LAPORAN TEKNIS

TUGAS KELOMPOK ANALISIS NUMERIK

Studi Penggunaan Bézier Curves dalam Pembuatan Ilustrasi Gambar yang Sudah Ada

Disusun oleh Kelompok C10:

Argya Farel Kasyara	2306152424
Alexander William Lim	2306207505
Jason Kent Winata	2206081313
Daffa Abhipraya Putra	2306245131

Fakultas Ilmu Komputer
Universitas Indonesia

2025

"Dengan ini, kami menyatakan bahwa tugas ini adalah hasil pekerjaan kelompok sendiri."

Argya Farel Kasyara

NPM: 2306152424

Alexander William Lim

NPM: 2306207505

Jason Kent Winata

NPM: 2206081313

Daffa Abhipraya Putra

NPM: 2306245131

Rangkuman

Laporan ini membahas implementasi metode numerik untuk merepresentasikan gambar raster menjadi ilustrasi vektor menggunakan kurva Bézier kubik. Fokus utama laporan ini adalah metode **Pure Interpolation**, di mana setiap titik kontur yang diekstraksi dari citra raster dihubungkan secara langsung menggunakan kurva Bézier. Selain itu, kami juga membandingkan metode ini dengan pendekatan optimisasi menggunakan *Least Squares Fitting* untuk menganalisis perbedaan efisiensi dan kualitas representasi.

Daftar Isi

1. Pendahuluan

1.1 Latar Belakang

Kurva Bézier merupakan salah satu primitif grafis terpenting dalam *Computer Assisted Design* (CAD) dan grafis vektor. Kemampuannya untuk memodelkan bentuk lengkung yang halus dengan hanya beberapa titik kontrol menjadikannya sangat efisien dibandingkan dengan representasi poligon atau raster. Dalam proyek ini, kami mengeksplorasi bagaimana mengonversi citra raster statis menjadi representasi matematis menggunakan kurva Bézier, sebuah proses yang dikenal sebagai *vectorization* atau *image tracing*.

1.2 Rumusan Masalah

Permasalahan utama yang dikaji adalah:

1. Bagaimana mengekstraksi titik-titik data yang merepresentasikan bentuk dari sebuah citra raster?
2. Bagaimana merepresentasikan sekumpulan titik data tersebut menggunakan kurva Bézier dengan metode interpolasi murni?
3. Bagaimana membandingkan efisiensi metode interpolasi murni dengan metode optimisasi *Least Squares*?
4. Bagaimana cara menghasilkan berkas keluaran standar (PDF) yang dapat memvisualisasikan hasil kurva tersebut?

1.3 Tujuan Penulisan

1. Mengekstraksi titik-titik data kontur dari citra raster.
2. Mengimplementasikan algoritma *Pure Interpolation* untuk menghubungkan setiap titik kontur dengan kurva Bézier.
3. Menganalisis perbandingan antara metode *Pure Interpolation* dan *Least Squares Fitting*.
4. Menghasilkan berkas PDF vektor dari hasil perhitungan kurva.

1.4 Batasan Masalah

- Kurva yang digunakan adalah Bézier kubik (derajat 3).
- Masukan berupa citra raster (PNG/JPG).
- Keluaran berupa berkas PDF.

2. Dasar Teori

2.1 Dasar Teori

Kurva Bézier derajat n didefinisikan oleh persamaan parametrik menggunakan polinomial Bernstein:

$$B(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i P_i, \quad 0 \leq t \leq 1$$

Untuk kasus Cubic Bézier ($n = 3$), persamaannya adalah:

$$B(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t) t^2 P_2 + t^3 P_3$$

Representasi ini menjamin bahwa kurva dimulai di P_0 (saat $t = 0$) dan berakhir di P_3 (saat $t = 1$). Garis singgung di P_0 searah dengan vektor $\vec{P_0 P_1}$, dan di P_3 searah dengan $\vec{P_2 P_3}$.

2.2 Pure Interpolation

Dalam metode *Pure Interpolation*, tujuan utamanya adalah membuat kurva yang melewati setiap titik data yang tersedia secara eksak. Jika diberikan sekumpulan titik $D_0, D_1, \ldots, D_k$, kita membuat segmen kurva Bézier antara setiap pasangan titik berurutan D_i dan D_{i+1}.

