Reliability-based G^1 Continuous Arc Spline Approximation

Jinhwan Jeon[a], Yoonjin Hwang[a,b], Seibum B. Choi[a,*]

[a] *Department of Mechanical Engineering, KAIST, 291 Daehak-ro Yuseong-gu, Daejeon, 34141, Republic of Korea*
[b] *Hankook & Company, 286 Pangyo-ro Bundang-gu, Seongnam, 13494, Republic of Korea*

Abstract

This paper introduces an algorithm for approximating a set of data points with G^1 continuous arcs, leveraging covariance data associated with the points. Prior approaches to arc spline approximation typically assumed equal contribution from all data points, resulting in potential algorithmic instability when outliers are present. To address this challenge, we propose a robust method for arc spline approximation, taking into account the 2D covariance of each data point. Beginning with the definition of models and parameters for **single-arc approximation**, we extend the framework to support **multiple-arc approximation** for broader applicability. Finally, we validate the proposed algorithm using both synthetic noisy data and real-world data collected through vehicle experiments conducted in Sejong City, South Korea.

Keywords:
Reliability, Arc Splines, G^1 Continuity, Constrained Nonlinear Least Squares, Optimization

1. Introduction

Various approaches have been developed to analyze point data geometry or to smooth sequences of points using different curve families. These techniques find applications in fields such as numerically controlled (NC) machining [1–3], curve reconstruction [4], and road lane data parameterization [5–7].

* Corresponding Author

Email addresses: jordan98@kaist.ac.kr (Jinhwan Jeon), yoonjinh@kaist.ac.kr, yoonjin.hwang@hankookn.com (Yoonjin Hwang), sbchoi@kaist.ac.kr (Seibum B. Choi)

Numerous curve types have been proposed for these purposes [8–10], among which the use of arcs for data approximation stands out due to its simplicity and translation/rotation invariant properties.

1.1. Literature Review and Problem Statement

Significant research has been conducted on arc spline approximation utilizing biarcs[11–15]. For instance, in the study by [15], a single 3D arc was parameterized by a position vector ($\mathbb{R}^3$), two length-related parameters ($\mathbb{R}$), and ZXZ-Euler angles to form a rotational matrix $\in SO(3)$. With these parameters, a single arc could be optimized through unconstrained optimization. By employing biarc interpolation between two arcs, [15] successfully parameterized data points into multiple arc segments, avoiding the need for constrained optimization. However, this approach had its limitations. When data points closely approximated a line, the length parameter became infinitesimally small, leading to near-singularity issues during Gaussian step computation. Additionally, when performing approximation using [15], on average, one arc segment was allocated for every 4 data points, which did not effectively fulfill the original objective of compact data representation.

Some studies encountered algorithmic instability due to data noise. While RANSAC[16] is a common method for removing outlier or noisy points[6], it has drawbacks: no time limit for robust regression and no assurance of complete outlier removal. Other approaches like [7, 14, 17] defined tolerance channels as lateral offsets of data points. These studies generated arc splines that were kept within the tolerance channels but assumed accurate data (low noise levels). Moreover, since tolerance channels applied equal lateral offsets to all data points, noise can significantly compromise algorithm performance and stability.

1.2. Overview of Our Approach

Our goal is to address arc spline approximation challenges with noisy data in a compact and robust manner. To overcome limitations in existing approaches, we emphasize the varying importance of each data point in the approximation process. By incorporating 2D covariance information for each data point, we devised an optimization problem reflecting this notion.

The paper is structured as follows: **Section 2** details our method for single arc approximation, including parameter definitions and cost function modeling. We evaluate this approach using both synthetic and real-world datasets. In **Section 3**, we extend the single-arc models to the multiple-arc approximation

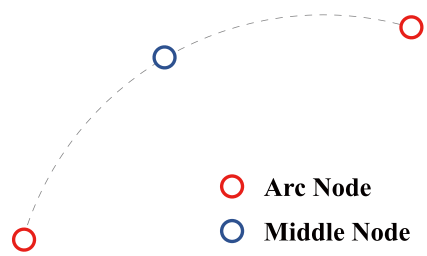

Figure 1: 3 Points are set as parameters(optimization variables) to define a **single arc**

framework. From arc parameter initialization, cost function for optimization and validation/update procedure will be discussed. Next, in **Section 4**, evaluation of the multiple-arc approximation framework using generated and real-world data is presented. Finally, **Section 5** explores potential applications and future research directions.

1.3. Contributions

Our main contributions are as follows:

- Cost function/constraint models for **Single Arc Approximation**
- Arc parameter initialization for **Multiple Arc Approximation**
- Cost function/constraint models for **Multiple Arc Approximation**
- Arc parameter validation/update for **Multiple Arc Approximation**

2. Single Arc Approximation

In this section, an algorithm for single-arc approximation is discussed. We define arc parameters and introduce a novel cost function and constraint model, which will later be adapted for multiple-arc approximation with modifications.

2.1. Parameters for Single Arc

Various parameter combinations exist to represent a single arc [7, 10, 15, 17]. To ensure stable and accurate convergence in our nonlinear optimization problem for data approximation, it is crucial to have optimization variables on similar numerical scales. Extreme discrepancies in variable scales can lead to significant numerical errors, potentially causing algorithm divergence or reduced accuracy.

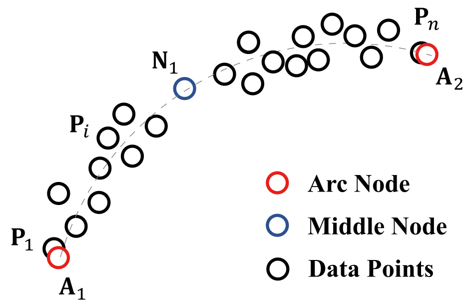

Figure 2: Anchor Model for Single Arc Approximation: Arc Nodes ($\mathbf{A}_1, \mathbf{A}_2$) are matched with first and last data points ($\mathbf{P}_1, \mathbf{P}_n$) respectively. Middle node is not included in the anchor model.

Thus, we define an arc with three points as optimization variables, as shown in Figure 1. These points—two red points representing **arc nodes** and a blue point representing the **middle node**—have similar numerical scales. By refining their positions during optimization, we aim to achieve the best-fit single arc for the given data points and covariance.

2.2. Optimization Models

To accurately approximate data points using arc parameters(Figure 1), a well-modeled cost function is crucial.In this section, we present three models that form the complete cost function for **single arc approximation**.

2.2.1. Single Arc Anchor Model

The first cost function model, the **anchor model**, is essential for stabilizing optimization by aligning the arc nodes with the first and last data points, as shown in Figure 2. Assuming ordered data points, it matches the first and second arc nodes with the first and last data points, respectively. The model cost is calculated as the sum of squared, weighted Euclidean distances between matched points, which can be written as follows:

$$\begin{aligned}\mathscr{L}_{\mathrm{AC}} &= \|\mathbf{P}_1 - \mathbf{A}_1\|^2_{\Sigma_{\mathrm{AC}}} + \|\mathbf{P}_n - \mathbf{A}_2\|^2_{\Sigma_{\mathrm{AC}}} \\ &= (\mathbf{P}_1 - \mathbf{A}_1)^\top {\Sigma_{\mathrm{AC}}}^{-1} (\mathbf{P}_1 - \mathbf{A}_1) + (\mathbf{P}_n - \mathbf{A}_2)^\top {\Sigma_{\mathrm{AC}}}^{-1} (\mathbf{P}_n - \mathbf{A}_2)\end{aligned} \tag{1}$$

$\mathbf{P}_1$ and $\mathbf{P}_n$ are the first and the last points in the dataset, $\mathbf{A}_1$ and $\mathbf{A}_2$ stand for the first and second arc nodes respectively. The notation $\|\cdot\|^2_\Sigma = (\cdot)^\top \Sigma^{-1} (\cdot)$ denotes the squared Mahalanobis Distance, which can be thought of as the square of weighted Euclidean distance. The weight is reflected in the cost by additionally

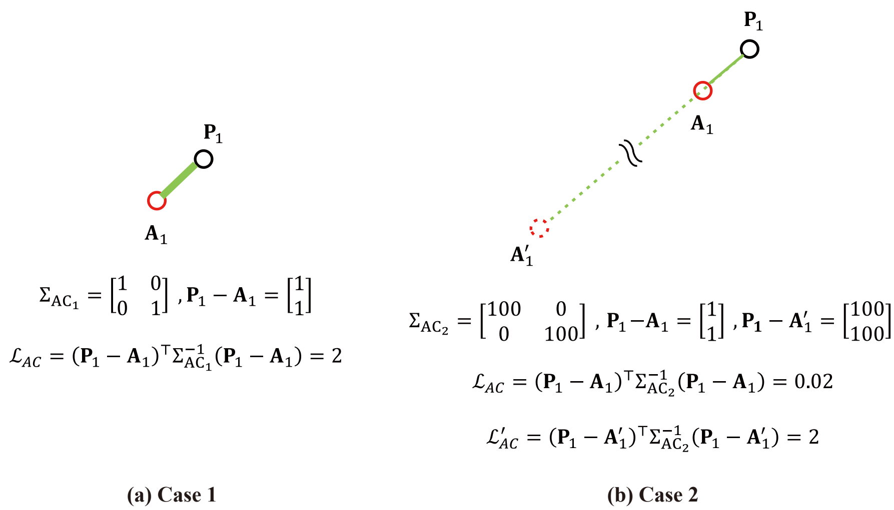

Figure 3: Anchor Model showcasing two scenarios with varying anchor covariance.

multiplying the inverse of the covariance matrix Σ_{AC} to the squared 2-norm of the original residual vectors $\mathbf{P}_1 - \mathbf{A}_1$ and $\mathbf{P}_2 - \mathbf{A}_2$, as shown in equation 1.

