RESEARCH ARTICLE

Filling gaps of cartographic polylines by using an encoder–decoder model

Wenhao Yu\textsuperscript{a,b} and Yujie Chen\textsuperscript{a}

\textsuperscript{a}School of Geography and Information Engineering, China University of Geosciences, Wuhan, China; \textsuperscript{b}National Engineering Research Center for Geographic Information System, China University of Geosciences, Wuhan, China

ABSTRACT

Geospatial studies must address spatial data quality, especially in data-driven research. An essential concern is how to fill spatial data gaps (missing data), such as for cartographic polylines. Recent advances in deep learning have shown promise in filling holes in images with semantically plausible and context-aware details. In this paper, we propose an effective framework for vector-structured polyline completion using a generative model. The model is trained to generate the contents of missing polylines of different sizes and shapes conditioned on the contexts. Specifically, the generator can compute the content of the entire polyline sample globally and produce a plausible prediction for local gaps. The proposed model was applied to contour data for validation. The experiments generated gaps of random sizes at random locations along with the polyline samples. Qualitative and quantitative evaluations show that our model can fill missing points with high perceptual quality and adaptively handle a range of gaps. In addition to the simulation experiment, two case studies with map vectorization and trajectory filling illustrate the application prospects of our model.

ARTICLE HISTORY

Received 28 June 2021
Accepted 13 March 2022

KEYWORDS

Filling gap; cartographic polylines; spatial data quality; encoder–decoder model

1. Introduction

The quality of spatial data is a major concern in geographic information science. Recently, volunteered geographic information (VGI) data have fundamentally enriched the information content of geospatial research and raised concerns about data quality (Basiri et al. 2019). Bauer (2013) compared different running programs and showed how GPS in smartphones generates low-quality traces. Mooney et al. (2016) proposed that problems such as non-professional volunteers and the absence of protocols can lead to low data quality and missing data.

Missing data is a common problem in spatial data management which directly affects the reliability of map products (Hong and Huang 2017). Missing data or data gaps may occur during many cartographic processes, including data collection, data management and data merging. For example, the instability of measuring instruments often results in an unexpected failure to record data, for example outliers in Global
Navigation Satellite System (GNSS) trajectories. Ivanovic et al. (2019) argued that both outliers and secondary trajectory features should be filtered to highlight the main trajectories for movement analysis. After the filtering operation, the polylines of the trajectory data can have gaps (Figure 1(a)). In addition, historical maps are stored in paper form, which may be stained or damaged during transportation and management. Annotations and other map features can also cover the polylines of interest (Figure 1(b)). Another situation occurs in the edge-matching process of adjacent sheets, during which graphic conflicts such as feature disconnection may occur (Ge et al. 2019, Hui and Tong 2000, Zhang et al. 2001). Notably, even a complete polyline on a printed map can be damaged during digitization. Nevertheless, not all lines with

![Figure 1](image_url)

*Figure 1.* Cases of missing data for polylines. Missing data may occur during the extraction of the main traces (a). In the printed historical maps (b), missing data may occur when the polylines were overlaid by annotations and other features or when the paper is damaged.
missing parts need to be filled, and some dotted/dashed lines are designed to improve the quality of the map, such as supplementary contour lines (Samsonov et al. 2019). Only those lines with gaps that cause a loss of data quality need to be filled.

Filling the missing data on polylines is a challenge because the length and shape of gap portions are highly uncertain. In practical applications, there are two main solutions for filling the gaps. The first solution is to re-collect the corresponding data, for example field measurements, and extract feature information from remote sensing images. However, field measurements are time-consuming and laborious, and remote sensing images are constrained by spatial and temporal scales. In addition, the premise of these solutions is that the data are available, a condition which may not be satisfied for problems such as historical map vectorization and GNSS trajectory filling. The second solution is to predict the gap content according to the known part of the cartographic polylines, for example manual polyline drawing and function-based fitting. The polylines filled by cartographers are reliable but influence the time efficiency. The function-based approach (e.g. Bezier curve fitting) is efficient, but it also has limited performance with respect to complex polyline shapes. Therefore, quickly and adaptively generating reasonable content to fill the missing regions on polylines remains a challenge in practical applications.

A similar problem with gaps has been widely studied in the domain of computer vision. Researchers have focused on filling holes in images, a task known as image inpainting or image completion. Many image-inpainting models have been proposed using neural network structures, for example using the encoder–decoder structure and generative adversarial network (GAN) structure (Pathak et al. 2016, Iizuka et al. 2017, Li et al. 2017, Yang et al. 2017). The research suggested that neural network models can learn semantic priors and adaptively repair missing content according to the known content. Rather than waiting for human experts to individually repair the images of interest, neural networks can automatically fill the images in batches.

The successful performance of neural network models in the image-inpainting problem demonstrates their ability to mine the potential patterns of input samples. Despite their diversity, images are highly structured and have regular patterns, whether they represent natural scenes or specific objects. The strong context relationships between the known regions and the missing regions of images make it possible to predict the latter from the former. The cartographic polylines also have a strong context, for example, road networks have straight segments and vertical intersections, and the rivers and contour lines have smooth polylines with continuous changes in curvature. In fact, neural network models have been increasingly used to process geospatial data from a data-driven perspective, as they can successfully extract underlying patterns given complex spatial contexts (Janowicz et al. 2020, Li 2020, Zhu et al. 2020).

Consequently, we propose a polyline-completion framework using a generator model based on an encoder–decoder structure. This involves novel research challenges, such as the creation of an appropriate training set from vector datasets. We compare our method with the function-based method (i.e. Bezier curve fitting) because both methods can generate results automatically. Both qualitative and quantitative experiments are used to demonstrate the advantages of our model.

The main contributions of this work can be summarized as:
1. Unlike function-based methods that choose different functions to fit polylines of different shapes, our model can adapt to various polylines and produce visually realistic predictions that incorporate structural characteristics with the known portion of the original polylines.

