Animal Trajectory Imputation Using Deep Learning

Kehui Yao[1]

[1]Department of Statistics, University of Wisconsin-Madison

Abstract

Imputing missing data in animal trajectories is crucial for understanding their movements during unobserved periods. Traditional methods such as linear interpolation and the continuous-time correlated random walk model often fail to capture the complexity of animal movements, making the imputation less accurate. Our study introduces a deep learning approach to animal trajectory imputation, using the Conditional Score-based Diffusion Model (CSDI) originally designed for multivariate time series data. Unlike traditional methods, CSDI uses observed trajectories and external covariates to impute missing trajectories, capturing periodic patterns and the influence of covariates on animal movements, which leads to more accurate imputations. We develop a pipeline to segment trajectories from hundreds of deer and train the model to learn general movement patterns from this collective dataset. When tested on a dataset of 402 female deer trajectories in Wisconsin, our method not only achieves more accurate deterministic imputations but also more reliable probabilistic imputations than existing methods.

keywords: Animal Trajectory Imputation, Deep Learning, Conditional Scored-based Diffusion Model

1 Introduction

Animal movement data often includes spatial positions recorded at specific time points using GPS telemetry technology. Although these locations are usually sampled at regular intervals, data gaps are inevitable due to collection issues or noise. A variety of statistical and machine learning methods have been used in an effort to fill in such missing data.

One common method, linear interpolation, estimates missing values by drawing straight lines between observed data points. Essentially, it assumes a linear relationship between consecutive points and interpolates the missing values accordingly. However, this method has several drawbacks. First, the estimation is too simple, failing to capture the complex dynamics of animal movement. Secondly, the imputation is deterministic, which means it fails to account for the inherent uncertainty in predicting the path between observed locations. Beyond linear interpolation, several statistical methods are used to analyze animal movement data. Hidden Markov Models (HMMs) and State-Space Models (SSMs) are effective in analyzing datasets with consistent time intervals (Holzmann et al., 2006; Patterson et al., 2009; McKellar et al., 2015; Towner et al., 2016; Breed et al., 2012). Continuous time models are more suitable for datasets with irregular or continuous time intervals (Johnson et al., 2008; Buderman et al., 2016). These methods are parametric. Once fitted to the observed data, predict the missing locations of animals at specified time points. Additionally, these models naturally incorporate uncertainty. However, these statistical models have limitations. First, these methods often require expert knowledge of animal movement patterns and ecological insights because accurate model specification is crucial. They often lack the flexibility to identify patterns or to understand complex, nonlinear environmental interactions directly from the observed data. Second, those methods typically model the movement of only one animal at a time and do not share information across individuals. Previously, this was reasonable due to limited data availability. However, modern GPS technology now enables the simultaneous tracking of multiple animals, providing the opportunity to leverage collective data for enhanced imputation.

Machine learning and deep learning methods often provide superior pattern extraction capabilities compared to traditional statistical methods. Researchers have explored these methods for animal trajectory analysis. Examples include the RandomForest approach by Rew et al. (2019), inverse reinforcement learning by Hirakawa et al. (2018), and the LSTM encoder-decoder network by Li et al. (2021). Despite their potential, deep learning's adoption for animal trajectory imputation remains limited. This limited adoption may be due to several factors. A major concern is that most deep learning models produce deterministic predictions, while a probabilistic approach providing a range of possible locations might better predict an animal's potential positions. Furthermore, deep learning requires extensive training data. In situations with limited data, such as tracking a few animals, these models may learn more noise

than underlying patterns. Lastly, methods like LSTM and RandomForest, designed primarily for datasets with regular time intervals, are not suitable for those with irregular intervals.

In our study, we apply the Conditional Score-based Diffusion Models (CSDI) (Tashiro et al., 2021), originally designed for probabilistic time series imputation, to the context of animal trajectory imputation. We prepare a large training dataset by segmenting the trajectories of 402 deer. The model is trained to impute missing trajectories based on observed locations and by incorporating the deer's habitat landcover and temporal information. Our study shows that the model significantly outperforms traditional trajectory imputation methods, such as linear interpolation and the continuous-time correlated random walk model (Johnson et al., 2008) in both mean absolute error (MAE) and continuous ranked probability score (CRPS) metrics. Additionally, this model addresses the two main challenges previously outlined: it can impute data with irregular time intervals and produce probabilistic imputations. To our knowledge, this is the first work to apply generative models to animal trajectory imputation.

2 Notations for Animal Trajectory Data

Consider a collection of N trajectories with missing data, each associated with either a single animal or a group of animals. Telemetry data is represented as $\boldsymbol{X} \in \mathbb{R}^{2\times L}$, where "2" denotes the two spatial coordinates (longitude and latitude), and L represents the trajectory's length. To distinguish between missing and observed data, we use an observation mask $\boldsymbol{M} \in \{0,1\}^{2\times L}$. Here, $\boldsymbol{M}[c,l] = 0$ indicates that $\boldsymbol{X}[c,l]$ is missing, and $\boldsymbol{M}[c,l] = 1$ indicates that $\boldsymbol{X}[c,l]$ is observed, for $c \in \{1,2\}$ and $l \in \{1,\ldots,L\}$. It is important to note that the time intervals between consecutive data points may vary, denoted by $\boldsymbol{s} \in \mathbb{R}^L$. Additionally, we include relevant information for each trajectory in $\boldsymbol{V} \in \mathbb{R}^{L\times K}$, where K is the dimension of information. In summary, each trajectory is represented by the set $\{\boldsymbol{X}, \boldsymbol{M}, \boldsymbol{V}, \boldsymbol{s}\}$. We define $\boldsymbol{X}^{\text{obs}} = \boldsymbol{X} \odot \boldsymbol{M}$ as the observed and $\boldsymbol{X}^{\text{mis}} = \boldsymbol{X} \odot (\boldsymbol{1} - \boldsymbol{M})$ as the missing that requires imputation.

3 CSDI Method

Tashiro et al. (2021) introduced Conditional Score-based Diffusion models for multivariate time series imputation (CSDI). Since animal trajectories can be considered as bivariate time series with additional variables, the idea is to use CSDI in the context of animal trajectory

imputation. The following sections will provide an overview of how CSDI works for time series imputation. We start by explaining the denoising diffusion probabilistic model (DDPM) and its conditional variant. Next, we will show how the conditional DDPM can be applied to time series imputation.

3.1 Denoising Diffusion Probabilistic Models

Assume we want to estimate a true data distribution $q(\boldsymbol{X}_0)$. The objective of DDPM is to train a model $p_\theta(\boldsymbol{X}_0)$, to closely match the true data distribution $q(\boldsymbol{X}_0)$ by working with a sequence of latent variables $\boldsymbol{X}_t$, (for $t = 1$ to T), which exist in the same sample space as $\boldsymbol{X}_0$. Following the diffusion probabilistic model framework introduced by Ho et al. (2020), the approach consists of two main stages: a forward process and a reverse process.

Forward Diffusion Process

The forward process begins by sampling a data point $\boldsymbol{X}_0$ from an actual data distribution $q(\boldsymbol{X}_0)$, and then incrementally introduces Gaussian noise over T steps, yielding a series of increasingly noisy samples $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_T$. The step sizes are controlled by a predetermined sequence of variances, $\{\beta_t \in (0,1)\}_{t=1}^T$. The forward diffusion process can be expressed as:

$$q(\boldsymbol{X}_{1:T}|\boldsymbol{X}_0) = \prod_{t=1}^{T} q(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}) \tag{1}$$

$$q(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}\boldsymbol{X}_{t-1}, \beta_t\mathbf{I}). \tag{2}$$

Based on the forward process, we can sample $\boldsymbol{X}_t$ at any arbitrary time step t directly from the initial data point $\boldsymbol{X}_0$, using the distribution $q(\boldsymbol{X}_t|\boldsymbol{X}_0) = \mathcal{N}(\boldsymbol{X}_t; \sqrt{\alpha_t}\boldsymbol{X}_0, (1-\alpha_t)\mathbf{I})$, where $\alpha_t = \prod_{i=1}^{t}(1-\beta_i)$. Therefore, $\boldsymbol{X}_t = \sqrt{\alpha_t}\boldsymbol{X}_0 + (1-\alpha_t)\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$.

Reverse Diffusion Process

The objective of reversing the diffusion process is to reconstruct the original data point starting from Gaussian noise, denoted as $\boldsymbol{X}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. However, directly calculating the reverse conditional probabilities $q(\boldsymbol{X}_{t-1}|\boldsymbol{X}_t)$ is challenging. As a solution, a model p_θ is trained to approximate these conditional probabilities and to execute the reverse diffusion. The reverse

diffusion process can be expressed as:

$$p_\theta(\boldsymbol{X}_{0:T}) = p(\boldsymbol{X}_T)\prod_{t=1}^{T} p_\theta(\boldsymbol{X}_{t-1}|\boldsymbol{X}_t) \tag{3}$$

$$p(\boldsymbol{X}_T) \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{4}$$

$$p_\theta(\boldsymbol{X}_{t-1}|\boldsymbol{X}_t) = \mathcal{N}(\boldsymbol{X}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{X}_t, t), \sigma_\theta(\boldsymbol{X}_t, t)\mathbf{I}) \tag{5}$$

(Ho et al., 2020) considers the following parameterization of $\boldsymbol{\mu}_\theta(\boldsymbol{X}_t, t)$ and $\sigma_\theta(\boldsymbol{X}_t, t)$:

$$\boldsymbol{\mu}_\theta(\boldsymbol{X}_t, t) = \frac{1}{\alpha_t}\{\boldsymbol{X}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{X}_t, t)\} \tag{6}$$

$$\sigma_\theta(\boldsymbol{X}_t, t) = \tilde{\beta}_t^{1/2} \tag{7}$$

$$\tilde{\beta}_t = \begin{cases} \frac{1-\alpha_{t-1}}{1-\alpha_t}\beta_t & t > 1 \\ \beta_1 & t = 1 \end{cases} \tag{8}$$

where $\boldsymbol{\epsilon}_\theta$ is a trainable denoising function. We denote $\boldsymbol{\mu}_\theta(\boldsymbol{X}_t, t)$ and $\sigma_\theta(\boldsymbol{X}_t, t)$ in Eq. 6 as $\boldsymbol{\mu}^{\text{DDPM}}(\boldsymbol{X}_t, t, \boldsymbol{\epsilon}_\theta(\boldsymbol{X}_t, t))$ and $\sigma^{\text{DDPM}}(\boldsymbol{X}_t, t)$, respectively. Under this parameterization, Ho et al. (2020) have shown that the reverse process can be trained by solving the following optimization problem:

$$\min_\theta \mathcal{L}(\theta) = \min_\theta \mathbb{E}_{t\sim[1,T],\boldsymbol{X}_0,\boldsymbol{\epsilon}}||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{X}_t, t)||_2^2 \tag{9}$$

$$\text{where } \boldsymbol{X}_t = \sqrt{\alpha_t}\boldsymbol{X}_0 + (1-\alpha_t)\boldsymbol{\epsilon}. \tag{10}$$

This optimization function implies that the denoising function $\boldsymbol{\epsilon}_\theta$ estimates the noise vector $\boldsymbol{\epsilon}$ that was added to its noisy input $\boldsymbol{X}_t$. Once trained, we can sample $\boldsymbol{X}_0$ from Eq. 3. More details of DDPM can be found in Appendix A.