Remarks on Anchor Model Covariance Σ_{AC}

The anchor model cost, controlled by the covariance matrix Σ_{AC} in Equation 1, can be adjusted to influence the optimization process. For instance, starting with an identity matrix for Σ_{AC_1} (Case 1 in Figure 3), increasing its diagonal terms to 100 (Case 2) reduces the cost by a factor of $\frac{1}{100}$ due to the inverse relationship in Equation 1. This adjustment allows arc nodes to move further from matched data points without significantly increasing the cost. Notably, the exclusion of the middle node (labeled $\mathbf{N}_1$ in Figure 2) from the anchor model cost ensures its flexibility during optimization, essential for determining the arc's radius accurately.

2.2.2. Single Arc Measurement Model

The second cost function model, the arc measurement model minimizes the discrepancy between data points and the arc approximation by adjusting the middle node's position, refining the arc shape.

Cost Computation

The process of computing the arc measurement model cost is as follows. Refer to Figure 4 for graphical understanding.

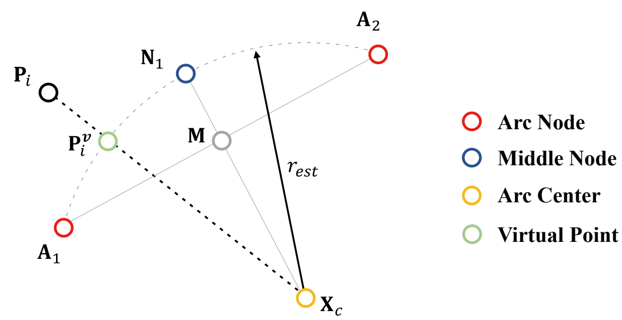

Figure 4: Arc Measurement Model for Single Arc Approximation: Residual is computed by obtaining the difference of the data point $\mathbf{P}_i$ and matched virtual point $\mathbf{P}_i^v$. Point **M** is the middle point of **A1** and **A2**, which will be used when explaining the third cost model.

1. Compute the position of arc center $\mathbf{X_C}$ from the positions of two Arc Nodes and the Middle Node, using simple geometry.
2. Set loop variable i to iterate from 1 to n (data size). For example, point $\mathbf{P}_i$ is chosen for explanation.
3. Find the intersection of line $\mathbf{P}_i\mathbf{X}_C$ and arc $\overset{\frown}{A_1 N_1 A_2}$, and set this point as the virtual point $\mathbf{P}_i^v$.
4. Arc measurement model residual $\mathbf{r}_{\text{ME}}^i$ is defined as the difference between data point $\mathbf{P}_i$ and virtual point $\mathbf{P}_i^v$: $\mathbf{r}_{\text{ME}}^i = \mathbf{P}_i^v - \mathbf{P}_i$.
5. Using the covariance Σ_{ME}^i of point $\mathbf{P}_i$ obtained beforehand, residual $\mathbf{r}_{\text{ME}}^i$ is weighted (squared Mahalanobis distance).
6. Steps 3 to 5 are repeated for the whole dataset(i iterating from 1 to n). The arc measurement model sums up all the costs computed in step 5.

The arc measurement residual introduced above is derived algebraically as:

$$\begin{aligned}
\mathbf{r}_{\text{ME}}^i =& \mathbf{P}_i^v - \mathbf{P}_i \\
=& \mathbf{X}_c + r_{est} \frac{\mathbf{P}_i - \mathbf{X}_c}{\|\mathbf{P}_i - \mathbf{X}_c\|} - \mathbf{P}_i \\
=& \left(\frac{r_{est}}{\|\mathbf{P}_i - \mathbf{X}_c\|} - 1 \right) (\mathbf{P}_i - \mathbf{X}_c)
\end{aligned} \tag{2}$$

where, r_{est} (estimated arc radius), $\mathbf{X}_c$ are computed using two arc nodes and the middle node. Finally, with the derived residual $\mathbf{r}_{\text{ME}}^i$ for each data point $\mathbf{P}_i$, we can compute the cost function for the arc measurement model as follows:

$$
\begin{aligned}
\mathscr{L}_{\mathrm{ME}} &= \sum_{i=1}^{n} \|\mathbf{r}_{\mathrm{ME}}^{i}\|_{\Sigma_{\mathrm{ME}}^{i}}^{2} \\
&\sum_{i=1}^{n} \|\mathbf{P}_{i}^{v} - \mathbf{P}_{i}\|_{\Sigma_{\mathrm{ME}}^{i}}^{2} \\
&= \sum_{i=1}^{n} \|\mathbf{r}_{\mathrm{ME}}^{i}(\mathbf{A}_1, \mathbf{A}_2, \mathbf{N}_1, \mathbf{P}_i)\|_{\Sigma_{\mathrm{ME}}^{i}}^{2}
\end{aligned}
\tag{3}
$$

Since the virtual point $\mathbf{P}_i^v$ is derived from $\mathbf{A}_1, \mathbf{A}_2, \mathbf{N}_1$ and point $\mathbf{P}_i$, we can write the residual $\mathbf{r}_{\mathrm{ME}}^i$ as a function of $\mathbf{A}_1, \mathbf{A}_2, \mathbf{N}_1$ and point $\mathbf{P}_i$ in equation 3. Here, the squared Mahalanobis Distance is used again for weighting each residual with covariance matrix Σ_{ME}^i. Also, note that all the data points have different covariance matrices Σ_{ME}^i, and therefore the arc will be optimized so that approximation error can be reduced further for data points with higher reliability.

2.2.3. Single Arc Equality Constraint 1: Middle Node

The final model included in the cost function is an equality constraint that restricts the relative positions of optimization variables $\mathbf{A}_1, \mathbf{A}_2, \mathbf{N}_1$. Other than the two arc nodes that represent both ends of the arc, we have set the middle point of the arc as one of the arc parameters (optimization variable). The middle node $\mathbf{N}_1$ should lie on the perpendicular bisector of line segment $\overline{\mathbf{A}_1\mathbf{A}_2}$. This can be implemented by taking the inner product of vectors $\overrightarrow{\mathbf{A}_1\mathbf{A}_2}$ and $\overrightarrow{\mathbf{M}\mathbf{N}_1}$ in Figure 4, and equating the result to zero.

$$
\mathbf{r}_{\mathrm{Eq1}} = (\mathbf{A}_2 - \mathbf{A}_1)^{\top} (\mathbf{N}_1 - \mathbf{M}) = (\mathbf{A}_2 - \mathbf{A}_1)^{\top} \left(\mathbf{N}_1 - \frac{1}{2} (\mathbf{A}_1 + \mathbf{A}_2) \right) \tag{4}
$$

The equality constraint (equation 4) will be added to the original cost function together with the Lagrange multiplier during optimization.

2.3. Single Arc Approximation: Augmented Cost Function and Optimization

Wrapping up the proposed cost function models and equality constraint model, we can rewrite the optimization problem as follows.

$$
\begin{aligned}
\min_{\mathbf{A}_1, \mathbf{A}_2, \mathbf{N}_1} \mathscr{L} &= \mathscr{L}_{\mathrm{AC}} + \mathscr{L}_{\mathrm{ME}} \\
&= \|\mathbf{P}_1 - \mathbf{A}_1\|_{\Sigma_{\mathrm{AC}}}^{2} + \|\mathbf{P}_n - \mathbf{A}_2\|_{\Sigma_{\mathrm{AC}}}^{2} \\
&\quad + \sum_{i=1}^{n} \|\mathbf{r}_{\mathrm{ME}}(\mathbf{A}_1, \mathbf{A}_2, \mathbf{N}_1, \mathbf{P}_i)\|_{\Sigma_{\mathrm{ME}}^{i}}^{2} \\
\text{s.t.} \quad & \mathbf{r}_{\mathrm{Eq1}} = 0
\end{aligned}
\tag{5}
$$

Equation 5 is the final cost function with an equality constraint, and the optimization variables are the two arc nodes $\mathbf{A}_1, \mathbf{A}_2$ and the middle node $\mathbf{N}_1$. This type of optimization problem can be classified as a typical constrained nonlinear least squares(CNLS) optimization problem. Note that the cost function in equation 5 can be balanced by controlling the anchor model covariance Σ_{AC}.

2.3.1. Typical Method of Solving Nonlinear Least Squares (NLS) Problem

Before solving the constrained version of nonlinear least squares, we first introduce briefly on solving unconstrained nonlinear least squares.

Unconstrained Nonlinear Least Squares

Let $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$ be a vector function with $m \geq n$. The main objective is to minimize $\|\mathbf{f}\|$, or equivalently to find

$$\mathbf{x}^* = \mathrm{argmin}_{\mathrm{x}}\, F(\mathbf{x}) \tag{6}$$

where

$$F(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{m}\left(f_i(\mathbf{x})\right)^2 = \frac{1}{2}\|\mathbf{f}(\mathbf{x})\|^2 = \frac{1}{2}\mathbf{f}(\mathbf{x})^\top \mathbf{f}(\mathbf{x}) \tag{7}$$

The detailed derivation of the well-known Gauss-Newton method for solving unconstrained NLS problems is introduced in [18]. For each iterative optimization, the optimization variable (vector) $\mathbf{x}$ is updated as

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha \mathbf{h}_{\mathrm{gn}}^k \tag{8}$$

The equation above describes the variable update at kth iteration. Here, α is the step size, which is set as 1 in the classical Gauss-Newton method. For other advanced methods, various line search methods are used for finding the value of α. Other than the Gauss-Newton method, Levenberg-Marquardt method, and Powell's Dog-Leg method are most widely used when solving general unconstrained NLS problems. The core difference between these three algorithms lies in how the update vector $\mathbf{h}$ is calculated.

2.3.2. Solving Constrained Nonlinear Least Squares (CNLS) Problem

Equality or inequality constraints are added to the previously introduced NLS to form the CNLS problem. For solving CNLS, advanced techniques such as Barrier / Penalty method, Broyden - Fletcher - Goldfarb - Shanno(BFGS) Hessian Approximation and Lagrange Multiplier are needed.