2. Rather than filling gaps with fixed sizes and locations, our model combined with the decision tree generates effective contextual results for gaps of different locations and sizes on the sample, making our model applicable in complex conditions.

The remainder of this paper proceeds as follows. Related work is introduced in Section 2, while Section 3 details the proposed framework. Section 4 presents the filling capacity of our model and compares our results with the results of the Bezier curve method from both qualitative and quantitative aspects. In addition, two practical cases are presented in Section 4. Section 5 concludes the paper and discusses potential improvements to the proposed approach.

2. Related work

In the domain of spatial data quality, missing data involve the problem of completeness. Volunteered Geographic Data (VGI) data often serve as auxiliary data to fill in missing data regions, especially for road networks. For example, Mahabir et al. (2017) compared the quality of authoritative and non-authoritative road data and suggested that non-authoritative data are a good supplementary spatial data source, especially for internal road data. Minaei (2020) analyzed the spatial pattern, evolution, density and diversity of OpenStreetMap (OSM) road networks in Iran and proposed that VGI data can effectively supplement the gaps in authoritative data, especially in developing countries. However, the quality of VGI data remains a concern because of the lack of professional volunteers and protocols (Mooney et al. 2016, Basiri et al. 2019). In addition, remote sensing images contain rich feature information, and many studies have successfully extracted coastlines, rivers and roads from images (Modava and Akbarizadeh 2017, Li et al. 2019, Wang et al. 2020). However, remote sensing images are limited by spatial and temporal scales.

Choosing an appropriate function to fit polylines is also a potential method to enrich data. Martin et al. (2008) used Bezier curves to generate smooth polylines for line generalization. Zhao et al. (2012) generated smooth paths according to the start and parking locations to provide an optimized scheme for automatic parking. Zakaria et al. (2019) described a Bezier curve model based on path generation for interpolation on a road map. However, the shapes of the fitting polylines are controlled by the parameters or control points, which can limit the filling performance of the function-fitting methods. In recent years, researchers have been attempting to develop models to learn patterns in existing data and proposed a range of end-to-end generation models to predict missing data. To solve the problem of missing GPS data in human mobility analysis, Zhao et al. (2021) proposed a framework for imputing data gaps based on frequent-pattern mining and time geography. However, instead of
predicting the values of $X, Y$ coordinates of the GPS trajectory points, they only imputed plausible travel behaviors (e.g. activities and trips) based on the sizes of gaps.

Many missing data prediction models are available in the computer vision domain, and the most relevant task is image inpainting. Researchers have attempted to predict the gap content directly from the known context of images (Yu et al. 2018, Elharrouss et al. 2020, Rojas et al. 2020). For example, Pathak et al. (2016) proposed a context encoder to predict the missing regions of images based on the encoder–decoder structure. In their study, an image with a missing region was fed into the encoder to obtain low-dimensional features; next, the decoder was used to generate the missing image content from these low-dimensional features. The model was optimized by calculating the difference between the generated and true values of the missing region. Their results showed that the context encoder was effective for natural images. Yang et al. (2017) argued that the inpainting results of the context encoder were sometimes missing fine texture details, which created visible artifacts around the border of the hole. They then proposed a texture network to generate high-frequency details. In their model, the context encoder provided strong priors about the semantics and global structures, and the global results predicted by the context encoder were then fed into the texture network for local neural patch matching. Iizuka et al. (2017) proposed a novel image completion model by combining an encoder–decoder structure with GANs. They used an encoder–decoder structure as the generator to replace the original generator of the GAN model, which started directly from a noise vector. A global discriminator and a local discriminator were trained to distinguish the whole and locally generated contents as real or fake. In contrast with the neural patch-based texture network, their model can complete the missing regions naturally that do not appear in the image. Li et al. (2017) used a similar model for face image inpainting. They segmented the face components (e.g. eyes and mouths) and trained a parsing network based on semantic segmentation to make the predicted results more consistent and authentic.

Deep-learning-based inpainting has also been used for satellite images. Dong et al. (2019) used GAN to recover sea surface temperature (SST) images with cloud occlusion. In their work, the generator network generated images from random noise vectors as accurately as possible. The discriminator network generated the nearest vector representations of the corrupted images, which were then fed into the generator to generate clean SST images. Lou et al. (2018) proposed a modified GAN consisting of one generator and two discriminators. However, due to the limitations of sample data and the spatial heterogeneity, the performance of their proposed model is unsatisfactory in complex scenes.

Most of these data filling models are based on encoder–decoder structure for feature extraction and data reconstruction. For a data set with gaps, the encoder of the structure can learn high-level abstract features, and the decoder can generate data for the missing region as similar as possible to the true-value. Such a model has been successfully applied to natural and remote sensing images, and it seems that the encoder–decoder model can also provide new solutions for filling gaps of cartographic polylines in vector maps. However, the irregular shape of polylines and the high uncertainty of gap contents make the filling tasks challenging in practical applications.
3. A gap-filling model for cartographic polylines

The framework includes two main stages: polyline data processing and gap content prediction (Figure 2). Gap content prediction is the core of the model and is implemented by the encoder–decoder based generator (Section 3.1). In the model training stage (Stage 3 in Figure 2), the polylines with gaps and the corresponding unbroken polylines are input to the model in pairs, and the loss between the truth values (i.e. the unbroken polylines) and the model results are calculated to optimize the model. With the trained model, the test sample with gaps can be filled adaptively in the model application stage (Stage 4 in Figure 2). Data processing includes training sample generation (Stage 1 in Figure 2, and more details are in Section 3.2) and test sample generation (Stage 2 in Figure 2, and more details are in Section 3.3). For the training samples, polylines with equal lengths are first intercepted from the original polyline database, which is converted into sample point sets (Step a in Figure 2). Next, different binary masks are used for each sample point set to generate the training samples (Step b in Figure 2). For the test data, the features of the gaps are extracted (Step c in Figure 2) and fed into a well-trained decision tree model to predict the number of points to be filled (Step d in Figure 2). Thus, the test samples are obtained by intercepting the original test data according to the inferred length (Step e in Figure 2).