3.2 Imputation with Conditional Diffusion Models

Let's assume the missing data $\boldsymbol{X}^{\text{mis}}$ follows a distribution $q(\boldsymbol{X}^{\text{mis}}|\boldsymbol{X}^{\text{obs}}, \boldsymbol{V}, \boldsymbol{s})$, which is conditioned on the observed data $\boldsymbol{X}^{\text{obs}}$ and additional variables $\boldsymbol{V}$ and $\boldsymbol{s}$. Our goal is to construct a model denoted by $p_\theta(\boldsymbol{X}^{\text{mis}}|\boldsymbol{X}^{\text{obs}}, \boldsymbol{V}, \boldsymbol{s})$ that can closely approximate this true distribution q. Consider modeling $p_\theta(\boldsymbol{X}_0^{\text{mis}}|\boldsymbol{X}_0^{\text{obs}}, \boldsymbol{V}, \boldsymbol{s})$ with a diffusion model by extending the reverse

process in Eq. 3 to a conditional format:

$$p_\theta(\boldsymbol{X}_{0:T}^{\text{mis}}|\boldsymbol{X}_0^{\text{obs}},\boldsymbol{V},\boldsymbol{s}) = p(\boldsymbol{X}_T^{\text{mis}})\prod_{t=1}^{T} p_\theta(\boldsymbol{X}_{t-1}^{\text{mis}}|\boldsymbol{X}_t^{\text{mis}},\boldsymbol{X}_0^{\text{obs}},\boldsymbol{V},\boldsymbol{s}) \tag{11}$$

$$\boldsymbol{X}_T^{\text{mis}} \sim \mathcal{N}(\mathbf{0},\mathbf{I}) \tag{12}$$

$$p_\theta(\boldsymbol{X}_{t-1}^{\text{mis}}|\boldsymbol{X}_t^{\text{mis}},\boldsymbol{X}_0^{\text{obs}}) = \mathcal{N}\{\boldsymbol{X}_{t-1}^{\text{mis}};\boldsymbol{\mu}_\theta(\boldsymbol{X}_t^{\text{mis}},t|\boldsymbol{X}_0^{\text{obs}},\boldsymbol{V},\boldsymbol{s}),\sigma_\theta(\boldsymbol{X}_t^{\text{mis}},t|\boldsymbol{X}_0^{\text{obs}},\boldsymbol{V},\boldsymbol{s})\mathbf{I}\} \tag{13}$$

Based on Eq. 11, we aim to model the conditional distribution $p_\theta(\boldsymbol{X}_{t-1}^{\text{mis}}|\boldsymbol{X}_t^{\text{mis}},\boldsymbol{X}_0^{\text{obs}},\boldsymbol{V},\boldsymbol{s})$. Specifically, we extend the unconditional DDPM in Eq. 6 to the following conditional case.

$$\boldsymbol{\mu}_\theta(\boldsymbol{X}_t^{\text{mis}},t|\boldsymbol{X}_0^{\text{obs}},\boldsymbol{V},\boldsymbol{s}) = \boldsymbol{\mu}^{\text{DDPM}}(\boldsymbol{X}_t^{\text{mis}},t,\boldsymbol{\epsilon}_\theta(\boldsymbol{X}_t^{\text{mis}},t|\boldsymbol{X}_0^{\text{obs}},\boldsymbol{V},\boldsymbol{s})) \tag{14}$$

$$\sigma_\theta(\boldsymbol{X}_t^{\text{mis}},t|\boldsymbol{X}_0^{\text{obs}},\boldsymbol{V},\boldsymbol{s}) = \sigma^{\text{DDPM}}(\boldsymbol{X}_t^{\text{mis}},t) \tag{15}$$

where $\boldsymbol{\epsilon}_\theta$ becomes the conditional denoising function that takes conditional observations $\boldsymbol{X}_0^{\text{obs}}$ and side information $\boldsymbol{V}$ and $\boldsymbol{s}$ as inputs. $\boldsymbol{\mu}^{\text{DDPM}}$ and σ^{DDPM} in Eq. 14 are the same functions defined in Eq. 6.

Once trained, given the function $\boldsymbol{\epsilon}_\theta$, $\boldsymbol{X}_0^{\text{obs}}$, $\boldsymbol{V}$, and $\boldsymbol{s}$, we can sample $\boldsymbol{X}_0^{\text{mis}}$ based on the reverse process in Eq. 11.

3.3 Training of CSDI

It is important to note that the form of $\boldsymbol{\epsilon}_\theta$ is the only difference between the parametrization in Eq. 14 and Eq. 6. Consequently, the training process for the conditional denoising function follows the same procedure as the unconditional model in Section 3.1. Given conditional observations $\boldsymbol{X}_0^{\text{obs}}$, $\boldsymbol{V}$, $\boldsymbol{s}$ and imputation targets $\boldsymbol{X}_0^{\text{mis}}$, we sample noisy targets $\boldsymbol{X}_t^{\text{mis}} = \sqrt{\alpha_t}\boldsymbol{X}_0^{\text{mis}} + (1-\alpha_t)\boldsymbol{\epsilon}$, and train $\boldsymbol{\epsilon}_\theta$ by minimizing the following loss function:

$$\min_\theta \mathcal{L}(\theta) = \min_\theta \mathbb{E}_{t\sim[1,T],\boldsymbol{X}_0^{\text{mis}},\boldsymbol{\epsilon}}||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{X}_t^{\text{mis}},t|\boldsymbol{X}_0^{\text{obs}},\boldsymbol{V},\boldsymbol{s})||_2^2, \tag{16}$$

However, in practical scenarios, the ground-truth values for $\boldsymbol{X}_0^{\text{mis}}$ are typically unknown during training. To address this, CSDI uses a self-supervised learning strategy, that is, for a given sample $\boldsymbol{X}_0$, the observed part of $\boldsymbol{X}_0^{\text{obs}}$ are further separated into two parts, one serving as imputation targets, denoted as $\boldsymbol{X}_0^{\text{ta}}$, and the other as conditional observations $\boldsymbol{X}_0^{\text{co}}$.

Once we obtain $\boldsymbol{X}_0^{\text{ta}}$ and $\boldsymbol{X}_0^{\text{co}}$, we proceed by substituting $\boldsymbol{X}_0^{\text{mis}}$ with $\boldsymbol{X}_0^{\text{ta}}$, $\boldsymbol{X}_0^{\text{obs}}$ with $\boldsymbol{X}_0^{\text{co}}$, and train $\boldsymbol{\epsilon}_\theta$ in Eq. 16. The details of the training algorithm are shown in Algorithm 1.

Algorithm 1 Training of CSDI

Input: Training data distribution $q(\cdot)$, masking strategy $\mathcal{T}$, the number of iterations n_{iter}, and the sequence of noise levels $\{\alpha_t\}$.

Output: Trained denoising function $\boldsymbol{\epsilon}_\theta$.

for $i = 1$ **to** n_{iter} **do**

  Sample t from Uniform($\{1, \ldots, T\}$), Sample $\{\boldsymbol{X}_0, \boldsymbol{M}, \boldsymbol{V}, \boldsymbol{s}\}$ from $q(\cdot)$.

  Apply the strategy $\mathcal{T}$ to separate $\boldsymbol{X}_0$ into conditional observations $\boldsymbol{X}_0^{\text{co}}$ and imputation targets $\boldsymbol{X}_0^{\text{ta}}$.

  Sample $\boldsymbol{\epsilon}$ from $\mathcal{N}(\mathbf{0}, \mathrm{I})$, where $\boldsymbol{\epsilon} \in \mathcal{X}$.

  Compute noisy targets $\boldsymbol{X}_t^{\text{ta}} = \sqrt{\alpha}_t \boldsymbol{X}_0^{\text{ta}} + (1 - \alpha_t)\boldsymbol{\epsilon}$.

  Update $\boldsymbol{\epsilon}_\theta$ by minimizing the squared error loss $\left\| \left(\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta \left(\mathbf{X}_t^{\text{ta}}, t \mid \boldsymbol{X}_0^{\text{co}}, \boldsymbol{V}_j, \boldsymbol{s}_j \right) \right) \right\|_2^2$.

end for

3.4 Imputation of CDSI

Once the $\boldsymbol{\epsilon}_\theta$ is trained, we follow Eq. 11 to generate $\boldsymbol{X}_0^{\text{mis}}$. The steps involved in the imputation process are outlined in Algorithm 2.

Algorithm 2 Imputation (Sampling) with CSDI

Input: a data sample $\{\boldsymbol{X}_0, \boldsymbol{M}, \boldsymbol{V}, \boldsymbol{s}\}$, a trained denoising function $\boldsymbol{\epsilon}_\theta$.

Output: Imputed missing values $\boldsymbol{X}_0^{\text{mis}}$.

Initialize $\boldsymbol{X}_T^{\text{mis}}$ by sampling from $\mathcal{N}(\mathbf{0}, \mathbf{I})$, where $\boldsymbol{X}_T^{\text{mis}} \in \mathcal{X}$.

for $t = T$ **to** 1 **do**

  Compute $\boldsymbol{X}_{t-1}^{\text{mis}}$ using Eq. 11 and Eq. 14.

end for

Output the final imputed values $\boldsymbol{X}_0^{\text{mis}}$.

3.5 Architecture of the Denoising Function

In this section, we briefly describe the architecture of the denoising function $\boldsymbol{\epsilon}_\theta$. At its core, $\boldsymbol{\epsilon}_\theta$ is designed as a neural network with multiple residual layers and channels. The complete structure of $\boldsymbol{\epsilon}_\theta$ is detailed in Tashiro et al. (2021).

The model inputs include the observed data $\boldsymbol{X}_0^{\text{obs}}$, the missing data $\boldsymbol{X}_t^{\text{mis}}$, covariate $\boldsymbol{V}$,

time intervals $\boldsymbol{s}$, and a diffusion step t. The model uses a 128-dimensional embedding for the diffusion step t and $\boldsymbol{s}$ (Vaswani et al., 2017) and uses a 64-dimensional feature embedding for $\boldsymbol{V}$. The temporal embedding for diffusion step t and time interval $\boldsymbol{s}$ are expressed as follows:

$$\text{DiffusionEmbed}(t) = \{\sin(10^{0\cdot 4/63}t), \ldots, \sin(10^{63\cdot 4/63}t), \cos(10^{0\cdot 4/63}t), \ldots, \cos(10^{63\cdot 4/63}t)\}. \tag{17}$$

$$\text{TimeEmbed}(s_l) = \{\sin(s_l/\tau^{0/64}), \ldots, \sin(s_l/\tau^{63/64}), \cos(s_l/\tau^{0/64}), \ldots, \cos(s_l/\tau^{63/64})\}, \tag{18}$$

where $\tau = 10000$.

Within each residual layer, the model uses a single-layer Transformer Encoder. This encoder includes a multi-head attention layer, fully connected layers, and layer normalization. These components are designed to effectively capture both temporal and feature-based dependencies within multivariate time series data.

4 Case Study

4.1 Dataset

Our dataset consists of telemetry data for 402 female deer tracked across Wisconsin. For each deer, locations were recorded at multiple time points. Here, locations are recorded as pairs of geographic coordinates and times are recorded in Julian days, beginning on January 1, 2017. Time points are generally spaced 4 hours apart, with gaps indicating missing data for those intervals. On average, approximately 2700 observations were recorded for each deer. Besides location tracking, the dataset includes information on the landscapes traversed by the deer, categorized by the National Land Cover Database (NLCD) classifications (Anderson, 1976). Additionally, each observation's timing is detailed down to the month, day, and hour, with the month variable being a critical indicator for various seasonal activities like breeding, fawning, nonbreeding, or post-fawning.