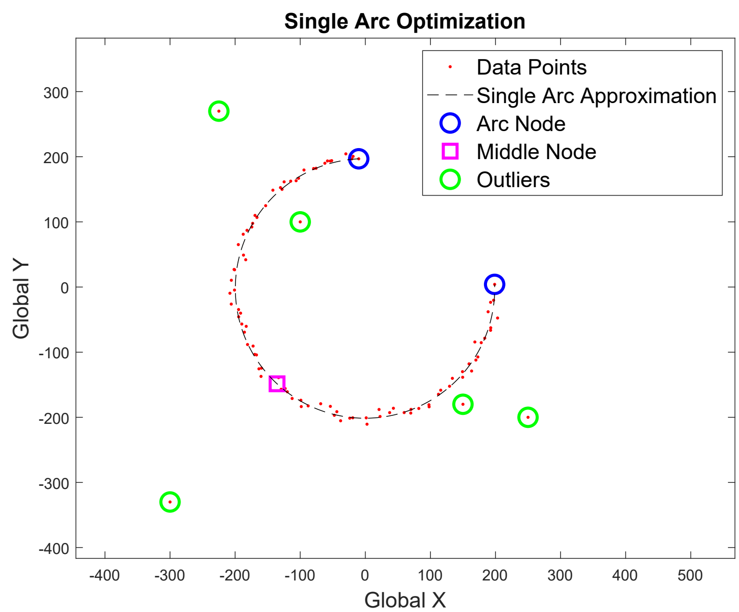

Figure 5: Single Arc Approximation Example 1 (Generated Data Points with Outliers)

Returning to our original problem introduced in equation 5, optimization variables for single arc approximation are the two arc nodes $\mathbf{A}_1, \mathbf{A}_2$ and the middle node $\mathbf{N}_1$. These arc parameters are augmented as a column vector $\mathbf{x}$, and will be iteratively updated in the CNLS solver. The optimal solution of the proposed CNLS problem in equation 5 is obtained using 'lsqnonlin.m' of MATLAB optimization toolbox [19].

2.4. Single Arc Approximation: Examples

Before moving on to multiple arc approximation, we test the proposed single arc approximation with generated data points and covariance. For data generation, white Gaussian noise was added to true points on the arc. The covariance matrix for each data point Σ^i_{ME} was set to have random diagonal elements from 1^2 to 30^2. Moreover, covariance of anchor model was set to have diagonal elements of 0.01^2 throughout this paper.

As we can observe from Figure 5, noisy generated data points with varying covariance and even outlier points are well-fitted into a single arc. We assumed here that the outlier points have large covariance values (50^2 to 200^2, low reliability). Other than generated data, the single arc approximation is also tested with real-world collected data points.

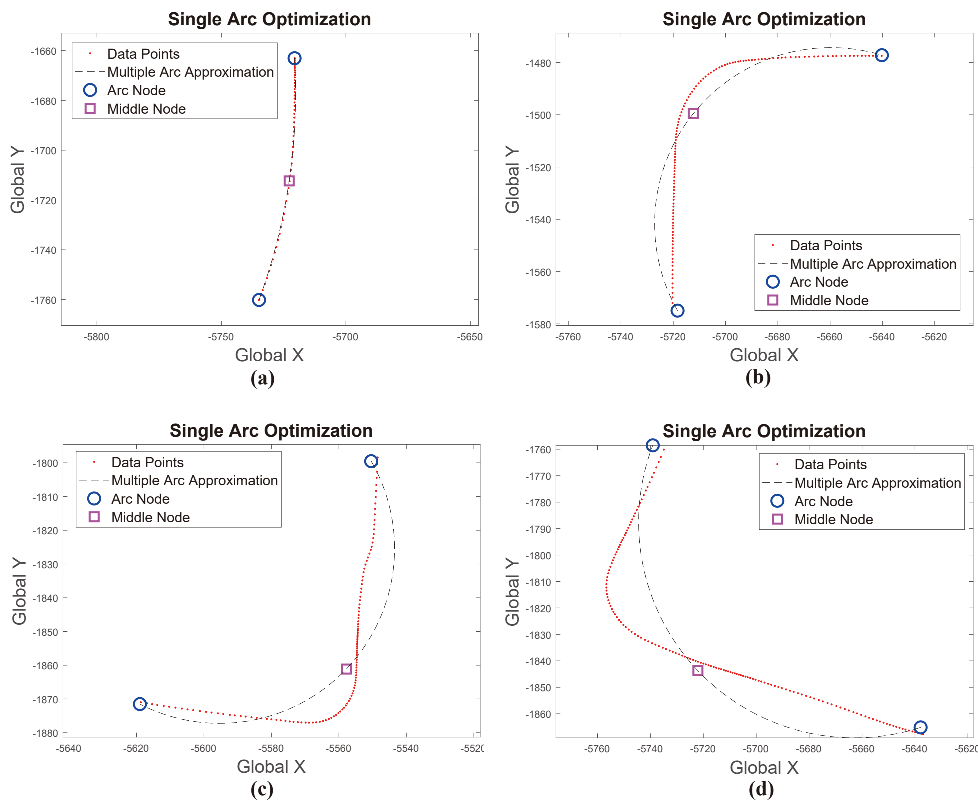

Figure 6: Single Arc Approximation Example 2 (Real-World collected data points from vehicle experiment in Sejong city, South Korea)

Data points introduced in Figure 6 are computed by fusing vehicle trajectory and lane detection results, in Sejong city, South Korea. The detailed process of obtaining data point covariance is introduced in [20]. While single arc approximation seems to be acceptable for cases (a), it is quite obvious that data approximation with only one arc is not enough for cases (b), (c) and (d). In order to tackle the limits of single-arc approximation, reliability-based multiple-arc approximation will be covered in Section 3.

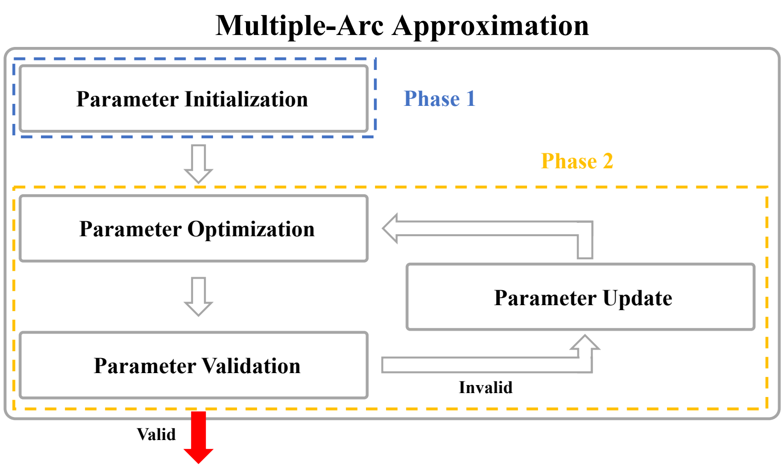

Figure 7: Multiple-Arc Approximation framework: (Phase 1) Initialization of arc parameters, (Phase 2) Obtaining the set of arc parameters that satisfies the approximation error condition.

3. Multiple Arc Approximation

In this section, we extend the concept of the single-arc approximation to multiple-arc approximation. Although the idea seems straightforward, there are several more key factors to consider, as shown below.

- Parameters of arc segments should be initialized for stable convergence.
- Arc nodes overlap for adjacent arc segments.
- (Data point - arc segment) matching is needed for optimization.
- All arc segments should satisfy G^1 continuity.
- A validating process of arc parameters is needed.
- A determination process of when to end the approximation is needed.

The multiple-arc approximation framework will be designed in a way that reflects all the arguments mentioned above. Modified cost functions / constraints will be explained, and the proposed framework will be tested on real-world collected data points and covariance.

3.1. Multiple Arc Approximation Framework

The overall framework for multiple-arc approximation is presented in Figure 7. The data approximation process can be divided into 2 phases.

In **phase 1**, the initial number of arc segments is determined, and corresponding arc parameters (arc nodes and middle nodes) are initialized by utilizing single-arc approximation discussed in Section 2. The purpose of initialization is to obtain initial parameter values of adequate quality to avoid divergence during the optimization step in phase 2.

Then in **phase 2**, CNLS optimization is performed based on several cost function and constraint models. The main difference between single-arc and multiple-arc approximation occurs directly after the arc parameter optimization. While the single-arc approximation ends right away, the multiple-arc approximation framework performs additional arc parameter validation using arc approximation errors and covariance of each data point. If the current arc parameter (arc nodes and middle nodes) set is acceptable after the validation process, optimization ends. If not, the number of segments is increased by one and the parameter optimization step is repeated until all the arc segments are valid.

3.2. Multiple Arc Approximation Phase 1: ***Parameter Initialization***

In the phase 1: parameter initialization step, the initial number of arc segments needed is computed. Using this information, the arc parameters are initialized separately using single-arc approximation proposed in Section 2.

3.2.1. Recursive Linear Approximation of Data Points

To determine the initial number of arc segments for approximating data points, the rough shape of the given points should be known. Assuming that the points are well ordered, we perform recursive linearization to approximate data points into polylines(i.e. multiple connected lines). Here, the **divide-and-conquer** algorithm is implemented for the recursive data point linearization.

A brief explanation of the algorithm is as follows. We assume that there are a total of n data points.

Step 1. Set initial interval of interest as [1, n]

Step 2. Connect the first index and the last index data point with a single line.

Step 3. Find the index of the point(**Idx**) with the largest linear fit error.

Step 4. Check if the largest error is above the threshold (Yes/No).