Secara matematis, untuk menjamin kontinuitas C^0 (kurva tersambung), titik akhir segmen ke-i harus sama dengan titik awal segmen ke-$(i+1)$. Ini sudah terpenuhi dengan menetapkan $P_0 = D_i$ dan $P_3 = D_{i+1}$.

Untuk menentukan titik kontrol internal P_1 dan P_2, kami menggunakan pendekatan interpolasi linear sederhana. Pendekatan ini menempatkan titik kontrol pada garis lurus yang menghubungkan D_i dan D_{i+1} dengan pembagian segmen yang merata (1/3 dan 2/3 jarak).

Rumus yang digunakan adalah:

$$P_1 = D_i + \frac{1}{3}(D_{i+1} - D_i) = \frac{2}{3} D_i + \frac{1}{3} D_{i+1}$$
$$P_2 = D_i + \frac{2}{3}(D_{i+1} - D_i) = \frac{1}{3} D_i + \frac{2}{3} D_{i+1}$$

Dengan konfigurasi ini, kurva Bézier kubik akan berimpit dengan garis lurus yang menghubungkan D_i dan D_{i+1}. Meskipun secara teknis ini adalah kurva derajat 3, bentuk geometrisnya adalah segmen garis lurus. Hal ini dipilih karena titik-titik kontur yang dihasilkan dari ekstraksi citra (pixel-based) sudah sangat rapat, sehingga interpolasi linear antar piksel sudah memberikan representasi visual yang sangat akurat tanpa perlu menebak kelengkungan antar piksel.

3. Metodologi Eksperimen

3.1 Pengolahan Citra (Image Processing)

Kami menggunakan pustaka `cv2` (OpenCV) untuk tahap pra-pemrosesan. Kami menggunakan *Canny Edge Detection* untuk mendeteksi tepi berdasarkan gradien intensitas. Metode ini bekerja dengan mencari area di mana intensitas piksel berubah drastis (misalnya dari hitam ke putih). Hasil dari tahap ini adalah sekumpulan titik koordinat (x, y) yang merepresentasikan garis tepi gambar.

3.2 Pure Interpolation Algorithm

Algoritma utama kami untuk metode ini (`fit_curve_pure_interpolation`) mengiterasi setiap titik kontur dan membentuk kurva Bézier.

Pseudocode:

```
Algorithm: Pure Interpolation Curve Fitting
Input: List of contour points D = [D_0, D_1, ..., D_k]
Output: List of Bezier curves

1. Initialize empty list 'curves'
2. If length(D) < 2:
       Return empty list

3. For i from 0 to k-1:
       // Set endpoints
       P0 = D[i]
       P3 = D[i+1]

       // Calculate internal control points (Linear Interpolation)
       // P1 is 1/3 of the way from P0 to P3
       P1.x = (2/3) * P0.x + (1/3) * P3.x
       P1.y = (2/3) * P0.y + (1/3) * P3.y

       // P2 is 2/3 of the way from P0 to P3
       P2.x = (1/3) * P0.x + (2/3) * P3.x
       P2.y = (1/3) * P0.y + (2/3) * P3.y

       // Create curve segment
       curve = [P0, P1, P2, P3]
       Append curve to 'curves'

4. Return 'curves'
```

3.3 Pembuatan PDF

Kami mengembangkan generator PDF (`pdf_generator.py`) yang menulis perintah-perintah grafis PDF secara manual.

Pseudocode:

```
Algorithm: Generate PDF
Input: List of Bezier curves, Output filename

1. Initialize PDF Structure:
   Write the PDF Header ("%PDF-1.4").

2. Construct Content Stream:
   For each curve in the list:
       Append "Move To" command (m) for P0.
       Append "Curve To" command (c) using P1, P2, P3.
       Append "Stroke" command (S).

3. Finalize PDF:
   Write Object Catalog, Pages, and Page objects.
   Write Cross-Reference Table (xref).
   Write Trailer and EOF.
```

4. Hasil dan Analisis (Pure Interpolation)

Pada bagian ini, kami menyajikan hasil eksperimen menggunakan metode **Pure Interpolation** pada beberapa jenis gambar.