4.2 Notations

Let N represent the total number of deer trajectories. For each deer j, let n_j be the number of observed time points, $\boldsymbol{t}_j = \{t_{j1}, \ldots, t_{jn_j}\}$ the specific observation times, $\boldsymbol{Y}_j \in \mathbb{R}^{2\times n_j}$ the location coordinates, and $\boldsymbol{Z}_j \in \mathbb{R}^{n_j \times K}$ the side information. The side information includes

landscape and time-related variables. The landscape variable is categorized into 15 unique land cover types. Each category is transformed into a 15-dimensional one-hot encoded vector, with each dimension representing a distinct land cover type. Additionally, time-related variables like the month, day, and hour of observation contribute three more dimensions to $\boldsymbol{Z}_j$, resulting in a total of 18 dimensions ($K = 18$). Here, the complete dataset for deer j is represented by $\{\boldsymbol{Y}_j, \boldsymbol{Z}_j, \boldsymbol{t}_j\}$.

Our ultimate goal is to reconstruct the paths of each deer to ensure their movements are recorded at regular 4-hour intervals. To achieve this, $\boldsymbol{t}_j$ is expanded into an augmented series $\boldsymbol{t}_j^{\text{aug}} = \{t_{j1}^{\text{aug}}, \ldots, t_{jm_j}^{\text{aug}}\}$, with n_j^{aug} denoting the series length. Define $t_{j1}^{\text{aug}} = t_{j1}$, $t_{jn_j^{\text{aug}}}^{\text{aug}} = t_{jn_j}$, $\boldsymbol{t}_j \subseteq \boldsymbol{t}_j^{\text{aug}}$, and the interval between t_{ji}^{aug} and $t_{j,i-1}^{\text{aug}}$ as 4 hours. Additionally, the deer's location data for the augmented time points is denoted by $\boldsymbol{Y}_j^{\text{aug}} \in \mathbb{R}^{2\times n_j^{\text{aug}}}$, and the associated side information by $\boldsymbol{Z}_j^{\text{aug}}$. Time points in $\boldsymbol{t}_j^{\text{aug}}$ not found in the original $\boldsymbol{t}_j$ indicate intervals with missing location data we aim to reconstruct. Define $\boldsymbol{P}_j^{\text{aug}} \in \mathbb{R}^{2\times n_j^{\text{aug}}}$ as the mask for identifying missing versus observed data. Therefore, our task is to impute missing values in $\boldsymbol{Y}_j^{\text{aug}}$.

Evaluating the effectiveness of various imputation methods for $\boldsymbol{Y}_j^{\text{aug}}$ is challenging due to the absence of true missing data values, making direct evaluation infeasible. We address this challenge using a self-supervised learning approach. We first partition our dataset into training and testing subsets. Specifically, we randomly select 80% (331 individuals) to form the training dataset, denoted as $\mathcal{D}_{\text{train}}$. The remaining 20% (71 individuals) form the testing dataset, denoted as $\mathcal{D}_{\text{test}}$. For each deer j in $\mathcal{D}_{\text{test}}$, we randomly select p% of the time points from $\boldsymbol{t}_j$ and mark the corresponding location data $\boldsymbol{Y}_j$ and associated covariates $\boldsymbol{Z}_j$ as missing. The resulting datasets are defined as $\{\boldsymbol{Y}_j^{\text{test}}, \boldsymbol{Z}_j^{\text{test}}, \boldsymbol{t}_j, \boldsymbol{P}_j^{\text{test}}\}$, with $\boldsymbol{P}_j^{\text{test}}$ as the mask identifying observed versus missing data. We then use various methods to impute $\boldsymbol{Y}_j^{\text{test}}$ and evaluate their performance by comparing the imputed data with the true data of $\boldsymbol{Y}_j$. Let $\hat{\boldsymbol{Y}}_j^{\text{test}}$ denote the deterministic imputation of $\boldsymbol{Y}_j^{\text{test}}$. The Mean Absolute Error (MAE) is defined as:

$$\mathcal{L} = \frac{\sum_{j\in\mathcal{D}_{\text{test}}} \sum \left\{|\hat{\boldsymbol{Y}}_j^{\text{test}} - \boldsymbol{Y}_j| \odot (\boldsymbol{1} - \boldsymbol{P}_j^{\text{test}})\right\}}{\sum_{j\in\mathcal{D}_{\text{test}}} \sum (\boldsymbol{1} - \boldsymbol{P}_j^{\text{test}})}. \tag{19}$$

In probabilistic imputation, treat the imputed test matrix $\boldsymbol{Y}_j^{\text{test}}$, denoted by $\tilde{\boldsymbol{Y}}_j^{\text{test}}$, as a matrix of random variables following a specific probability distribution $\boldsymbol{F}_j^{\text{test}}$. To assess probabilistic imputation accuracy, we use the continuous ranked probability score (CRPS)(Matheson and Winkler, 1976). CRPS calculates the integral of the quantile loss, $\Lambda_\alpha(q, z) = (\alpha - \mathbb{I}_{z<q})(z - q)$,

for a scalar y and its estimated probability distribution F, across all quantile levels α in the interval $[0, 1]$:

$$\text{CRPS}\left(F, y\right) = \int_0^1 2\Lambda_\alpha \left(F^{-1}(\alpha), y\right) \mathrm{d}\alpha \tag{20}$$

If F is not tractable, we generate 100 samples to approximate it for each missing value, following Tashiro et al. (2021). We calculate quantile losses at discrete quantile levels, using increments of 0.05:

$$\text{CRPS}\left(F, y\right) \simeq \sum_{i=1}^{19} 2\Lambda_{i*0.05} \left(F^{-1}(i * 0.05), y\right) / 19 \tag{21}$$

If $\boldsymbol{Y}$ is a matrix of random variables with $\boldsymbol{F}$ as its elementwise probability distribution, where F_{ij} is the distribution for y_{ij}, the CRPS calculation is:

$$\text{CRPS}\left(\boldsymbol{F}, \boldsymbol{y}\right) = \frac{\sum_{i,j} \text{CRPS}(F_{i,j}^{-1}, y_{i,j})}{\sum_{i,j} |y_{i,j}|} \tag{22}$$

Therefore, the CRPS for our testing dataset is calculated as:

$$\mathcal{L} = \frac{\sum_{j \in \mathcal{D}_{\text{test}}} \sum \left\{\text{CRPS}(\boldsymbol{F}_j^{\text{test}}, \boldsymbol{Y}_j) \odot (\boldsymbol{1} - \boldsymbol{P}_j^{\text{test}})\right\}}{\sum_{j \in \mathcal{D}_{\text{test}}} \sum \{\boldsymbol{Y}_j^{\text{test}} \odot (\boldsymbol{1} - \boldsymbol{P}_j^{\text{test}})\}}, \tag{23}$$

4.3 Imputation using CTCRW

Johnson et al. (2008) introduced the continuous-time correlated random walk (CTCRW) model, specifically designed to analyze animal telemetry data collected at irregular time intervals. At its core, the model uses the continuous-time Ornstein-Uhlenbeck process to model the animal's movement velocity. To estimate movement parameters and predict unobserved locations, the model uses a state-space framework.

We define each animal's bivariate location data at time t as $\boldsymbol{\mu}(t) = [\mu_1(t), \mu_2(t)]'$. Define $d\boldsymbol{\mu}(t) = \boldsymbol{v}(t)dt$, where $\boldsymbol{v}(t)$ is the instantaneous rate of location change (velocity). For each coordinate axis ($c = 1, 2$), we define the Ornstien-Uhlenbeck (OU) process $v_c(t)$ as follows:

$$v_c(t + \Delta) = \gamma_c + e^{-\beta\Delta}[v_c(t) - \gamma_c] + \zeta_c(\Delta), \tag{24}$$

where γ_c is the mean velocity, β is an autocorrelation parameter, and $\zeta_c(\Delta)$ is a zero mean normal random variable with variance $\sigma^2[1 - \exp(-2\beta\Delta)]/2\beta$. The parameter σ controls the overall variability in velocity.

Using the velocity process, we obtain the continuous-time location process $\boldsymbol{\mu}(t)$ by integration as shown:

$$\boldsymbol{\mu}(t) = \boldsymbol{\mu}(0) + \int_0^t \boldsymbol{v}(u)du. \tag{25}$$

Equations 24 and 25 define the basic continuous-time correlated random walk model (CTCRW).

However, the continuous path of an animal can only be observed at sampled times. Assume locations $\boldsymbol{y}_i = [y_{1i}, y_{2i}]$ are measured at times $t_1, \ldots, t_n$, and that the true locations $\boldsymbol{\mu}_i = [\mu_{1i}, \mu_{2i}]'$ of the animal at each time t_i allow us to represent CTCRW within a state-space model framework.

$$y_{ci} = Z_i' \boldsymbol{\alpha}_{ci} + \epsilon_{ci} \tag{26}$$

$$\boldsymbol{\alpha}_{c,i+1} = T_i \boldsymbol{\alpha}_{ci} + \boldsymbol{\eta}_{ci}, \tag{27}$$

where $Z_i = [1, 0]'$, $\boldsymbol{\alpha}_{ci} = [\mu_{ci}, v_{ci}]'$, $\boldsymbol{\eta}_i = [\zeta_{ci}, \xi_{ci}]'$, $H_{ci} = H(x_i)$, where x_i is a known location quality covariate; and ξ_{ci} are normal errors with variance

$$V[\xi_{ci}] = \sigma^2/\beta^2\{\Delta_i - 2/\beta(1 - e^{-\beta\Delta_i}) + 1/(2\beta)(1 - e^{-2\beta\Delta_i})\}. \tag{28}$$

$$\mathbf{T}_i = \begin{bmatrix} 1 & \left(1 - e^{-\beta\Delta_i}\right)/\beta \\ 0 & e^{-\beta\Delta_i} \end{bmatrix} \tag{29}$$

$$\mathbf{Q}_{ci} = \begin{Bmatrix} V\left[\xi_{ci}\right] & C\left[\xi_{ci}, \zeta_{ci}\right] \\ C\left[\xi_{ci}, \zeta_{ci}\right] & V\left[\zeta_{ci}\right] \end{Bmatrix} \tag{30}$$

$$C[\zeta_{ci}, \xi_{ci}] = \sigma^2/(2\beta^2)(1 - 2e^{-\beta\Delta_i} + e^{-2\beta\Delta_i}) \tag{31}$$

We use the maximum likelihood method for parameter estimation and location prediction. A key advantage of the Kalman filter and smoother (KFS) is its natural handling of missing observations. To estimate the value of $\boldsymbol{\alpha}_{ci}$ at a time t_i with no observed location, simply augment the dataset with $y_{ci} = \text{NA}$ for that time. The KFS automatically filters and smooths the data, providing estimates for location $\hat{\boldsymbol{\alpha}}_{ci}$ and variance $\hat{V}_{ci}$. We implement this using R Statistical Software (v4.2.1) (R Core Team, 2022) and the R package "crawl" (Johnson and London, 2018).

4.4 Imputation using CSDI

4.4.1 Preparation for Model Input

This section explains the preparation of training data for the CSDI model, using deer trajectory data from the dataset $[\{\boldsymbol{Y}_j, \boldsymbol{Z}_j, \boldsymbol{t}_j\}, j \in \mathcal{D}_{\text{train}}]$. Direct use of $\{\boldsymbol{Y}_j, \boldsymbol{Z}_j, \boldsymbol{t}_j\}$ poses three challenges.

First, the temporal length of each $\boldsymbol{Y}_j$ varies, whereas the model requires inputs of uniform length. Second, the model finds the temporal length of $\boldsymbol{Y}_j$ excessively long, as it focuses on attention between any two time points, while animal movements usually lack long-term dependencies. Lastly, treating each of the 331 deer trajectories as an individual training sample fails to produce a sufficiently large dataset for effective training.