Step 5. (a) (Yes) Divide intervals into [1, **Idx**], [**Idx**, n] and repeat from Step 2.
(b) (No) Return current intervals (Will be propagated back)

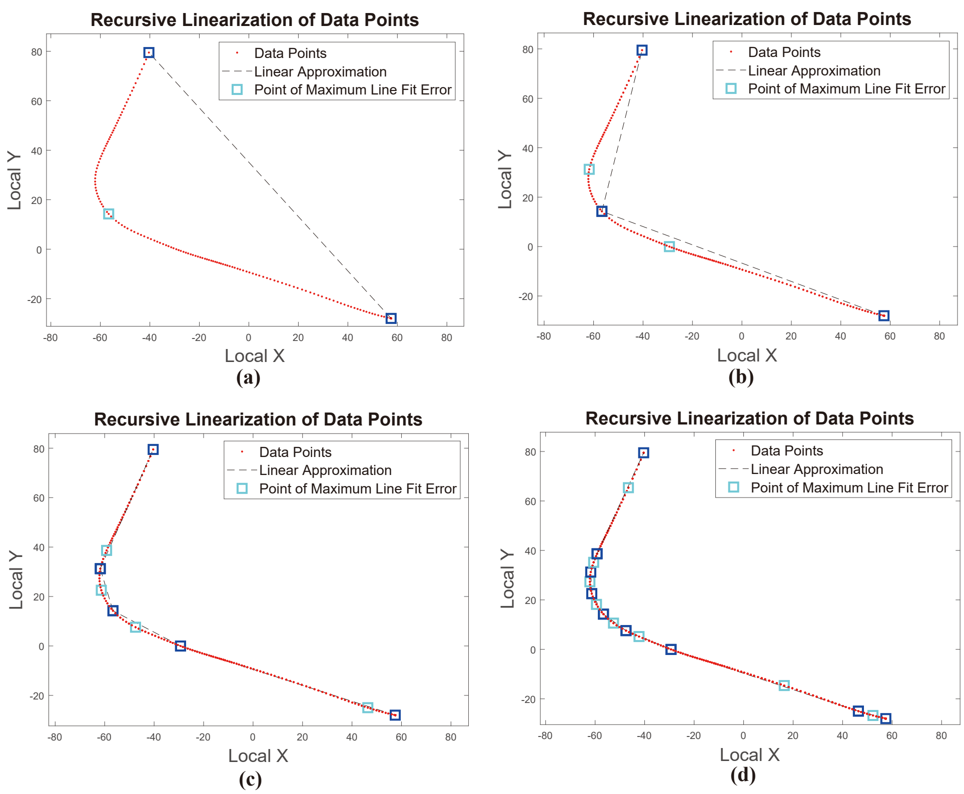

Figure 8: Example of Recursive Linearization: Tested on sample data points (from (a) to (d))

A sample result of recursive linear approximation is shown in Figure 8. As a result of recursive linear approximation, data point intervals for piecewise linear approximation can be obtained. Since these linear approximation intervals contain the rough shape of given data points, we can now determine the initial number of arc segments(i.e. the number of arc nodes and middle nodes needed for creating the initial optimization variable).

3.2.2. Determining Initial Number of Arc Segments

To determine the initial number of arc segments, we iteratively merge previously obtained linear approximation intervals and evaluate the validity of single-arc approximations (explained in Section 3.4). This process, depicted in Figure 9, utilizes vertical lines to mark boundaries of linear approximation indices, with **lb** and **ub** denoting the lower and upper bounds of data, respectively.

Initially, at step (a), data points between **lb** and **ub** form a single line. By po-

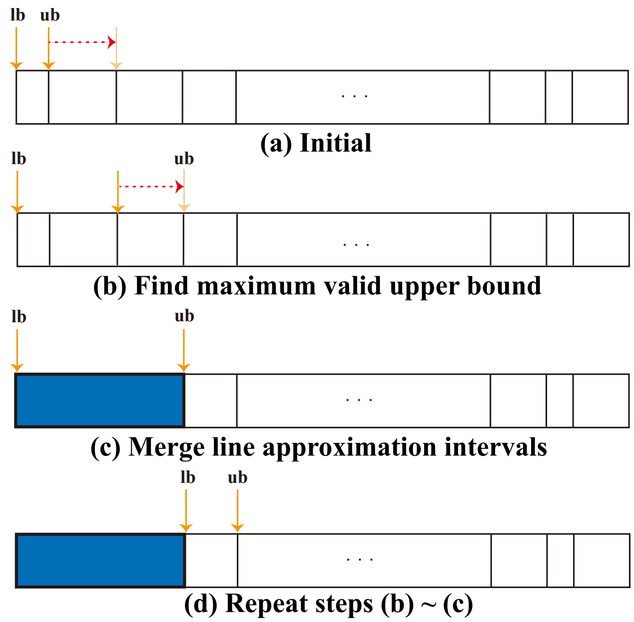

Figure 9: Merge Linear Approximation Intervals for Single Arc Approximation

sitioning the middle node (N1) close to the midpoint of the two arc nodes (A1, A2), our single-arc approximation creates an arc segment resembling a line, ensuring the approximation's validity for the initial interval [**lb**, **ub**]. Subsequently, at step (b), the upper bound **ub** is incremented along the linear approximation interval boundaries until the single-arc approximation becomes invalid for the data points between **lb** and **ub**. When this occurs, **ub** returns to the most recently valid boundary index. In step (c), the linear approximation intervals from **lb** to **ub** are merged, indicating that these data points will be fitted as a single arc during the initialization phase.

Steps (b) to (c) are repeated until **ub** reaches the final data point. This iterative process computes the initial number of arc segments and the initial arc approximation intervals required to represent the given data points. Such an approach prevents algorithmic inefficiencies by avoiding fixed initialization with a single arc segment.

3.2.3. Multiple Arc Parameter Initialization

After obtaining initial intervals for multiple arc approximation, we initialize arc parameters (arc nodes and middle nodes) for each segment via single-arc approximation. For adjacent segments, overlapping arc nodes are addressed

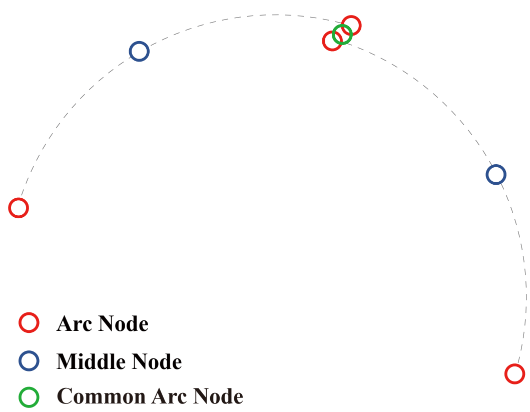

Figure 10: Parameter Initialization for Multiple Arcs: To consider two arcs as a single set of arc parameters, the common arc node (green point) is defined to be shared between two arcs.

by assigning a common arc node. This common node's position is determined by averaging the positions of the overlapping nodes. For example, Figure 10 illustrates this, where 3 arc nodes (2 original, 1 common) and 2 middle nodes are initialized as optimization variables. Subsequently, these parameters will be optimized based on the cost/constraint models for multiple-arc approximation in phase 2.

3.3. Multiple Arc Approximation Phase 2: ***Parameter Optimization***

Moving on to multiple-arc approximation framework phase 2, as shown in Figure 7, the initialized arc parameters from phase 1 will be optimized with modified cost function models and constraints, and will also be validated using arc approximation error and each data point's covariance matrix. If all the arc segments are acceptable after evaluation, the optimization loop ends. On the other hand, if there are some invalid arc segments, the number of arc segments is increased by one, and the optimization loop is repeated. Here, note that the number of arc segments is fixed within the arc parameter optimization process (left top block of phase 2 in Figure 7).

Focusing on cost function models, slight modifications were made to the original cost function models and equality constraint introduced in Section 2. Moreover, 2 more constraint models were added due to the properties of the multiple-arc approximation framework.

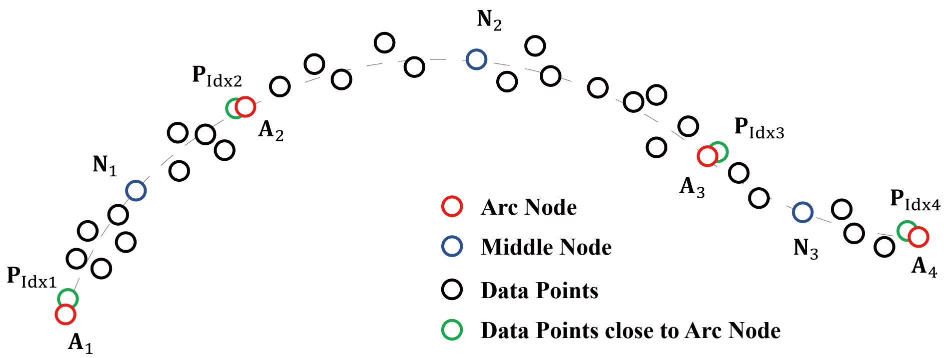

Figure 11: Data Association for 3 Arcs: (1) Find the closest data point to each arc node (marked green) (2) Data points between index Idx i to Idx $i+1$ are matched to arc segment number i. For example, points between $\mathbf{P}_{\mathrm{Idx1}}$ and $\mathbf{P}_{\mathrm{Idx2}}$ are matched to the first arc segment. (3) After each optimization iteration, the optimization variables $\mathbf{A}_1$, $\mathbf{A}_2$, and so forth, are updated. Subsequently, the association process from step (1) to step (2) is reiterated.

3.3.1. Data Association

Before moving on to the cost function and constraint model explanation, we first handle data association, which is the process of matching data points and arc segments. Unlike single-arc approximation, where all the data points are matched to one arc, the matching relationship between data points and multiple arc segments may become ambiguous during the multi-arc approximation. Therefore, for a particular arc segment, we need to decide which data points are going to be matched to the arc segment in the data association step. For example in Figure 11, since there are 3 arc segments, we need to divide data points into 3 groups during data association. Assuming that the data points are well-sorted, fast data association can be performed by using the index of data points (will be written as **Idx**) that are closest to arc nodes. In the case of starting ($\mathbf{A}_1$) and ending ($\mathbf{A}_4$) arc nodes, the first and the last data points will be used (Index number 1 and n) respectively. As a result, as shown in Figure 11, data points that have indices between **Idx(1)** (= 1) and **Idx(2)** are linked to the first arc segment. This is the same for the remaining two arc segments. Other than the example addressed previously, the same logic can be applied to various cases with different numbers of arc segments.

The result of data association will be used in the extended cost function and constraint models. Note that data association matching results may critically affect the approximation performance, especially in the arc measurement model. Approximation using wrong data points will lead to convergence failure, instability, and large approximation errors.

Remarks on Data Association

In phase 2 parameter optimization step, the positions of multiple arc nodes and middle nodes are changed for each optimization iteration. Since data association is done based on the positions of arc nodes, it is necessary to perform data association after every optimization iteration.

