technische universität
dortmund

fakultät für informatik

Master thesis

Filling Gaps in Incomplete Satellite Images with Compressable Spatio-Temporal Random Fields

Raphael Fischer

March 28, 2019

Gutachter:
Prof. Dr. Katharina Morik
Dr. Nico Piatkowski

Computer Science VIII
Artificial Intelligence Group
TU Dortmund

Contents

Acronyms

AG *Asymmetric Gaussian* 10

BP *Belief propagation* 14, 16

DL *Double logistic* 10

DTW *Dynamic time warping* 48

EM *Expectation-maximization* 18, 24, 25, 29, 30, 36–39, 41, 42, 47

ERR *Error* 35, 37–41, 45

GD *Gradient descent* 17, 22, 25, 30, 48

IBS *Intensity-based similarity* 25, 26, 37, 40

JSON *JavaScript Object Notation* 31

JT *Junction tree* 15, 16

L2Z *L2 Zero* 25, 37, 40

LBP *Loopy belief propagation* 15, 16, 30, 48

LIDAR *Light detection and ranging* 4

LTP *Log transition probability* 26, 27, 37–41

MAE *Mean absolute error* 35–42, 45

MAJA *MACCS-ATCOR Joint Algorithm* 32, 33

MAP *Maximum a posteriori* 14, 16–18, 21, 22, 24, 25, 30, 41, 48

MCMC *Markov chain Monte Carlo* 15

ML *Maximum likelihood* 16, 17

MRF *Markov random field* 11–14, 17, 18, 20–22, 24, 25, 27, 31, 47

NCRIS *National Collaborative Research Infrastructure Strategy* 33

NN *Nearest-neighbor* 8

OSGeo *Open Source Geospatial Foundation* 31, 33

OTB *Orfeo ToolBox* 10, 31, 35

PDE *Partial differential equation* 9

PSCS *Parameter series cluster sharing* 27, 29, 31, 43, 45–48

RADAR *Radio detection and ranging* 4

RGB *Red-Green-Blue* 5, 7

SD *Standard deviation* 35–41, 45

SITS *Satellite image time series* 5, 20–22, 24

SOM *Self-organizing map* 11

STRF *Spatio-temporal random field* 18, 22, 27–30, 32, 48

VQ *Vector quantization* 12, 18, 19, 24, 29

List of Figures

List of Tables

Chapter 1

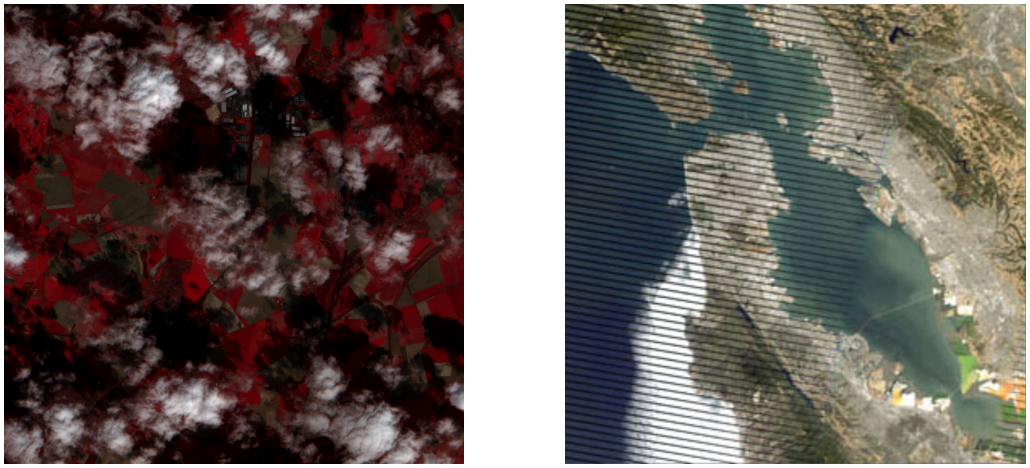
Introduction

Many different complex processes wind their way through our modern world. Understanding those procedures has been a central aspiration of mankind for ages. Insight into global economy, climate forecasting, or analysis of the population growth are just some exemplary tasks of present time, which all require comprehension of underlying processes. The key to gain that understanding can only be found within the observed information from the process itself.
Data from processes can be captured in various ways, and observation from afar is a popular choice. However resulting datasets such as satellite images often suffer from faulty entries, which are rooted in technical malfunctions, distortion issues or occlusion. This thesis investigates how machine learning methods can be applied for filling the resulting data gaps. In the following we explain our motivation for the topic and shortly discuss our chosen approach. Afterwards the specific thesis objectives are determined. This first chapter ends with a short overview of the thesis structure.

1.1 Motivation

Remote sensing is a commonly known example for visual process observation, typically resulting in satellite or drone images. It provides humans with insight into important developments and evolutions by capturing states of large scale processes. As an example climate change and population growth are under the greatest dangers to mankind, according to a recent expert survey [22]. At the same time they are perfect examples for processes, where remote sensing observation is not only possible, but leads to unique discoveries [84] [89]. Improving our understanding of the processes is vital to stem those threats. Therefore, the demand for high-quality methods dealing with remote sensing data is stronger than ever.
According to the United Nations Register of Objects Launched into Outer Space[1] nearly 5000 satellites orbit earth in 2019, with about 800 of them dedicated to making observations and taking pictures of earth and space. As a result, hundreds of remote sensing datasets are available and can be used for analysis. However remote sensing data often suffers from various disturbances occurring during image capturing, which reduces their utility. Figure 1.1 depicts examples for the two most common issues, namely cloud cover and sensor failure, which often result in large parts of faulty and hence unusable data.

[1] http://www.unoosa.org/oosa/en/spaceobjectregister/index.html



(a) Satellite image with cloud cover

(b) Satellite image with sensor failure

Figure 1.1: Erroneous satellite image data resulting from cloud cover (a) (THEIA L2A Sentinel-2 data) and sensor failure (b) (Landsat 7 data with SLC failure, taken from [68]).

For many application scenarios it is necessary to fill those gaps, i.e. to replace the erroneous values with more appropriate data. Appropriateness here means that replacing values ideally resemble the data which would have been observed without any issues. A lot of work has already been done on this task[65] [32], however in some cases those approaches provide unsatisfactory results.
We present a novel approach to reconstruct the missing data in remote sensing images by utilizing well-founded probabilistic machine learning techniques or, more specifically, generative Markov models [81] [57]. They are especially popular due to their high customizability and their extensive theoretical background. Gap filling with a fully probabilistic approach has (to the best of our knowledge) never been attempted before.
Our methodology allows to use an arbitrary mass of the available multidimensional data for fine-tuning the probabilistic model, which is then able to predict likely values for gaps. A major benefit of this approach is the influence of non-local data during gap filling, as every pixel in the training data affects the whole set of parameters, and thus, every prediction. At the same time a specified amount of locally available information also affects the probabilistic prediction. Besides that we explore a novel approach to compress probabilistic models. Our experiments show that these methods are well suited for their task, and we are indeed able to outperform other gap filling approaches. Moreover the proposed methodology could be applied to arbitrary incomplete image datasets, with satellite images serving as a proof-of-concept example.
However utilizing probabilistic models for gap filling on satellite images is not straightforward and leads to some special problems, which require smart solutions. The thesis discusses fundamental theories, the specific difficulties, the respective solutions and experimental results of our novel gap filling approach.
This work is a research cooperation between the Competence Center Machine Learning Rhine-Ruhr (ML2R)[2] and a research team located at Monash University [3]. Piatkowski's

[2] `https://www.ml2r.de/en/landingpage/`

[3] `https://research.monash.edu/en/projects/time-series-classification-for-new-generation-earth-observation-s`

work on probabilistic models and exponential families [57] is the fundamental backbone for the machine learning methodology of this thesis. Petitjean and Webb on the other hand provided insight into the remote sensing context and the associated satellite data, allowing us to explore a novel use case for probabilistic models. In terms of the collaboration I spent six months at Monash University, working closely together with both groups and thus linking research interests.

1.2 Objectives and Structure

The thesis investigates how gaps in satellite images can be filled with probabilistic machine learning. We intend to inform about all underlying fundamental aspects, convey the comprehensive methodology, and discuss the benefits and drawbacks in a comprehensible way. We hope that our work contributes to the state-of-the-art research in remote sensing and computer science.
Chapter 2 introduces the remote sensing context and gives an overview of popular gap filling approaches. It also conveys the fundamentals of probabilistic machine learning and generally explores all underlying theoretical aspects for our approach.
Chapter 3 contains the core methodology which allows to use probabilistic machine learning for gap filling. We here discuss the different emerging subproblems and their respective solutions. Besides the gap filling approach we also introduce our novel compression procedure and give a short overview of our proof-of-concept implementations.
The implemented methods were tested in various experiments, which are discussed in Chapter 4. Their results have been evaluated in a qualitative and quantitative way, showing feasibility of the presented methodology. We also explore how the subproblems' different solutions affect the results.
The thesis ends with a conclusion (Chapter 5), where we sum up the methodological content. We also provide an overview of possible extensions and future work.

Chapter 2

Fundamental Concepts

This chapter introduces topics and theories which are necessary for understanding the remainder of this thesis. As outlined earlier, furthering the understanding of processes is a common goal, and can be achieved by capturing and evaluating data from the process. Following up, the possibilities of observation via remote sensing are explored (Section 2.1). However the measured information can be erroneous (like data in general). Several approaches have emerged for filling resulting gaps in data, some of them specifically for the satellite data context. They are explored in Section 2.2. This thesis later on investigates how specialized probabilistic models can be used for remote sensing gap filling. Section 2.3 introduces these models and their fundamental theory.

2.1 Image Series as Observations of Spatio-Temporal Processes

Data from a process can be captured by various types of sensors, which somehow measure the process state. Many commonly known sensory devices such as wind gauges or seismometers are able to capture these information locally, but there are also a wide range of remote sensors in use.

2.1.1 Introduction to Remote Sensing

In these days, remote sensing generally refers to collecting data with satellite- or aircraft-based sensory technologies. Even though this is not beneficial in every scenario, remote sensing generally can be very helpful, e.g. for observing processes in dangerous or inaccessible areas. In addition looking at the big picture, literally, can also help with understanding large-scale developments. Remote sensing follows the *inverse problem principle*, assuming that observations can be used for calculating the causal factors which produced them.
Various sensor types are used for remote sensing. They can be roughly divided into *passive* and *active* approaches. While passive collectors only gather emitted reflections or radiance from the target, active remote sensors emit energy and then measure the reflected radiance. Film photography, infrared, and radiometers are examples for passive sensors, with sunlight reflection being the most commonly measured radiation. Examples of active remote sensors include *radio detection and ranging* (RADAR) and *light detection and ranging* (LIDAR), which measure the time delay between emission and return. With this information one could for example compute the location, speed, and

direction of an object. This is also an obvious example for how the inverse problem principle is applied in the field of remote sensing [7].
Remote sensing technologies can be quite complex, and the resulting data is not always easy to use and interpret. Basically it has a *spatial*, *temporal*, *spectral* (wavelengths of frequency bands), and *radiometric* (different distinguishable values) dimension. The respective dimension ranges and resolutions can highly vary between different sensor technologies, and define the resulting data dimensions. Single numeric entries within datasets are mostly *reflectances*, which are computed from sensor measurements.
It is often necessary to *preprocess* the data before it can be used in applications, with routines like *georeferencing*, *radiometric calibration*, and *terrain correction* [64]. As already outlined earlier *cloud cover* is a frequent problem in the evaluation of remote sensing data. For example, Cihlar estimated that up to 80% of Canada's surface can be obscured by clouds during mid-morning [12]. Cloud cover often leads to gaps in remotely sensed data, as the clouded reflectances are highly faulty. *Sensor failure* is another example which can lead to erroneous or missing data [8]. In most cases identifying and reconstructing affected data (i.e. the so-called *gap filling*) needs to take place before the data can be used in applications.

2.1.2 Interpreting Remote Sensing Data As Images

The interpretation of satellite data as *images* is quite common, due to the fact that many remote sensing datasets are acquired or stored as matrices and tables. Therefore the term *intensity* is often used as a synonym for the reflectance values. Interpreted as an image, a remotely sensed dataset $\mathfrak{I}$ consists of b-dimensional intensity values $\{\boldsymbol{I}(x,y) \in \mathbb{R}^b\}$ which are identified by pixel locations (x,y) with $1 \leq x \leq W, 1 \leq y \leq H$. Here W(idth) and H(eight) denote the image size, and b is the number of bands in the spectral dimension of the dataset. If data is acquired at T different times (or dates) it is also possible to interpret it as an *satellite image time series* (SITS) $\mathfrak{I}$ with dimensions (T,W,H). It consists of multidimensional intensities $\{\boldsymbol{I}(t,x,y)\}$ with $1 \leq t \leq T$. This allows the interpretation of intensities for a single pixel (x,y) at different times as a *time series* $\boldsymbol{\mathcal{I}}(x,y) = \{\boldsymbol{I}(1,x,y), \boldsymbol{I}(2,x,y), ..., \boldsymbol{I}(T,x,y)\}$. An exemplary visualization of this interpretation is displayed in Figure 2.1.
Usually a mask $M(t,x,y)$ identifies erroneous data entries in the SITS. Some masks feature a probability that the data at (t,x,y) is faulty, i.e. $0 \leq M(t,x,y) \leq 1$, some of them simply act as a flag, i.e. $M(t,x,y) \in \{0,1\}$. Obviously some preprocessing steps (as mentioned above) are crucial for such a visual interpretation, because a certain pixel location at different times needs to describe the same spatial area of the captured terrain over time.
For visualizing data some of the b bands are selected, based on their importance for the application scenario. Usually the intensities per band are *normalized* to the desired color channel intensity range (reflectance outliers are often excluded for normalization, e.g. highest and lowest 2%). Accordingly, one could choose three bands and normalize them to the interval $[0,255]$ for displaying the remotely sensed data at a certain time in an *Red-Green-Blue* (RGB) image.
The experiments of this thesis were run on data from the Sentinel-2 earth observation mission [16] (see Section 4.1.1 for more information on the specific data product). Figure 2.2 depicts a grayscale and RGB visualization of Sentinel-2 data, as well as the same region on a date with clouds and the corresponding cloud mask.