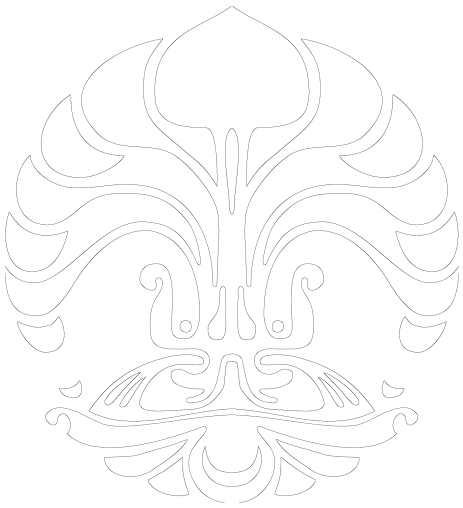
I. Percobaan 1: Makara UI (Kompleks)

Kami melakukan percobaan pada gambar logo Makara UI (`makara.png`) yang memiliki geometri kompleks.

Tabel 4.1 Statistik Pure Interpolation pada Makara UI

Metrik	Nilai
Dimensi Gambar	1890×2065 piksel
Total Kontur	46
Total Titik Asli	74,615 titik
Total Kurva Bézier	**74,569 kurva**
Rasio Kompresi	**1 kurva per 1.00 titik**

Visualisasi Hasil:

Input	Output (Pure Interpolation)
	

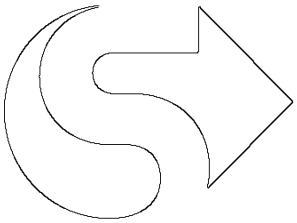
II. Percobaan 2: Logo Superbank (Sederhana)

Kami juga menguji pada logo Superbank (`superbank.png`) yang memiliki bentuk geometris sederhana.

Tabel 4.2 Statistik Pure Interpolation pada Logo Superbank

Metrik	Nilai
Dimensi Gambar	512×512 piksel
Total Kontur	3
Total Titik Asli	2,546 titik
Total Kurva Bézier	**2,543 kurva**
Rasio Kompresi	**1 kurva per 1.00 titik**

Visualisasi Hasil:

Input	Output (Pure Interpolation)
	

III. Percobaan 3: Gradient (Failure Case)

Kami melakukan percobaan pada gambar `gradient.png` di mana warna objek memiliki gradien yang halus menyatu dengan latar belakang atau perubahan warna yang tidak tajam.

Tabel 4.3 Statistik Pure Interpolation pada Gradient

Metrik	Nilai
Dimensi Gambar	1000×1000 piksel
Total Kontur	**0**
Total Titik Asli	**0 titik**
Total Kurva Bézier	**0 kurva**
Rasio Kompresi	**N/A**

Visualisasi Hasil:

Input	Output (Pure Interpolation)
	

Analisis Kegagalan: Pada percobaan ini, output PDF yang dihasilkan **kosong (blank)**. Hal ini disebabkan oleh cara kerja algoritma *Canny Edge Detection* yang bergantung pada perubahan intensitas piksel yang tajam (gradien tinggi) untuk mendeteksi tepi. Pada gambar dengan gradien warna yang halus, tidak terdapat batas tegas yang dapat diidentifikasi sebagai "tepi" oleh algoritma, sehingga tidak ada titik kontur yang diekstraksi. Akibatnya, tidak ada kurva yang dapat dibentuk.

IV. Analisis Umum Pure Interpolation

1. **Akurasi Sempurna:** Metode ini menjamin akurasi visual yang sangat tinggi karena setiap titik kontur direpresentasikan.
2. **Inefisiensi Ukuran:** Rasio kompresi konsisten pada angka 1:1. Jumlah kurva yang dihasilkan sangat masif (74 ribu untuk Makara), membuat ukuran file tidak efisien.
3. **Efek "Jaggedness" pada Zoom:** Meskipun akurat pada skala 1:1, metode ini sebenarnya melakukan aproksimasi linear antar piksel. Karena titik kontrol ditempatkan segaris dengan titik ujung, kurva Bézier yang dihasilkan efektif menjadi garis lurus. Jika gambar diperbesar (*zoom in*), area yang seharusnya melengkung halus akan terlihat patah-patah (*jagged*) atau seperti tangga (*staircase effect*), mengikuti grid piksel asli. Ini berbeda dengan kurva Bézier sejati yang tetap mulus pada pembesaran berapapun.
4. **Kompleksitas Gambar:** Perbandingan antara Makara UI dan Superbank menunjukkan korelasi langsung antara kompleksitas visual dan jumlah kurva. Makara UI memiliki bentuk organik dengan banyak ornamen dan lekukan, menghasilkan keliling (perimeter) yang sangat panjang, sehingga terdeteksi 74 ribu titik tepi. Sebaliknya, Superbank yang terdiri dari bentuk geometris sederhana memiliki keliling yang jauh lebih pendek, hanya menghasilkan sekitar 2.500 titik. Karena rasio kompresi adalah 1:1, jumlah kurva yang dihasilkan murni bergantung pada jumlah piksel tepi yang terdeteksi.