3.3.2. Anchor Model

The structure of the anchor model for multiple-arc approximation mirrors that of single-arc approximation: the positional difference between an arc node and the matched data point is weighted by a covariance matrix. The first two terms in Equation 9 fix the first and last arc nodes to the first and last data points, respectively, while the remaining terms anchor the remaining arc nodes to their corresponding data points found via **Data Association**. To ensure stability, small covariance values Σ_{AC_1} constrain the initial and final arc nodes, while larger covariance values Σ_{AC_2} are assigned to the remaining arc nodes to accommodate potential variations during optimization, allowing them to explore the solution space more freely. With m arc segments yielding $m+1$ arc nodes, the anchor model cost function is expressed as:

$$\mathscr{L}_{\mathrm{AC}} = \|\mathbf{P}_{\mathrm{Idx}(1)} - \mathbf{A}_1\|^2_{\Sigma_{\mathrm{AC}_1}} + \|\mathbf{P}_{\mathrm{Idx}(m+1)} - \mathbf{A}_{m+1}\|^2_{\Sigma_{\mathrm{AC}_1}} + \sum_{i=2}^{m} \|\mathbf{P}_{\mathrm{Idx}(i)} - \mathbf{A}_i\|^2_{\Sigma_{\mathrm{AC}_2}} \tag{9}$$

In Equation 9, $\mathbf{P}$ denotes the data point vector and $\mathbf{Idx}(i)$ represents the data point index obtained from **Data Association**. Specifically, $\mathbf{Idx}(1) = 1$ and $\mathbf{Idx}(m+1) = n$, where n is the total number of data points.

3.3.3. Arc Measurement Model

For the arc measurement model in multiple-arc approximation, equation 3 derived in the single-arc approximation is repeatedly computed for multiple arcs. In general, for arc segment i, the arc measurement model cost is computed for data points of index $\mathbf{Idx}(i)$ to $\mathbf{Idx}(i+1)$, using arc nodes $\mathbf{A}_i, \mathbf{A}_{i+1}$, and middle node $\mathbf{N}_i$. The arc measurement model cost for multiple arcs is written below.

$$\mathscr{L}_{\mathrm{ME}} = \sum_{i=1}^{m} \sum_{j=\mathrm{Idx(i)}}^{\mathrm{Idx(i+1)}} \|\mathbf{r}_{\mathrm{ME}}\left(\mathbf{A}_i, \mathbf{A}_{i+1}, \mathbf{N}_i, \mathbf{P}_j\right)\|^2_{\Sigma^j_{\mathrm{ME}}} \tag{10}$$

In equation 10, note that the covariance matrix Σ_{ME}^{j} is uniquely defined for each data point (pre-computed).

3.3.4. Equality Constraint 1: Middle Node

Expanding the equality constraint introduced in single-arc approximation (2.2.3) to accommodate multiple arc segments, we compute Equation 4 for each segment in the multiple-arc framework. Specifically, for segment i, involving arc nodes $\mathbf{A}_i, \mathbf{A}_{i+1}$, and middle node $\mathbf{N}_i$, the middle node equality constraint is formulated as follows:

$$\mathbf{r}_{\text{Eq1}}(i) = (\mathbf{A}_{i+1} - \mathbf{A}_i)^\top \left(\mathbf{N}_i - \frac{1}{2}(\mathbf{A}_i + \mathbf{A}_{i+1})\right), \quad \text{for } i = 1 : m \tag{11}$$

Here, $\mathbf{r}_{\text{Eq1}}$ of size $\mathbb{R}^m$ is constrained as a zero vector during parameter optimization.

3.3.5. Equality Constraint 2: G^1 Continuity

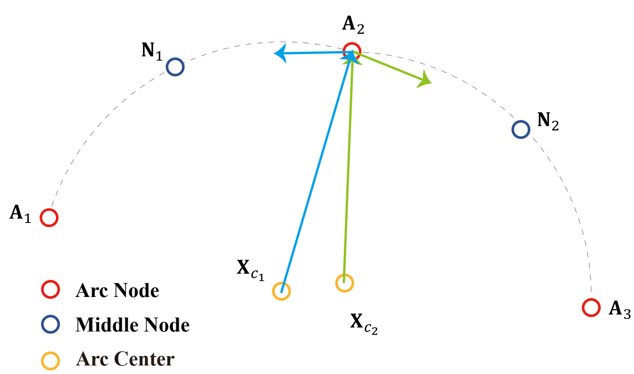

Figure 12: Equality Constraint for G^1 Continuity

When there are multiple arc segments, ensuring G^1 continuity between each adjacent pair of segments is necessary. Figure 12 illustrates that the two adjacent arc segments are not G^1 continuous. To achieve G^1 continuity between these segments, the following conditions must be met.

- (Orthogonality between Blue Vectors)
  Tangential vector of arc segment 2 at $\mathbf{A}_2$ is orthogonal to vector $\overrightarrow{\mathbf{X}_{c_1}\mathbf{A}_2}$
- (Orthogonality between Green Vectors)
  Tangential vector of arc segment 1 at $\mathbf{A}_2$ is orthogonal to vector $\overrightarrow{\mathbf{X}_{c_2}\mathbf{A}_2}$

However, since the satisfaction of either of the two conditions automatically leads to the satisfaction of the remaining condition, we opted for ensuring G^1 continuity by enforcing orthogonality between the blue vectors. Extending the case introduced in Figure 12, if we consider the G^1 continuity constraint between ith and $(i+1)$th arc segment, arc nodes $\mathbf{A}_i, \mathbf{A}_{i+1}$, and $\mathbf{A}_{i+2}$ correspond to $\mathbf{A}_1, \mathbf{A}_2$, and $\mathbf{A}_3$ in Figure 12 respectively. Then, the condition above can be expressed as follows.

$$\begin{aligned}
\mathbf{v}_{b_1} &= \left[(\mathbf{A}_{i+1})_y - \left(\mathbf{X}_{c_i}\right)_y ;\ \left(\mathbf{X}_{c_i}\right)_x - (\mathbf{A}_{i+1})_x \right] \\
\mathbf{v}_{b_2} &= \left[(\mathbf{A}_{i+1})_x - \left(\mathbf{X}_{c_{i+1}}\right)_x ;\ (\mathbf{A}_{i+1})_y - \left(\mathbf{X}_{c_{i+1}}\right)_y \right] \\
\mathbf{r}_{\text{Eq2}}(i) &= {\mathbf{v}_{b_1}}^\top \mathbf{v}_{b_2} \text{, for } i = 1 : m-1
\end{aligned} \tag{12}$$

In the equation, $(\mathbf{v})_x$ and $(\mathbf{v})_y$ denote the x and y components of vector $\mathbf{v}$ respectively. The inner product ${\mathbf{v}_{b_1}}^\top \mathbf{v}_{b_2}$ represents orthogonality between blue vectors. For m arc segments, the equality constraint residual vector $\mathbf{r}_{\text{Eq2}}$ is constrained to be a zero vector during optimization, with a size of $\mathbb{R}^{m-1}$.

3.3.6. Inequality Constraint 1: Minimum Arc Length

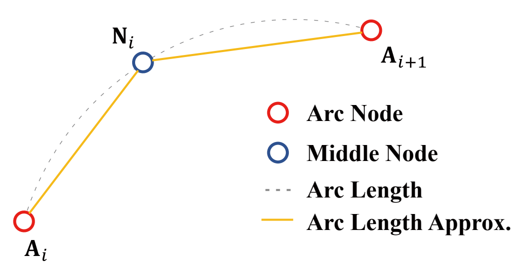

Figure 13: Inequality Constraint for Minimum Arc Length: The true arc length is approximated with the length of orange lines. $\overline{\mathbf{A}_i\mathbf{N}_i} + \overline{\mathbf{N}_i\mathbf{A}_{i+1}} \geq \text{L}_{\text{min}}$ is set as the inequality constraint.

The final constraint model enforces the arc segments to have a minimum length of L_{min}. The model aims to prevent arc segments from collapsing to a single point(i.e. 2 arc nodes and the middle node converging to the same point), which causes singularity problems during the optimization process.

For example in Figure 13, if we compute the true arc length of segment i using $\mathbf{A}_i, \mathbf{A}_{i+1}$, and $\mathbf{N}_i$, the value would be severely nonlinear. Setting the true arc length to be larger than L_{min} would cause the CNLS solver to slow down, or even fail in extreme cases. A way around this problem is to find some simple approximation of the arc length. One method is to approximate the arc length by summing the lengths of $\overline{\mathbf{A}_i\mathbf{N}_i}$ and $\overline{\mathbf{N}_i\mathbf{A}_{i+1}}$, as shown in Figure 13. Note that the true arc length is always greater than the sum of the length of two line segments $\overline{\mathbf{A}_i\mathbf{N}_i}$ and $\overline{\mathbf{N}_i\mathbf{A}_{i+1}}$ geometrically. Therefore if we set $\|\mathbf{A}_i - \mathbf{N}_i\| + \|\mathbf{N}_i - \mathbf{A}_{i+1}\|$ to be larger than $\mathrm{L_{min}}$, the true arc length will be constrained to have larger value than $\mathrm{L_{min}}$. The inequality constraint residual vector of size $\mathbb{R}^m$ is written as follows.

$$\mathbf{r}_{\mathrm{InEq1}}(i) = 1 - \frac{\|\mathbf{A}_i - \mathbf{N}_i\| + \|\mathbf{N}_i - \mathbf{A}_{i+1}\|}{\mathrm{L_{min}}}, \text{ for } i = 1:m \tag{13}$$

The inequality above is computed for all the arc segments (from $i = 1$ to m). When performing optimization, the inequality constraint residual vector $\mathbf{r}_{\mathrm{InEq1}}$ is constrained to be less than or equal to a 0 vector.

