - y_r)\cos\theta$$
$$x' = x_r + (x - x_r)\cos\theta - (y - y_r)\sin\theta \quad (6)$$

Sama halnya dengan translasi, rotasi merupakan *rigid-body transformations* yang memindahkan obyek tanpa merusaknya. Setiap titik pada sebuah obyek dirotasi dengan sudut yang sama.

c. Scaling

Transformasi *scaling* merupakan jenis transformasi yang memgubah ukuran dari obyek. Untuk melakukan *scaling* diperlukan faktor skala s_x untuk menskala obyek dalam arah x, dan s_y untuk menskala dalam arah y. Sebagai contoh, sebuah titik (x,y) akan di-*scaling* dengan skala s_x dan s_y. Koordinat titik yang baru (x',y') dapat diperoleh dengan menggunakan persamaan seperti pada persamaan (7).

$$x' = x. \cdot s_x, y' = y \cdot s_y \quad (7)$$

Dengan matrik scaling :

$$\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

Semua nilai numerik yang positif dapat di-*assign* untuk faktor skala s_x dan s_y. Nilai skala yang lebih kecil dari 1 akan mengurangi ukuran obyek, sedangkan nilai skala yang lebih besar dari akan

memperbesar ukuran obyek. Jika nilai s_x dan s_y sama-sama bernilai satu maka ukuran obyek tidak berubah. Jika nilai s_x dan s_y di-*assign* dengan nilai yang sama maka yang terjadi adalah *scaling uniform* dan jika di-*assign* dengan nilai yang berbeda maka yang terjadi adalah *scaling nonuniform*.

d. Transformasi dua titik centroid

Transformasi yang digunakan untuk algoritma *curve synthesis* adalah transformasi antara dua titik *centroid* dari sekumpulan(ini dapat dilihat pada paper **[2N-__]**. Misalkan ada sejumlah n titik yang diukur dalam 2 koordinat sistem yang berbeda, yaitu koordinat sistem kiri dan kanan. Untuk titik yang diukur dengan sistem koordinat kiri ditulis dengan notasi $\{r_{l,i}\}$ sedangkan untuk sistem koordinat kanan ditulis dengan notasi $\{r_{r,i}\}$.

dengan i = 1 sampai n

$r_{l,i}$ = notasi untuk sistem koordinat kiri

$r_{r,i}$ = notasi untuk sistem koordinat kanan

Untuk mencari transformasi antara 2 sistem koordinat tersebut dilakukan dengan langkah-langkah sebagai berikut:

1. Mencari titik pusat dari sekumpulan titik
2. Menghitung skala
   Untuk mencari skala di lakukan beberapa langkah yaitu :
   - Mencari koordinat baru dari masing-masing titik.
   - Menghitung panjang koordinat baru dari masing-masing titik
   - Menjumlahkan kuadrat panjang dari masing-masing titik.
3. Menghitung Translasi antara dua titik pusat

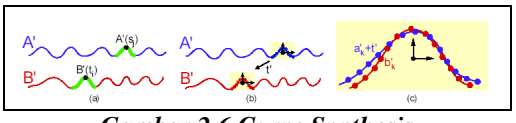
e. Curve Synthesis

Permasalahan dalam *curve synthesis* adalah bagaimana men-*generate* kurva baru B' dengan panjang tertentu L_B,. Bentuk kurva baru B' yang dihasilkan serupa dengan kurva contoh A', sebagaimana terlihat pada gambar 2.7. Algoritma *single-scale curve texture synthesis* yang digunakan pada permasalahan *curve synthesis* merupakan adaptasi dari algoritma *image texture synthesis* untuk memproses bentuk yang dikemukakan oleh Efros dan Leung**[EFR-99]**.

Untuk mensintesa kurva baru B' diperlukan pendefinisian *cost function. Cost function* tersebut didefinisikan seperti pada persamaan (8). Kurva baru B' harus dihasilkan dengan *minimal cost.* B' merupakan kurva output, A' merupakan kurva input, t_i merupakan indek titik kontrol dari kurva B' dan s_j merupakan indek titik kontrol dari kurva A'.

$$E(B') = \sum_i \min_j d(B', t_i, A', s_j) \qquad (8)$$

Persamaan (8) ini menyatakan bahwa *neighborhood* di sekitar tiap lokasi t_i di B' serupa dengan *neigborhood* disekitar beberapa s_j di A'. Dengan kata lain, *cost function* ini digunakan untuk mengukur perbedaan antara bentuk lokal dari B' disekitar t_i dan bentuk lokal dari A' disekitar s_j. Sampel t_i dan s_j di ambil dari sekumpulan sampel.