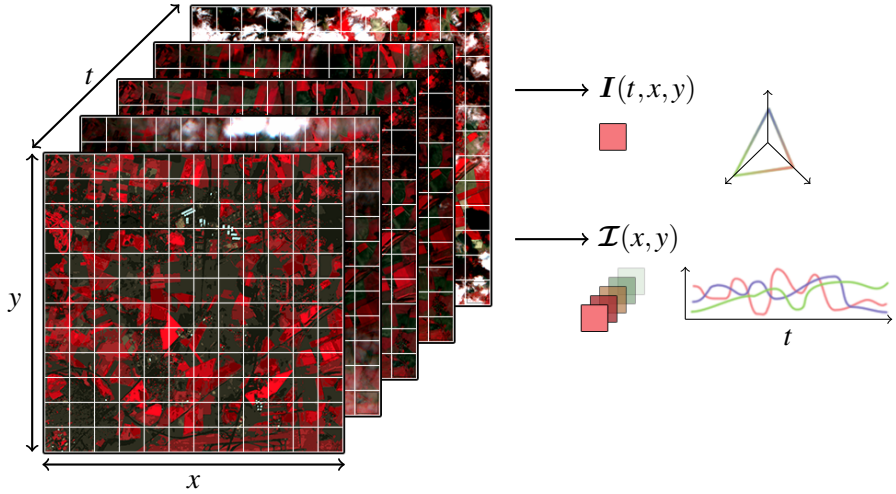
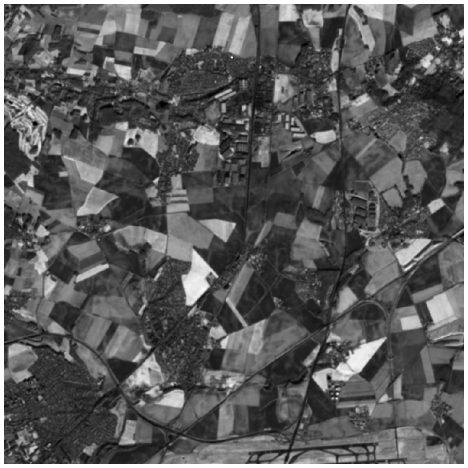


Figure 2.1: Remote sensing data interpreted as satellite image time series.

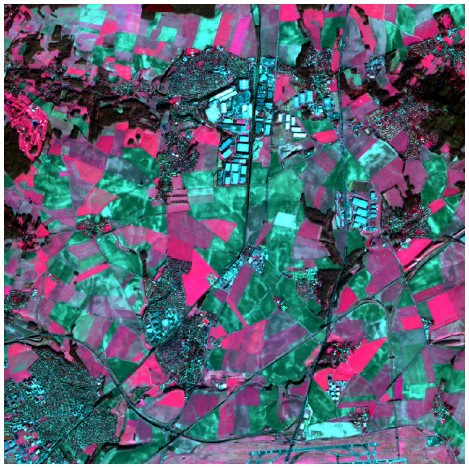
It is obvious that intensities affected by clouds will lead to faulty results when used in applications. Several approaches for gap filling in remote sensing have already been published, they are explored in the following section. Moreover, this thesis presents a novel way to fill gaps in satellite images with the help of probabilistic generative machine learning.

2.2 Background of Remote Sensing Gap Filling

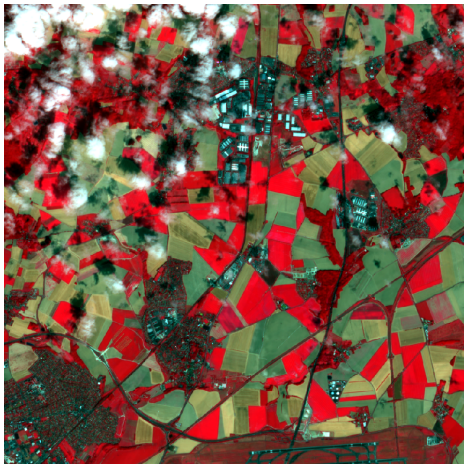
Missing data is a quite common problem in information technologies, with each field of research having their own specialized methods to solve it. In most use cases preprocessing steps identify parts of the data as missing, because the available information is faulty and such unusable. Following up some general approaches for gap filling in arbitrary temporal and spatial datasets are introduced.
In addition various specific methods have been published to fill gaps in the remote sensing context. All approaches use a certain part of the available data to compute hopefully realistic values for unobserved pixels. One can roughly divide them into categories, with methods that take available data along the spatial or temporal dimension into consideration. Some hybrids exist which even combine both (spatio-temporal methods). The remainder of this section is structured accordingly.
Some approaches also make use of ancillary information, such as valid data recorded by another sensor (spectral-based methods). This can be especially helpful for gaps resulting from sensor failure, as malfunctions often affect only some of the spectral bands. However those approaches cannot be applied when no ancillary information is available, e.g. for cloud cover, which usually affects all sensor bands. As the thesis experiment data suffers from cloud cover, those methods are not further discussed here. More extensive reviews of gap filling approaches have been published by Shen et al. and Kandasamy et al. [65] [32].



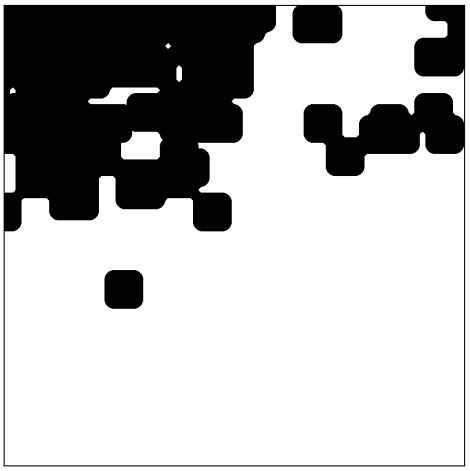
(a) Grayscale visualization of band eight



(b) RGB visualization of bands eight, three and two



(c) Region with cloud cover



(d) Cloud mask (black pixels identify clouds)

Figure 2.2: Example visualization of THEIA L2A Sentinel-2 satellite data in grayscale (a) and RGB (b), RGB visualization with cloud cover (c) and corresponding cloud mask (d).

2.2.1 Filling Gaps in General Spatial and Temporal Datasets

General spatial and temporal data, such as audio signals, weather data, and images, can contain faulty values, which creates demand for universal gap filling approaches. *Curve fitting* methods have been in use since antiquity [44] and can be roughly divided into interpolating or smoothing the available data. They assume that n observations $(x_1,y_1),(x_2,y_2),...,(x_n,y_n)$ are described by an unknown function $y = f(x)$. *Interpolation* provides the possibility to estimate the function value at arbitrary places x while resulting in the exact observations $y_i = f(x_i)$ for $1 <= i <= n$. *Smoothing* on the other hand assumes a certain noise on observations and thus tries to approximate the values as good as possible. Accordingly the function computes $f(x_i) = y_i + r_i$ for $1 <= i <= n$, with r_i being the assumed noise (or error) of the observation.
Curve fitting methods differ in complexity, accuracy, cost, and number of data points needed for successful calculation. A nearest-neighbor *(NN) interpolation* is the easiest method for interpolating, it returns the y-value of the data point whose x-value has the smallest distance to the queried x. *Linear interpolation* relies on only two data points, with the interpolant f for computing data being a simple linear function based on the data points. *Polynomial interpolation* generalizes this idea by using a polynomial function of a higher degree as interpolant. With n available data points there exists exactly one polynomial of degree at most $n-1$ such that all points are interpolated. The resulting interpolation function f is smoother and infinitely differentiable, while the interpolation error is proportional to the distance between data points to the power of n. *Spline interpolation* uses low-degree polynomials in each of the data point intervals. This allows efficient interpolation, especially if n is large, while the estimation and evaluation of a high-degree polynomial interpolation gets increasingly cumbersome. Other forms of interpolation choose a different class of interpolants (e.g. rational functions) or rely on knowledge about local derivatives [23].
Linear regression is an easy way to fit a smoothing curve f to observed data. With $n \geq 3$, fitting the linear function results in an overdetermined system of equations (as long as the data points are not collinear). A common method for solving overdetermined systems (and thus finding the best curve parameters) is *least squares*, which approximates the solution by minimizing the sum of squared errors. As with the interpolation, *polynomial regression* generalizes the linear approach by making f a polynomial function of a higher degree. Other regression approaches use complexer functions such as splines or sigmoids. *Smoothing splines* have a dedicated term for measuring (and possibly penalizing) the smoothness of the fitted function [78]. Instead of fitting the curve based on the y errors, some geometric approaches also try to minimize the orthogonal distance of points to the curve. Figure 2.3 depicts a few exemplary interpolation and smoothing results for a set of observations from a simple function. It shows that regressions results in a smooth curve that can be extrapolated, but interpolation here is closer to the original function.
More refined methods exist, such as utilizing *Gaussian processes* by interpreting the mapping of values as a random process (instead of a function). As one can see, the whole topic of curve fitting is closely related to estimation and probabilistic theory, and thus, to machine learning [33]. However more advanced methods are not discussed here, as they go beyond the scope of this thesis.

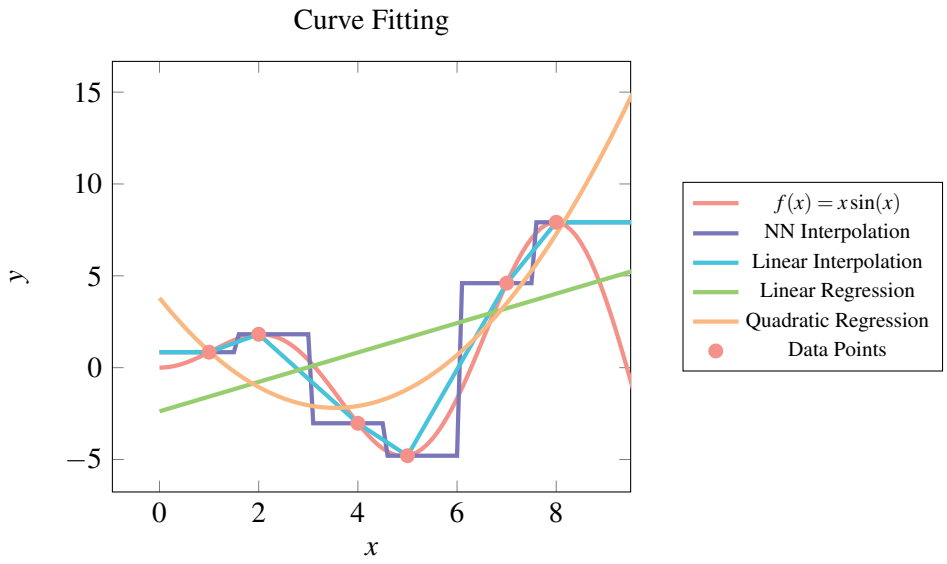


Figure 2.3: Results of several different curve fitting approaches, based on six data points evaluated from $f(x) = x\sin(x)$.

2.2.2 Spatial Gap Filling Approaches for Remote Sensing

The most simple and traditional approaches for satellite data gap filling are spatial methods, as only available information in spatially surrounding pixels is utilized. At the same time this can be a huge drawback, especially for images with clouds, which can result in enormous spatial data gaps. Obviously the methods can be applied to all kinds of incomplete images, another reason why they are well founded.

Interpolation methods are quite common and recover missing values as weighted averages of certain pixel intensities. The specific number and positions of interpolated pixels change from method to method, for example one could consider a basic bilinear interpolation in the 8-pixel neighborhood. Many approaches have been adapted from geostatistics [74], including *kriging* [51], one of the most famous techniques. Over the years it has been extensively explored in both geostatistical [60] and mathematical [67] ways, and also contributed in remote sensing [91] [88]. The central idea behind kriging is to estimate the pixel value as an unknown random-process mean together with the best linear unbiased estimator. It is being used and extended to the present day, for example by taking gap-free data into consideration [92] and extending it for nonlinear thermomechanical analysis [94].

The *propagated diffusion methods* propagate local information around gaps from the exterior to the interior, like in image inpainting. The propagation effect is formalized with *partial differential equation*s (PDEs), which also determine the direction for propagation [4]. Many variants have been presented with different PDE models, each favoring certain directions [50] [45]. However they are known for blurred results when large areas need to be reconstructed [42].