3.1. Structure of the gap-filling generator

The proposed generator is based on an encoder–decoder structure, also known as an autoencoder (AE) (Goodfellow et al. 2016). The encoder $f_{\theta_{AE}}$ transforms the input $x_{AE}$ into a short representation (code) $c_{AE}$, i.e., $c_{AE} = f_{\theta_{AE}}(x_{AE})$. The decoder $g_{\phi_{AE}}$ maps the...
code back to the output $y_{AE}$, i.e. $y_{AE} = g_{AE}(c_{AE})$. The encoder and decoder contain multiple hidden layers with the layer-wise propagation rule based on fully connected layer and nonlinear activation (Equation (1)). With the objective of reducing as much as possible the loss between the input $x_{AE}$ and the output $y_{AE}$, the AE is trained to have the ability for data reconstruction and feature compression (Equation (2)). Therefore, the AE is typically used for data compression and feature extraction.

$$H_{AE}^{i+1} = \begin{cases} x_{AE} \\ \sigma_{AE}(W_{AE}^i \cdot H_{AE}^i + b_{AE}^i) \end{cases} \quad (i = 0)$$

$$\theta_{AE} = \arg \min_{\theta_{AE}} \text{Loss} (x_{AE}, y_{AE}) \quad (i > 0) \quad (1)$$

where $\sigma_{AE}$ denotes a nonlinear activation function; $H_{AE}^i$ and $H_{AE}^{i+1}$ denote the $i$th input and $i$th output of the $i$th layer, respectively; $W_{AE}^i$ and $b_{AE}^i$ are the weight and bias for $i$th layer, respectively; $\theta_{AE} = \{W_{AE}, b_{AE}\}$ is the parameter set; Loss is the loss function (e.g. the mean squared error) penalizing $y_{AE}$ for being dissimilar from $x_{AE}$.

In addition, to avoid the extraction of useless features by the AE simply copying the original input, we can implement a de-noising autoencoder (Vincent et al. 2008, 2010, Xie et al. 2012). The input $x_{DAE}$ of the denoising autoencoder (DAE) is a corrupted version of the original data $x_{AE}$, and the output $y_{DAE}$ is obtained through the same process as AE. However, the goal of the DAE is to minimize the loss between the output $y_{DAE}$ and the original data $x_{AE}$. The goal of the general AE is to make the output as similar as possible to the input, and the goal of the DAE is to explore patterns and rules from the data with noise to restore the data to a clean state. The DAE model can be formalized as

$$H_{DAE}^{i+1} = \begin{cases} x_{DAE} \\ \sigma_{DAE}(W_{DAE}^i \cdot H_{DAE}^i + b_{DAE}^i) \end{cases} \quad (i = 0)$$

$$\theta_{DAE} = \arg \min_{\theta_{DAE}} \text{Loss} (x_{AE}, y_{DAE}) \quad (i > 0) \quad (2)$$

Based on the data generative ability of the DAE, our polyline gap-filling model was constructed as presented in Figure 3. Specifically, each original polyline sample $x_{AE}$ and its masked version sample $x_{DAE}$ (polyline with gap) are pairwise inputs to the generator. $x_{DAE}$ is fed into the encoder–decoder structure to generate the output $y_{DAE}$. Next, by minimizing the reconstruction loss between the output $y_{DAE}$ and the original polyline sample $x_{AE}$, the generator is optimized to generate a semantically consistent and visually realistic gap content of the polyline. Notably, this process corresponds to

![Figure 3. Training process of our generator architecture. The generator takes the masked sample as input and generates an output polyline as similar as possible to the original sample.](image-url)
the training of the model. With the trained model, we only need to input the polyline samples with gaps to the generator to obtain a polyline result with filled gaps.

3.2. Training sample preparation

For the same structure of a neural network, different training datasets can make the model have different abilities to learn features. Because our application context is significantly different from traditional image data learning, it is crucial for our model that we develop a polyline dataset that fulfills the requirements of the application. In this section, we introduce in detail the training data preparation in our proposed gap-filling framework.

- **Vector polyline to sample point set**

Neural network models require the dimensions of the samples to be consistent to share the weights for batch training. However, the complex shape and irregular layout of vector polylines mean it is difficult to directly use the inflection points in neural network models. Some studies have proposed transforming vector data into raster data to fulfill the neural network model requirements (Courtial et al. 2020, Du et al. 2021). However, determining the appropriate image size and resolution is uncertain and, for long and narrow polylines, the rasterization method usually suffers from data redundancy. In our model, we represent the polyline as a set of sample points for generating an equal-size vector for the model input (Figure 4). First, sub-polylines with equal lengths $L_0$ are randomly segmented from the original polyline as individual samples. Second, for each segmented polyline, a set of linear relative reference positions is taken along the polyline to generate the sample points. In practical applications, the length $L_0$ of a single polyline and the interval $l_0$ between any two adjacent reference points can be determined by the data distribution (Section 4). Finally, the $X, Y$ coordinates of the point sets were input into our model as the original features.

- **Original sample and masked sample**

In addition, we need to generate a masked version (polyline with gap) for each training sample. This process can be implemented using binary masks. For this research, the masked samples with gaps were generated by multiplying the binary masks by the original samples. In the masks, the ‘1’ values represent the known context regions, and the ‘0’ values represent the gap regions to be generated. Therefore, the count of the ‘0’ values in the masks represents the count of points in the gaps, and the

![Figure 4](image)

**Figure 4.** The process of obtaining the features of a vector polyline. The vector polyline is transformed into a set of sample points, whose $X, Y$ coordinate values are the input data of our model.
positions of the ‘0’ values in the masks correspond to the locations of the gaps in the samples. Notably, to simulate the real gap existing on the polyline, we set the ‘0’ values in the masks to be continuous. We set different counts and different locations of the ‘0’ values in the masks to enrich the training samples and improve the generalization ability of the model. More detailed information about settings is given in Section 4.1.