Gambar 2.6 Curve Synthesis

Dalam proses *curve synthesis,* hal yang paling penting adalah pendefinisian *neighborhood distance metric* d (B',t_i,A',s_i). Hal ini disebabkan karena *neighborhood distance metric* digunakan untuk mencari *cost.* Setiap *neighborhood* merupakan sekumpulan dari K sampel, yaitu $a'_k = A'(s_k)$ dan $b'_k = B'(t_k)$, k= {1…K}. Sampel-sampel tersebut diambil dari titik-titik sebelum dan sesudah titik kontrol. Dengan a'_k merupakan koordinat titik sampel yang ke-k dari *neighborhood* titik kontrol kurva A' dan b'_k merupakan koordinat titik sampel yang ke-k dari *neighborhood* titik kontrol kurva B'. Sebagai tambahan, digunakan fitur tangen untuk meng-*capture property* dari kurva supaya lebih bagus.

$$\Delta a'_k = \frac{(a'_k - a'_{k-1})}{\left\|a'_k - a'_{k-1}\right\|} \quad \text{dan} \qquad (9)$$

$$\Delta b'_k = \frac{(b'_k - b'_{k-1})}{\left\|b'_k - b'_{k-1}\right\|}$$

Secara umum, perbandingan neighborhood menggunakan invariant rigid transformation. Dari penjelasan di atas maka distance metric didefinisikan seperti persamaan (10).

$$d(B', t_i, A', s_j) = \min_{R',t'} \sum_k w_k \left(\left\| R' a'_k + t' - b'_k \right\|^2 + w\Delta \left\| R' \Delta a'_k - \Delta b'_k \right\|^2 \right) \qquad (10)$$

Copying patch dari tekstur dapat memperbaiki kualitas pada tekstur *image* **[HER-01]**, hal ini juga terjadi pada kurva. Oleh karena itu , dengan cara yang sama dilakukan *copy-ing coherent segment* dari A' ke B' (gambar 2.8). Karena kurva B' dibangun dari titik kontrol (t_i,p_i) maka $B(t_i) = p_i$. Jika S(i) menjadi *source index* untuk setiap titik kontrol di B', maka didefinisikan $S(i) = \text{argmin}_j\, d(B',t_i,A',s_j)$. Titik kontrol (ti,pi) di B' adalah *coherent* dengan titik kontrol sebelumnya (t_{i-1}, p_{i-1}) jika $S(i-1)<S(i)$ dan kurva segment antara A'(S(i-1)) dan A'(S(i)) adalah identik untuk kurva segment antara B'(t_{i-1}) dan B(t_i). Untuk mengetahui *coherence* atau tidaknya maka

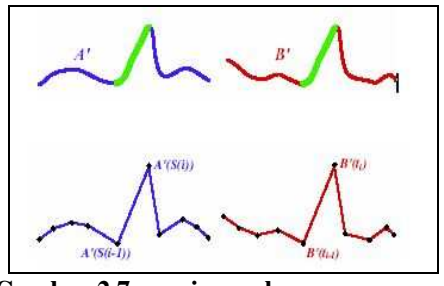
yang dilakukan adalah menguji secara aproksimasi dengan mengukur apakah panjang busur yang terkait sama atau tidak dengan menggunakan persamaan (11) . Jika tidak *coherence* maka nilai *cost*-nya dikalikan dengan (1+k/D), dimana D = aproksimasi jarak titik kontrol dan k adalah parameter.

$$\left\|A'(S(i)) - A'(S(i-1))\right\| = \frac{3}{2}\left\|B'(t_i) - B'(t_{i-1})\right\| \qquad (11)$$

Keterangan:

A'(S(i)) = titik kontrol kurva A' yang ke-i

$B'(t_i)$ = titik control kurva B' yang ke-i. $B'(t_i)$ diperoleh dari nilai p_j (yang merupakan titik kontrol kandidat untuk kurva B') .