To tackle these challenges, we partition each dataset $\{\boldsymbol{Y}_j, \boldsymbol{Z}_j, \boldsymbol{t}_j\}$ into shorter segments along the temporal dimension using a fixed window size L. In practice, L is set to 72. This method ensures consistent model input and generates more training data. The partition process is as follows:

$$\boldsymbol{X}_{jk} = \boldsymbol{Y}_j\big[1:2, (k-1)\cdot L+1 : k\cdot L\big] \in \mathbb{R}^{2\times L} \tag{32}$$

$$\boldsymbol{V}_{jk} = \boldsymbol{Z}_j\big[(k-1)\cdot L+1 : k\cdot L, 1:K\big] \in \mathbb{R}^{L\times K} \tag{33}$$

$$\boldsymbol{M}_{jk} = \mathbf{1} \in \mathbb{R}^{2\times L} \tag{34}$$

$$\boldsymbol{s}_{jk} = \{t_{j,(k-1)\cdot L+1} - t_{j,(k-1)\cdot L}, \ldots, t_{j,k\cdot L} - t_{j,k\cdot L-1}\} \in \mathbb{R}^{L} \tag{35}$$

where $k = 1, 2, \ldots, \lceil \frac{n_j}{L} \rceil$. Each $\boldsymbol{X}_{jk}$ or $\boldsymbol{V}_{jk}$ is created by selecting a non-overlapping subset of $\boldsymbol{Y}_j$ or $\boldsymbol{Z}_j$, starting from $(j-1)\cdot L+1$ to $j\cdot L$. $\boldsymbol{s}_{jk}$ denotes the time interval between two consecutive observations. Thus, we divide each deer's dataset $\{\boldsymbol{Y}_j, \boldsymbol{Z}_j, \boldsymbol{t}_j\}$ into multiple samples $\{\boldsymbol{X}_{jk}, \boldsymbol{M}_{jk}, \boldsymbol{V}_{jk}, \boldsymbol{s}_{jk}\}$. We partition each deer's trajectory in the training dataset to form $\mathcal{C}_{\text{train}} = \big[\{\boldsymbol{X}_{jk}, \boldsymbol{M}_{jk}, \boldsymbol{V}_{jk}, \boldsymbol{s}_{jk}\}, k \in \{1, \ldots, \lceil \frac{n_j}{L} \rceil\}, j \in D_{\text{train}}\big]$.

Another challenge is the significant variability in location data among and within deer movements, causing large variations in the model input $\boldsymbol{X}_{jk}$. This variability may lead to issues like model overfitting, where it memorizes specific trajectories, and distribution shift problems when applied to different regions. To address this, we apply min-max normalization to each $\boldsymbol{X}_{jk}$ within $\mathcal{C}_{\text{train}}$. This normalization technique scales each time series $\boldsymbol{X}_{jk}$ to a consistent range from 0 to 1 along the temporal dimension, improving the models training stability and generalization across regions. Details of the min-max normalization algorithm are provided in Algorithm 3.

It's important to note that normalization may lead to the loss of original spatial location details. To address this, the model includes landscape variables for each location to consider the spatial environment.

Algorithm 3 Min-Max Normalization

Input: a data sample $\boldsymbol{X} \in \mathbb{R}^{2\times L}$, a mask $\boldsymbol{M} \in \{0,1\}^{2\times L}$

Output: a normalized data sample $\boldsymbol{X}^{\text{norm}} \in \mathbb{R}^{2\times L}$, minimum value $\mathbf{min} \in \mathbb{R}^2$, maximum value $\mathbf{max} \in \mathbb{R}^2$.

Apply mask to $\boldsymbol{X}$ to obtain observed data $\boldsymbol{X}^{\text{obs}} = \boldsymbol{X} \odot \boldsymbol{M}$

Calculate the number of observed data points per coordinate: $\boldsymbol{n}[i] = \sum_{j=1}^{L} \boldsymbol{M}[i,j]$, $i = \{1,2\}$.

Calculate the minimum and maximum of observed values per coordinate:

$\mathbf{min}[i] = \min_{j:\boldsymbol{M}[i,j]=1} \boldsymbol{X}^{\text{obs}}[i,j]$, $i = \{1,2\}$.

$\mathbf{max}[i] = \max_{j:\boldsymbol{M}[i,j]=1} \boldsymbol{X}^{\text{obs}}[i,j]$, $i = \{1,2\}$.

Normalize the observed data: $\boldsymbol{X}^{\text{norm}}[i,j] = (\boldsymbol{X}[i,j] - \mathbf{min}[i])/(\mathbf{max}[i] - \mathbf{min}[i])$, for each observed value $i = \{1,2\}$, $j = \{1,\ldots,L\}$

Apply the mask to the normalized data to preserve missing values: $\boldsymbol{X}^{\text{norm}} = \boldsymbol{X}^{\text{norm}} \odot \boldsymbol{M}$

return $\boldsymbol{X}^{\text{norm}}$, **min**, **max**

4.4.2 Model Training

We use Algorithm 1 for model training. The distribution $q(\cdot)$ randomly selects samples from the training dataset $\mathcal{C}_{\text{train}}$, while the masking strategy $\mathcal{T}$ randomly selects a proportion of time points to mask the corresponding location data. This strategy replicates the pattern of missing data in animal movement, where an animal's absence at certain times means both location coordinates are missing. Algorithm 4 describes the implementation of the random masking strategy. The training and model hyperparameters follow the configuration in Tashiro et al. (2021).

4.4.3 Model Imputation

After training, we apply the model to the testing dataset $\{\boldsymbol{Y}_j^{\text{test}}, \boldsymbol{Z}_j^{\text{test}}, \boldsymbol{t}_j, \boldsymbol{P}_j^{\text{test}}\}$. The imputation process involves preprocessing the data for the model, running the model to get the output, and then applying inverse preprocessing to revert the data to its original format. Detailed steps of this process are provided in Algorithm 5.

We apply Algorithm 5 to the set $\{\boldsymbol{Y}_j^{\text{test}}, \boldsymbol{Z}_j^{\text{test}}, \boldsymbol{t}_j, \boldsymbol{P}_j^{\text{test}}\}$ and generate B imputed samples, denoted by $\{\boldsymbol{Y}_j^{\text{test},b}, b = 1,\ldots,B\}$. In practice, we set $B = 100$. To derive a single determin-

Algorithm 4 Separate observed values into conditional observations and imputation target

Input: a training sample $\boldsymbol{X}_0 \in \mathbb{R}^{2\times L}$, a mask $\boldsymbol{M} \in \{0,1\}^{2\times L}$.
Output: a conditional observation $\boldsymbol{X}_0^{\text{co}}$, an imputation target $\boldsymbol{X}_0^{\text{ta}}$.
Initialize conditional observation mask $\boldsymbol{M}^{\text{co}}$ as $\boldsymbol{M}$.
Randomly select a ratio r from Uniform($\{0.2, 0.5, 0.8\}$).
for each time step $l = 1$ to L **do**
    Generate a random number q from Uniform$(0, 1)$.
    if $q < r$ **then**
        Set $\boldsymbol{M}^{\text{co}}[1:2, l] = \boldsymbol{0}$.
    end if
end for
Compute conditional observation $\boldsymbol{X}_0^{\text{co}} = \boldsymbol{X}_0 \odot \boldsymbol{M}^{\text{co}}$.
Compute imputation target $\boldsymbol{X}_0^{\text{ta}} = \boldsymbol{X}_0 \odot (\boldsymbol{M} - \boldsymbol{M}^{\text{co}})$.

Algorithm 5 Animal Trajectory Imputation

Input: $\{\boldsymbol{Y}_j, \boldsymbol{Z}_j, \boldsymbol{t}_j, \boldsymbol{P}_j\}$, trained denoising function $\boldsymbol{\epsilon}_\theta$, number of samples B.
Output: $\{\boldsymbol{Y}_j^b\}$, $b = 1, \ldots, B$
for $b = 1$ **to** B **do**
    Follow the input processing procedure, which includes partitioning and min-max normalization as described in Section 4.4.1, to divide $\{\boldsymbol{Y}_j, \boldsymbol{Z}_j, \boldsymbol{t}_j\}$ into sequences $\{\boldsymbol{X}_{jk}, \boldsymbol{M}_{jk}, \boldsymbol{V}_{jk}, \boldsymbol{s}_{jk}\}$.
    for $k = 1, 2, \ldots, \lceil \frac{n_j}{L} \rceil$ **do**
        Set $\boldsymbol{M}_{jk} = \boldsymbol{P}_j[1:2, (k-1)\cdot L + 1 : k \cdot L]$.
        Apply Algorithm 2 using the inputs $\{\boldsymbol{X}_{jk}, \boldsymbol{M}_{jk}, \boldsymbol{V}_{jk}, \boldsymbol{s}_{jk}\}$ and set $\boldsymbol{X}_{jk}^b$ as the imputed target.
        Reverse the min-max normalization for $\boldsymbol{X}_{jk}^b$ by recalculating: $\boldsymbol{X}_{jk}^b = \boldsymbol{X}_{jk}^b \cdot (\textbf{max} - \textbf{min}) + \textbf{min}$.
    end for
    Reassemble the partitions to obtain $\boldsymbol{Y}_j^b$.
end for

istic imputation $\hat{\boldsymbol{Y}}_j^{\text{test}}$, we average all B samples, calculated as $\hat{\boldsymbol{Y}}_j^{\text{test}} = \frac{1}{B}\sum_{b=1}^{B} \boldsymbol{Y}_j^{\text{test},b}$. For probabilistic imputation, these 100 samples collectively constitute the empirical distribution of $\tilde{\boldsymbol{Y}}_j^{\text{test}}$.

4.5 Model Comparison

In this analysis, we compare the CSDI method with two traditional approaches: Linear nterpolation and Continuous-Time Correlated Random Walk (CTCRW), using the testing dataset. Linear interpolation is a straightforward method that draws a straight line between two known points to estimate missing values for any intervening points. This method is advantageous due to its simplicity and minimal computational demands. Linear interpolation does not rely on any model and is applied directly to the testing data. Similarly, CTCRW builds a parametric model from observed trajectory data and is directly applicable to the testing data. In contrast, the CSDI method requires initial training on the training dataset before application to the testing dataset.

To evaluate the testing dataset, we established three scenarios with varying levels of missing data: 20%, 50%, and 80%. These methods were applied to each scenario, and their performance is detailed in Table 1. The results show that CSDI consistently outperforms the other methods in all scenarios in terms of MAE. In terms of CRPS, CSDI also outperforms CTCRW. Since linear interpolation is a deterministic imputation method, it does not provide CRPS values. Additionally, the MAE difference between CSDI and the second-best method grows more significant as the amount of missing data increases. These results highlight the superior robustness and accuracy of the CSDI approach compared to other methods.

Visualizations of the imputation results for a single deer from the test data, across various missing data ratios using both interpolation and the CSDI method, are presented. These visualizations are shown in Figure 1, 2, 3, 4, 5, 6, 7, 8, and 9. Each figure is organized into rows of subplots, with each row depicting the deer's trajectories over different time segments. Within each row, the left subplot displays the X coordinate (longitude) and the right subplot shows the Y coordinate (latitude), both plotted against time (in Julian days).

In each subplot, black "x" marks represent observed data points, while unfilled black "o" circles indicate evaluation data points, and solid green lines depict imputed values. For the CSDI method, shaded green areas represent the 5% and 95% quantiles of probabilistic impu-

Scenario	Method	Metrics	
		MAE	**CRPS**
$p = 20\%$	**CSDI**	123	$4.4e^{-5}$
	CTCRW	137	$4.6e^{-5}$
	Interpolation	141	/
$p = 50\%$	**CSDI**	136	$4.8e^{-5}$
	CTCRW	155	$5.1e^{-5}$
	Interpolation	157	/
$p = 80\%$	**CSDI**	164	$6.0e^{-5}$
	CTCRW	187	$6.2e^{-5}$
	Interpolation	183	/

Table 1: Performance Comparison Across Scenarios and Methods

tations.