3.3. Test sample generation

After the model is trained, a polyline with a gap can be input into the model for filling. However, there remains a challenge in test sample generation (Figure 5). For traditional image inpainting applications, each image is composed of a set of pixels with a fixed size (e.g. 128 × 128); thus, the size of the missing part (i.e. the count of the missing pixels) in the image can be deduced easily. By comparison, the gaps in the vector polylines are uncertain. The sizes of gaps and the complex bends they contain cannot be fixed; that is, the number of points to be filled in the gap (i.e. the count of ‘0’ values in the mask) cannot be set directly. Therefore, we propose using a decision tree model to predict the approximate number of points that should be filled for each gap. The purpose of this step is to preset the size of the points to be filled; then, we

![Figure 5](image_url). Comparisons between the images with missing regions (a and b) and the vector polylines with gaps (c and d). Notably, the gaps in the images always have the same size for a certain application, and different polylines can have different sizes of gaps.
can obtain the feature points from the upstream and downstream of the gap along the polyline to form a complete test sample. Thus, we can ensure that the sample size is the same as that of the training samples.

**Decision tree**

A decision tree can learn to predict the number of points in the gap according to the relevant features. It exists as a tree containing the root node (the dataset), the decision node (the judgment condition) and the leaf node (the final category). The formalization is as follows:

\[ D = \{(x_{DT1}, y_{DT1}), (x_{DT2}, y_{DT2}), ..., (x_{DTn}, y_{DTn})\} \]  

\[ x_{DTi} = (x_{DTi1}, x_{DTi2}, ..., x_{DTin})^T, \quad i = 1, 2, ..., N \]  

\[ y_{DTi} \in \{1, 2, ..., K\}, \quad i = 1, 2, ..., N \]  

where \( D \) is the dataset, \( x_{DTi} \) is the feature vector, \( y_{DTi} \) is the class label, \( N \) is the size of the dataset, \( n \) is the count of the features, \( K \) is the value set of labels. Given an input \( x_{DTi} \), a classified value \( y_{DTi} \) can be predicted by the decision tree model. Under our context, \( y_{DTi} \) corresponds to the count of points in the gaps, denoted as \( C \) in our model. \( x_{DTi} \) are the features of gaps for the decision tree.

The features of a gap comprise the attributes of the gap and its context. As noted in **Section 3.2**, because the vector polylines are difficult to describe, we also use the features of the sample points to represent the polyline features. The gap in **Figure 6** is used as an example to illustrate our operations. First, two features can be determined by the attributes of the gap, that is, the relationship between the start point \( P_{11} \) and the end point \( P_{21} \) of the gap (e.g. the distance \( DIS_0 \) and direction \( DIR_0 \)). We then recorded the curvatures from the upstream and downstream of the gap along the polyline to represent its context. To facilitate implementation, we use the slopes \( kMi \) of any two adjacent sample points to approximate the curvatures of the polyline. The formalization is as follows:

\[ DIS_0 = \sqrt{(X_{P11} - X_{P21})^2 + (Y_{P11} - Y_{P21})^2} \]  

**Figure 6.** Example of extracting features of a gap using its upstream points (i.e. \( P_{11}, P_{12}, P_{13}, ..., P_{1n} \)) and downstream points (i.e. \( P_{21}, P_{22}, P_{23}, ..., P_{2n} \)) along the polyline.
where the ‘1’ value of $M$ indicates the upstream of the gap, and the ‘2’ value of $M$ indicates the downstream of the gap along the polyline. In our experiment, the features of each gap were set to $[\text{DIS}_0, \text{DIR}_0, k_{11}, k_{12}, k_{21}, k_{22}]$, and the classified value was set to $C$ (i.e. number of points in the gap).

- **Test sample generation**

By predicting the count of points ($C$) that should be filled in the gap, we can choose one of the corresponding masks (i.e. the mask with the count of ‘0’ values equal to $C$) to generate the test sample. Figure 7 shows the process of generating test samples. The first step is to choose an appropriate mask. For example, the mask in Figure 7(a) indicates that we must take $m$ sample points from the upstream and $n$ sample points along the polyline.
from the downstream of the gap of the original polyline (Figure 7(b)). Because the neural network models cannot accept NaN values, we use the auxiliary point $P_0$ to occupy the bits, and the $X, Y$ coordinate values of $P_0$ are set to the default value of 0. Then, we can obtain the initial test samples with the same feature dimensions as the training data (Figure 7(c)). Finally, by inputting the initial test samples into our generator, the content of gaps (e.g. shape) can be predicted.

4. Experiment and evaluation

4.1. Datasets and parameters

In the experiment, we used contour lines as our polyline dataset. Contour lines were extracted from 90-m resolution SRTM (Shuttle Radar Topography Mission) digital elevation model (DEM) data (Farr et al. 2007) using ArcMap. The DEM was collected by the National Aeronautics and Space Administration. As described in Section 3.2, the vector polyline data must be transformed into sample points for the generator model. In our experiment, we set $L_0 = 2000m$ to ensure that each sample polyline contains a sufficient number of features, and set $l_0 = 5m$ by considering the sampling loss and the training speed of the model. In addition, we normalize the $X, Y$ coordinate values of each sample to the interval $[0,1]$ for model training. To make the model adaptive for different sized gaps, the proposed model must be trained with a sufficient number and diversity of samples. In the experiment, we set the sizes of gaps to a value between 5% and 50% of the sample length, thus the number of sample points in the gaps ranged from 20 to 200. To accelerate the optimization of the model, we selected several mask templates by setting the count of sample points to a value from 20 to 200 with a step size of 10 (Table 1). We randomly generated 10,000 sample polylines with length $L_0$ from the contour line dataset and obtained 190,000 masked samples as our training dataset. In addition, we randomly generated 10,000 masked samples as our test dataset following the same process of generating training data. Note that, the test samples were not used in the training data.