Variation-based methods fill gaps by performing image regularization, which is commonly formulated as a variational problem. It can be solved by minimizing a global energy function based on the observed data, a mask identifying the missing data and a clear target image. Regularization is added as a prior model for the target data, imposing

smoothness [27], sharpness [82], or non-local similarity [10] constraints on the images. Variation-based methods in general have satisfactory results with a chance of blurriness, depending on length of training, regularization, size of gaps, and data complexity [42].
Based on texture synthesis, *exemplar-based methods* copy similar known pixels to recover missing values [14]. Gap filling can be performed stepwise for single pixels or whole patches, the similarity is based on neighborhood pixels. Algorithms usually are run in a greedy way, therefore ordering has a high impact on the result. Global image coherence can be achieved by post-processing the inpainting [28] [35]. Those methods are able to restore large regions with detailed textures, however their accuracy is not always satisfactory, especially when only a small amount of data is observed.
Spatial approaches generally have problems if large parts of images are missing (e.g. due to cloud cover). Therefore it makes sense to also rely on temporal information, if it is available.

2.2.3 Temporal Gap Filling Approaches for Remote Sensing

Temporal-based approaches intend to fill gaps by looking at supplementary information for regions with missing data at different times. Obviously areas of interest can change over time, due to regular periodic geographical changes or abrupt, standalone incidents (caused by nature or humans). Good temporal gap filling methods hence should try to take those evolutions into consideration.
Replacement methods fill gaps by replacing the erroneous values with other measurements in the same region at a different time (also referred to as mosaicing [29]). Some methods first transform the observed intensities with a function to deal with temporal differences and changes that should also affect the replacement [38]. Replacing can commence pixel-by-pixel, patch-by-patch, or for the whole unobserved region, and is very similar to the spatial exemplar-based methods.
Temporal filter methods generally are used to process one-dimensional signals (e.g. for reducing noise), and thus allow the use of methods mentioned in Section 2.2.1. They can easily be applied in the remote sensing context be interpreting the intensities at a certain pixel position over time as a time series (as described in Section 2.1.2). Filtering can be applied locally or to the whole series. Common techniques for local filtering use sliding windows, the values within the window enclosing the gap are then used to compute the filling value [76] [63]. Curve fitting methods that only rely on some local data points for estimating the function can also be interpreted as sliding window approaches (here the window size is determined by the interpolated points). As an example basic curve fitting is used in the gap filling module[1] for *Orfeo ToolBox* (OTB) (a collection of applications for working with remote sensing data[2]). It computes values for missing data with the help of linear and spline interpolation methods, depending on the available surrounding data. Despite of its simplicity even a basic linear interpolation can still capture changes over time, as long as they are more or less linear. Several more advanced curve fitting methods have been explored, such as the *asymmetric Gaussian* (AG) model [31] or the *double logistic* (DL) technique [5], which assume that the temporal profile follows a Gaussian distribution. Some methods filter within the *frequency domain* by Fourier or wavelet transforming the original signal. It allows to use amplitude and phase informa-

[1] `http://tully.ups-tlse.fr/jordi/temporalgapfilling/blob/master/README.org/`
[2] `https://www.orfeo-toolbox.org/`

tion of the signal frequencies for gap filling, which can be especially promising for time series with periodical behavior [31] [61].
Only few approaches were made of *temporally learning a model* for the gap filling task. They utilize genetic optimization [42], dictionary learning [39], and Kohonen's *self-organizing map*s (SOMs) [37]. This thesis adds a novel approach to the short list of temporal learning models, utilizing *Markov random field*s (MRFs).
Kandasamy et al. provided an informative review on other temporal gap filling techniques [32]. The authors conclude that temporal smoothing and gap filling by Verger, Baret, and Weiss [75] provides the best overall performances when there is more than 20% missing data.
In general applying temporal gap filling approaches to every pixel of the series can lead to visible hard edges in the resulting images, as the novel pixel values might not blend well with their spatial neighborhood.

2.2.4 Spatio-Temporal Gap Filling Approaches for Remote Sensing

Methods that only work spatially or temporally have certain advantages and drawbacks. Some attempts have been made that combine both approaches.
A simple approach is to use spectral and temporal methods successively, such that the results of the previous method are further improved [90]. However simply pipelining the techniques does not necessarily capture correlations in both dimensions well, as possibly better results might be overridden or not even computed in the first place. Another approach uses spatial completion with temporal guidance, combining spatial exemplar-based replacement with a spatio-temporal variational smoothing technique [9]. Only pixels in the same picture are considered for replacing, while local spatio-temporal smoothness is achieved with the help of a MRF. While this method scored great results in Cheng et al.'s experiments, it might struggle when no spatial data is available at all.
As already stated earlier, some methods utilize available information in the spectral dimension. Similarly to spatio-temporal approaches, some hybrids exist that rely on spectral and temporal (or spatial) information [73] [93]. They are not further discussed here because spectral computations have limited use with clouded satellite data. Nevertheless Shen et al. think that spatio-spectro-temporal approaches might become the "trend in missing information reconstruction" [65] (as long as such data is available).

2.3 Background on Probabilistic Machine Learning

Machine learning techniques can be used to model and thus better understand processes. A learned model is usually represented by its parameters, which are changed during training to best fit sampled data from the process. Trained models can be used for various tasks such as classification or generation of new data.
This thesis explores the use of machine learning techniques for gap filling in remote sensing. More specifically, the idea is to train a model with some available data, and then use it to generate likely data for the gaps. It is possible to formulate the machine learning task with the help of probability theory (this formulation can be found in Section 3.1.1). Accordingly this thesis utilizes probabilistic models for solving the task. One should note that various other machine learning techniques might be applicable for gap

filling, however this thesis and the machine learning fundamentals section only explores probabilistic models.
Probabilistic models have extensive fundamental theory which is introduced in the following, with used notation and terminology mostly inspired by Piatkowski's work [57]. First the possibilities of using a graph as underlying *conditional dependency structure* for a random process variable are explored. Model parametrization can be achieved by utilizing the theory of *exponential families*. *Probabilistic inference* techniques allow to compute specific probability quantities, which are necessary to train and use a model. Possibilities of training and recent work on spatio-temporal models are discussed afterwards. Lastly a short introduction to *vector quantization* (VQ) is given, which, although not directly related to probabilistic machine learning, will be useful for our probabilistic gap filling approach.

2.3.1 Probabilistic Models and Graphical Structures

A multivariate random variable $\boldsymbol{X} = (\boldsymbol{X}_1, \boldsymbol{X}_2, ..., \boldsymbol{X}_n)$ can be used to probabilistically model a set of observations (or missing data). Here $\boldsymbol{X}_i$ addresses the i-th component of $\boldsymbol{X}$ (i can also be a set of indices for addressing multiple components). Each single component $\boldsymbol{X}_i$ has a specific state space $\boldsymbol{\mathcal{X}}_i$, which is the set of all its possible realizations $x_1, x_2, ...x_{k_i}$, meaning that $\boldsymbol{X}_i$ has k_i different realizations. The full state space of $\boldsymbol{X}$ is defined as the Cartesian product of all component state spaces $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{X}}_1 \times \boldsymbol{\mathcal{X}}_2 \times ... \times \boldsymbol{\mathcal{X}}_n$. In a practical sense, the state space $\boldsymbol{\mathcal{X}}$ defines which observations can be modeled with $\boldsymbol{X}$. The probability density of the event that $\boldsymbol{X}$ takes the value $\boldsymbol{x} \in \boldsymbol{\mathcal{X}}$ is denoted as $p(\boldsymbol{x}) = p(\boldsymbol{X} = \boldsymbol{x})$. For computing probabilistic inference it is also necessary to work with conditional probabilities, denoted as $p(\boldsymbol{X} = \boldsymbol{x}|O)$ for an arbitrary event O.
Independence between components of an arbitrary n-dimensional random variable $\boldsymbol{X}$ is denoted as $\boldsymbol{X}_i \perp\!\!\!\perp \boldsymbol{X}_j$, with $I(\boldsymbol{X})$ containing all independences of $\boldsymbol{X}$. Declaring a graphical dependency structure $G = (V, E)$ with vertices V and edges E is an easy and intuitive way to describe many conditional independence assumptions. Dependency structures in form of directed acyclic graphs are commonly known as *Bayesian networks*, while undirected graphs are generally called MRFs. In theory both approaches can be used as a graphical dependency structure [57], however only undirected graphs will be discussed in the following. Different forms of MRF models have been explored in the past, for example their use in resource-constrained environments has been investigated [57].
Let G be an undirected graph with vertices V and an associated edge set E. With interpreting G as a graphical independence structure for the random variable $\boldsymbol{X}$, the edges of G encode conditional independences of each vertex given its neighbors $N(v) = \{u \in V | (v,u) \in E\}$, i.e. each node is independent from all other nodes given its neighbors. The set of all conditional independences can be formulated as

$$\forall u \in V : \boldsymbol{X}_u \perp\!\!\!\perp \boldsymbol{X}_{V \setminus (\{v\} \cup N(v))} | \boldsymbol{X}_{N(v)}. \tag{2.1}$$

Together these independence assertions are called local Markov properties, denoted as $I(G)$. Besides the intuitive representation of dependencies, the graph G and its cliques $\mathcal{C}(G)$ can be used to factorize the probability density of $\boldsymbol{X}$ [26]:

$$p(\boldsymbol{x}) = \frac{1}{Z}\psi(\mathbf{x}) = \frac{1}{Z}\prod_{C \in \mathcal{C}(G)} \psi_C(\boldsymbol{x}_C), \text{ as long as } I(G) \subseteq I(\boldsymbol{X}) \tag{2.2}$$

Here $Z = \int_{\mathcal{X}} \psi(\boldsymbol{x}) \mathrm{d}\nu(\boldsymbol{x})$ is a constant for normalization known as *partition function*, and ψ, which factorizes into positive functions ψ_C, is the *potential function* of the graphical model. Any non-negative function ψ for n variables induces a certain undirected graphical model, the possible factorization of ψ defines the cliques and thus the connectivity of the graph. It does not even have to contain all independences of $\boldsymbol{X}$ in order to yield a valid factorization, as normally multiple factorizations of ψ exist. However more independences (and thus, less edges in E) will lower the complexity for computing probabilistic inference (see Section 2.3.3). Hence a factorization in terms of cliques with minimal factors $\bar{\mathcal{C}}(G) \subseteq \mathcal{C}(G)$ is desirable, and will be considered in the following [57]. Practically, the real statistical dependencies are unknown, and different graphical dependency structures can be used depending on the task. Some algorithms exist for discovering associations and deriving dependency structures from available data. They usually result in triangulated graphs [55] [54] [46] [72] or trees [11]. In some processes sensor nodes (which are represented by components of $\boldsymbol{X}$) might influence adjacent sensors, and a dependency graph can be derived from sensor placement, hopefully approximating the real statistical dependencies. An example for that would be a city street network and traffic density sensors at different places, with the road infrastructure possibly indicating dependencies in the measured data [40].

2.3.2 Exponential Families and Probabilistic Models

The goal of probabilistic machine learning is to build an adaptable model for the density of a multivariate random variable $\boldsymbol{X}$. For adaptability, this density needs to be parametrized by a *parameter vector* $\boldsymbol{\Theta} \in \mathbb{R}^d$, which leads to exploring exponential family densities. The parametrized density $p_{\boldsymbol{\Theta}}$ is part of the exponential family of densities, if it can be written as follows:

$$p_{\boldsymbol{\Theta}}(\boldsymbol{X} = \boldsymbol{x}) = \exp(\langle \boldsymbol{\Theta}, \phi(\boldsymbol{x}) \rangle - A(\boldsymbol{\Theta})) = \frac{\psi(\boldsymbol{x})}{Z(\boldsymbol{\Theta})} \tag{2.3}$$

Here $\phi(\boldsymbol{x})$ are the *sufficient statistics* based on the training data, $\psi(\boldsymbol{x}) = \exp(\langle \boldsymbol{\Theta}, \phi(\boldsymbol{x}) \rangle)$ denotes the potential function, and $A(\boldsymbol{\Theta}) = \log Z(\boldsymbol{\Theta})$ is the logarithmized partition function.
$\langle \boldsymbol{\Theta}, \phi(\boldsymbol{x}) \rangle$ denotes the Euclidean inner product of the vectors Θ and $\phi(\boldsymbol{x})$, which both take values in $\mathbb{R}^d$. $A(\Theta)$ assures the normalization of $p_{\boldsymbol{\Theta}}$, such that $\sum_{\boldsymbol{x} \in \mathcal{X}} p_{\boldsymbol{\Theta}}(\boldsymbol{X} = \boldsymbol{x}) = 1$. According to the definition of Z, it is defined as $A(\boldsymbol{\Theta}) = \log \int_{\mathcal{X}} \langle \boldsymbol{\Theta}, \phi(\boldsymbol{x}) \rangle \mathrm{d}\nu(\boldsymbol{x})$.
The exponential family formulation is rather simple and well suited for computations, however factorization is needed to use it with the previously introduced graphical models. A canonical representation with indicator functions is available for discrete MRFs, achieving factorization of $p_{\boldsymbol{\Theta}}$. For the discrete, n-dimensional random variable $\boldsymbol{X}$, its state space $\boldsymbol{\mathcal{X}}$, its undirected conditional independence structure $G = (V, E)$, and the set of cliques with minimum factors $\bar{\mathcal{C}}(G)$, the overcomplete sufficient statistic $\phi : \boldsymbol{\mathcal{X}} \to \mathbb{R}^d$ is

$$\phi(\boldsymbol{x}) = (\phi_C(\boldsymbol{x})^\top : \forall C \in \bar{\mathcal{C}}(G))^\top. \tag{2.4}$$

