Variational Methods in Trajectory Optimization: A Calculus-Based Approach to Stroke Smoothing

Bhavesh Adhikari

Jan, 2024

1. Introduction and Problem Definition

In the domain of Human-Computer Interaction (HCI) and Digital Ink processing, the capture of hand-drawn trajectories is fundamentally a stochastic process. When a user draws a line, the resulting discrete signal $P = \{p_0, p_1, \ldots, p_n\}$ is a combination of the user's intent and high-frequency noise.

1.1. The Nature of Noise in Discrete Curves

Noise in discrete trajectories manifests as sharp discontinuities in the tangent and curvature vectors. In a physical sense, if the curve represented the path of a particle, these "jags" would imply infinite acceleration at discrete time steps. To produce an aesthetically pleasing and "natural" stroke, we must filter these accelerations while preserving the low-frequency positional data.

2. Mathematical Foundations

2.1. The Discrete Laplacian (Measuring Smoothness)

To quantify "smoothness," we utilize the second derivative of the trajectory. From the Taylor Series Expansion, we can express the values of a continuous function $f(x)$ at small offsets h:

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + O(h^3) \tag{1}$$

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) - O(h^3) \tag{2}$$

By summing equations (1) and (2), the first-order derivative terms $hf'(x)$ cancel out due to symmetry. Solving for $f''(x)$, we arrive at the **Central Difference Formula**:

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \tag{3}$$

In a discrete point cloud, we assume a unit distance between indices ($h = 1$). Thus, the "roughness" or curvature at index i is defined by the **Discrete Laplacian**:

$$\Delta P_i = P_{i+1} - 2P_i + P_{i-1} \tag{4}$$

2.2. Geometric Interpretation

The Laplacian expression can be algebraically rearranged to provide geometric intuition:

$$\Delta P_i = 2\left(\frac{P_{i+1} + P_{i-1}}{2} - P_i\right) \tag{5}$$

This formulation reveals that the second derivative is proportional to the distance between the current point P_i and the **midpoint** of its immediate neighbors. Consequently, if a point lies exactly on the segment connecting its neighbors, the Laplacian is zero, representing a state of local perfect smoothness.

3. Proposed Solution

3.1. Laplacian Iterative Smoothing

The most direct implementation of the Laplacian is a "Diffusion" approach. We iteratively update each point P_i by nudging it in the direction of the Laplacian:

$$P_i^{(next)} = P_i^{(curr)} + \gamma \Delta P_i \tag{6}$$

Where, γ represents the step size or smoothing factor.

Over successive iterations, high-frequency noise (analogous to thermal spikes) dissipates, causing the trajectory to converge toward a smoother state.But Runing this for higher iterations blindly, introduces a shrinkage problem so instead of this simple filter, we propose a variational framework. We seek a set of points Q that minimizes a total energy function $J(Q)$.

3.2. The Objective Function (Loss)

We define $J(Q)$ as a weighted sum of two competing energies:

$$J(Q) = \mathcal{E}_{data}(Q) + \lambda \mathcal{E}_{smooth}(Q) \tag{7}$$

- **Data Fidelity ($\mathcal{E}_{data}$):** Measured as the L_2 norm between optimized points q_i and raw points p_i. This prevents the curve from collapsing or drifting.
- **Smoothness ($\mathcal{E}_{smooth}$):** Measured as the squared norm of the Laplacian. This penalizes high curvature.

Expanding the terms:

$$J(Q) = \sum_{i=1}^{n} \|q_i - p_i\|^2 + \lambda \sum_{i=2}^{n-1} \|q_{i+1} - 2q_i + q_{i-1}\|^2 \tag{8}$$

4. Optimization Algorithm: Gradient Descent

To find the minimum of $J(Q)$, we use an iterative Gradient Descent solver. This is the same logic used in training neural networks.

4.1. Gradient Derivation

For a point q_i, the gradient ∇J is the partial derivative:

$$\frac{\partial J}{\partial q_i} = 2(q_i - p_i) + \lambda \text{ (High order terms from the Laplacian sum)} \tag{9}$$

By moving points in the direction of $-\nabla J$, we simultaneously pull the points toward their neighbors and toward their original "noisy" positions.

5. Refinement: Cubic Spline Representation

Once the point cloud Q is optimized, it remains a discrete set. To achieve C^2 continuity (where the second derivative is continuous everywhere), we fit a Cubic Spline. A cubic spline is a piecewise polynomial $S_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i$. By solving for coefficients that ensure the first and second derivatives match at every point, we generate a curve that is mathematically "perfect" for visual rendering and ensures:

- The curve passes exactly through the optimized anchor points.
- The first derivative $S'(t)$ (tangent/velocity) is continuous.
- The second derivative $S''(t)$ (curvature) is continuous.

This final interpolation step transforms the discrete point cloud into a professional-grade, smooth vector stroke.

6. Computational Algorithm

The smoothing objective is solved using an iterative Gradient Descent approach. By differentiating the total energy $J(P)$, we derive the update rule that balances positional accuracy with curvature minimization.

Algorithm 1 Laplacian-Regularized Gradient Descent for Stroke Smoothing

Input: Raw coordinate sequence $\mathbf{P} \in \mathbb{R}^{n \times 2}$, smoothing coefficient λ, learning rate η, max iterations T
Initialize: $\mathbf{Q}^{(0)} \leftarrow \mathbf{P}$

for $t = 0$ **to** $T - 1$ **do**
    Compute Residuals (Fidelity Gradient):
    $\mathbf{G}_f \leftarrow \mathbf{Q}^{(t)} - \mathbf{P}$ ▷ Force pulling point to original location
    Compute Curvature (Smoothing Gradient):
    for $i = 1$ **to** $n - 1$ **do**
        $\mathbf{L}_i \leftarrow \mathbf{Q}_{i+1}^{(t)} - 2\mathbf{Q}_i^{(t)} + \mathbf{Q}_{i-1}^{(t)}$ ▷ The Discrete Laplacian
    end for
    Update Internal Points:
    for $i = 1$ **to** $n - 1$ **do**
        $\mathbf{Q}_i^{(t+1)} \leftarrow \mathbf{Q}_i^{(t)} - \eta \cdot (\mathbf{G}_{f,i} - \lambda \mathbf{L}_i)$
    end for
    Enforce Boundary Conditions:
    $\mathbf{Q}_0^{(t+1)} \leftarrow \mathbf{P}_0, \quad \mathbf{Q}_n^{(t+1)} \leftarrow \mathbf{P}_n$ ▷ Keep endpoints fixed
end for

Output: Optimized smooth trajectory $\mathbf{Q}^{(T)}$

7. Conclusion

This report demonstrates that line smoothing is not merely a visual trick, but a formal optimization problem rooted in Taylor calculus. By balancing data fidelity and Laplacian curvature, we can effectively reconstruct human intent from noisy digital signals.