After these settings, it can be calculated that we had 401 sample points in each sample sub-polyline. Since we used the X and Y coordinates of points as features, the size of the input feature vector was $1 \times 802$. The structure of the generator model was set as 802-600-400-600-802. Because the encoder-decoder structure in our model has

| The count of ‘0’ value | The mask template |
|------------------------|-------------------|
| 20                     | ![Mask Template 20](image) |
| 30                     | ![Mask Template 30](image) |
| ...                    | ![Mask Template ...](image) |
| 200                    | ![Mask Template 200](image) |
different functions from the traditional feature compression, the dimension of the hidden layer does not need to be set to be small. This setting can help avoid the loss of too many features of the polylines by the hidden layers. The ReLU function is applied to activate the hidden layers, and the mean squared error was applied to calculate the loss between the outputs and the original samples. The model was optimized using the Adam optimizer with a learning rate of 0.0001. The model was implemented using PyTorch, a deep learning framework in Python, with GPU acceleration. The experimental environment was a computer with 10 CPU cores (Intel [R] Core [TM] i9-10900X CPU @ 3.70 GHz [20 processors]), 64 GB RAM and 2 GPU [NVIDIA GeForce RTX 2080 Ti]).

4.2. Experimental results

We only show the loss change after 100 times to clearly show the convergence of the loss (Figure 8). A series of results of our generator model with corresponding input masked samples are listed for visual comparison (Figure 9). Our generator can produce visually realistic results regardless of the bend categories (e.g. convex and concave bends), bend counts and bend sizes. This finding indicates that our method is aware of the contextual structure of the polyline and can adaptively extract information from the surroundings. Therefore, our model can promote the synthesis and generation of the polyline gaps. Although our results are not identical to the original polylines, the model can still produce realistic content by smoothly connecting the gap content to the known parts.

The results show that that our proposed model can robustly manage gaps of different sizes and shapes (Figure 10). The model has an aptitude for learning the underlying patterns from the known contents of the given samples, to predict the unknown contents of the gaps. In addition, as the size of the gaps increases, the uncertainty of the gap contents increases, and it becomes more difficult to predict the polyline structure accurately. Nevertheless, the performance of our model does not decrease much

![Figure 8](image_url). Model training error. The training loss is the mean squared error (MSE), and the epoch is the iteration number during the training process.
Figure 9. Gap completion results by using our method.
with a larger gap size, and the results at each scale can still present visually realistic and context-aware content.

For gaps of unknown size, we use the decision tree to estimate the number of points that should be filled in the gap space, for which the accuracies are given in Table 2. Among the predicted results, 70.08% were completely consistent with the truth value, and 2.19% had errors of more than 20 points. In general, the decision tree model can guarantee that more than 90% of the predicted results have errors of less than 10 points compared with the ground truth.

We also conducted experiments to demonstrate the impact of this error on the filling results. We compared the filling results of the model by using three different templates. For template 1, we set the predicted result of the decision tree as the truth value, that is, $P_0 = \text{truth value}$. For template 2, we set the predicted result to be a value larger than the truth value, that is, $P_0 = \text{truth value} + 10$. For template 3, we set the predicted result to be a value smaller than the truth value, that is, $P_0 = \text{truth value} - 10$. We use these three templates to generate the initial vector features for the polyline gaps and then input the generated masked samples into our generator model. Figure 11 shows the final prediction results (i.e. completion results 1, 2,

Table 2. Error statistical result for predicting the counts of points in the gaps by the decision tree model.

| Type                        | Percentage (%) |
|-----------------------------|----------------|
| $|\text{Truth} - \text{Predicted result}| = 0$            | 70.08          |
| $|\text{Truth} - \text{Predicted result}| \leq 5$         | 86.69          |
| $|\text{Truth} - \text{Predicted result}| \leq 10$        | 92.88          |
| $|\text{Truth} - \text{Predicted result}| \leq 20$        | 97.81          |
and 3). The samples constructed using the three templates differ slightly in their known parts, which also leads to a slight difference in the length of the generated content. For example, template 2 has 10 more filling points than template 1 in the gap. Correspondingly, the filling content of completion result 2 is longer than that of completion result 1. In general, the filling results for the three templates were similar in shape and structure. Therefore, our model is robust with different numbers of gap sample points.

4.3. Comparisons

Next, we compare our results with those of the Bezier curve method. The Bezier curve is an important parameter curve in computer graphics, which is constructed using a given start point, a given end point and a series of control points. The Bezier curve does need to pass through the control points, but the count and locations of the control points determine the shape of the polyline.

We designed three strategies for generating input points for the Bezier curve. The first strategy uses all the sample points of a polyline as the input points (Figure 12(a)). The second strategy selects only a small number of points in the context of the gap as the input to the Bezier curve (Figure 12(b)). The third strategy uses the start and end points of the gap as the input, and the control points need to be calculated according to the known gap context. For example, in Figure 12(c), point $PK_1$ is on the extension line of points $P_1$ and point $P_1$, and the distance between points $PK_1$ and the point $P_1$ is $n$ times the length unit $l_0$ of the sample, that is, $D_K = n \times l_0$. Point $PK_2$ was obtained in the same manner on the other side of the gap.

Figure 11. Comparison of the completion results with different mask templates for the same gap. Source polylines and the known parts of sample are displayed in gray and black, respectively.
These three Bezier methods have different performances in our context. In the first method, \( P_1 \) and \( P_2 \) are the starting point and the end point, respectively, and the other points are the control points. The result only maintains the positions of the start and end points, and the bends contained in the original polyline can be over fitted (Figure 13(a)). In the second method, because the control points (i.e. \( P_{11} \) and \( P_{21} \)) are too close to the start point \( P_{12} \) or the end point \( P_{22} \), the generated Bezier curve can be under fitted (Figure 13(b)). The third method has a better performance when the value of parameter \( n \) is large (Figure 13(c,d)). The results of the third method can be connected to known portions without misalignment, and they can maintain an appropriate degree of fitting.