$\phi_C(\boldsymbol{x})$ is a $|\boldsymbol{\mathcal{X}}_C|$-dimensional vector of indicator bits representing the possible states of the variables that belong to C. For a given $\boldsymbol{x}$ exactly one bit in $\phi_C(\boldsymbol{x})$ is set. Accordingly $\phi(\boldsymbol{x})$ is a d-dimensional vector containing the indicator functions for all cliques, with $d = \sum_{C \in \bar{\mathcal{C}}(G)} |\boldsymbol{\mathcal{X}}_C|$. Figure 2.4 shows a simple example for sufficient statistics.

$$\phi_U(\boldsymbol{x}_U^1) = \begin{pmatrix} 0 \\ \mathbf{1} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad \phi_U(\boldsymbol{x}_U^2) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \mathbf{1} \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad \phi_U(\boldsymbol{x}_U^3) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \mathbf{1} \\ 0 \end{pmatrix} \qquad \boldsymbol{\mathcal{X}}_U = \begin{matrix} \{(A,0,+1), & \\ (A,0,-1), & \leftarrow \boldsymbol{x}_U^1 \\ (A,1,+1), & \\ (A,1,-1), & \\ (B,0,+1), & \\ (B,0,-1), & \\ (B,1,+1), & \\ (B,1,-1), & \leftarrow \boldsymbol{x}_U^2 \\ (C,0,+1), & \\ (C,0,-1), & \\ (C,1,+1), & \leftarrow \boldsymbol{x}_U^3 \\ (C,1,-1)\} & \end{matrix}$$

Figure 2.4: Sufficient statistics example of a vertex set $U = \{u, v, w\}$ with vertex state spaces $\boldsymbol{\mathcal{X}}_u = \{A, B, C\}$, $\boldsymbol{\mathcal{X}}_v = \{0, 1\}$, $\boldsymbol{\mathcal{X}}_w = \{-1, +1\}$ and three realizations $\boldsymbol{x}_U^1 = (A, 0, -1)$, $\boldsymbol{x}_U^2 = (B, 1, +1)$, and $\boldsymbol{x}_U^3 = (C, 1, -1)$. Each entry in ϕ_U is an indicator function for a specific state in $\boldsymbol{\mathcal{X}}_U = \boldsymbol{\mathcal{X}}_u \times \boldsymbol{\mathcal{X}}_v \times \boldsymbol{\mathcal{X}}_w$, as listed on the right. Example is taken from [57].

Each weight in the d-dimensional parameter vector Θ represents a certain clique state, the order of both $\phi(\boldsymbol{x})$ and $\boldsymbol{\Theta}$ is arbitrary but fixed. A sufficient statistic like this can be constructed for any discrete random variable. For continuous variables no universal definition of sufficient statistics exists, which is why discrete MRF models will be used and explored in this thesis. Exponential family representations of graphical models can also be motivated through the principle of maximum entropy [81].

2.3.3 Probabilistic Inference

Inference is the general term for computing certain quantities from probability measures, such as the value of a partition function $Z(\boldsymbol{\Theta})$, marginal probabilities $p(\boldsymbol{X}_i = \boldsymbol{x}_i)$, or the *maximum a posteriori* (MAP) state $\max_{\boldsymbol{x} \in \boldsymbol{\mathcal{X}}} p(\boldsymbol{x})$.
As already stated earlier, the partition function ensures normalization of the probability density p_ψ for an arbitrary non-negative potential function $\psi : \boldsymbol{\mathcal{X}} \to \mathbb{R}_+$:

$$p(\boldsymbol{x}) = \frac{1}{\int_{\boldsymbol{\mathcal{X}}} \psi \mathrm{d}\nu} \psi(\boldsymbol{x}) = \frac{1}{Z} \psi(\boldsymbol{x}) \text{ , with } Z \text{ being the partition function of } p \tag{2.5}$$

It is closely related to the computation of marginal probabilities, which can be computed by only partially carrying out the integration for unfixed variables U [57]:

$$p_{\bar{U}}(\boldsymbol{x}_{\bar{U}}) = \frac{1}{Z} \int_{\boldsymbol{\mathcal{X}}_U} \psi(\boldsymbol{x}_U \, , \boldsymbol{x}_{\bar{U}}) \mathrm{d}\nu_U(\boldsymbol{x}_U) \text{ , } U \subset \{1, 2, ..., n\}, \text{ variables } \bar{U} \text{ fixed to } \boldsymbol{x}_{\bar{U}} \tag{2.6}$$

For discrete random variables with a conditional independence structure the naive computation of the partition function is not feasible, as it requires summation over the whole state space and multiplication over the cliques (more specifically it is a **#P**-complete problem [6]). However the *belief propagation* (BP) algorithm (occasionally called sum-product message passing) allows to evaluate Z in polynomial time, as long as the underlying graphical independence structure G is acyclic [52]. BP uses a message passing

technique on the tree graph G. Messages are send inwards starting at the leaves, meaning that each subtree root obtains messages from all adjoining nodes and then sends them to its unambiguous parent. Probabilistically speaking, the partition function can be recursively reformulated by cutting edges between any leaf v and its parent node u, and computing the potential for the subtree rooted at u without the former edge to v [57]. In a tree every node has a unique edge towards the root, and it is guaranteed that each node can obtain messages from all adjoining nodes before it passes them on. As soon as all messages were collected at the root node, all messages can be sent back in the opposite direction, starting at the root. This is only feasible if G is a tree, as otherwise there is at least one subgraph that does not contain leaf vertices.
The heuristic inference *loopy belief propagation* (LBP) algorithm uses a similar approach for arbitrary graphs [36]. Here the messages are recomputed several times, as incoming messages can possibly change due to cycles. LBP converges for trees, however convergence and correctness for general graphs is neither well understood nor guaranteed. Despite LBP's heuristic nature and lack of approximation bounds it is still commonly used in practice for computing approximate inference, and often provides good results [57].
Another approach for evaluation of the partition function is the *junction tree* (JT) algorithm. It is not based on the summation over the whole state space of the graph, instead it computes inference on its JT (which can be constructed for any loopy graph) [81] [57]. The computational effort for inference with a JT depends on the state spaces and neighborhood sizes of the vertices and can be feasible for simple graphs [34]. However in most cases LBP delivers sufficient results while running significantly faster.
Methods of *variational inference* can be used to modify the conditional independence structure. The goal is to reduce the complexity for probabilistic inference without loosing too many details and thus quality [81]. For example certain variational approximations allow to bound the partition function [80] [79].
Sampling methods can be used to alleviate the computational effort of enumerating a large dataset for computing inference (such as the vertex state space). The idea here is to draw N random samples $\boldsymbol{x}$ from the set and extrapolate the solutions for those samples onto the whole set. In other words, the wanted quantity is *estimated* from the samples. Different estimators could be used for computing the expected value (e.g. computing the average value over all sample function values). In probabilistic inference, sampling allows to estimate the partition function, marginal probabilities, or the most likely joint state [57]. Those techniques are based on the Monte Carlo principle [2]. *Markov chain Monte Carlo* (MCMC) methods generate samples according to the Markov property, such that each sampling depends on the previously drawn sample. One popular example for MCMC methods is the *Gibbs sampler* [19]. It uses the Markov property $p(\boldsymbol{x}_v|\boldsymbol{x}_{V\setminus\{v\}}) = p_v(\boldsymbol{x}_v|\boldsymbol{x}_{N(v)})$ for any vertex v to compute conditional marginal probabilities. With $\mathcal{C}(v)$ being the set of cliques containing v, the conditional marginal is

$$p_v(\boldsymbol{x}_v|\boldsymbol{x}_{N(v)}) = \frac{\prod_{C\in\mathcal{C}(v)} \psi_C(\boldsymbol{x}_v, \boldsymbol{x}_{N(v)})}{\sum_{y\in\boldsymbol{\mathcal{X}}_v} \prod_{C\in\mathcal{C}(v)} \psi_C(y, \boldsymbol{x}_{N(v)})}. \tag{2.7}$$

Gibbs sampling starts with some initial joint state $\boldsymbol{x}^0$ and constructs a new state by sampling each component of $\boldsymbol{x}^0$ according to Equation 2.7. By repeating the sampling several times it is possible to obtain a valid sample state $\boldsymbol{x}$, which is independent from $\boldsymbol{x}^0$. The generic and lightweight nature makes Gibbs sampling a great choice for generating

samples from densities.
When samples are generated with a model, certain joint states will have a higher chance to appear depending on the parameters. The so-called MAP state $\boldsymbol{x}^*$ is the most likely joint state that might be sampled:

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}\in\boldsymbol{\mathcal{X}}} p_{\boldsymbol{\Theta}}(\boldsymbol{X}=\boldsymbol{x}) = \arg\max_{\boldsymbol{x}\in\boldsymbol{\mathcal{X}}} \langle \boldsymbol{\Theta}, \boldsymbol{\phi}(\boldsymbol{x})\rangle. \tag{2.8}$$

If only a subset of the variables $O \subset V$ are observed and the rest $H = V \setminus U$ is hidden (or missing), it is possible to compute the most likely state given the observation:

$$\boldsymbol{x}_H^* = \arg\max_{\boldsymbol{x}_H\in\boldsymbol{\mathcal{X}}_H} p_{\boldsymbol{\Theta}}(\boldsymbol{X}_H = \boldsymbol{x}_H | \boldsymbol{X}_O = \boldsymbol{x}_O) \tag{2.9}$$

This is called *conditional MAP* problem, which allows to use probabilistic models for conditionally predicting missing data (or classification, with variables in O denoting features and H being class variables.) [57]. Computing conditional MAP states is also **NP**-hard, however the inference techniques which have been introduced above can be adapted for solving (BP, JT) or approximating (LBP, variational principle, sampling) the problem [85] [36] [83] [57]. Instead of finding the conditional most probable joint state, it can also be interesting to find the top-M solutions for Equation 2.9, and methods have been adapted accordingly [86] [18].

2.3.4 Training a Probabilistic Model

After introducing the idea of graphical dependency structures, parametrized discrete probabilistic models, and practical approaches for computing inference with those models, the question remains how they can be trained.
Training is based on a dataset D with samples from $\boldsymbol{X}$ and results in parameters $\hat{\boldsymbol{\Theta}}$ which hopefully fit the data well. To measure the model quality for training data it is necessary to specify some sort of *loss function* $\mathcal{L}$, with $\hat{\boldsymbol{\Theta}}$ minimizing the loss. For probability densities it is common to use the *maximum likelihood* (ML) principle, which assumes that samples in the dataset are independent from each other. The probability density of D therefore factorizes over the samples, i.e. $p_{\boldsymbol{\Theta}}(D) = \prod_{\boldsymbol{x}\in D} p_{\boldsymbol{\Theta}}(\boldsymbol{x})$. The most likely parameters given the data are

$$\hat{\boldsymbol{\Theta}} \in \arg\max_{\boldsymbol{\Theta}\in\mathbb{R}^d} p(\boldsymbol{\Theta}|D) = \arg\max_{\boldsymbol{\Theta}\in\mathbb{R}^d} \frac{p(D|\boldsymbol{\Theta})p(\boldsymbol{\Theta})}{p(D)}. \tag{2.10}$$

For now $p(\boldsymbol{\Theta})$ is a constant, and dividing by $p(D)$ is irrelevant for maximization. It is numerically convenient to work with logarithmized probabilities, and minimization of the negative average log-density $-(1/|D|)\log p(D|\boldsymbol{\Theta})$ preserves the intuitive idea of minimizing loss. As $p(D|\boldsymbol{\Theta}) = p_{\boldsymbol{\Theta}}(D)$ is part of the exponential family of densities, the obtained ML loss function (with plugging in Equation 2.3) is

$$\mathcal{L}(\boldsymbol{\Theta}; D) = -\frac{1}{|D|}\sum_{\boldsymbol{x}\in D} (\langle \boldsymbol{\Theta}, \boldsymbol{\phi}(\boldsymbol{x})\rangle - A(\boldsymbol{\Theta}) = -\langle \boldsymbol{\Theta}, \tilde{\boldsymbol{\mu}}\rangle + A(\boldsymbol{\Theta}). \tag{2.11}$$

Here $\tilde{\boldsymbol{\mu}}$ is the sample average estimator for the sufficient statistics, and due to the linearity of the dot product the last equality holds [57]. The gradient is $\nabla\mathcal{L}(\boldsymbol{\Theta}; D) = \nabla A(\boldsymbol{\Theta}) - \tilde{\boldsymbol{\mu}}$,

and the optimal parameters that minimize the loss are $\hat{\boldsymbol{\Theta}} = \arg\min_{\boldsymbol{\Theta}\in\mathbb{R}^d} \mathcal{L}(\boldsymbol{\Theta}; D)$. Obviously Equation 2.11 requires computation of the #**P**-hard (log) partition function, therefore exact computation is cumbersome and in most cases not feasible. Efficient algorithms for inference, computation of loss and gradient, and parameter estimation problems are available for tree structured models, in other cases the approximate inference techniques mentioned above come in handy [81]. Various first- or second-order methods can be applied for parameter estimation if no closed-form solution exists, they vary in resource consumption, convergence speed, and complexity [49]. For the proof-of-concept experiments in this thesis a basic first-order parameter estimation technique was used.
First-order methods iteratively improve the parameters, starting with an arbitrary vector $\boldsymbol{\Theta}^0$ and generating $\boldsymbol{\Theta}^1, \boldsymbol{\Theta}^2, \ldots$. The simplest approach, known as *gradient descent* (GD), generates the series by improving $\boldsymbol{\Theta}$ step by step, with respect to the (approximated) loss gradient, i.e.