3.3.7. Multiple Arc Approximation: Augmented Cost Function and Constraints

Merging the cost function models and equality/inequality constraint models, we obtain the full CNLS problem structure. Assuming we have m arc segments, the augmented cost function can be written as follows.

$$\begin{aligned}
\min_{\mathbf{A}_1,\cdots\mathbf{A}_{m+1},\mathbf{N}_1,\cdots\mathbf{N}_m} \mathscr{L} &= \mathscr{L}_{\mathrm{AC}} + \mathscr{L}_{\mathrm{ME}} \\
&= \|\mathbf{P}_1 - \mathbf{A}_1\|^2_{\Sigma_{\mathrm{AC}_1}} + \|\mathbf{P}_n - \mathbf{A}_{m+1}\|^2_{\Sigma_{\mathrm{AC}_1}} \\
&+ \sum_{i=2}^{m} \|\mathbf{P}_{\mathrm{Idx}(i)} - \mathbf{A}_i\|^2_{\Sigma_{\mathrm{AC}_2}} \\
&+ \sum_{i=1}^{m} \sum_{j=\mathrm{Idx(i)}}^{\mathrm{Idx(i+1)}} \|\mathbf{r}_{\mathrm{ME}}\left(\mathbf{A}_i, \mathbf{A}_{i+1}, \mathbf{N}_i, \mathbf{P}_j\right)\|^2_{\Sigma^j_{\mathrm{ME}}} \\
\text{s.t.} \quad & \mathbf{r}_{\mathrm{Eq1}} = \mathbf{0},\ \mathbf{r}_{\mathrm{Eq2}} = \mathbf{0},\ \mathbf{r}_{\mathrm{InEq1}} \leq \mathbf{0}
\end{aligned} \tag{14}$$

Similar to single-arc optimization, the cost function balancing can be done by tuning covariance matrix $\Sigma_{\mathrm{AC}_1}, \Sigma_{\mathrm{AC}_2}$. Having initialized arc nodes and middle nodes (computed in 3.2) as the input to the CNLS solver, the output will be the optimized positions of arc nodes and middle nodes. The CNLS problem given as equation 14 is solved by using the interior point method [21] implemented in 'lsqnonlin.m' of MATLAB optimization toolbox. The mathematical procedure for attaining the optimal solution mirrors the explanation provided in section 2.3.

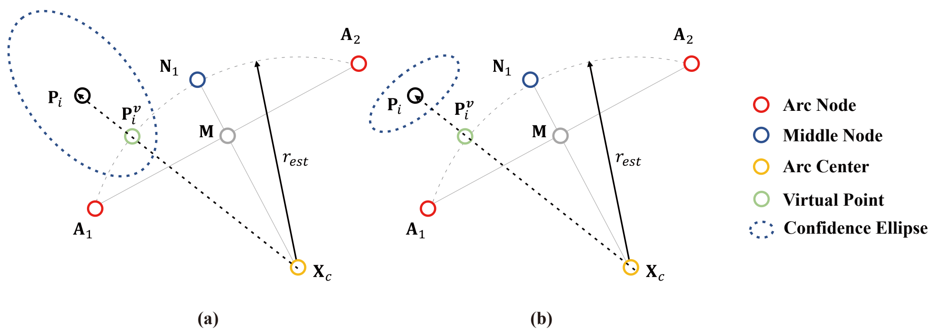

Figure 14: Determining the validity of arc approximation: Assuming that we are performing analysis on arc segment no. 1 and data point $\mathbf{P}_i$, we check if the virtual point $\mathbf{P}_i^v$ (marked green) is inside the confidence ellipse. This validation is done for all the data points and their matched arc segments. Figures (a) Inside: Valid Approximation (b) Outside: Invalid Approximation

3.4. Multiple Arc Approximation Phase 2: ***Parameter Validation***

While many curve-fitting/approximation algorithms use simple RMSE for evaluating approximations, naively using the RMSE value is an inappropriate approach if covariance matrices of data points are given. In our research, instead of RMSE, **Chi-squared**(χ^2) **test**[22] is conducted for all data points to determine whether the arc approximation of each arc segment is acceptable or not. The validation steps for an arc segment are:

Step 1. For an arc segment, obtain the matched data point indices from the data association process.

Step 2. Compute arc measurement residual (section 2.2.2) for each data point. For arc segment i and point index j, $\mathbf{r}_{\mathrm{ME}}\left(\mathbf{A}_i, \mathbf{A}_{i+1}, \mathbf{N}_i, \mathbf{P}_j\right)$ is computed.

Step 3. Compute the squared Mahalanobis distance using the residual from Step 2 and test whether this value is larger than the Chi-squared test threshold (Larger/Smaller).

Step 4. (a) (Larger) Arc approximation is invalid for the data point $\mathbf{P}_j$.
(b) (Smaller) Arc approximation is valid for the data point $\mathbf{P}_j$.

Step 5. If the total number of invalid arc approximations exceeds the threshold N, the corresponding arc segment is considered invalid.

Typically, 99% confidence level is chosen for the Chi-squared test threshold value. This means that the arc approximation is considered invalid, only if it is placed outside of the 99% confidence ellipse(drawn using covariance matrix), as shown in Figure 14.

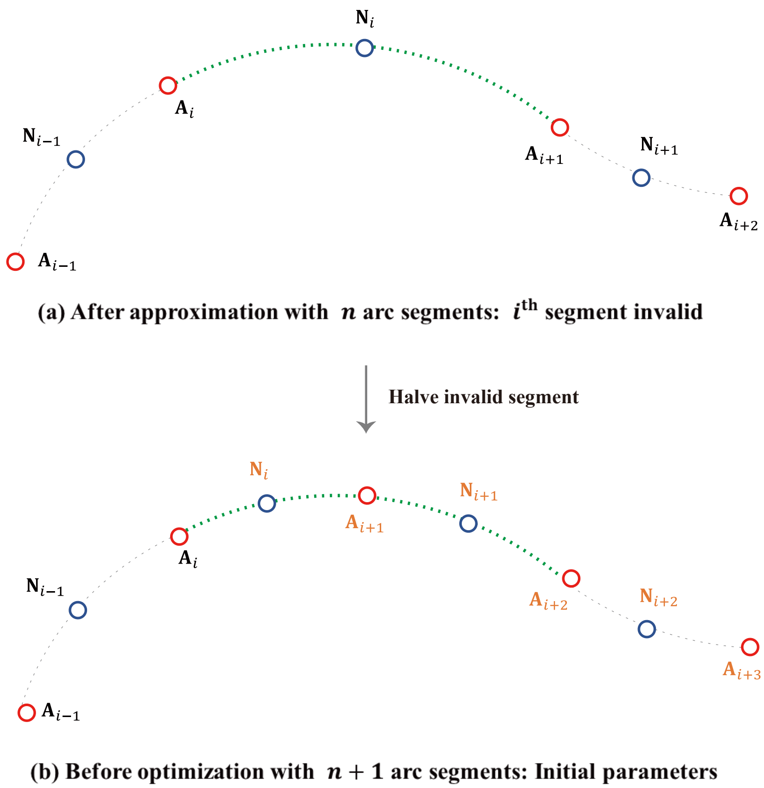

Figure 15: Parameter update process: (a) After optimization with n arc segments, i^{th} segment is the most invalid segment(green dash). (b) Halve the i^{th} arc segment to create the initial parameter values for optimization with $n+1$ arc segments.

Ultimately, in order to assess the validity of the approximated arc segment, we count the number of invalid data point approximations for each arc segment (Step 5). If the total number of invalid data point approximations surpasses the threshold N, a tuning variable, the present arc segment is deemed invalid. It is crucial to note that the value of N requires careful tuning. If N is set too high, arc segments with substantial approximation errors might be accepted. Conversely, if N is set too low, a greater number of arc segments may be necessary to accurately approximate data points.

3.5. *Multiple Arc Approximation Phase 2: **Parameter Update***

The parameter update step is performed if there exists any invalid arc segment after the **Parameter Validation** (3.4) step. We increase the number of arc segments by halving the arc segment with the most number of invalid data point approximations, as shown in Figure 15. Original arc nodes and the middle node are used to generate the arc nodes and the middle node of the newly generated arc segment without changing the cost function value. Starting from the updated initial parameters in Figure 15-(b), optimization is then performed for $n+1$ arc segments.

It is important to note that due to the low anchor covariance values, the initially halved two arc segments will be adjusted appropriately during the optimization steps. Therefore simply halving the invalid segment will not cause stability problems.

3.6. Simple Proof on Iterative Convergence of the Proposed Framework

Based on the parameter update process, we can show that our framework ensures that the arc parameters will iteratively converge to a local minimum. Let us denote $\mathbf{X}_n^f = [\mathbf{A}_1, \cdots, \mathbf{A}_{n+1}, \mathbf{N}_1, \cdots, \mathbf{N}_n]$ as the arc parameters **after optimization with** n **arc segments** and $\mathbf{X}_{n+1}^o = \left[\tilde{\mathbf{A}}_1, \cdots, \tilde{\mathbf{A}}_{n+2}, \tilde{\mathbf{N}}_1, \cdots, \tilde{\mathbf{N}}_{n+1}\right]$ as the arc parameters **before optimization with** $n+1$ **arc segments**. Then the parameters in Figure 15-(a) are $\mathbf{X}_n^f$ and the parameters in Figure 15-(b) are $\mathbf{X}_{n+1}^o$.

Even though $\mathbf{X}_{n+1}^o$ includes an additional arc segment, the augmented cost function $\mathscr{L}\left(\mathbf{X}_{n+1}^o\right)$ remains identical to $\mathscr{L}\left(\mathbf{X}_n^f\right)$ due to the consistent geometry, as shown in Figure 15. If further cost reduction is possible after increasing the number of arc segments, the CNLS solver (Section 3.3.7) adjusts node positions, gradually approaching the local minima. Notably, the CNLS solver only accepts updates that reduce the cost function $\mathscr{L}$.

The relationship of the cost function values across iterations is as follows:

$$\cdots \mathscr{L}\left(\mathbf{X}_n^o\right) > \mathscr{L}\left(\mathbf{X}_n^f\right) = \mathscr{L}\left(\mathbf{X}_{n+1}^o\right) > \mathscr{L}\left(\mathbf{X}_{n+1}^f\right) = \cdots \tag{15}$$

Since the cost function value monotonically decreases with increasing number of arc segments, we have proved the iterative convergence of our multiple-arc approximation framework.