Because the results of the Bezier curve depend on parameter \( n \), we examined the performance with different parameter settings (Figure 14). In general, our results are closer to the ground truth values than those of the Bezier curves, regardless of the parameter. For different input samples, our method can adaptively fill the gaps, and the results have high qualities and complex shapes. For example, the first row of data shows that our model can generate significant bends and recover the main shape of the original polyline. The results of the Bezier curve are relatively simple, and the optimal parameters differ across input samples. For example, for the first three rows of the data in Figure 14, the results with \( n = 60 \) show obviously inappropriate abrupt angularities, whereas for the fourth row, the results with \( n = 60 \) are significantly better than those with \( n = 20 \) and \( n = 40 \). Although adding control points can improve the ability of the Bezier curve to simulate complex bends, determining the optimal locations and count of control points remains a challenge in practical applications. Therefore, although both our model and the Bezier curve model can achieve content prediction for gaps, our model does not require complex parameters and is more generally applicable. In addition, our model can examine the sample polyline from a global perspective, explore the underlying patterns of the samples and generate content highly similar to the ground truth in an end-to-end fashion.

For quantitative comparisons, we chose three measures to evaluate the results. First, the Hausdorff distance and Fréchet distance were used to evaluate the difference between the predicted and original lines. Next, the simple buffer method (SBM) proposed by Goodchild and Hunter (1997) was used to measure the accuracy of the generated polyline with respect to the ground truth. This method is widely used in DEM
analysis to evaluate the accuracy of streamlines (Yu et al. 2021), and its formula is as follows:

$$SBM = \frac{L_{\text{calculated}}}{L_{\text{true}}}$$

where $L_{\text{true}}$ is the length of the ground truth polyline and $L_{\text{calculated}}$ is the length of the calculated polylines that are consistent with the ground truth. More specifically, $L_{\text{calculated}}$ is calculated in terms of the length of the generated polyline falling in the buffer of the ground truth polyline.

All masked samples in the test dataset were used to evaluate our model quantitatively. Comparisons between our results and those of the Bezier curves ($n = 40$) are shown in Table 3. The parameter $h$ represents the width of the buffer in the SBM, and larger SBM values indicate the calculated polyline is more similar to the original polyline. The calculated polylines of our model are more similar to the original polylines than those of the Bezier curve model. The smaller values of the Hausdorff distance

Figure 13. Results of the three Bezier methods: (a, b) the result of the first and second methods, (c) the third method extending $PK_1$ and $PK_2$ within the intersection of upstream portion and downstream portion, and (d) the third method extending $PK_1$ and $PK_2$ beyond the intersection of the upstream portion and downstream portion.
and Fréchet distance also suggest that our model performs better than the Bezier curve model. Because we normalized the $X$, $Y$ coordinate values to the interval $[0,1]$, the values of $h$, Hausdorff distance, and Fréchet distance are dimensionless.

**Figure 14.** Comparisons of the results of our model and the Bezier curve model.

**Table 3.** Quantitative evaluation of the results of our model and the Bezier curve model.

| Method               | $h = 0.01$ | $h = 0.02$ | $h = 0.03$ | $h = 0.04$ | Hausdorff distance | Fréchet distance |
|----------------------|------------|------------|------------|------------|--------------------|------------------|
| Our proposed model   | 62.50      | 85.12      | 93.10      | 95.71      | 0.0230             | 0.0231           |
| Bezier curve model   | 45.01      | 66.55      | 77.49      | 84.37      | 0.0385             | 0.0414           |
4.4. Example application scenarios

Next, we conducted two experiments to demonstrate the applicability of our model. We choose two application scenarios: the gaps in the automatic vectorization of printed historical maps and the gaps in the GNSS main trace after deleting outliers and secondary trace points. More application scenarios and comparisons are provided in Supplementary Material.

- **Scenario 1–Filling the gaps in automatic vectorization of historical maps**
We collected a printed historical topographical map of a part of Washington D.C. from 1861 from the Library of Congress website (https://www.loc.gov/item/88694009/). We used the ArcScan extension in ArcGIS to vectorize the image after first reclassifying it to binary using the Jenks Natural Breaks Classification method (Jenks 1967). Then, we obtained the vectorized polylines by the centerline vectorization method (Figure 15(b)). We observed that even complete polylines in the printed map can have gaps after vectorization. We inferred the sample length $L_0$ and the interval of sample points $l_0$ from the shape of sample polyline. We extracted the context features of each gap, namely $[DIS_0, DIR_0, k_{11}, k_{12}, k_{21}, k_{22}]$ (see Section 3.3), and input the features into the

![Figure 15. Gap-filling results of our model for the automatic vectorization of polylines on a historical map. From left to right, top to bottom: printed historical map (a), polylines after automatic vectorization (b), our filling results (c), and overlay of our results and the map (d).]
decision tree to predict the number of points that should be filled in the gap. Then, according to the number of predicted points, we chose a suitable mask template to cover the source data to obtain the sample data for the gap. Note that our well-trained model requires each sample to contain only one gap. Therefore, for those samples whose upstream polyline length or downstream polyline length was less than the required length in the mask template, we reduced the sampling interval for the source data to ensure that the number of points of the upstream and downstream was consistent with the number of points of the mask template. Our model can successfully fill the gaps in the polylines (Figure 15(c)), and the shapes of the predicted results were consistent with those of the real polylines (Figure 15(d)). The advantage of our model is that it can be used to fill local bends with different structures (Figure 16).

**Scenario 2—Filling the gaps in main traces after deleting outliers and secondary trace features**

In the behavior analysis of moving objects, outliers and secondary trace features should be deleted to highlight the main patterns (Ivanovic et al. 2019), which can lead to various gaps in the main traces. We downloaded GNSS point data from the Movebank website (https://www.movebank.org/), representing the grazing movements of cattle in the Far North Region of Cameroon. Data are provided by Moritz et al. (2010; data at https://www.doi.org/10.5441/001/1.j682ds56). We connected the GNSS points in chronological order and used the Polynomial Approximation with Exponential Kernel (PAEK) algorithm (Bodansky et al. 2002) with an appropriate tolerance to smooth the traces. Then, we manually eliminated outliers and secondary features to obtain the trace curves with gaps. Finally, we generated the input samples as described in Scenario 1. The results suggest that our model can adaptively fill in the gaps in trace data (Figure 17). Furthermore, the predicted results of the model can be used as a simplified form of the original traces, which is conducive to the pattern analysis of movement behaviors.