Gambar 2.7 copying coherence segment

e. Curve Synthesis dengan Constraint

Ada dua macam *constraint* dalam *curve synthesis* yaitu :

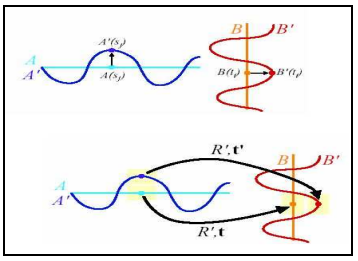
- *Soft Contraints* menyatakan bahwa kurva seharusnya mendekati nilai tertentu. Setiap *soft constraint* menambahkan nilai $w_c\left\|B(t_c) - q_c\right\|^2$ ke *cost function* dimana c adalah indek dari *constraint*.
- *Hard Constraints* menyatakan bahwa kurva hasil harus melalui posisi tertentu q_c. *Hard constraint* dispesifikasikan sebagai $B'(t_c) = q_c$ dan digunakan sebagai batas untuk *soft constrain* karena $w_c \to \infty$

f. Curve Analogies

Style kurva baru didesain dengan pendekatan *curve analogies* yang didasarkan pada algoritma dalam *image texture synthesis,* dimana setiap piksel *neighborhood* tekstur baru yang dihasilkan serupa dengan *neighborhood* ditekstur contoh.

Ada banyak masalah *curve analogies*, untuk tugas akhir ini permasalahannya digambarkan sebagai berikut : misalkan ada contoh kurva A dan contoh kurva A', maka bagaimana menghasilkan kurva baru B' dari kurva B dengan acuan kurva A dan A' (gambar 2.8).

Algoritma *curve analogies* tidak jauh berbeda dengan algoritma *curve syhthesis*. Dalam *curve analogies* juga dilakukan proses yang dilakukan *curve synthesis* dalam men-*generate* kurva baru B'. Yang membedakan keduanya adalah adanya *hard constraint* pada *curve analogies.* Algoritma *curve analogies* menganalogikan kurva garis dari input dengan kurva contoh garis dan kurva contoh detil (gambar 2.8). Proses untuk menganalogikan kurva garis dari input dengan kurva contoh garis dan kurva contoh detail dilakukan dengan mencari transformasi kurva garis input dan kurva contoh garis.



Gambar 2.8 Analogi antara kurva A dengan kurva B

3. DESAIN PERANGKAT LUNAK

a. Gambaran umum

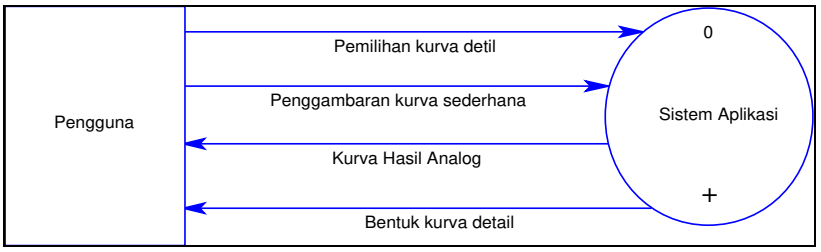
Secara umum perangkat lunak ini digunakan untuk mempermudah pengguna dalam membuat sketsa karena pengguna cukup memasukkan kurva garis saja. Aplikasi ini tidak menuntut pengguna mempunyai kemampuan dalam menggambar dengan menggunakan mouse. Pengguna dapat menggambar bentuk global yang lebih sederhana dari kurva yang diinginkan dan sistem akan menerapkan detil yang di-capture. Misalnya untuk membuat sketsa rumput, bisa diterapkan detail lengkung kecil pada kurva berbentuk garis memanjang.

Kurva yang digunakan untuk detil ini adalah kurva polyline dan disimpan dalam file berekstensi txt. Untuk memilih kurva detil, pengguna mengklik daftar dari kurva detil yang telah ada dalam sistem. Pengguna dapat melihat bentuk dari kurva detail dengan memilih submenu edit sample pada menu edit. Pada submenu editsampel pengguna dapat menambahkan kurva detail sesuai dengan yang diinginkan. Kurva detail ini diaplikasikan ke kurva yang diinputkan pengguna Untuk mengaplikasikannya dilakukan proses analogi dari kurva yang dinputkan pengguna dengan kurva detail yang dipilih. Dalam proses analogi dilakukan proses tranformasi dan synthesis.