Although interpolation follows the general direction of movement between observed points, it fails to capture subtle patterns or variations in the data, potentially oversimplifying the animal's path. The imputed path from CSDI is smoother and captures the underlying periodicity and trends in the deer's movement more effectively. The green shaded areas suggest that the CSDI method offers a measure of uncertainty or confidence intervals around the imputations, absent in the interpolation method.

4.6 Trajectory Imputation Using CSDI

In this section, we use the CSDI method to impute $\boldsymbol{Y}_j^{\text{aug}}$, following the procedure outlined for deer trajectory imputation in the test dataset, as detailed in Section 4.4.3. For any j in $\mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{test}}$, we apply Algorithm 5 to the set $\{\boldsymbol{Y}_j^{\text{aug}}, \boldsymbol{Z}_j^{\text{aug}}, \boldsymbol{t}_j^{\text{aug}}, \boldsymbol{P}_j^{\text{aug}}\}$ and obtain B imputed samples, denoted by $\{\boldsymbol{Y}_j^{\text{aug},b}, b = 1, \ldots, B\}$. For deterministic imputation, we calculate the average of all B samples, given by $\hat{\boldsymbol{Y}}_j^{\text{aug}} = \frac{1}{B}\sum_{b=1}^{B} \boldsymbol{Y}_j^{\text{aug},b}$. For probabilistic imputation, we calculate the 5% and 95% quantiles of the samples to estimate the "range". Visualizations of the imputation for one deer are presented in Figure 10.

5 Conclusion

Our study introduces the Conditional Score-based Diffusion Model (CSDI) for animal trajectory imputation. The model can handle irregular time intervals and performs probabilistic imputation. Additionally, unlike traditional models that are applied to individual animals, this model can learn various movement patterns and conditions from a large dataset of hundreds of deer. We provide an end-to-end example from collecting animal trajectory data, training the CSDI model, comparing its performance with other methods, to finally filling the gaps in missing trajectories. In tests using Wisconsin female deer trajectory data, we showed that CSDI outperforms two traditional methods: interpolation and the continuous time random walk method in both deterministic and probabilistic imputation. Once well-trained, the model can be directly applied to the same species of animals without further training, offering a convenient tool for improved animal trajectory imputation. Since the model learns the patterns of animal movement, it can do more than imputation; for example, it can perform predictions and other analyses based on animal movement. Its potential is not limited.

References

Anderson, J. R. (1976). *A land use and land cover classification system for use with remote sensor data*, volume 964. US Government Printing Office.

Breed, G. A., Costa, D. P., Jonsen, I. D., Robinson, P. W., and Mills-Flemming, J. (2012). State-space methods for more completely capturing behavioral dynamics from animal tracks. *Ecological Modelling*, 235:49–58.

Buderman, F. E., Hooten, M. B., Ivan, J. S., and Shenk, T. M. (2016). A functional model for characterizing long-distance movement behaviour. *Methods in Ecology and Evolution*, 7(3):264–273.

Hirakawa, T., Yamashita, T., Tamaki, T., Fujiyoshi, H., Umezu, Y., Takeuchi, I., Matsumoto, S., and Yoda, K. (2018). Can ai predict animal movements? filling gaps in animal trajectories using inverse reinforcement learning. *Ecosphere*, 9(10):e02447.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

Holzmann, H., Munk, A., Suster, M., and Zucchini, W. (2006). Hidden markov models for circular and linear-circular time series. *Environmental and Ecological Statistics*, 13:325–347.

Johnson, D. and London, J. (2018). crawl: an r package for fitting continuous-time correlated random walk models to animal movement data. *Zenodo*, 10.

Johnson, D. S., London, J. M., Lea, M.-A., and Durban, J. W. (2008). Continuous-time correlated random walk model for animal telemetry data. *Ecology*, 89(5):1208–1215.

Li, X., Sindihebura, T. T., Zhou, L., Duarte, C. M., Costa, D. P., Hindell, M. A., McMahon, C., Muelbert, M. M., Zhang, X., and Peng, C. (2021). A prediction and imputation method for marine animal movement data. *PeerJ Computer Science*, 7:e656.

Matheson, J. E. and Winkler, R. L. (1976). Scoring rules for continuous probability distributions. *Management science*, 22(10):1087–1096.

McKellar, A. E., Langrock, R., Walters, J. R., and Kesler, D. C. (2015). Using mixed hidden markov models to examine behavioral states in a cooperatively breeding bird. *Behavioral Ecology*, 26(1):148–157.

Patterson, T. A., Basson, M., Bravington, M. V., and Gunn, J. S. (2009). Classifying movement behaviour in relation to environmental conditions using hidden markov models. *Journal of Animal Ecology*, 78(6):1113–1123.

R Core Team (2022). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rew, J., Park, S., Cho, Y., Jung, S., and Hwang, E. (2019). Animal movement prediction based on predictive recurrent neural network. *Sensors*, 19(20):4411.

Tashiro, Y., Song, J., Song, Y., and Ermon, S. (2021). Csdi: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in Neural Information Processing Systems*, 34:24804–24816.

Towner, A. V., Leos-Barajas, V., Langrock, R., Schick, R. S., Smale, M. J., Kaschke, T., Jewell, O. J., and Papastamatiou, Y. P. (2016). Sex-specific and individual preferences for hunting strategies in white sharks. *Functional Ecology*, 30(8):1397–1407.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

A Details of Denoising Diffusion Probabilistic Models

This section details the denoising diffusion probabilistic models discussed in Section 3.1. Introduced in Equations 1 and 3, diffusion models are latent variable models comprising two main processes: the forward process, which gradually adds noise to the data, and the reverse process, which removes this noise to either reconstruct the original data or generate new data. The objective is to adjust the parameters θ to maximize the likelihood's variational lower bound (ELBO), expressed as $p_\theta(\mathbf{X}_{0:T})$:

$$\mathbb{E}_{q(\mathbf{X}_0)}\left\{\log p_\theta(\mathbf{X}_0)\right\} \geq \mathbb{E}_{q(\mathbf{X}_0,\mathbf{X}_1,\ldots,\mathbf{X}_T)}\left\{\log p_\theta(\mathbf{X}_{0:T}) - \log q(\mathbf{X}_{1:T}|\mathbf{X}_0)\right\} := \text{ELBO}. \tag{36}$$

Ho et al. (2020) developed denoising diffusion probabilistic models (DDPM) and introduced a parametrization that simplifies the Evidence Lower Bound (ELBO). Using this parametrization, they showed that the ELBO satisfies:

$$-\text{ELBO} = c + \sum_{t=1}^{T} \kappa_t \mathbb{E}_{\mathbf{X}_0\sim q(\mathbf{X}_0),\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})} \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\alpha_t}\mathbf{X}_0 + (1-\alpha_t)\boldsymbol{\epsilon}, t\right)\right\|_2^2, \tag{37}$$

Here, c is a constant, and $\{\kappa_{1:T}\}$ are positive coefficients that depend on $\alpha_{1:T}$ and $\beta_{1:T}$. This formulation implies that training the diffusion process requires minimizing the ELBO.

Furthermore, Ho et al. (2020) found that minimizing an unweighted version of the ELBO yields high-quality samples:

$$\min_\theta \mathcal{L}(\theta) := \min_\theta \mathbb{E}_{\mathbf{X}_0\sim q(\mathbf{X}_0),\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I}),t} \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\alpha_t}\mathbf{X}_0 + (1-\alpha_t)\boldsymbol{\epsilon}, t\right)\right\|_2^2, \tag{38}$$

Here, the function $\boldsymbol{\epsilon}_\theta$ is trained to estimate the noise $\boldsymbol{\epsilon}$ in the noisy input. Once trained, the model samples $\mathbf{X}_0$ using the reverse diffusion equation 3.