**5. Conclusion and future work**

In this work, we propose a framework for gap completion of polylines, including cartographic data processing and model training. In data processing, the polylines are transformed into samples with uniform feature dimensions that are acceptable to the neural network model. For gaps of unknown size, the decision tree is used to predict the number of points that should be filled in the gaps. The introduction of a decision tree facilitates the generation of polyline samples and improves the final predictions of the gaps. The key part of the framework is a generative network that implements an encoder–decoder model as a generator. Such a model can successfully synthesize semantically valid, visually plausible content for the missing regions by considering the context of gaps. Compared with the Bezier curve method, our method can generate polylines with complex shapes and structures. In addition, two practical applications demonstrate the filling ability of our model in different contexts.

However, according to our review of the literature, our model is the first attempt to use neural network models to fill gaps in vector polylines. Notably, our model has
Figure 16. Details of the local filling results. In each sub-figure (a-f), from left to right, top to bottom: printed historical map (i), result after automatic vectorization (ii), our filling result (iii), and overlay of our results and the map (iv).
limitations. For example, our model is limited in size and number of gaps at this stage, and the training dataset is critical to the performance of our model. In further research, we plan to enrich the size and count of gaps in each simulated sample and randomly design the gaps to improve the generalization ability of the model. In addition, attempting to add geographical knowledge to the input feature vector could improve the results. For example, the shape of adjacent contour lines may help in filling the gaps of contour lines with preserved terrain features.

In further research, we plan to improve our work from different perspectives. First, cartographic data usually show multiscale features, and the corresponding polylines can also have bends with different details. We plan to incorporate these multiscale features into our generator model. Thus, the results are promising for multiscale mapping and spatial database establishment. From the perspective of models, more neural networks can be used in our task, for example GANs and attention-based mechanisms.

Figure 17. Filling results of our model for the main traces with gaps. In each sub-figure (a-c), from left to right, top to bottom: original GNSS points (i), trace lines (ii), main trace lines after deleting outliers and secondary trace features (iii), and filling result of our model (iv). The points represent the trace data.
We can refer to the structure of GANs and add a discriminator after our generator module to judge the generated local polylines as real or fake.

**Data and codes availability statement**

The data and codes that support the findings of this study are available with a DOI at [https://doi.org/10.6084/m9.figshare.14742717](https://doi.org/10.6084/m9.figshare.14742717).

**Acknowledgments**

The authors are grateful to the associate editor, Shawn Laffan, and the anonymous referees for their valuable comments and suggestions.

**Author contributions**

Wenhao Yu: Conceptualization, methodology, formal analysis, validation, writing—original draft preparation, writing—review and editing, supervision, project administration, funding acquisition; Yujie Chen: Methodology, validation, formal analysis, investigation, writing—original draft preparation, writing—review and editing, visualization. All authors have read and agreed to the published version of the manuscript.

**Disclosure statement**

No potential conflict of interest was reported by the author(s).

**Funding**

The project was supported by the National Natural Science Foundation of China (Grant No. 42071442), and the Fundamental Research Funds for the Central Universities, China University of Geosciences (Wuhan) (No. CUG170640).

**Notes on contributors**

*Wenhao Yu* received the BS and PhD degrees in Geoinformatics from the Wuhan University, Wuhan, China, in 2010 and 2015, respectively. He is a professor at China University of Geosciences, Wuhan, China (CUG). His research interests include spatial data mining, map generalization and deep learning.

*Yujie Chen* is a master student in the School of Geography and Information Engineering, China University of Geosciences, Wuhan, China (CUG). Her research interests include map generalization and spatial analysis.

**ORCID**

Wenhao Yu [http://orcid.org/0000-0003-1521-2674](http://orcid.org/0000-0003-1521-2674)

**References**

Basiri, A., *et al.*, 2019. Crowdsourced geospatial data quality: challenges and future directions. *International Journal of Geographical Information Science*, 33 (8), 1588–1593.
Bauer, C., 2013. On the (In-)accuracy of GPS measures of smartphones: a study of running tracking applications. In: Proceedings of international conference on advances in mobile computing & multimedia. New York: ACM, 335–341.

Bodansky, E., Gribov, A., and Pilouk, M., 2002. Smoothing and compression of lines obtained by raster-to-vector conversion. Graphics Recognition Algorithms and Applications, 2390, 256–265.

Courtial, A., et al., 2020. Exploring the potential of deep learning segmentation for mountain roads generalisation. ISPRS International Journal of Geo-Information, 9 (5), 338.

Dong, J., et al., 2019. Inpainting of remote sensing sst images with deep convolutional generative adversarial network. IEEE Geoscience and Remote Sensing Letters, 16 (2), 173–175.

Du, J., et al., 2021. Segmentation and sampling method for complex polyline generalization based on a generative adversarial network. Geocarto International, 3, 1–23.

Elharrouss, O., et al., 2020. Image inpainting: a review. Neural Processing Letters, 51 (2), 2007–2028.

Farr, T.G., et al., 2007. The shuttle radar topography mission. Reviews of Geophysics., 45, RG2004.

Ge, Y., et al., 2019. Study on automatic edge match checking for geographical conditions survey data based on ArcEngine. Geomatics & Spatial Information Technology, 42 (2), 58–64.

Goodchild, M., and Hunter, G., 1997. A simple positional accuracy for linear features. International Journal of Geographical Information Science, 11 (3), 299–306.

Goodfellow, I., Bengio, Y., and Courville, A., 2016. Deep learning. Cambridge, MA: MIT Press.