$$\boldsymbol{\Theta}^{i+1} = \boldsymbol{\Theta}^{\boldsymbol{i}} - \kappa_i \nabla \mathcal{L}(\boldsymbol{\Theta}^i; D). \tag{2.12}$$

For linear objective functions, the step width κ_0 can be chosen as 1, resulting in the optimal solution after one step. Otherwise $\boldsymbol{\Theta}^i$ eventually converges to a stationary point, depending on the chosen step widths $\kappa_0, \kappa_1, \kappa_2, \ldots$. Several more advanced methods have been developed based on the GD idea [48] [62], and its convergence to the global minimum is guaranteed for the introduced ML loss function [57] [47].
It is questionable if a perfect fit for training data (i.e. convergence to a stationary Θ^i) is desirable, as it cannot be assumed that training data perfectly represents all use case scenarios. For training a generalizing model, it is common to stop the optimization as soon as the improvement drops below a specified threshold ρ [87] [1]. An absolute $\mathcal{L}(\boldsymbol{\Theta}^i; D) \leq \rho$ or relative $\mathcal{L}(\boldsymbol{\Theta}^i; D) - \mathcal{L}(\boldsymbol{\Theta}^{i+1}; D) \leq \rho$ stop criteria is used to check for improvement. With cyclic MRFs the loss function and gradient of probabilistic models are usually approximated in every iteration, resulting in a slight fluctuating of the objective function. This means that a single iteration might not improve the loss, although further improvement could be achieved when the approximation is run again. For optimization termination, this is usually countered by only stopping after I iterations without improvement.
Regularization can be helpful for incorporating additional information into the optimization, especially when the problem is ill-posed. It is a way to impose certain characteristics on parameters (e.g. sparseness or smoothness), although it can reduce their optimality regarding Equation 2.11. It is usually achieved by adding penalty terms to the loss, which depend on $\boldsymbol{\Theta}$. The resulting minimization problem for any function f is then

$$\min_{\boldsymbol{\Theta}\in\mathbb{R}^d} f(\boldsymbol{\Theta}) + \lambda R(\boldsymbol{\Theta}). \tag{2.13}$$

The *regularization weight* λ controls the impact of the regularizer R during optimization. Powers of l_p-norms are common examples for regularizers:

$$R(\boldsymbol{\Theta}) = ||\boldsymbol{\Theta}||_p = (\sum_{i=1}^{d} |\boldsymbol{\Theta}_i|^p)^{\frac{1}{p}}, \tag{2.14}$$

they allow the popular l_1-regularization $||\boldsymbol{\Theta}||_1$ or squared-l_2-regularization $||\boldsymbol{\Theta}||_2^2$. Adding regularization to optimization can also be seen as imposing some sort of prior density on the parameters. This can be achieved with MAP estimation, by making $p(\boldsymbol{\Theta})$ in

Equation 2.10 dependent on some hyperparameters $\boldsymbol{\lambda} \in \mathbb{R}^k$:

$$\hat{\boldsymbol{\Theta}} \in \underset{\boldsymbol{\Theta} \in \mathbb{R}^d}{\arg\max}\, p(D|\boldsymbol{\Theta}) p(\boldsymbol{\Theta}|\boldsymbol{\lambda}), \tag{2.15}$$

leading to the negative average log-likelihood

$$\mathcal{L}_{MAP}(\boldsymbol{\Theta}; D) = -\langle \boldsymbol{\Theta}, \tilde{\boldsymbol{\mu}} \rangle + A(\boldsymbol{\Theta}) - \log p(\boldsymbol{\Theta}|\boldsymbol{\lambda}). \tag{2.16}$$

In fact it is possible to choose densities for $p(\boldsymbol{\Theta}|\boldsymbol{\lambda})$ which result in regular l_1- or l_2-regularized objective functions [57]. We later extend the idea of regularization based on prior knowledge about parameters Section 3.1.5.
Training exponential family models usually demands fully observed data in D. Yet it is possible to support incomplete data during training by running it as an *expectation-maximization* (EM) algorithm [15]. EM algorithms support missing data by treating them as latent variables, and has been successfully applied in the context of exponential family models [70] [69] In the expectation step, the missing data is estimated based on the current parameters (e.g. by solving the conditional MAP problem). After that the model is optimized based on the observed and estimated data (maximization step). By repeating both steps in an alternating fashion, it is possible to train an exponential family model with missing data.

2.3.5 Spatio-Temporal Random Fields

Piatkowski, Lee, and Morik applied Markov models in specific use cases with their *spatio-temporal random field* (STRF) approach [58]. The authors provide a generic method to construct a MRF from a given spatial dependency structure. The resulting models can be used to analyze spatio-temporal processes, such as traffic density in cities [40].
STRFs are based on a spatial structure $G_0 = (V_0, E_0)$, that is being replicated for each time step $1 \leq t \leq T$ (i.e. the process is modeled for T different times). The replicas G_t of G_0 are connected with $(T-1) \cdot (|V_0| + |E_0|)$ temporal edges E_{temp_t}, i.e. the final model has temporal edges for each node and each spatial edge. Accordingly the whole STRF is $G = (\bigcup_t V_t, \bigcup_t (E_t \bigcup E_{\text{temp}_t}))$, with $E_{\text{temp}_T} = \emptyset$ (only $T-1$ connections between the T replicas are needed). This structure also allows a temporal reparametrization which can be used for compression and regularization [58].

2.3.6 Vector Quantization

Obviously data complexity can highly affect the complexity of machine learning models, and hence the runtime for training and prediction. For the discussed probabilistic models the state space size $|\boldsymbol{\mathcal{X}}|$ affects the number of parameters and pretty much every computation. Therefore it can be helpful to compress data and thus lower its complexity. VQ is a classical approach for data compression, that has been explored for many years [21]. Compressing data is achieved by subdividing the input space into convex groups, which are represented by their centroid points. During quantization the centroids are iteratively moved depending on the data, resulting in groups that have approximately the same number of vectors closest to them. A popular choice for VQ is the *k-means* clustering algorithm [43], a specialization of Lloyd's algorithm [41]. It chooses the first

k points as centroids and iteratively adds the remaining points, each time shifting the assigned centroid. Several algorithmic extensions have been published [3] [77] [30], and implementations exist in various languages and frameworks.
Obviously VQ and clustering can also be used for machine learning, e.g. in pattern recognition [66]. More extensive information can be found in [21] [20].
All fundamental theories for gap filling with probabilistic models have now been introduced. The next chapter conveys information about the specific problems of this task, and how they can be overcome.

Chapter 3

Methodology and Implementation

Our methodology applies machine learning in the remote sensing context. The main contribution is our fully probabilistic approach for filling gaps in SITS. Section 3.1 discusses the resulting non-trivial subproblems and how we intend to solve them. In addition we present a novel technique for compression of probabilistic models, it is explored in Section 3.2. These methods have been implemented and tested in experiments to show their applicability and usefulness, Section 3.3 gives an overview over the implementation details.

3.1 Probabilistic Approach for Gap Filling in Satellite Images

Utilizing probabilistic machine learning for gap filling in remote sensing has many advantages compared to other approaches. The prediction with a trained model can take local and global available information in both temporal and spatial dimension into consideration, depending on the chosen graphical dependency structure. Various routines and hyperparameters make the whole process customizable to many use cases with specific requirements. However several more or less complicated obstacles need to be overcome for solving this ambitious task.
We first formulate the gap filling task in a probabilistic way and explore how MRF models can be utilized for this task. After that we address the complexity problem, which is rooted in the nature of remote sensing data. Next a optimization routine is introduced, which is able to handle missing data entries in the training dataset. The remainder of this section explores how parameter prior knowledge can be incorporated into the training process.

3.1.1 Probabilistic Interpretation of the Gap Filling Task

The following probabilistic interpretation of gap filling is based on the visual interpretation of remote sensing data introduced in Section 2.1.2. Let $\mathfrak{I} = \{\boldsymbol{I}(t,x,y) \in \mathbb{R}^b\}$ with $1 \leq t \leq T, 1 \leq x \leq W, 1 \leq y \leq H$ be an arbitrary SITS with $M(t,x,y) \in \{0,1\}$ identifying missing or erroneous data entries. More specifically, $\boldsymbol{I}$ takes values in $\boldsymbol{\Omega} = (\Omega_0 \times \Omega_1 \times ... \times \Omega_b) \subseteq \mathbb{R}^b$. $\boldsymbol{\Omega}$ is the Cartesian product of the different reflectance values Ω_i that can be measured on each sensor band $1 \leq i \leq b$.
$\mathfrak{I}$ can be interpreted as a multivariate random variable $\boldsymbol{X} = (\boldsymbol{X}_1, \boldsymbol{X}_2, ..., \boldsymbol{X}_N)$, with $N = T \cdot W \cdot H$. Note that all components of $\boldsymbol{X}$ have the same state space $\boldsymbol{\Omega}$, i.e. $\boldsymbol{X}$

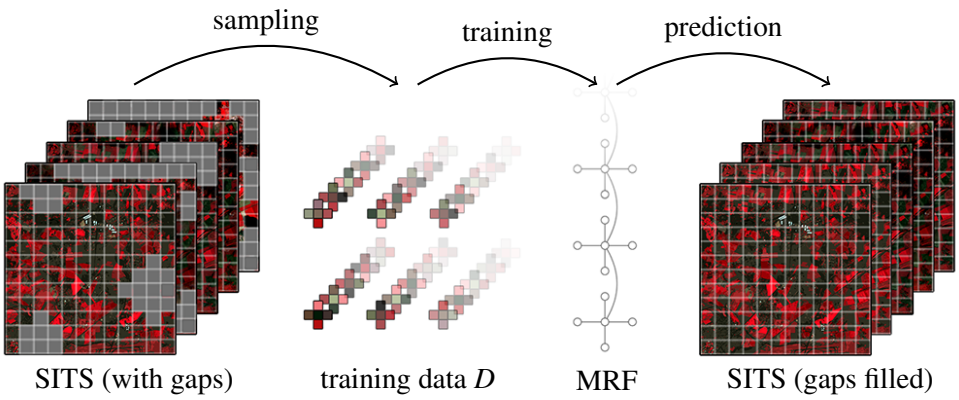


Figure 3.1: Pipeline with steps of the novel probabilistic gap filling approach.

takes values $\boldsymbol{x} \in \boldsymbol{\Omega}^N$. It is safe to assume that some components of $\boldsymbol{X}$ affect others, e.g. reflectances are possibly influenced by local spatial neighborhood and the past and following reflectances over time. A graphical dependency structure $G = (V, E)$ could be used to approximate the real dependencies. M identifies certain variables $O \subset V$ as observed and the rest $H = V \setminus U$ as missing. Accordingly the gap filling task can be interpreted as a conditional MAP problem (see Equation 2.9).
The fundamental theories in Section 2.3 allow to train a spatio-temporal probabilistic model, which can be used for solving the conditional MAP problem. However it does not really make sense to model the complete SITS, as only a single training sample would be available. Moreover the number of parameters for such a model would be enormous (e.g. a million vertices in the MRF for a single image of size $W = H = 1000$). Despite this underwhelming realization, it is still possible to use probabilistic models for gap filling.

3.1.2 Probabilistic Models for Filling Gaps

Our central idea is to subdivide the data into small patches, define a random variable $\boldsymbol{Y} \subset \boldsymbol{X}$ describing such a patch, and train a MRF for $\boldsymbol{Y}$. The model can be trained with an arbitrary amount of data which is randomly sampled from the whole dataset. Filling gaps can be achieved by solving the conditional MAP problem for every patch which contains missing entries. Practically speaking the model is slid over the whole dataset for filling all gaps, Figure 3.1 displays the individual steps of our gap filling approach.
The patch form and the graphical dependency structure determine which local information is taken into consideration during prediction. One should note that due to the training with randomly sampled data, an arbitrary amount of globally available information affects the model parameters, and thus, the prediction. This is a big advantage over other methods, which only rely on local information.
Our approach might perform poorly if there are completely unobserved patches in the dataset. Without any observed data (i.e. $O = \emptyset$), the conditional MAP prediction would result in the most likely state, i.e. the average data seen during training. However this is not really a disadvantage, as all gap filling approaches struggle when no local reference data is available. Depending on the data of interest it makes sense to choose different model structures. As an example models for filling gaps which stem from cloud cover

should make use of available temporal information, as there are hardly any places on earth that are always clouded. Simply put these (temporal) models should simulate how any single pixel in the data (and optionally its local spatial neighborhood) can change over time. For sensor failure gaps it might make sense to construct a model that considers spatially available data. As the thesis experiments deal with cloud covered data we will only explore temporal models here.