5. Menggunakan Least Square untuk Optimisasi

Sebagai perbandingan dan solusi atas inefisiensi metode *Pure Interpolation*, kami juga mengimplementasikan metode **Hybrid** yang menggunakan **Least Squares Fitting**.

5.1 Metodologi (Recursive Curve Fitting)

Algoritma ini menggunakan pendekatan *divide and conquer* untuk menemukan jumlah kurva minimal yang diperlukan untuk merepresentasikan kontur dengan tingkat kesalahan tertentu.

Prinsip Kerja:

1. **Fitting Awal:** Algoritma mencoba mencocokkan *satu* kurva Bézier kubik pada seluruh rangkaian titik kontur menggunakan metode kuadrat terkecil (*Least Squares*). Metode ini mencari posisi titik kontrol P_1 dan P_2 yang meminimalkan jumlah kuadrat jarak antara titik-titik data asli dan kurva hasil fitting.
2. **Evaluasi Galat:** Setelah fitting, algoritma menghitung jarak maksimum (*max error*) antara titik data asli dan kurva yang dihasilkan.
3. **Keputusan Rekursif:**
    - Jika *max error* berada di bawah ambang batas (*threshold*) yang ditentukan (misalnya 2.0 piksel), maka kurva tersebut diterima sebagai representasi yang valid.
    - Jika *max error* melebihi ambang batas, berarti satu kurva tidak cukup untuk merepresentasikan bentuk tersebut dengan akurat. Rangkaian titik kemudian dipecah (*split*) menjadi dua bagian pada titik dengan kesalahan terbesar.
4. **Rekursi:** Proses ini diulang secara rekursif untuk kedua bagian (kiri dan kanan) hingga seluruh segmen memenuhi kriteria galat.

Pseudocode:

```
Function fit_recursive(points, threshold):
    // Base case: not enough points to fit
    If length(points) < 2:
        Return empty list

    // Try to fit a single Bezier curve to all points
    curve = fit_cubic_bezier_least_squares(points)

    // Calculate the maximum deviation of points from this curve
    max_error, split_index = calculate_max_error(points, curve)

    // Check if the fit is good enough
    If max_error < threshold:
        Return [curve]
    Else:
```

```
// Fit is poor, split points at the point of max error
left_points = points[0 to split_index]
right_points = points[split_index to end]

// Recursively fit both halves
left_curves = fit_recursive(left_points, threshold)
right_curves = fit_recursive(right_points, threshold)

Return concatenate(left_curves, right_curves)
```

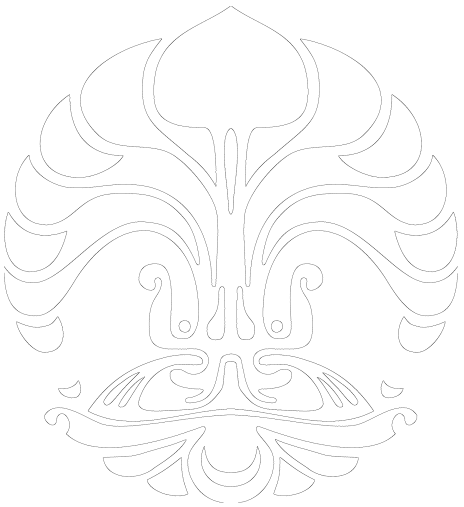
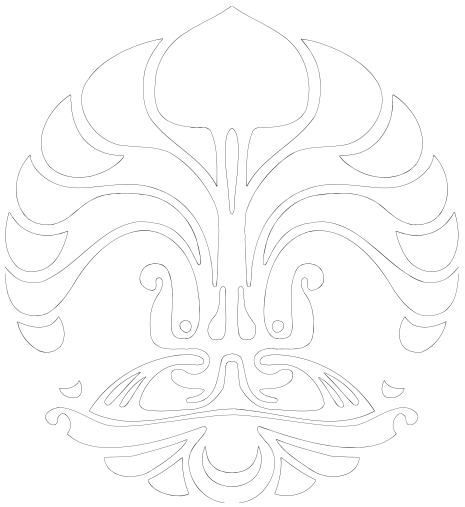
5.2 Hasil Eksperimen (Least Squares)

Kami menjalankan algoritma ini pada gambar Makara UI (`makara.png`) untuk membandingkan hasilnya.