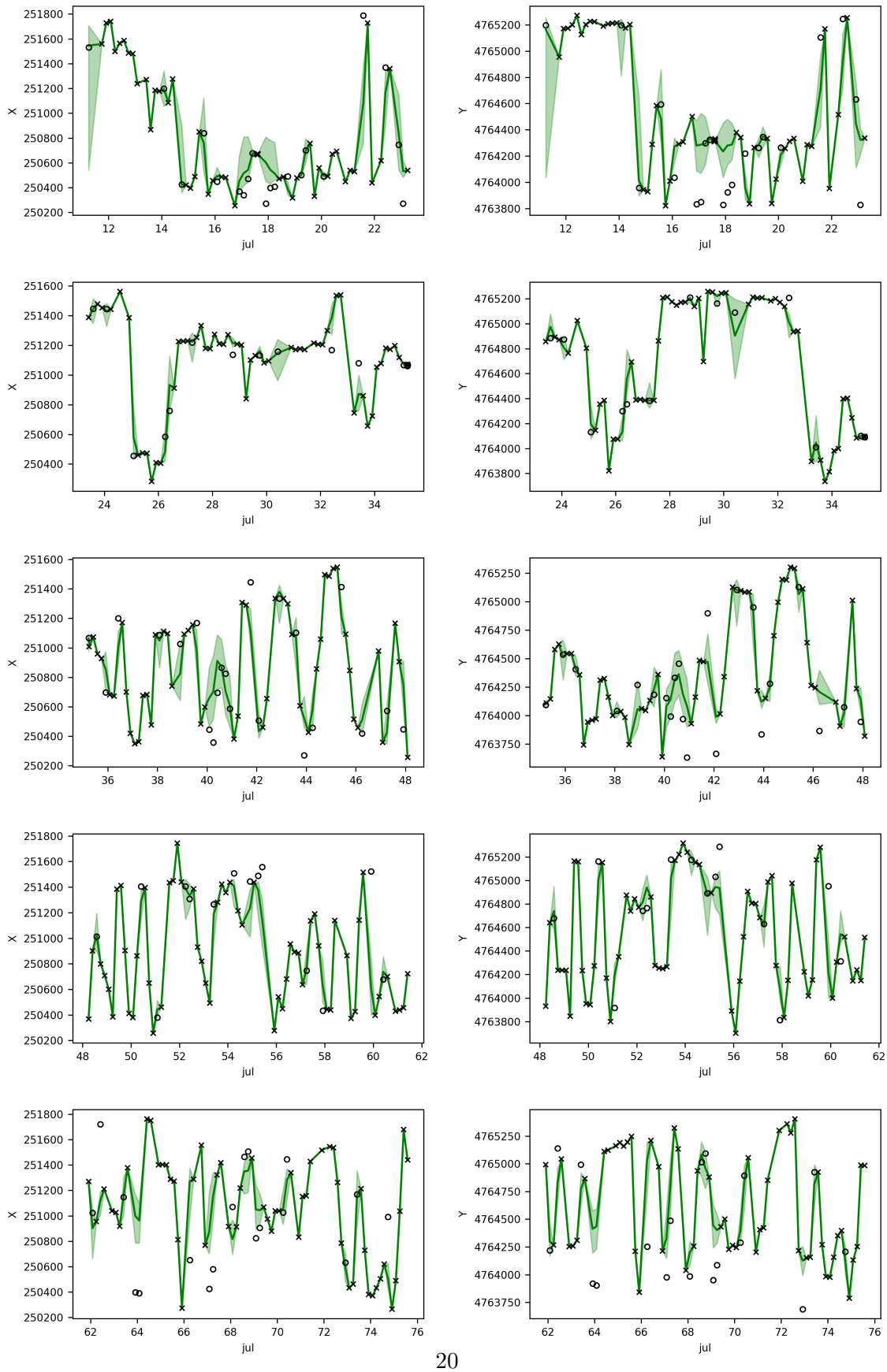

Figure 1: CSDI, p=20%

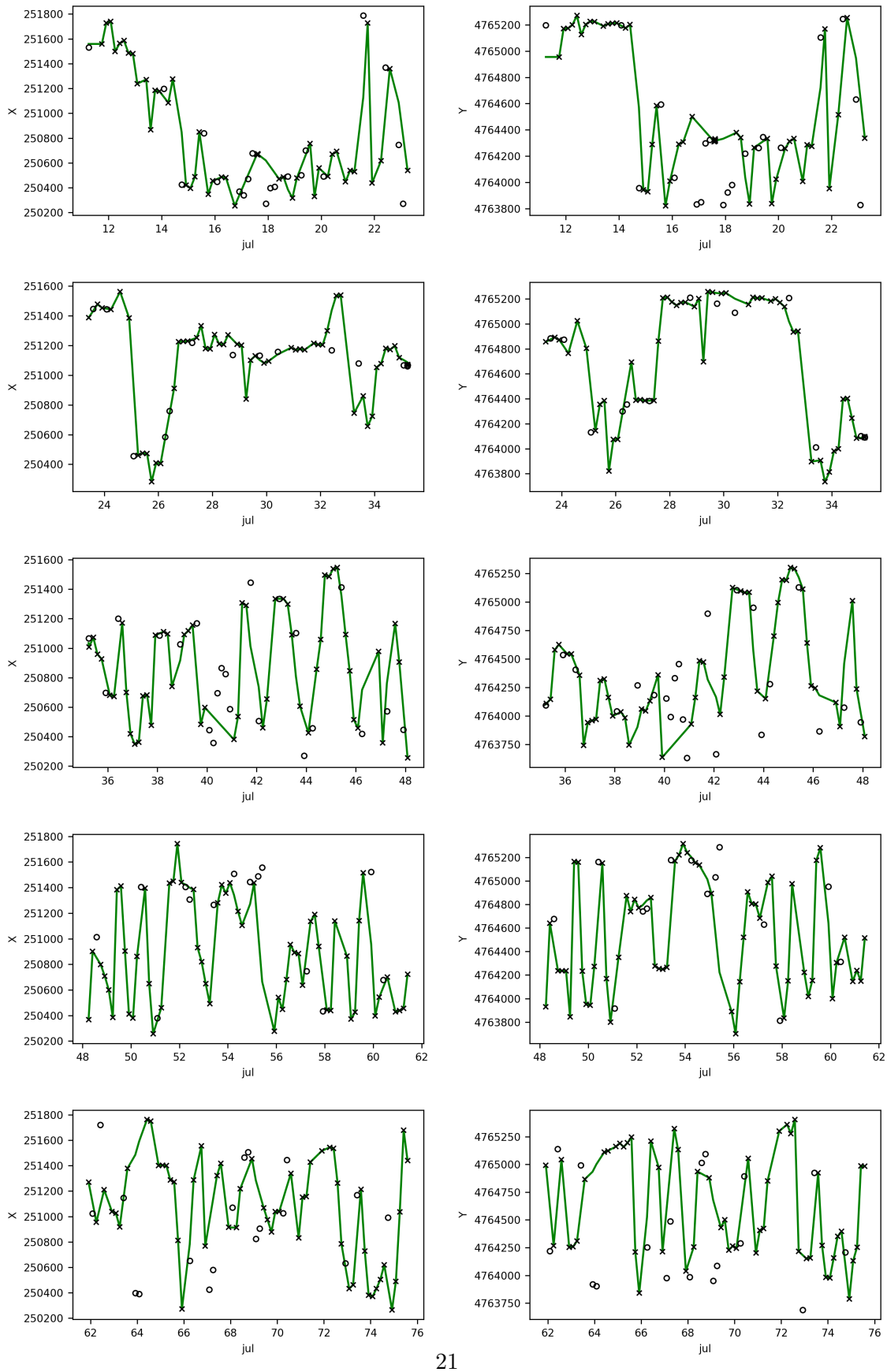

Figure 2: Interpolation, p=20%

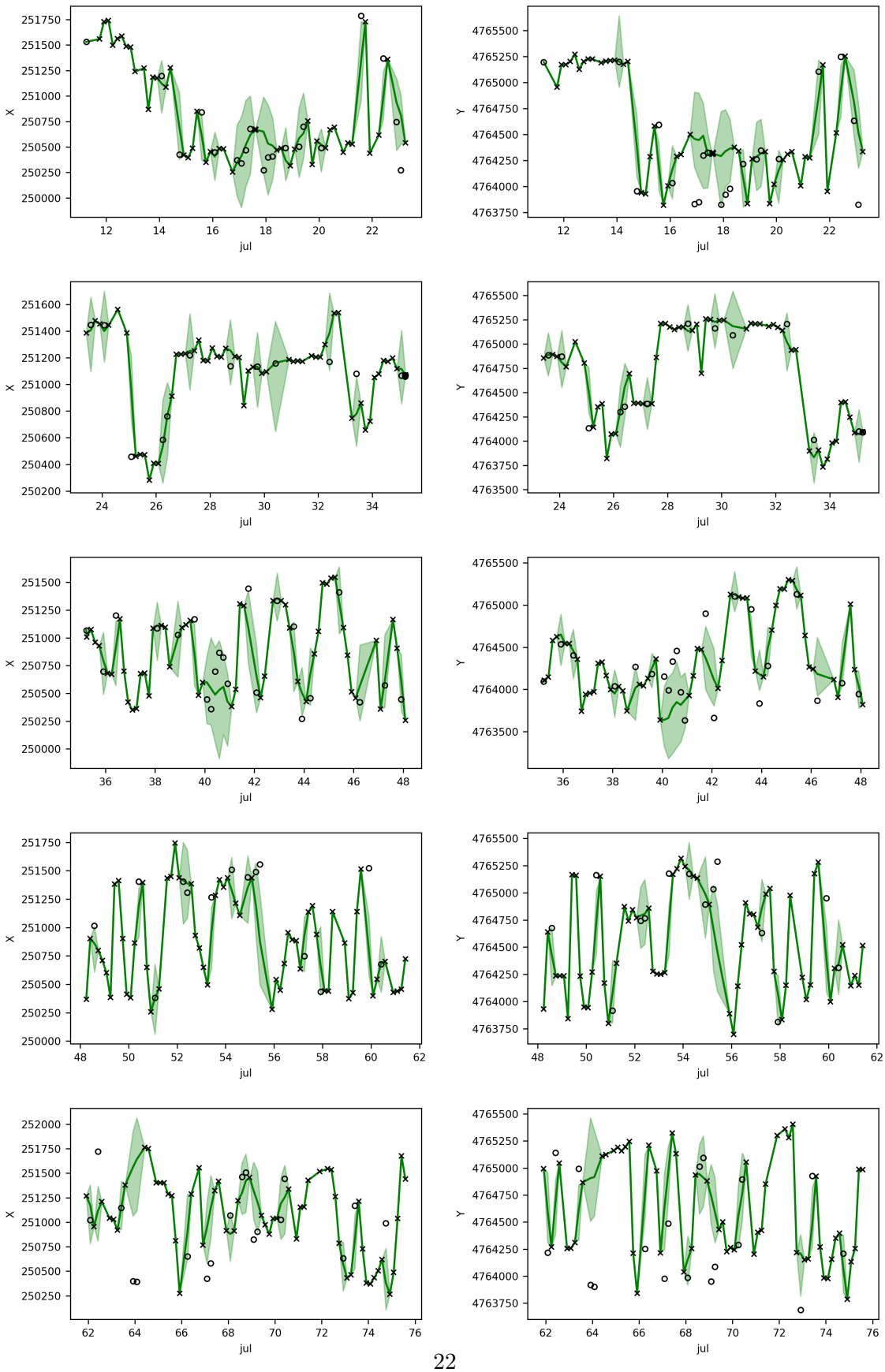

Figure 3: CTCRW, p=20%

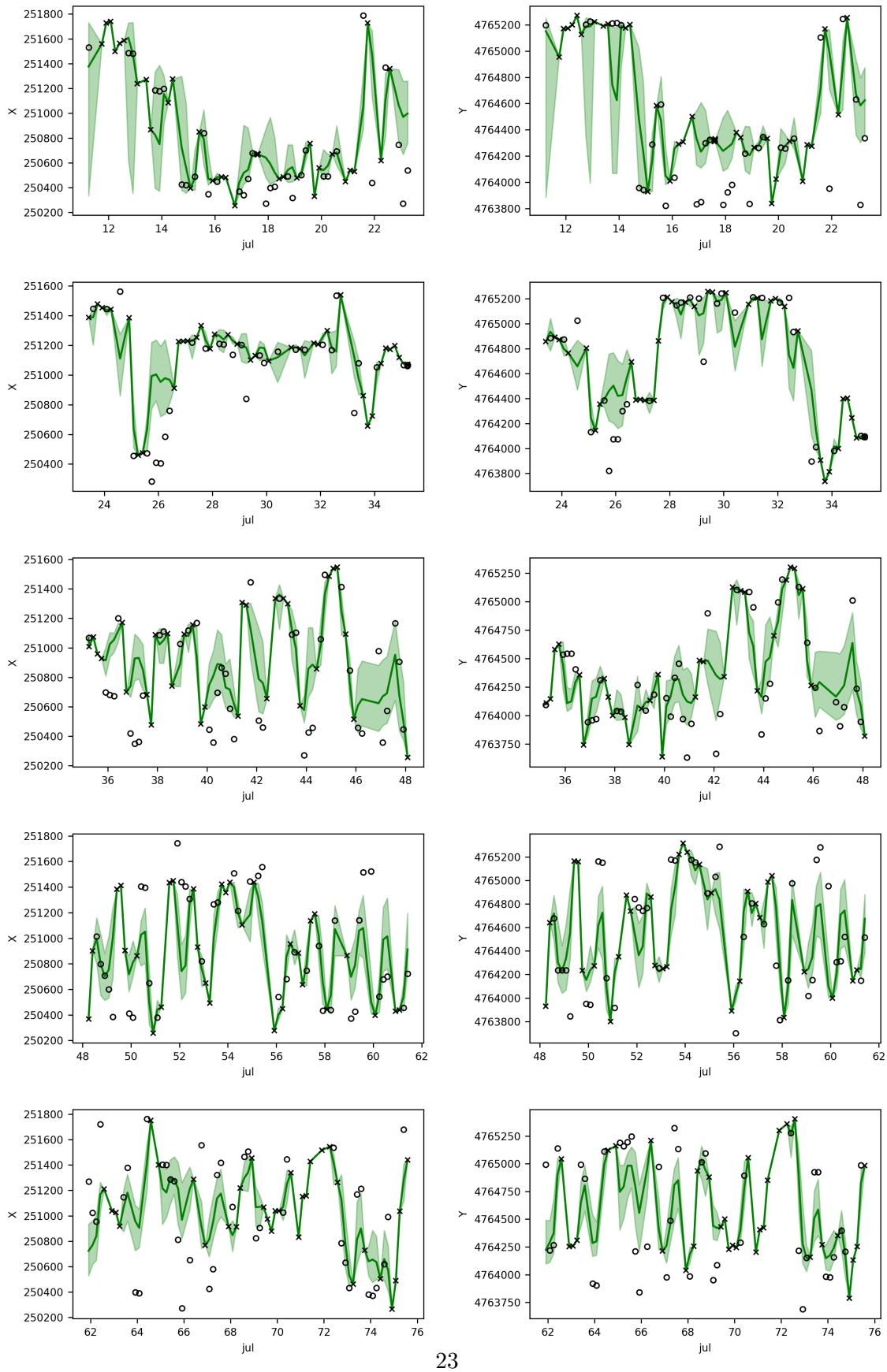