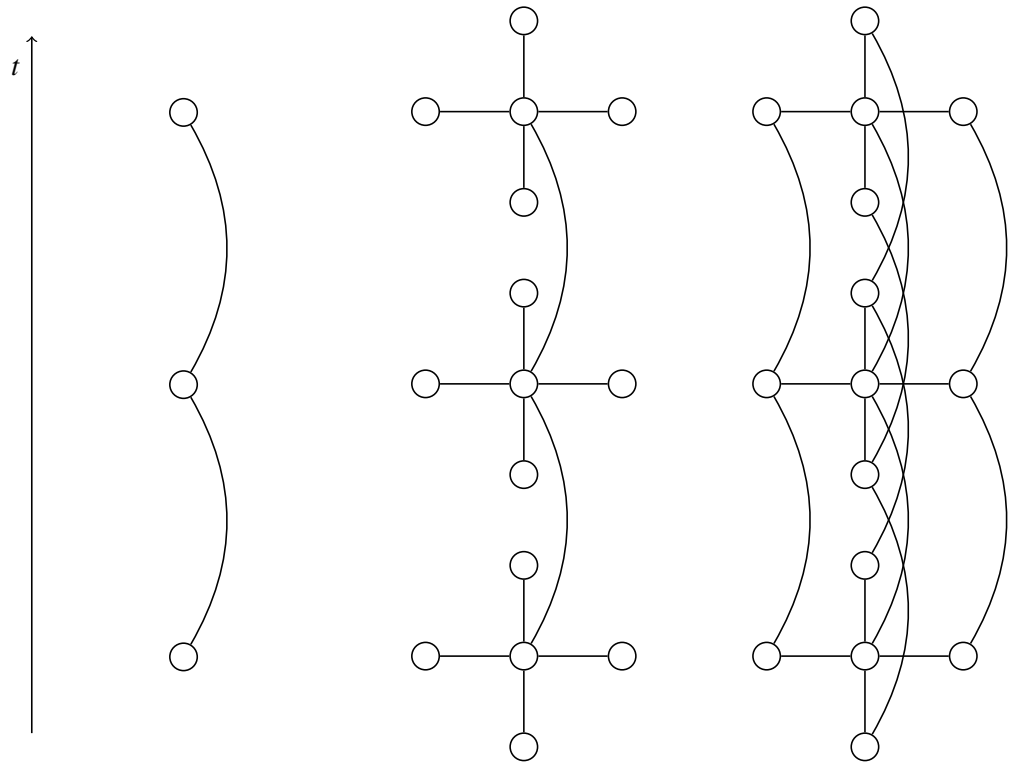
A *Chain* model might be the simplest model for gap filling on cloud covered data. In this case $\boldsymbol{Y}$ has T components, and the underlying graphical dependency structure is a chain-like MRF with T vertices and $T-1$ edges connecting them. It only considers temporally available information, accordingly the patches are simply intensity time series as described in Section 2.1.2. $|D|$ pixel locations (x,y) within the image are randomly sampled and the associated data slices $\boldsymbol{\mathcal{I}}(x,y)$ are extracted from the dataset, forming the training data D. It will be later discussed how training with missing data can commence, but for now it is assumed that training slices are fully observed. The model is trained by minimizing the loss function (Equation 2.11), e.g. with running GD. Afterwards gaps are filled by solving the conditional MAP problem (e.g. with Gibbs sampling) for every slice in the dataset that features missing data.

The Chain model can be easily extended to also consider local spatial information during predictions. The idea here is to add neighboring nodes to each of the T nodes, forming a *Cross* (4-pixel neighborhood) model. As a result $\boldsymbol{Y}$ has $5 \cdot T$ components, while $(T-1) \cdot 30 + 4 \cdot T$ edges indicate the conditional independences.

A MRF that captures spatio-temporal dependencies like this might encounter problems when large spatial patches are unobserved (e.g. with cloud covered data). In this case most gap pixels would also have an unobserved neighborhood, and the randomness of Gibbs sampling predictions for these pixels would probably dominate the available temporal information. To still make use of spatial data, one could utilize the simpler Chain model for predicting neighboring pixels. This means that a Chain prediction for the whole SITS is computed first, and for Cross predictions the corresponding neighboring pixels are taken from the Chain result. Another option would be to also connect the neighborhood nodes in the MRF with temporal edges, leading to the *T-Cross* structure. Here all pixels are affected by the temporally available information, which could results in less random behavior. Figure 3.2 depicts possible graphical dependency structures of a Chain, Cross, and T-Cross model as described above, with $T = 3$.

These models obviously capture spatio-temporal dependencies, similar to the original STRFs which were introduced in Section 2.3.5. In fact the T-Cross model is a STRF (just without diagonal temporal edges), with the underlying spatial structure resembling a cross (hence the name). The STRF structures originally model independences in the whole process, as several captured spatio-temporal process states are available for training. On the other hand, the models introduced above compute inference on small patches within the data, and are slided over the dataset for filling all gaps. However the novel models are just special adaptions, and the classic STRFs could also be used for gap filling with our sliding prediction approach. This means that our models could also be temporally reparametrized [58]. Based on this idea we further exploit the spatio-temporal structure in the compression approach (Section 3.2). By using a grid-like STRF one could also predict several pixels at once, and slide the model but patch-per-patch (instead of pixel-per-pixel).

Obviously even more complex patch forms and graphical dependency structures could



(a) Chain model (b) Cross model (c) T-Cross

Figure 3.2: Graphs of different temporal models for the gap filling task on remote sensing data with $T = 3$. The Chain model (a) only considers temporally available information during prediction, while the Cross model also relies on the spatial 4-pixel neighborhood (Cross model) (b). The 4-pixel T-Cross model (c) features more temporal edges, as each node is temporally connected.

be chosen here, however it is questionable if the increase in complexity really pays off. More edges allow to consider more local information for gap filling, however it also increases the model complexity and thus, the time needed for training and prediction. It should also be noted that some MRF structures (including our T-Cross) deny the exact computation of probabilistic inference due to cycles (see Section 2.3.3). A comparison of gap filling results with different model structures can be found in Section 4.2.

3.1.3 Reducing the Computational Complexity

Looking at the computational effort of probabilistic gap filling reveals a remaining problem. The high radiometric resolution of sensor bands leads to a colossal state space size, which has a high impact on the model complexity. For example considering a model which processes very simple satellite data of five bands, with each band featuring $\Omega_i = [0,255]$ different reflectance values. Accordingly the state space size of each component of $\boldsymbol{X}$ would be $|\boldsymbol{\Omega}| = 256^5 \approx 1.1\text{e}12$. This means that each edge in the constructed MRF model would be represented by $256^{10} \approx 1.2\text{e}24$ parameters, which makes the computation of probabilistic inference practically impossible. In remote sensing the radiometric resolution is usually way higher, and most data products feature more than five bands.
The solution for this is to drastically reduce the satellite data complexity, for example with VQ (shortly discussed in Section 2.3.6). For this scenario it makes sense to only quantize the valid intensities, as faulty pixels are not presented to the model and would probably deteriorate the quantization results. By replacing each valid multidimensional intensity value $\boldsymbol{I}(t,x,y)$ in the SITS with one of the k prototype intensities $\{\tilde{\boldsymbol{I}}_1, \tilde{\boldsymbol{I}}_2, ...\tilde{\boldsymbol{I}}_k\}$, the number of parameters per edge is reduced to k^2. Accordingly quantization results in a new image series $\tilde{\boldsymbol{\mathfrak{I}}} = \{\tilde{\boldsymbol{I}}(t,x,y) \in \mathbb{R}^b\}$. The predicted quantized values could even be further improved by replacing them with better estimations, based on more information from the quantization process or locally observed data. Consequences of quantization are shortly discussed in Section 4.2.1.

3.1.4 Training Probabilistic Models with Missing Data

A big problem arises from data incompleteness. It would be desirable to have a probabilistic gap filling approach which can be applied in all scenarios, no matter how much of the data is missing. However the random sampling of training data might result in partially unobserved patches. Limiting the sampling to fully observed patches does not really help, as it can highly restrict the data for training. In some cases it might even be impossible to find patches which do not feature any missing data.
Generally it cannot be guaranteed that patches sampled for training are completely observed, which creates demand for a training routine supporting partially missing data. Unfortunately this disqualifies simply minimizing the loss function (Equation 2.11), as the estimated average sufficient statistics $\tilde{\boldsymbol{\mu}}$ cannot be computed if parts of the data are unobserved.
This problem can be solved by running the whole training as an EM algorithm, as introduced in Section 2.3.4. The expectation step can be executed by (approximately) solving the conditional MAP problem (Equation 2.9) for the unobserved training data in D, e.g. with Gibbs sampling. As intended this prediction is based on the current model parameters. The average sufficient statistics $\tilde{\boldsymbol{\mu}}$ are re-estimated after every expectation step. The

maximization is then performed by running a regular training routine that demands fully observed data, e.g. minimizing the loss with GD. The number of outer EM iterations Q (i.e. number of expectation and maximization steps) affects the amount of time needed for training, as well as the quality of the resulting model. Experiments in Chapter 4 inspect the impact of EM training in terms of convergence, training runtime, and model quality.

3.1.5 Incorporating Parameter Priors for Optimization

Even with EM training for supporting missing data, a specific problematic scenario remains. Let us assume that we have a spatio-temporal dataset $\mathfrak{I}$ that is completely unobserved on a specific acquisition date, e.g. due to total failure of sensors. We want to train a Chain MRF model for this data, i.e. the model shall utilize only available temporal information for a pixel. However all sampled training patches have missing intensities for pixels at this unobserved acquisition date, i.e. one node of our model is always unobserved. The weights are initialized randomly, so the first conditional MAP state of this random variable component (computed during the first EM iteration) is totally random. Accordingly the model will then learn parameters that fit this random state (or rather, the estimated average sufficient statistics resulting from the conditional MAP state). This might already be problematic if acquisition dates exist where the vast majority of pixels is unobserved, because here the sufficient statistics are dominated by the few observed pixels.
Obviously this scenario is also problematic for other spatial gap filling approaches. However most temporal methods would simply compute interpolated intensities based on the temporally available information, instead of random values. To pave the way for similar behavior of probabilistic models, it is necessary to incorporate prior knowledge about parameters. This can be achieved by adding a regularization to the objective function, as described in Section 2.3.4 (Equation 2.13). The commonly used squared-l_2-regularization can be interpreted as enforcing prior knowledge about the parameters, such that all parameters take values near zero:

$$R(\boldsymbol{\Theta}) = ||\boldsymbol{\Theta}||_2^2 = ||\boldsymbol{\Theta} - \mathbf{0}||_2^2 = ||\boldsymbol{\Theta} - \boldsymbol{\sigma}||_2^2, \tag{3.1}$$

with $\mathbf{0}$ being the d-dimensional zero vector and $\boldsymbol{\sigma} \in \mathbb{R}^d$ being the *prior vector*. As shown above prior knowledge can be incorporated by choosing values for $\boldsymbol{\sigma}$, which contains a prior value for each parameter $\boldsymbol{\Theta}_i \in \boldsymbol{\Theta}$. In the remainder of this thesis normal squared-l_2-regularization is referred to as *L2 Zero* (L2Z) prior. Simply put prior knowledge regularization will push the model parameters towards $\boldsymbol{\sigma}$, as it is costly (in terms of loss) to move away from the prior values.
The question remains how values for $\boldsymbol{\sigma}$ can be computed. In general each parameter describes a specific state transition, either over time or spatially (depending on the associated edge in the MRF). Accordingly each prior value $\boldsymbol{\sigma}_i \in \boldsymbol{\sigma}$ functions as a guideline for the (logarithmized) probability of this state transition. In the following we make some suggestions on how prior values can be computed.
One approach is to prioritize state transitions based on state similarity. In the remote sensing context the k states are the b-dimensional quantized centroid intensities, hence we named this the *intensity-based similarity* (IBS) prior. Similarity can be measured by

first computing the Euclidean distances between the centroid vectors:

$$D = \begin{bmatrix} 0 & d_{12}^2 & d_{13}^2 & \dots & d_{1k}^2 \\ d_{21}^2 & 0 & d_{23}^2 & \dots & d_{2k}^2 \\ d_{31}^2 & d_{32}^2 & 0 & \dots & d_{3k}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{k1}^2 & d_{k2}^2 & d_{k3}^2 & \dots & 0 \end{bmatrix}, \text{ with } d_{ij}^2 = ||\tilde{\boldsymbol{I}}_i - \tilde{\boldsymbol{I}}_j||_2^2. \tag{3.2}$$