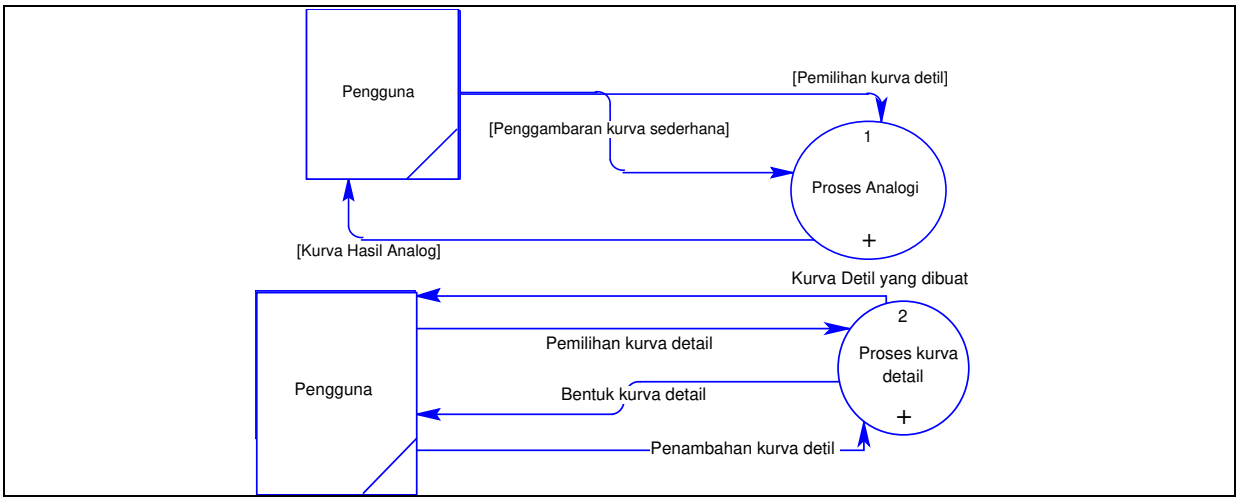
b. Desain Proses

Perancangan proses perangkat lunak digambarkan dengan menggunakan Data Flow Diagram (DFD). Secara garis besar DFD level 0 dari perangkat lunak dapat dilihat pada gambar 3.1.

Pada DFD level 1 perangkat lunak aplikasi editor didekomposisi menjadi dua sub-proses yaitu proses analogi dan proses kurva detail seperti yang terlihat pada gambar 3.2.



Gambar 3.1. DFD Level 0



Gambar 3.2. DFD Level 1

Proses analogi merupakan proses utama yang digunakan untuk menganalogikan antara kurva input dengan kurva detail. Pada proses analogi ini ada beberapa proses yang dilakukan meliputi proses penyimpanan, proses penyamaan parameter, proses pengambilan kurva detail, proses transformasi, proses membuat kurva baru, proses *curve synthesis*, proses rotasi curve synthesis. Input dari proses ini adalah penggambarn kurva sederhana dan Kurva detail yang dipilih oleh pengguna. Sedangkan output dari proses analogi ini adalah kurva hasil analogi.

Proses kurva detail merupakan proses pembuatan kurva detail yang menjadi kurva contoh untuk proses analogi dan untuk menampilkan kurva detail. Pada proses ini ada beberapa proses yang dilakukan meliputi proses pembacan data, proses penggambaran, proses pembuatan kurva, dan proses penyimpanan ke file. Input dari proses ini berupa kurva detail yang dipilih pengguna dan penambahan kurva detail. Sedangkan output dari proses ini adalah bentuk kurva detail yang dipilih pengguna atau bentuk kurva detail yang dibuat pengguna.

4. UJI COBA

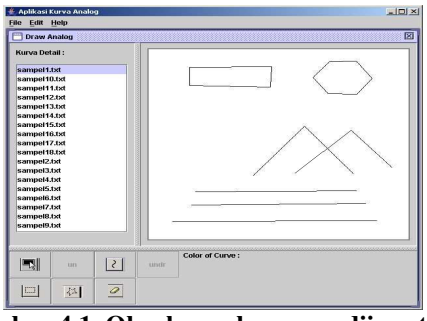
a. Lingkungan Ujicoba

Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam pengujian ini dapat dilihat pada table berikut ini :

Perangkat keras	Prosesor : AMD Duron Memori : 256 MB
Perangkat Lunak	Sistem Operasi : Windows Perangkat Lunak Pembangun : JCreator, J2sdk 1.4.2_01

b. Uji Coba

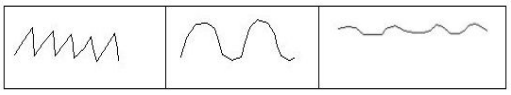
Skenario 1: Pengguna membuat lebih dari satu obyek gambar.