Tabel 5.1 Statistik Least Squares pada Makara UI

Metrik	Nilai
Dimensi Gambar	1890×2065 piksel
Total Kontur	46
Total Titik Asli	74,615 titik
Total Kurva Bézier	**797 kurva**
Rasio Kompresi	**1 kurva per 93.62 titik**

5.3 Perbandingan Visual dan Efisiensi

Output (Pure Interpolation)	Output (Least Squares / Hybrid)
	
74,569 Kurva	**797 Kurva**

Gambar 5.1 Perbandingan Visual Pure Interpolation vs Least Squares

Analisis Perbandingan:

1. **Efisiensi Ekstrem:** Metode *Least Squares* mampu mereduksi jumlah kurva dari **74,569** menjadi hanya **797**. Ini adalah pengurangan sebesar **98.9%** dalam jumlah primitif grafis yang diperlukan. Hal ini dimungkinkan karena satu kurva Bézier kubik memiliki fleksibilitas untuk membentuk lengkungan kompleks (seperti huruf 'S' atau busur panjang). Dalam metode interpolasi murni, lengkungan panjang ini harus dibentuk oleh ratusan segmen garis kecil. Dengan *Least Squares*, ratusan titik tersebut dapat diaproksimasi oleh satu persamaan matematis saja.
2. **Kualitas Visual (Smoothness vs Exactness):** Secara visual, hasil *Least Squares* seringkali terlihat *lebih baik* daripada interpolasi murni. Interpolasi murni menangkap "noise" atau ketidaksempurnaan diskritisasi piksel (efek tangga), sedangkan *Least Squares* melakukan *smoothing* atau penghalusan. Kurva hasil fitting akan mengalir mulus di antara titik-titik data, merepresentasikan bentuk ideal yang dimaksudkan oleh gambar asli, bukan sekadar menghubungkan kotak-kotak piksel. Inilah sebabnya mengapa jumlah kurva yang jauh lebih sedikit (797 vs 74 ribu) tetap dapat menghasilkan gambar yang secara perseptual sama bagusnya, bahkan lebih estetis karena sifat kontinuitas C^1 atau C^2 yang implisit dalam kurva Bézier tunggal.
3. **Kesimpulan:** Metode *Least Squares* jauh lebih superior untuk keperluan ilustrasi vektor karena menghasilkan file yang ringan dan efisien tanpa mengorbankan kualitas visual secara signifikan.

6. Kesimpulan

Proyek ini berhasil mendemonstrasikan dua pendekatan dalam vektorisasi citra menggunakan kurva Bézier:

1. **Pure Interpolation:** Menghubungkan setiap titik data secara langsung. Metode ini menjamin akurasi 100% terhadap data input tetapi sangat tidak efisien (rasio kompresi 1:1), menghasilkan puluhan ribu kurva untuk gambar kompleks. Analisis menunjukkan bahwa metode ini rentan terhadap efek *jaggedness* saat diperbesar karena sifatnya yang *piecewise linear*.
2. **Least Squares Fitting (Optimisasi):** Menggunakan pendekatan statistik rekursif untuk mencocokkan kurva terbaik. Metode ini terbukti sangat efektif, mampu merepresentasikan gambar yang sama dengan jumlah kurva yang jauh lebih sedikit (rasio kompresi ~1:93). Efisiensi ini dicapai karena kemampuan kurva Bézier untuk mengaproksimasi lengkungan panjang yang mulus, menggantikan ratusan segmen linear kecil.
3. **Keterbatasan Deteksi Tepi:** Kedua metode sangat bergantung pada kualitas deteksi tepi (*Canny*). Pada gambar dengan gradien halus, metode ini gagal mengekstraksi kontur, menyebabkan kegagalan vektorisasi.

Untuk aplikasi praktis seperti desain grafis dan penyimpanan aset digital, metode optimisasi dengan *Least Squares* adalah pilihan yang jauh lebih baik dibandingkan interpolasi murni.

Daftar Referensi

Sauer, T. (2012). *Numerical Analysis* (2nd ed.). Pearson.

Lampiran

Complete Source Code: https://github.com/absolutepraya/bezier-curves-python