4. Numerical Examples and Comparison

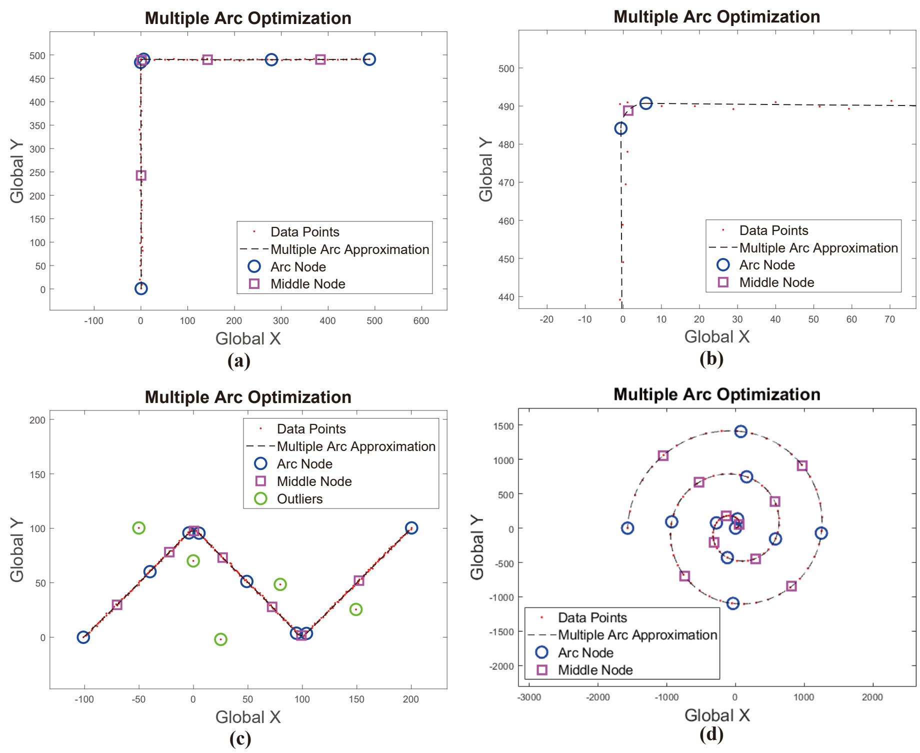

Figure 16: Multiple Arc Approximation of various data: (a) Two perpendicular lines are approximated using three arcs. (b) The sharp corner in (a) is approximated by employing an arc segment with a small radius. (c) Zig-zag lines(with outliers) and (d) spiral curve are approximated with 7 and 10 arc segments, respectively.

Integrating four blocks—parameter initialization (Chapter 3.2), optimization (Chapter 3.3), validation (Chapter 3.4), and update (Chapter 3.5)—in the multiple-arc framework (Figure 7), we evaluate its performance using two types of datasets. We first evaluate with several noisy, generated data point sets. Subsequently, the framework is tested on real-world lane data from a test drive in Sejong City [20].

4.1. Multiple Arc Approximation of Generated Noisy Data Points

We initiate our framework evaluation by using various generated data points, including smooth ones, alongside datasets containing outliers or sharp geometries, as depicted in Figure 16. Remarkably, we observe effective management

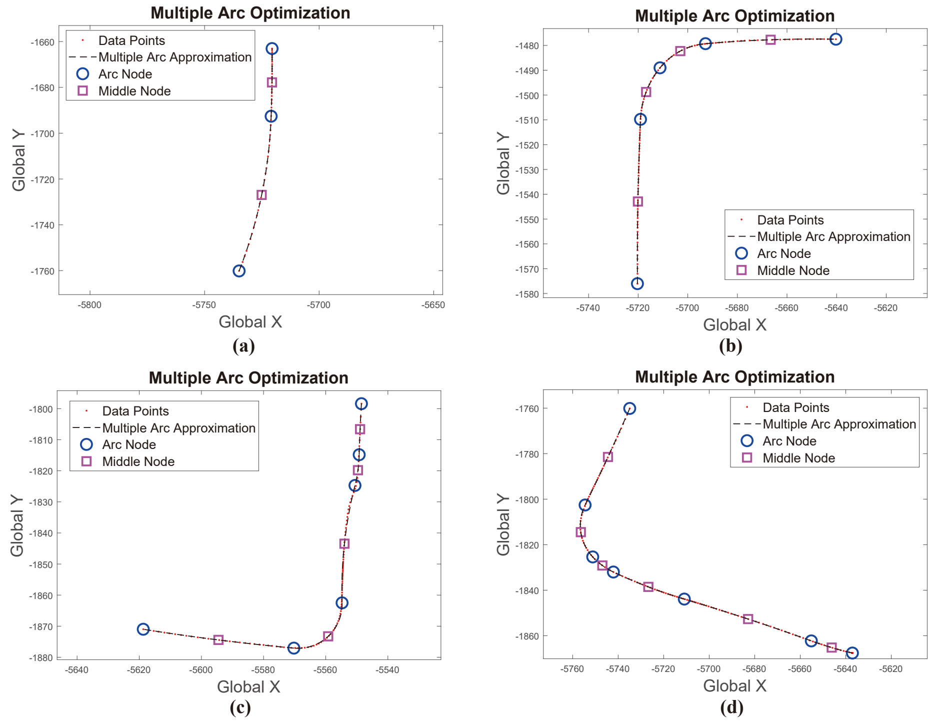

Figure 17: Multiple Arc Optimization Example 1 (Same data points as Figure 6)

of sharp corners using arc segments with small radii. Moreover, outlier points of low reliability are disregarded without compromising the approximation performance.

4.2. Multiple Arc Approximation of Real-World Collected Data Points

The multiple-arc optimization framework is applied to the same examples as in the single-arc optimization. Figure 17 demonstrates the optimization process for all data points using two or more valid arc segments. Notably, achieving a lower Root Mean Square Error (RMSE) doesn't always indicate a superior approximation, as it may accompany a higher number of invalid Chi-squared test samples. Employing pre-computed covariance matrices for all data points enables evaluation based on Chi-squared tests, which is considered more reasonable than solely relying on the RMSE for each arc segment.

4.3. Multiple Arc Approximation Application: Lane Map Parameterization

Finally, we introduce the multiple-arc approximation results of the left and right ego lanes of a vehicle trip in Sejong City, South Korea. The real-world

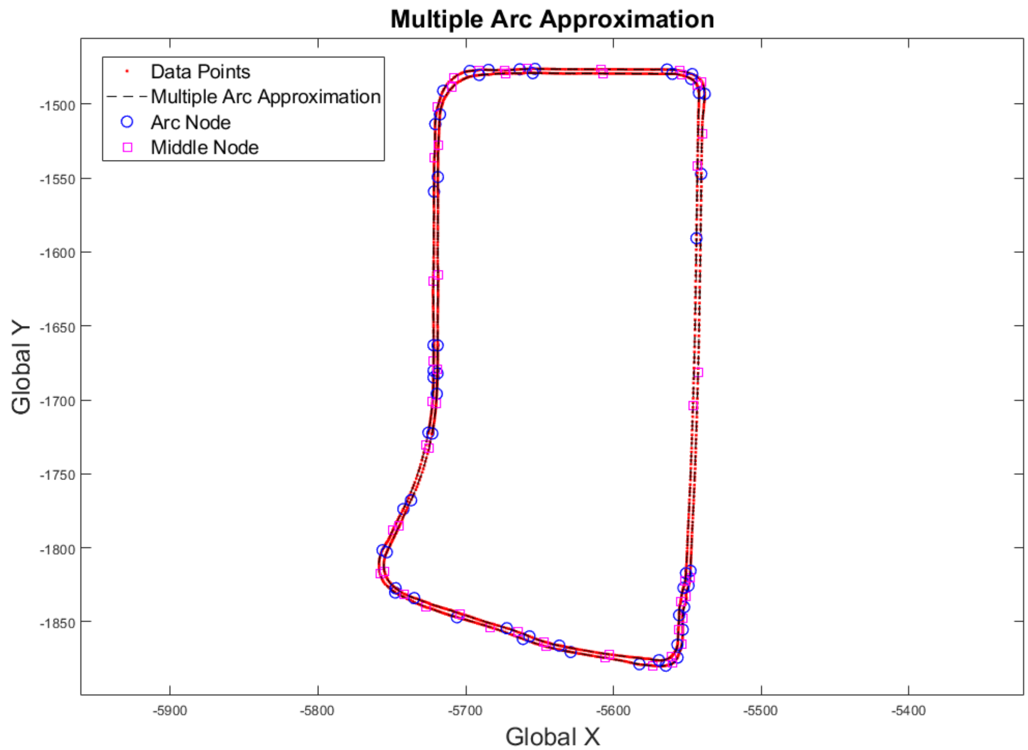

Figure 18: Multiple Arc Optimization Example 2 (Full data points from vehicle trip)

examples introduced at single-arc optimization and multiple-arc optimization previously in Figures 6 and 17 were partially sampled from this whole trip. In Figure 18, the left/right lane data points and arc nodes were drawn together, but optimized separately.