Obviously state transitions from the same state into the same should have the highest probability. Therefore D needs to be be inversely scaled, i.e. all distances are projected onto the interval $[p_l, p_h]$, with the highest distances obtaining the p_l-value and the smallest distances resulting in p_h. This gives the IBS matrix containing probability priors between $[p_l, p_h]$. Simply put, the IBS prior for a parameter is higher, if the centroid intensities of the associated states are more similar. Elements of $\boldsymbol{\sigma}$ take values from this matrix, based on the state transition that they describe.
Another approach is to compute a *log transition probability* (LTP) prior, which is based on the (logarithmized) empirical probabilities of state changes. Here it makes sense to compute different priors for temporal and spatial edges, as the empirical probabilities for temporal and spatial state changes are different. The logarithmized empirical temporal state transition probabilities can be computed from the quantized image series $\tilde{\boldsymbol{\mathfrak{I}}}$ as follows:

$$P_t = \begin{bmatrix} \hat{p}_{11} & \hat{p}_{12} & \dots & \hat{p}_{1k} \\ \hat{p}_{21} & \hat{p}_{22} & \dots & \hat{p}_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{p}_{k1} & \hat{p}_{k2} & \dots & \hat{p}_{kk} \end{bmatrix}, \text{ with } \hat{p}_{ij} = \log \frac{\sum_{t=2}^{T} \sum_{x}^{W} \sum_{y}^{H} \gamma_t(t,x,y,i,j)}{\sum_{t=2}^{T} \sum_{x}^{W} \sum_{y}^{H} \sum_{m}^{k} \sum_{n}^{k} \gamma_t(t,x,y,m,n)} \tag{3.3}$$

and $\gamma_t(t,x,y,m,n)$ indicating if a temporal state transition from $\tilde{\boldsymbol{I}}_m$ to $\tilde{\boldsymbol{I}}_n$ takes place:

$$\gamma_t(t,x,y,m,n) = \begin{cases} 1 & \text{if } \tilde{\boldsymbol{I}}(t-1,x,y) = \tilde{\boldsymbol{I}}_m \wedge \tilde{\boldsymbol{I}}(t,x,y) = \tilde{\boldsymbol{I}}_n \\ 0 & \text{else} \end{cases} \tag{3.4}$$

By normalizing each row in P_t (i.e. $\sum_j^k \hat{p}_{ij} = 1 \quad \forall 1 \leq i \leq k$), one obtains the temporal LTP prior matrix. In a similar fashion it is possible to compute the LTP prior for spatial edges, e.g. based on the empirical spatial state change probabilities for a 4-pixel neighborhood:

$$\hat{p}_{ij} = \log \frac{\sum_{t}^{T} \sum_{x=2}^{W} \sum_{y=2}^{H} \gamma_s(t,x,y,i,j)}{\sum_{t}^{T} \sum_{x=2}^{W} \sum_{y=2}^{H} \sum_{m}^{k} \sum_{n}^{k} \gamma_s(t,x,y,m,n)} \tag{3.5}$$

and

$$\gamma_s(t,x,y,m,n) = \begin{cases} 2 & \text{if } \tilde{\boldsymbol{I}}(t,x-1,y) = \tilde{\boldsymbol{I}}_m \wedge \tilde{\boldsymbol{I}}(t,x,y-1) = \tilde{\boldsymbol{I}}_m \wedge \tilde{\boldsymbol{I}}(t,x,y) = \tilde{\boldsymbol{I}}_n \\ 1 & \text{if } \tilde{\boldsymbol{I}}(t,x-1,y) = \tilde{\boldsymbol{I}}_m \wedge \tilde{\boldsymbol{I}}(t,x,y) = \tilde{\boldsymbol{I}}_n \\ 1 & \text{if } \tilde{\boldsymbol{I}}(t,x,y-1) = \tilde{\boldsymbol{I}}_m \wedge \tilde{\boldsymbol{I}}(t,x,y) = \tilde{\boldsymbol{I}}_n \\ 0 & \text{else} \end{cases} \tag{3.6}$$

The values in $\boldsymbol{\sigma}$ are taken from the LTP prior matrices, depending on the (spatial or temporal) state transition that the corresponding parameter describes.

As introduced earlier, regularization is commonly weighted with λ (see Equation 2.13). However when different priors for spatial and temporal parameters are used, it might make sense to weight the regularization with different values:

$$\min_{\boldsymbol{\Theta}\in\mathbb{R}^d} f(\boldsymbol{\Theta}) + \lambda R(\boldsymbol{\Theta}) = \min_{\boldsymbol{\Theta}\in\mathbb{R}^d} f(\boldsymbol{\Theta}) + \lambda ||\boldsymbol{\Theta} - \boldsymbol{\sigma}||_2^2 = \min_{\boldsymbol{\Theta}\in\mathbb{R}^d} f(\boldsymbol{\Theta}) \sum_i^d \lambda_i (\boldsymbol{\Theta}_i - \boldsymbol{\sigma}_i)^2 \quad (3.7)$$

This allows to weight each single parameter prior differently, and by assigning $\lambda_i \in \{\lambda_t, \lambda_s\}$ depending on the associated edge, the temporal and spatial regularization can be weighted differently. If the STRF model also contains diagonal temporal edges as described in [58], it might also make sense use another designated prior and weight it independently.
All those priors were tested in experiments with different weights, they are discussed in Section 4.2.2 and Section 4.2.4. Obviously one could test even more priors, e.g. by computing the LTP prior over a local temporal window instead of the full data. Easily incorporable prior knowledge is a big advantage of the probabilistic gap filling approach, as it allows a lot of fine-tuning. Moreover the prior regularization could be adapted to MRF extensions, e.g. Piatkowski, Lee, and Morik's STRF reparametrization. In this specific case the trained parameters are $\boldsymbol{Z}$ [58], and the prior values $\boldsymbol{\sigma}$ need to be recalculated in a similar fashion.

3.2 Model Compression with Parameter Series Cluster Sharing

The high complexity of probabilistic models is probably their highest drawback. However the temporal dimension of STRFs allows regularization [58] and compression by merging parameters and thus reducing their size. Following up we present our novel *parameter series cluster sharing* (PSCS) compression approach, which first subdivides the whole set of parameters and then compresses the resulting parameter series.

3.2.1 Temporal Parameter Series in Probabilistic Models

The parameters of any STRF $G = (V, E)$ with underlying spatial dependency structure $G_0 = (V_0, E_0)$ can be subdivided into parameter series. As described in Section 2.3.5 edges of these graphical structures can be divided into spatial and temporal edges.
A single parameter $\theta_{\text{spat}}(v_i, v_j, t) \in \boldsymbol{\Theta}$ of a spatial edge describes a specific state transition between replicas of nodes $v_i, v_j \in V_0$ at time $1 \leq t \leq T$. The temporal edges are represented by parameters $\theta_{\text{temp}}(v_i, v_j, t) \in \boldsymbol{\Theta}$. They describe the temporal transition from t to $t+1$ between replicas of nodes $v_i, v_j \in V_0$ with $1 \leq t \leq T-1$ ($v_i \neq v_j$ for diagonal edges). Accordingly each parameter belongs to a set of parameters $\boldsymbol{\Theta}_{\text{spat}}(v_i, v_j)$ or $\boldsymbol{\Theta}_{\text{temp}}(v_i, v_j)$, which describe the same data relation at different temporal sections, i.e. edges of the model.
This subdivision of $\boldsymbol{\Theta}$ is exemplary displayed in Figure 3.3. Each set $\boldsymbol{\Theta}_{\text{spat}}(v_i, v_j)$ or $\boldsymbol{\Theta}_{\text{temp}}(v_i, v_j)$ contains T or $T-1$ parameters, and can be interpreted as a time series of parameters (hence the name *parameter series*).

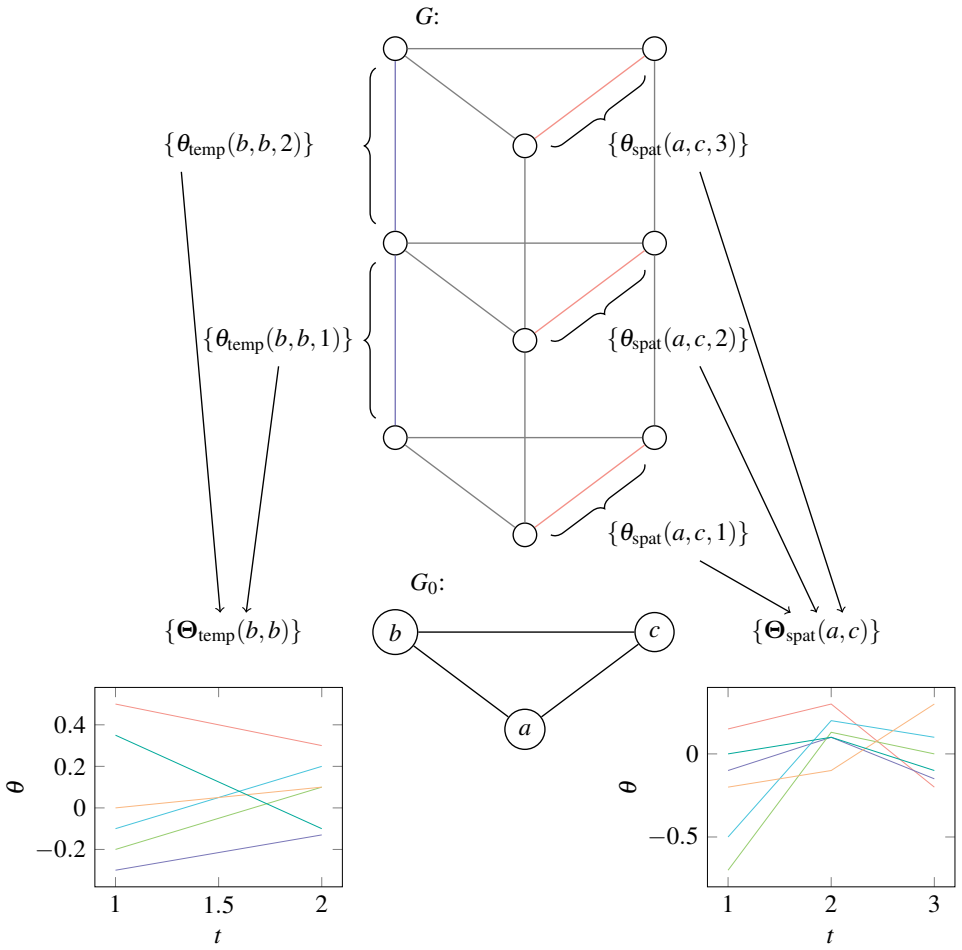


Figure 3.3: Example for a STRF G based on a simple spatial graph G_0 with $T = 3$ (without diagonal edges), whose parameters can be interpreted as parameter series.

3.2.2 Cluster Sharing for Parameter Series

For complex models the parameter series of different edges and nodes will probably show similarities. This is particularly the case for STRFs which are used in a sliding way (as described in Section 3.1.2), as here multiple spatial nodes represent any arbitrary pixel in the image over time. Even with STRFs describing a whole process, there is a high chance that matters might evolve similarly in different regions of the model.
We therefore propose to compress the set of parameter series, meaning that each series $\Theta_{\text{spat}}(v_i, v_j)$ or $\Theta_{\text{temp}}(v_i, v_j)$ is replaced by a specific cluster centroid series $\tilde{\Theta}_{\text{spat}}(v_i, v_j)$ or $\tilde{\Theta}_{\text{temp}}(v_i, v_j)$. As a parameter series is essentially a T- or $(T-1)$-dimensional vector, regular methods of VQ can be easily applied for finding good centroids (see Section 2.3.6). In a model which has been compressed with this novel PSCS approach, several parameter series share the values of a clustered centroid parameter series.
PSCS compression can drastically decrease memory requirements of the model, without reducing the level of detail too much. Moreover higher compression results in a more general model, and hence might alleviate overfitting. The compression rate is parametrized by the number of temporal and spatial cluster centroids c. As an example $c = k^2$ would indicate that all spatial (or temporal) edges share the same state transition probabilities over time (with k^2 being the state space size per node). With the number of initial parameter series and the number of temporal and spatial cluster centroids c, it is possible to compute the compression rate. For example a model with 1024 different parameter series, which are being replaced by $c = 128$ clustered centroids, would be compressed to $128/1024 \cdot 100 = 12.5\%$ of its original complexity.

3.3 Implementation Details

The introduced methodology for probabilistic gap filling was implemented[1] and evaluated in experiments. Some implementation details are discussed in the following, starting off with the C++[2] implementations for probabilistic computations.
As this part of the software was not developed for the specific remote sensing context, several Python 3 scripts[3] were implemented, which process and shape the satellite data for the probabilistic computations. Information about this code is provided in the second part of this section. The thesis code is available on the data medium and might be uploaded publicly at a later time.