Figure 4: CSDI, p=50%

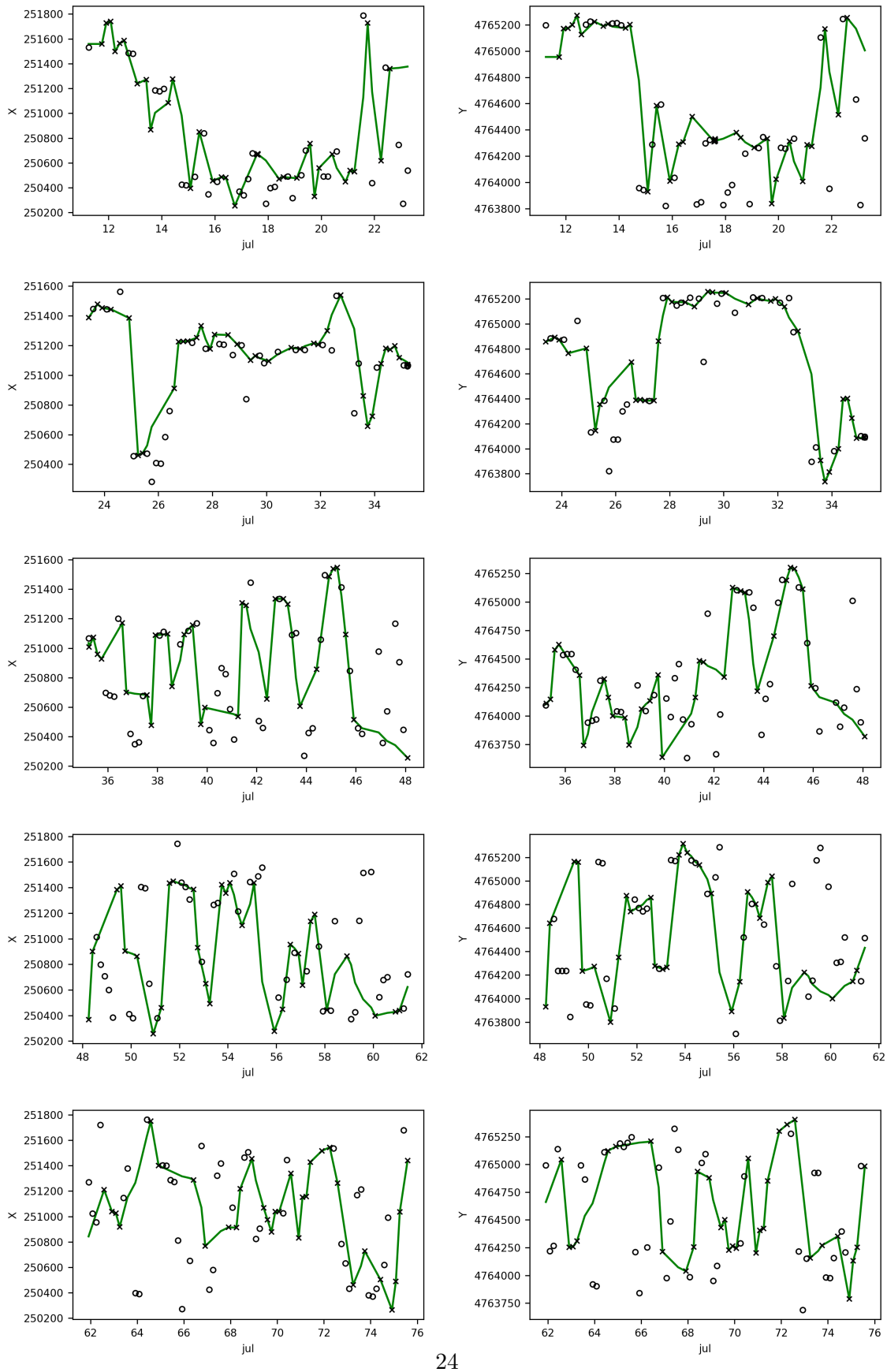

Figure 5: Interpolation, p=50%

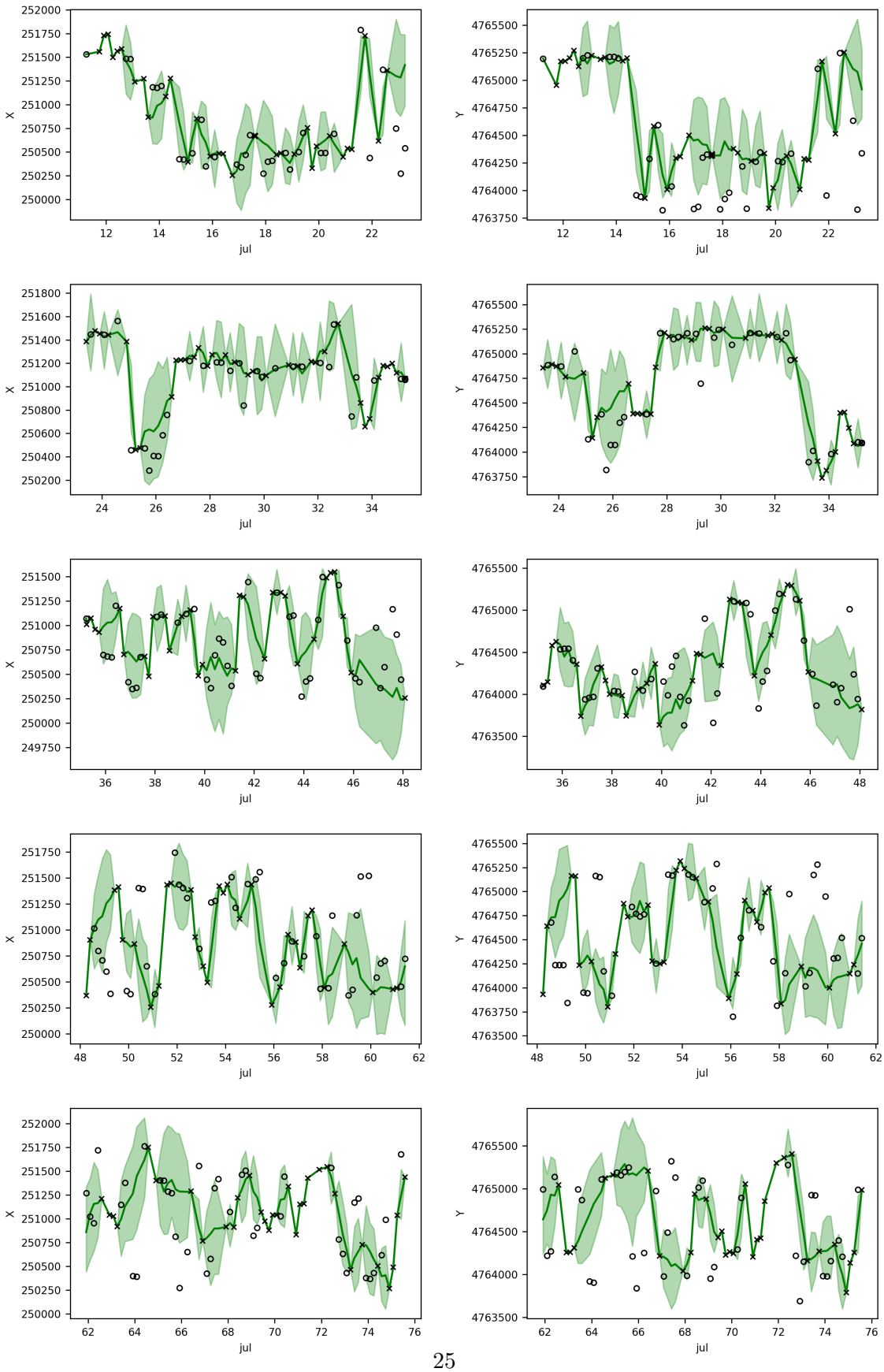

Figure 6: CTCRW, p=50%

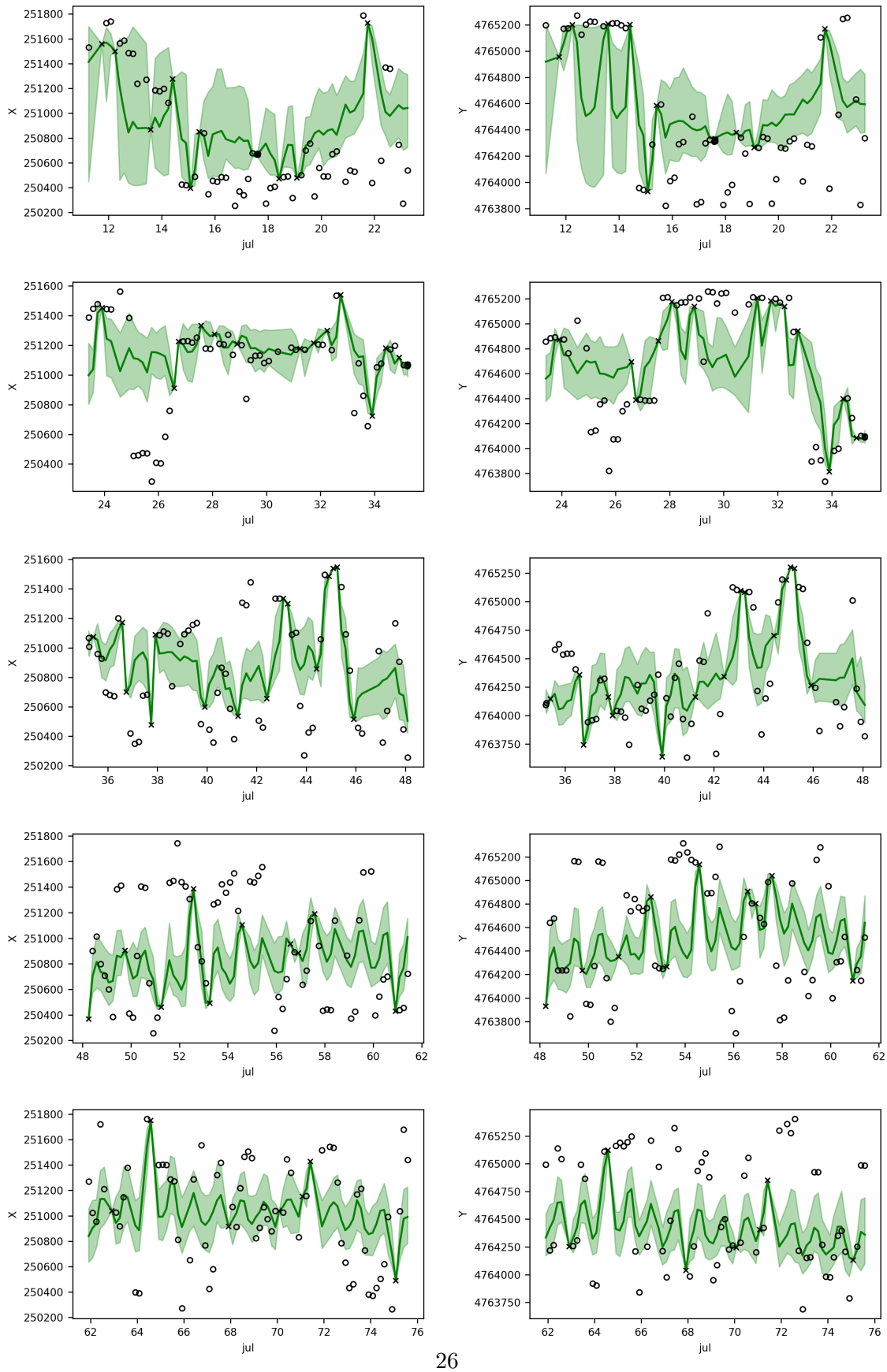

Figure 7: CSDI, p=80%