Direction	Total Arc Segment Length	Total Number of Segments
Left	1185.10 m	27
Right	1163.73 m	23

Table 1: Summary of Left and Right Ego Lane Arc Parameterization

A total of 1152 data points were parameterized into several arc segments for each left/right ego lane. A summary of multiple-arc approximation results is listed in table 1. If we analyze the results, 1152 data points from the left ego lane can be simply represented with 55 control points(i.e. 27 middle nodes + 28 arc nodes = 55 control points) and the right ego lane can be represented with

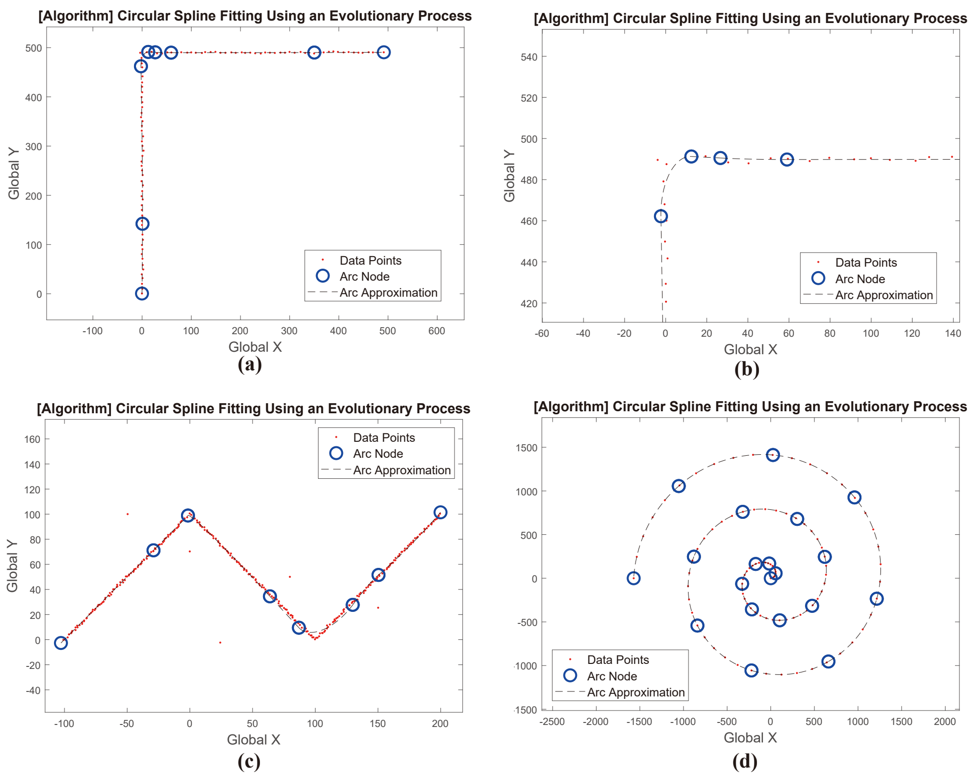

Figure 19: Multiple Arc Approximation of data points in Figure 16 by algorithm [15]

47 control points while obeying the reliability conditions(Section 3.4).

4.4. Comparison

For a meaningful comparison among methods capable of handling both noisy and outlier data points, we have selected the approximation algorithm proposed in [15]. As our primary focus in this paper is robust arc spline approximation, it is imperative that the data points used for evaluation contain noise and even outliers. Consequently, to the best of our knowledge, most arc spline approximation algorithms would fail under such conditions, underscoring the suitability of reference [15] for comparison with our framework in terms of algorithm robustness.

In Figure 19, the data point sets from Figure 16 serve as the basis for evaluating the comparison algorithm proposed by Song et al. [15]. For the sharp geometry case (a), our proposed framework utilizes 4 arc segments compared to Song et al.'s 7 arcs. In the case of zig-zag lines (c), both methods employ 7 arc segments, yet our approach yields a smaller error. Similarly, for the spiral geometry (d), our method employs 10 arc segments while Song et al. used 19 arcs. Notably, our algorithm demonstrates superior performance by reducing the number of arcs required to approximate the same data points. Additionally, Song et al.'s algorithm exhibits instability when the density of data points varies within the dataset. Given that raw data points are directly sampled for initial arc generation, datasets with significant noise or outliers pose substantial challenges for the comparison algorithm [15].

4.5. Analysis

Before concluding our research, we analyze the advantages and the possible limitations of our reliability-based arc spline approximation framework.

Advantages

- Robust to noisy data points
- Compact data approximation by multiple arcs

Limitations

- Data points should be well-ordered (sorted)
- Covariance for data points should be accurate
- Optimized arc parameters may be sub-optimal solutions

Note that the first and second drawbacks of the proposed framework can be mitigated by other pre-processing algorithms. For example, sorting unorganized points can be done by applying the 'moving least squares method' introduced in [4].

5. Conclusion

In this study, we propose novel optimization frameworks for single and multiple arc approximations. Departing from traditional methods focused on minimizing RMSE, our approach aims at determining statistically optimal arc parameters using data points and their covariance matrices. Evaluation across various datasets validates the effectiveness of our approach.

As demonstrated in Section 4.3, a possible application of our multiple-arc approximation framework is vehicle lane mapping. Considering that existing digital maps represent lane data using points and line segments, we anticipate notable improvements in data storage and management. Moreover, our reliability-based approach facilitates updating lane segment information following data collection from overlapping trips, presenting a distinct advantage over conventional arc spline methods. Therefore, future research will involve implementing and evaluating multiple-arc approximation across wider regions of Sejong City.

Acknowledgement

This research was supported by the BK21 FOUR Program of the National Research Foundation Korea(NRF) grant funded by the Ministry of Education(MOE); Autonomous Driving Technology Development Innovation Program (20018181, Development of Lv. 4+ autonomous driving vehicle platform based on point-to-point driving to logistic center for heavy trucks) funded by the Ministry of Trade, Industry & Energy(MOTIE, Korea) and Korea Evaluation Institute of Industrial Technology(KEIT); Technology Development for Future Automotives Tuning Parts (P0021036, Development of a smart damper system for electric vehicles capable of tuning damping force while driving) funded by the Ministry of Trade, Industry & Energy (MOTIE, Korea) and the Korea Institute for Advancement of Technology (KIAT); the Technology Innovation Program(20014983, Development of autonomous chassis platform for a modular vehicle)funded By the Ministry of Trade, Industry & Energy(MOTIE, Korea);

References

[1] D. Meek, D. Walton, Approximating smooth planar curves by arc splines, Journal of Computational and Applied Mathematics 59 (2) (1995) 221–231. doi:https://doi.org/10.1016/0377-0427(94)00029-Z

[2] S. Jee, T. Koo, Tool-path generation for nurbs surface machining, in: Proceedings of the 2003 American Control Conference, Vol. 3, 2003, pp. 2614-2619 vol.3. doi:10.1109/ACC.2003.1243471.

[3] D. J. Kim, V. T. Nguyen, Reduction of high frequency excitations in a cam profile by using modified smoothing

spline curves, Int J Automot Technol 8 (1) (2007) 59–66. URL http://www.ijat.net/journal/view.php?number=396

[4] I.-K. Lee, Curve reconstruction from unorganized points, Computer Aided Geometric Design 17 (2) (2000) 161–177. doi:https://doi.org/10.1016/S0167-8396(99)00044-8.

[5] A. Schindler, G. Maier, S. Pangerl, Exploiting arc splines for digital maps, in: 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), 2011, pp. 1–6. doi:10.1109/ITSC.2011.6082800.

[6] A. Schindler, G. Maier, F. Janda, Generation of high precision digital maps using circular arc splines, in: 2012 IEEE Intelligent Vehicles Symposium, 2012, pp. 246–251. doi:10.1109/IVS.2012.6232124.

[7] G. Maier, Optimal arc spline approximation, Computer Aided Geometric Design 31 (5) (2014) 211–226. doi:https://doi.org/10.1016/j.cagd.2014.02.011.

[8] R. Klass, An offset spline approximation for plane cubic splines, Computer-Aided Design 15 (5) (1983) 297–299. doi:https://doi.org/10.1016/0010-4485(83)90019-2.

[9] L. Piegl, W. Tiller, Curve and surface constructions using rational b-splines, Computer-Aided Design 19 (9) (1987) 485–498. doi:https://doi.org/10.1016/0010-4485(87)90234-X.

[10] J. Hoschek, Circular splines, Computer-Aided Design 24 (11) (1992) 611–618. doi:https://doi.org/10.1016/0010-4485(92)90072-I

[11] D. Meek, D. Walton, Approximation of discrete data by G1 arc splines, Computer-Aided Design 24 (6) (1992) 301–306. doi:https://doi.org/10.1016/0010-4485(92)90047-E

[12] S.-N. Yang, W.-C. Du, Numerical methods for approximating digitized curves by piecewise circular arcs, Journal of Computational and Applied Mathematics 66 (1) (1996) 557–569, proceedings of the Sixth International Congress on Computational and Applied Mathematics. doi:https://doi.org/10.1016/0377-0427(95)00191-3

[13] L. Piegl, W. Tiller, Data approximation using biarcs, Engineering With Computers 18 (2002) 59–65. doi:10.1007/s003660200005.

[14] M. Heimlich, M. Held, Biarc approximation, simplification and smoothing of polygonal curves by means of voronoi-based tolerance bands, International Journal of Computational Geometry & Applications 18 (03) (2008) 221–250. doi:https://doi.org/10.1142/S0218195908002593.

[15] X. Song, M. Aigner, F. Chen, B. Jüttler, Circular spline fitting using an evolution process, Journal of Computational and Applied Mathematics 231 (1) (2009) 423–433. doi:https://doi.org/10.1016/j.cam.2009.03.002.

[16] M. A. Fischler, R. C. Bolles, Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, Commun. ACM 24 (6) (1981) 381–395. doi:10.1145/358669.358692

[17] R. Drysdale, G. Rote, A. Sturm, Approximation of an open polygonal curve with a minimum number of circular arcs and biarcs, Comput. Geom. 41 (2008) 31–4

[18] K. Madsen, H. Nielsen, O. Tingleff, Methods for Non-Linear Least Squares Problems (2nd ed.), Technical University of Denmark, 2004.

[19] The MathWorks Inc., lsqnonlin : Nonlinear Least Squares Solver (2023). URL https://kr.mathworks.com/help/optim/ug/lsqnonlin.html

[20] J. Jeon, Robust vehicle trajectory reconstruction and parameterized lane map generation via multi-modal sensor fusion, Master's thesis, Korea Advanced Institute of Science and Technology(KAIST), Daejeon, South Korea (February 2023).

[21] T. F. Coleman, Y. Li, An interior trust region approach for nonlinear minimization subject to bounds, SIAM Journal on Optimization 6 (2) (1996) 418–445. doi:10.1137/0806023.

[22] K. Pearson, On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 50 (302) (1900) 157–175. doi:https://doi.org/10.1080/147864400094