Gambar 4.1. Obyek gambar yang diinputkan user

Setelah pengguna menginputkan kurva yang akan dianalogkan maka pengguna harus memilih

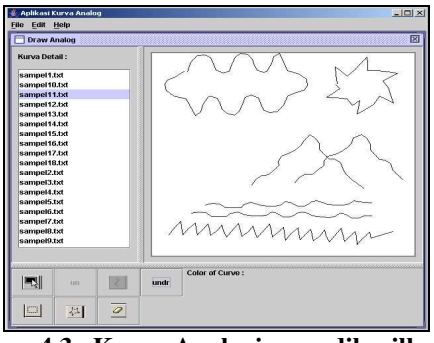
kurva detil yang akan dianalogkan ke kurva tersebut. Bentuk kurva detil yang dipilih untuk scenario yang ini adalah bentuk kurva yang terlihat pada gambar 4.2.



a b c

Gambar 4.2. Kurva detil yang digunakan

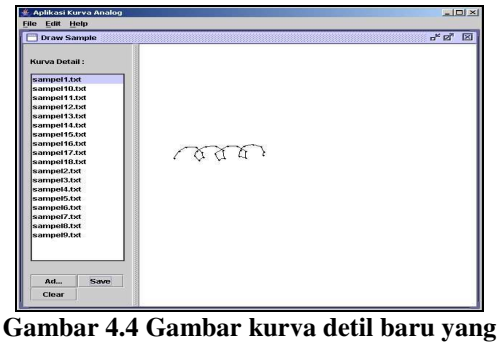
Dengan bentuk kurva detil yang dipilih pengguna seperti pada gambar 4.2 maka dengan proses analogi bentuk kurva outpunya adalah seperti yang terlihat pada gambar 4.3. Kurva detil yang dipilih pengguna di analogikan pada setiap segmen pada seluruh bagian dari obyek gambar yang terlihat pada gambar 4.1.



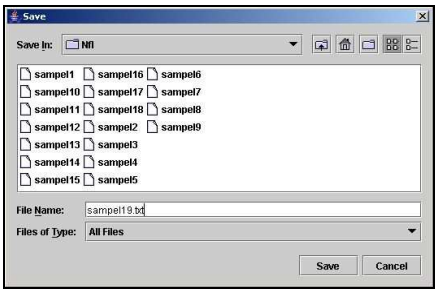
Gambar 4.3. Kurva Analogi yang dihasilkan

Skenario 2: pengguna menambah kurva detil baru dengan membuka form *drawSampel.*

Untuk menambah kurva detil baru pengguna harus menekan tombol add terlebih dahulu, kemudian menginputkan koordinatnya kurva melaui mouse. Misalkan pengguna menginputkan bentuk kurva detil pada gambar 4.4, kemudian pengguna melakukan proses penyimpanan dengan cara menekan tombol save dan tampil form yang terlihat pada gambar 4.5. Kurva detil yang dibuat pengguna bisa digunakan untuk proses analogi.



Gambar 4.4 Gambar kurva detil baru yang diinputkan pengguna

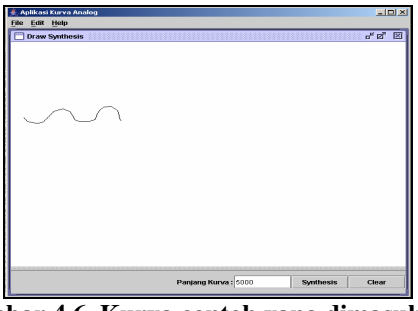


Gambar 4. 5. Form untuk menyimpan hasil kurva detil

Skenario 3: pengguna melakukan proses synthesis.