Hong, J., and Huang, M., 2017. Enabling smart data selection based on data completeness measures: a quality-aware approach. International Journal of Geographical Information Science, 31 (6), 1–1197.

Hui, H., and Tong, X., 2000. Automatic edge match of digital maps. Journal of Tongji University, 28 (1), 33–36.

Iizuka, S., Simo-Serra, E., and Ishikawa, H., 2017. Globally and locally consistent image completion. ACM Transactions on Graphics, 36 (4), 1–14.

Ivanovic, S., et al., 2019. A filtering-based approach for improving crowdsourced GNSS traces in a data update context. ISPRS International Journal of Geo-Information, 8 (9), 380.

Janowicz, K., et al., 2020. GeoAI: spatially explicit artificial intelligence techniques for geographic knowledge discovery and beyond. International Journal of Geographical Information Science, 34 (4), 625–636.

Jenks, G., 1967. The data model concept in statistical mapping. International Yearbook of Cartography, 7, 186–190.

Li, W., 2020. GeoAI: where machine learning and big data converge in GIScience. Journal of Spatial Information Science, 20 (20), 71–77.

Li, X., et al., 2019. An object-based river extraction method via optimized transductive support vector machine for multi-spectral remote-sensing images. IEEE Access., 7, 46165–46175.

Li, Y., et al., 2017. Generative face completion. In: Proceedings of 2017 IEEE conference on computer vision and pattern recognition (CVPR), Honolulu, HI, USA, 21–26 Jul. 2017, 5892–5900.

Lou, S., et al., 2018. Preliminary investigation on single remote sensing image inpainting through a modified GAN. In: Proceedings of IAPR Workshop on Pattern Recognition in Remote Sensing, Beijing, China, 19–20 Aug. 2018, 1–6.

Mahabir, R., et al., 2017. Authoritative and volunteered geographical information in a developing country: a comparative case study of road datasets in Nairobi, Kenya. ISPRS International Journal of Geo-Information, 6 (1), 24.

Martin, N., et al., 2008. Morphing polylines: a step towards continuous generalization. Computers, Environment and Urban Systems, 32 (4), 248–260.

Minaei, M., 2020. Evolution, density and completeness of OpenStreetMap road networks in developing countries: the case of Iran. Applied Geography, 119, 102246.

Modava, M., and Akbarizadeh, G., 2017. Coastline extraction from SAR images using spatial fuzzy clustering and the active contour method. International Journal of Remote Sensing, 38 (2), 355–370.

Mooney, P., et al., 2016. Towards a protocol for the collection of VGI vector data. ISPRS International Journal of Geo-Information, 5 (11), 217.
Moritz, M., et al., 2010. An integrated approach to modeling grazing pressure in pastoral systems: the case of the Logone Floodplain (Cameroon). *Human Ecology*, 38 (6), 775–789.

Pathak, D., et al., 2016. Context encoders: feature learning by inpainting. *In: Proceedings of the IEEE conference on computer vision and pattern recognition*, Las Vegas, NV, USA, 27–30 Jun. 2016, 2536–2544.

Rojas, D., Fernandes, B., and Fernandes, S., 2020. A review on image inpainting techniques and datasets. *In: Proceedings of 2020 33rd SIBGRAPI conference on graphics, patterns and images (SIBGRAPI)*, Recife/Porto de Galinhas, Brazil, 7–10 Nov. 2020.

Samsonov, T., et al., 2019. Automated placement of supplementary contour lines. *International Journal of Geographical Information Science*, 33 (10), 2072–2093.

Vincent, P., et al., 2008. Extracting and composing robust features with denoising autoencoders. *In: Proceedings of Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008)*, Helsinki, Finland, 5–9 June 2008, 1096–1103.

Vincent, P., et al., 2010. Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11 (12), 3371–3408.

Wang, Z., Zhang, M., and Liu, W., 2020. An effective road extraction method from remote sensing images based on self-adaptive threshold function. *In: Proceedings of IEEE 2019 Asia-Pacific signal and information processing association annual summit and conference (APSIPA ASC)*, Lanzhou, China, 18–21 Nov. 2019, 455–460.

Xie, J., Xu, L., and Chen, E., 2012. Image denoising and inpainting with deep neural networks. *Advances in Neural Information Processing Systems*, 1, 341–349.

Yang, C., et al., 2017. High-resolution image inpainting using multi-scale neural patch synthesis. *In: Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 21–26 Jul. 2017, 4076–4084.

Yu, J., et al., 2018. Generative image inpainting with contextual attention. *In: Proceedings of 2018 IEEE/CVF conference on computer vision and pattern recognition*, Salt Lake City, UT, USA, 18–23 Jun. 2018, 5505–5514.

Yu, W., et al., 2021. An integrated method for DEM simplification with terrain structural features and smooth morphology preserved. *International Journal of Geographical Information Science*, 35 (2), 273–295.

Zakaria, W., Ramli, A., and Ali, J., 2019. Bezier curves interpolation with end point constraints on road map. *In: Proceedings of the international conference on mathematical sciences and technology 2018 (MATHTECH2018): Innovative technologies for mathematics & mathematics for technological innovation*, Penang, Malaysia, 10–12 Dec. 2018.

Zhang, Z., Xia, Q., and Xu, Z., 2001. Automatic adjoining sheets based on line primitives. *Journal of Institute of Surveying and Mapping*, 18, 77–80.

Zhao, L., Zheng, G., and Li, J., 2012. Automatic parking path optimization based on Bezier curve fitting. *In: Proceeding of IEEE international conference on automation & logistics*, Zhengzhou, China, Aug. 2012.

Zhao, P., Jonietz, D., and Raubal, M., 2021. Applying frequent-pattern mining and time geography to impute gaps in smartphone-based human-movement data. *International Journal of Geographical Information Science*, 35 (11), 2187–2215.

Zhu, D., et al., 2020. Spatial interpolation using conditional generative adversarial neural networks. *International Journal of Geographical Information Science*, 34 (4), 735–758.