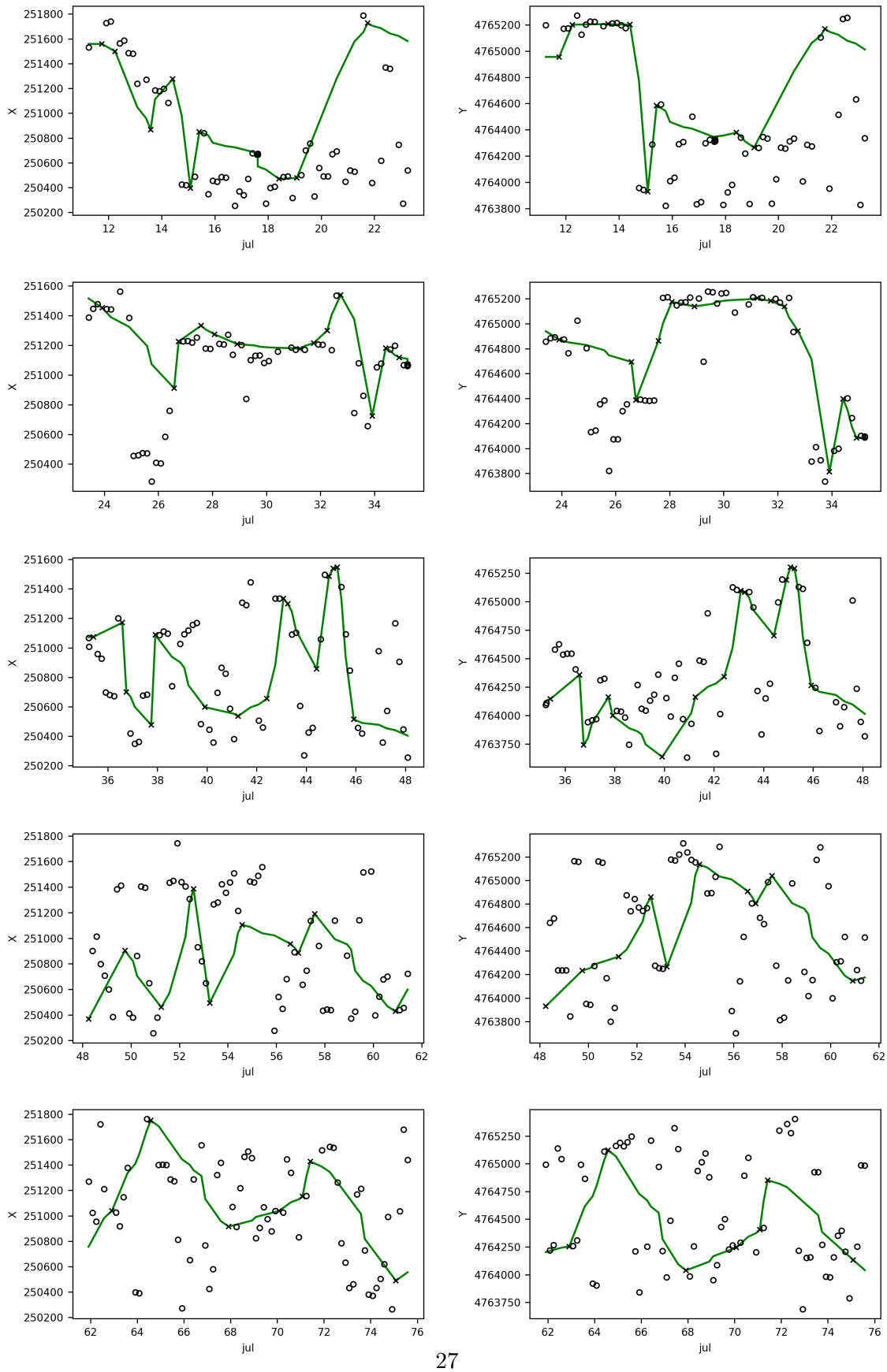

Figure 8: Interpolation, p=80%

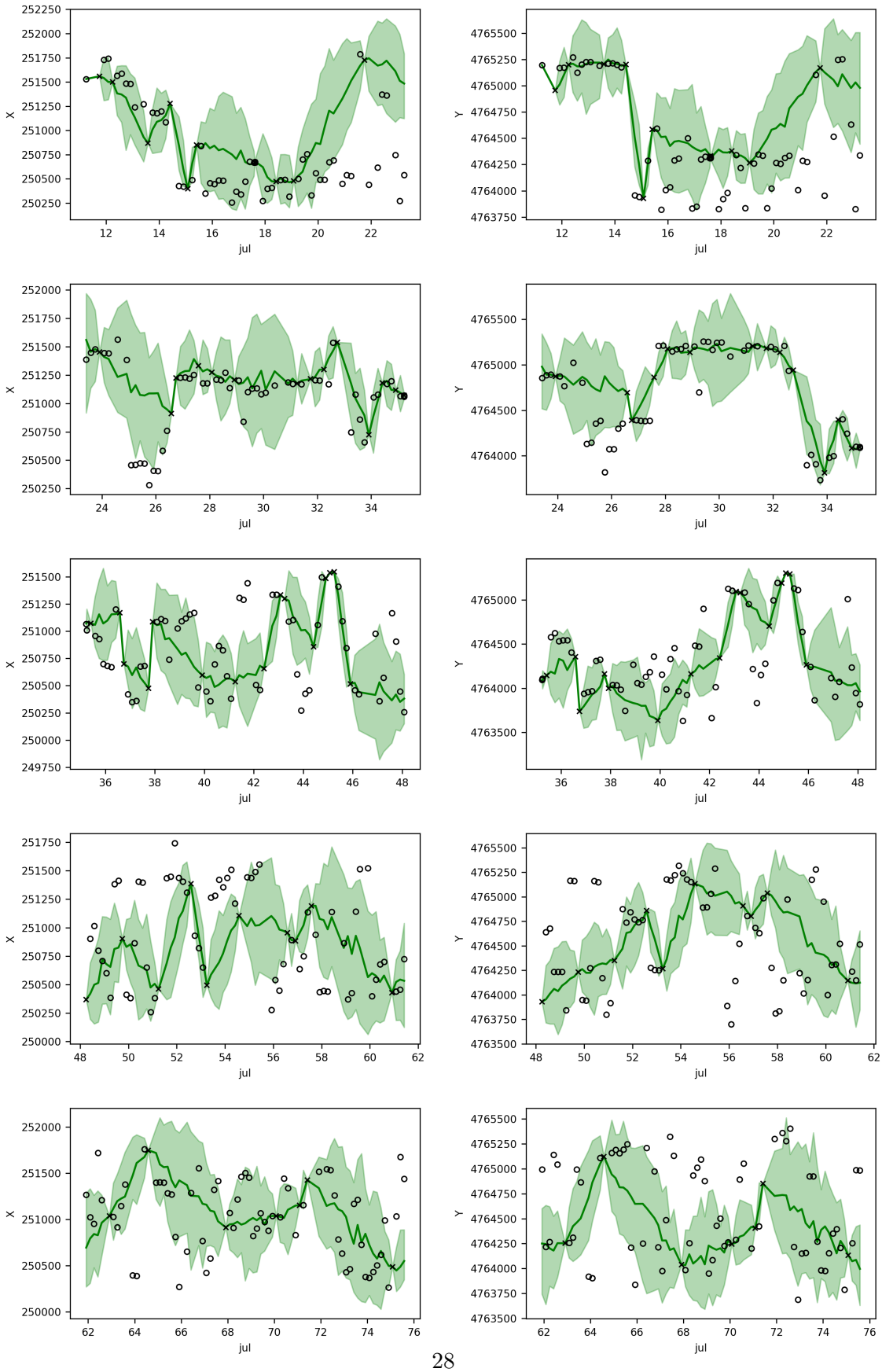

Figure 9: CTCRW, p=80%

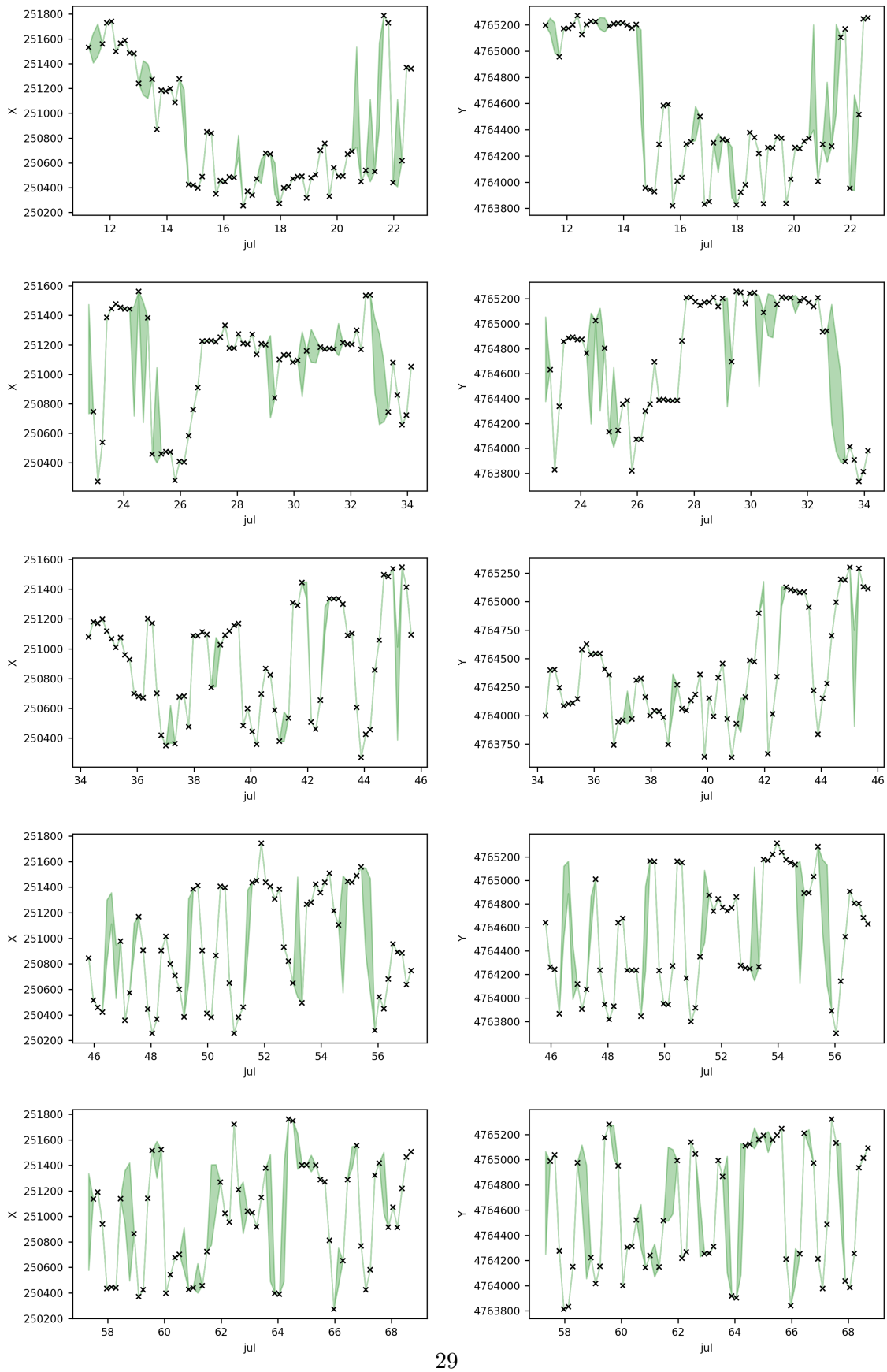

Figure 10: CSDI for deer trajectory imputation at 4-hour interval