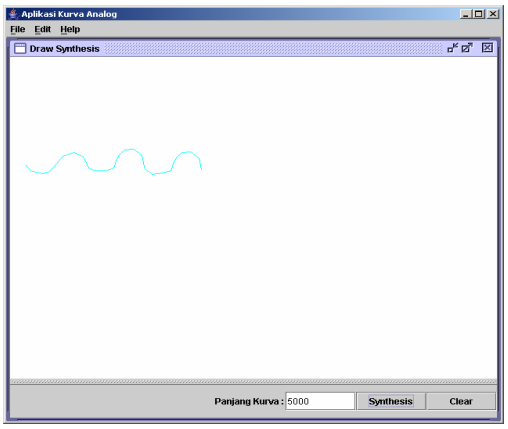
Pada skenario ini, pengguna dapat melakukan proses pembuatan kurva dengan algoritma *synthesis*. Langkah yang harus dilakukan pengguna dalam proses ini adalah sebagai berikut :

- pengguna memasukkan kurva contoh dengan menggunakan *mouse click*. Misalnya sebagaimana yang terlihat pada gambar 4.6.



Gambar 4.6. Kurva contoh yang dimasukkan pengguna

- selanjutnya pengguna memasukkan panjang kurva yang ingin disintesa. Misalnya 10000 (angka ini merepresentasikan jarak titik awal dengan titik akhir kurva yang akan disintesa).
- Hasilnya terlihat pada gambar 4.7. kurva hasil sintesa mempunyai bentuk yang mirip dengan kurva contoh. Jika kurva contoh dibuat dengan jumlah titik kontrol sedikit maka kurva hasil sintesa akan jelek.
- Panjang kurva yang akan disintesa akan mempengaruhi bentuk kurva.



Gambar 4.7. Kurva hasil sintesa

5. PENUTUP

a. Kesimpulan

- Adapun kesimpulan dari pembuatan Tugas Akhir ini adalah sebagai berikut:
- Dengan dasar metode *curve* analogies dapat dibuat editor kurva *Polyline* seperti yang dilakukan pada ujicoba skenario yang pertama dan kedua dengan hasil seperti pada gambar 4.3

b. Saran

- Saran untuk kemungkinan pengembangan lebih lanjut dari pembuatan tugas akhir ini adalah penggunaan model kurva lain selain kurva *polyline* misalnya kurva B-Spline, Bezier dan lain-lain untuk mendapatkan hasil sintesa kurva yang lebih baik dalam editor kurva *Polyline*.

6. DAFTAR PUSTAKA

1. Herztmann, Aaron, Nuria Oliver, Brian Curless, Steven M.Seitz, *Curve Analogies*, Washington, 2002.
2. 2n,Berthold K.P.,Hugh M. Hilden, Shahriar Negahdaripour, Closed-Form Solution of Absolute Orientation Using Orthonormal Matrices, Honolulu
3. Hearn, Donald, M. Pauline Baker, *Computer Graphics C Version*, Second Edition, Prentice Hall, New Jersey, 1997.
4. Efro, Alexei A. s,Thomas K. Leung, *Texture Synthesis by Non-parametric Sampling*, California, 1999.
5. Drakos, Nikos, *Differentiation in Parametric Form* http://www.maths.abdn.ac.uk/~igc/tch/ma1002/diff/node52.html, 2000
6. Herztmann, Aaron, Nuria Oliver, Brian Curless, Charles E.Jacoby, David H. Salesin, *Image Analogies*, Washington, 2002.
7. Soelaiman, Rully , Diktat Kuliah Grafika Komputer, Surabaya
8. Markosian, Lee, Michael A. Kowalski, Samuel J. Trychin, Lubomir D. Bourdev, Daniel Goldstein, John F. Hughes, *Real Time Nonphotorealistic Rendering*, SIGGRAPH '97, Conference Proceeding, Augustus 1997.
9. Ottmann, Thomas, Line simplification Algorithm a highlight on the DP-Star Algorithm, Universität zu Freiburg, Mei 2004.
10. Stewart, *Parametric curve*, workshop#2.
11. http://pcroot.cern.ch/TaligentDocs/TaligentOnline/DocumentRoot/1.0/Docs/books/Gs/Gs_208.hmtl
12. http://www.w3.org/1999/xhtml"><HEAD><TITLE>MySQL Manual | 19.2.4 Class Curve
13. Hertmann, Aaron, Charles E. Jacob, Nuria Oliver, Brian Curless and David H Salesin, *Image Analogies,* Proceeding of SIGGRAPH 2001, pages 327-340,2001.