3.3.1 Implementations for Probabilistic Computations

For working with probabilistic models Piatkowski's *PX* framework[4] was utilized and slightly extended.
PX is a collection of various classes and routines in raw C++, which essentially allow computations with discrete undirected probabilistic models. The software contains different approaches for computing probabilistic inference, implementations of graphical dependency structures, several optimization algorithms that can be used in an EM fashion, the computation of estimated average sufficient statistics, and much more. The

[1] `https://github.com/chaosdev94/mt-code/`
[2] `https://isocpp.org/`
[3] `https://www.python.org/`
[4] `https://bitbucket.org/np84/px/`

STRF approach is implemented as well, based on a spatial graph and a given value for T [58]. Users can choose between several forms of spatial structures, or simply specify a custom graph with an adjacency matrix. Training results in a binary model file that can be inspected or used for predictions. Training or prediction data is given in form of comma-separated value (`.csv`) files, with each line containing a full realization of the random variable $\boldsymbol{X}$ (the `?` character denotes unobserved components). In general PX is well optimized and large parts support parallelization over multiple cores. The framework core is compiled into a shared library, which is used by the different routines such as `px-train` or `px-predict`.
Small code changes were done for allowing code compilation on Windows with Microsoft Visual C++[5], such as adding corresponding compiler intrinsics. A new graph class which only connects a single node over time was added to `PXGRAPH` and can be used for Cross model computations. For prior regularization several adjustments were made in `PXMISC`, `px-train.cpp`, and `PXOPT`. The `GradientDescentL2EdgePrior` class in `PXOPT` is used for optimizing with a prior regularization. The changes in `PXMISC` and `px-train.cpp` allow to load prior matrices and weight the regularization differently, as described in Section 3.1.5. `px-manipulate.cpp` was added for compressing previously trained models (see Section 3.2). It loads a trained model and exports (or imports) temporal and spatial parameter series for compression.
PX features several options and routines, however only some of them were used for experiments. Training is run with the following command:

Listing 3.1: PX Training Command

```
./px-train -Y 1 -s 4 -I $I -p $p -Q $Q -T $T -o $plist
   -l $l_t $l_s $l_d $train $adj $mname    .
```

`-Y 1` assigns the state space of each node in the model to be similar. `-s 4` chooses the novel prior regularized GD as optimization routine, while `$I` and `$p` control the relative stop criteria I and ρ as described in Section 2.3.4. With setting `-Q` the training is run in an EM fashion with `$Q` outer iterations (see Section 3.1.4). `$T` determines the number of spatial structure copies which together form the spatio-temporal graph (i.e. the number of acquisition dates T). A comma separated list of prior matrices is given with `$plist`, the individual weights for temporal, spatial, and diagonal parameters are set to `$l_t`, `$l_s` and `$l_d`. `$train` is the file with data samples used for training, the spatial structure is generated based on the given adjacency matrix `$adj`. PX writes the model file `$mname` as soon as the training is finished. For other parameters the default values were used, i.e. LBP was used for approximating probabilistic inference with termination bound 100, and Gibbs sampling approximately solves the conditional MAP problems.
Predictions can be computed with the command

Listing 3.2: PX Prediction Command

```
./px-predict $pred $mname
```

with `$pred` being the `.csv` file for prediction and `$mname` being the model name. This routine results in a `.predicted` file where all missing values in `$pred` are filled. PX was designed to make a full prediction instead of sliding the trained model over a bigger

[5] `https://docs.microsoft.com/en-us/cpp/?view=vs-2017`

dataset, as it is the case with the gap filling task. One option to deal with this is to extract all patches with missing data, list them in the `$pred` file, run `px-predict`, and afterwards assemble the full prediction from all lines. A specialized script was implemented allowing the use of predicted data for following patches.

3.3.2 Implementations for Satellite Data Processing

The processing of remote sensing data was implemented in Python scripts. Each script is roughly discussed here, more detailed information is available by running scripts with the `-h` help argument.
Satellite data was accessed with the *Open Source Geospatial Foundation* (OSGeo) software[6] and the corresponding *osgeo* package for Python. The `SatDataAccess.py` script is responsible for loading the data, quantizing it with `scikit-learn`'s k-means clustering implementation [53], and adding artificial clouds based on a *JavaScript Object Notation* (JSON) file. It also reads data from OTB's linear temporal interpolation as described in Section 2.2.3. The script creates a range of `.npy` files which are used by other scripts, containing the data, cloud masks, and clustering information. `SatDataSampler.py` is used for sampling the data for training. It also generates a file containing the data for prediction, as well as the adjacency matrix describing the spatial model structure. The `SatDataPriorComputation.py` script allows to compute prior matrices as described in Section 3.1.5, which can be loaded with PX.
The whole training and prediction with MRF models is executed with the extended PX software, as described above. For simplicity only the centroid indices are used during the probabilistic computations. This means that the state space for each node in the MRF model is simply the set of centroid indices $\mathcal{X}_i = \{1, 2..., k\}$.
`PatchOverlapPrediction.py` is a specialized script which allows to use predicted data during following predictions. This is helpful for models with spatial neighborhood, as here the predicted intensities of the centered pixel should spatially influence the next prediction. It internally calls `px-predict` as a subroutine for each single patch.
The resulting `.predicted` files can be evaluated with `SatDataPredEvaluation.py`, which replaces the centroid indices with quantized intensities. The script also writes quality reports for the whole data and for each acquisition date, if artificial clouds were added. Moreover it generates pictures that show the visual results of gap filling, as well as another `.npy` file containing the predicted gap data. Rerunning `SatDataAccess.py` with the `-w` option allows to store the gap filled satellite data in the original format.
The `ModelCompressor.py` script executes the PSCS compression. It reads, clusters, and writes the quantized parameter series from and into the files which have been generated by `px-manipulate.cpp`. Clustering is again computed with `scikit-learn`'s k-means implementation.

[6] `https://www.osgeo.org/`

Chapter 4

Experiments

Several experiments were run with the implementations of our methodology (see Section 3.3 for details). The obtained results show the feasibility of our proposed methods and provided insight into issues and benefits. This chapter provides an extensive overview of the experiments, the obtained qualitative and quantitative results, as well as possible interpretations.
Starting off Section 4.1 conveys general experimental background information. Section 4.2 discusses the probabilistic gap filling experiments. This includes exploration of different hyperparameter choices, a comparison with a different gap filling approach, and the evaluation of quantitative and visual results. Lastly the experiments for our novel compression approach for STRFs are discussed in Section 4.3.

4.1 General Information on the Experiments

Some general information is crucial for understanding the experiments. Therefore this section introduces the experiment data, the technique for evaluation, and quality measures that are used later on.

4.1.1 Data and Environment for Gap Filling Experiments

The data used in the experiments originates from the Sentinel-2 earth observation mission [16]. It globally covers most land and sea surfaces, revisits each location every five days, and has a free data policy. The satellites' data features 13 bands in the visible, near infrared, and short wave infrared spectrum and is especially interesting because of its high spatial resolution (10, 20 and 60m). It makes Sentinel-2 data valuable for many use case scenarios, such as monitoring crops or vegetation [13], flood mapping (e.g. for disaster management) [17] or the generation of detailed land cover maps [71]. Different preprocessed products of the raw data are available. We chose THEIA's product format at level 2A[1] for our experiments, following up the corresponding preprocessing steps are shortly explained.
THEIA L2A data is already atmospherically corrected by the *MACCS-ATCOR Joint Algorithm* (MAJA) [25] and comes with cloud masks, which identify gaps. First step in THEIA L2A data correction is estimating the atmospheric water vapor content. It

[1] `https://theia.cnes.fr/atdistrib/rocket/`

is followed by the correction of gaseous absorption, which uses the SMAC model [59]. MAJA's classification of clouds is based on steep increases in the blue surface reflectance band over time and correlations in the local pixel neighborhood of preceding images [24]. After that the aerosol optical thickness is estimated, which is then used to retrieve the corrected surface reflectances. Final step in preprocessing is the correction of adjacency and terrain slope effects.
THEIA L2A data is subdivided into tiles of (10000×10000) pixels at T different acquisition dates (T varies from tile to tile). It features $d = 10$ of the original 13 Sentinel-2 bands and binary cloud information for each entry. The data of a single tile can be interpreted as an image series which allows uncomplicated visualization, for example with OSGeo[2] (exemplary visualizations were already shown in Figure 2.2). The thesis experiments were run on a subregion of (1000×1000) pixels in the *T31UDQ* tile. The featured data was measured for France on $T = 31$ acquisition dates in 2016. A single image in the dataset is referenced by its date

$$t \in \{2, 12, 25, 42, 65, 72, 75, 82, 92, 105, 122, 125, 172, 175, 192, 222, 225,$$
$$235, 245, 245, 252, 265, 272, 282, 285, 335, 342, 345, 352, 355, 362\}$$

which denotes the t-th day of 2016 (obviously it can be recomputed to the specific date). About half of the 31000000 pixels in the dataset are identified as clouded. Experiments were run remotely on NeCTAR Research Cloud[3], a collaborative Australian research platform which is supported by Australia's *National Collaborative Research Infrastructure Strategy* (NCRIS)[4].

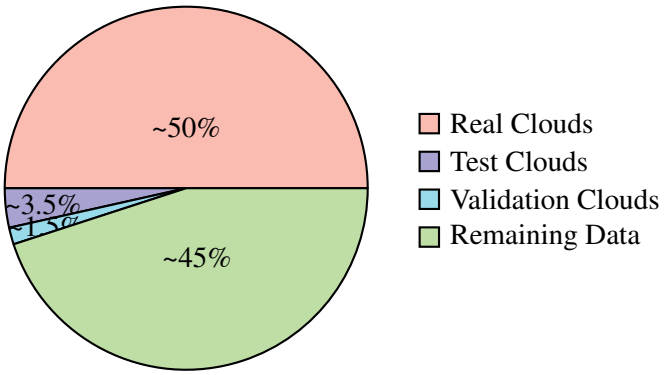
4.1.2 Adding Artificial Clouds for Evaluation

As described in Chapter 3, the introduced approach for gap filling is highly customizable, due to several hyperparameters and routines. This creates demand for measuring the quality of models with different hyperparametrizations. Artificial clouds were randomly added to the data, allowing the comparison of observed data with the model prediction and hence the computation of quality measures. Each synthetic cloud covers 6584 pixels and has the shape of a rectangle with rounded edges, the smallest cloud shape detected in THEIA L2A masks. Obviously the data for artificially concealed pixels is also unobserved during training.
Validation experiments were run to evaluate the impact of changing certain hyperparameters. The succeeding *test* experiments were run with the best parameters and led to final quantitative and qualitative results. About 3.5% of the intensities were artificially clouded for test experiments, and more clouds were added (concealing another 1.5%) for finding good hyperparameters. In total, 165 clouds were added for testing, and 70 clouds for validating. This means that roughly 10% of the originally unclouded data is artificially clouded for evaluation. Obviously the pixel data concealed for validation is observed during test experiments, leading to a slightly higher amount of available information. Figure 4.1 shows the proportion of real, test, and validation clouds over the whole dataset, for each acquisition date, and examples of resulting cloud masks.

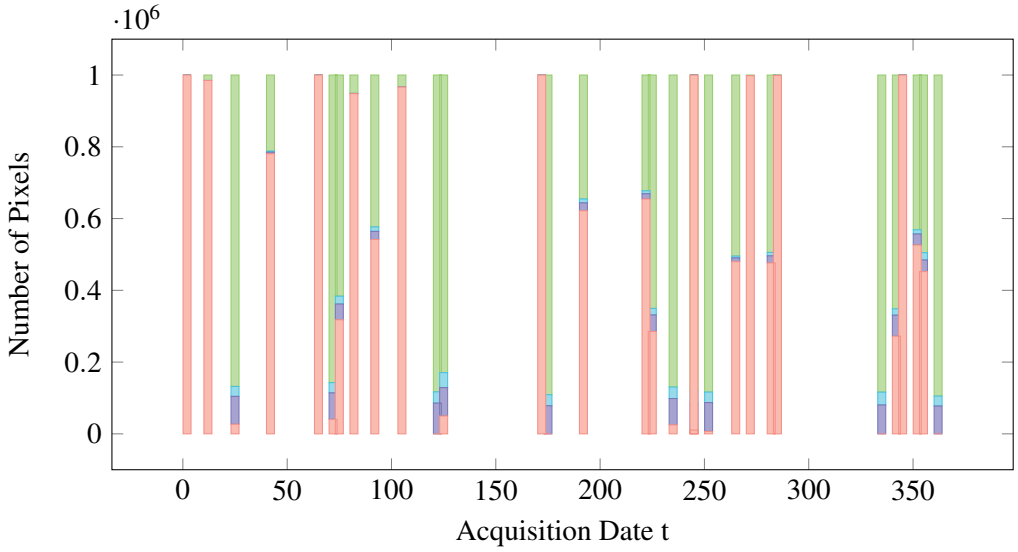
[2] `https://www.osgeo.org/`

[3] `https://nectar.org.au/research-cloud/`

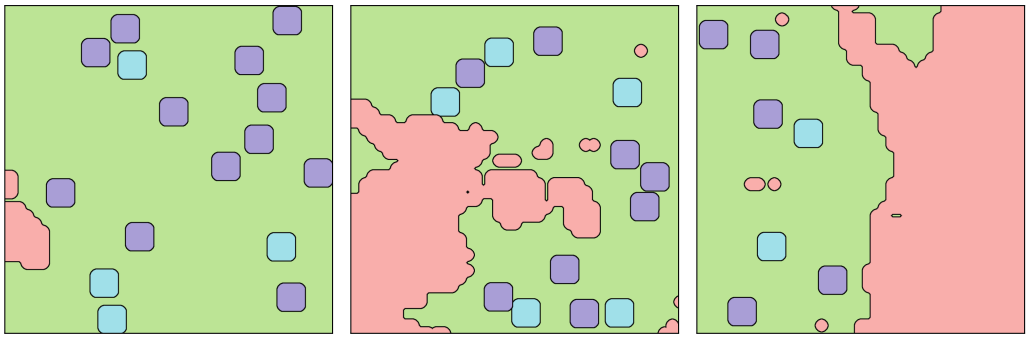
[4] `https://www.education.gov.au/national-collaborative-research-infrastructure-strategy-ncris`



(a) Overall proportion of clouds



(b) Cloud proportion for each acquisition date



(c) Cloud masks at $t = 235$ (d) Cloud masks at $t = 342$ (e) Cloud masks at $t = 355$

Figure 4.1: Artificial clouds were added randomly, allowing evaluation of results. The proportion of clouds in the data can be analyzed over the whole series (a) or for each acquisition date (b). Resulting exemplary cloud masks are shown for three different dates (c - e).

The artificial clouds also allow to run different gap filling approaches and compare their results. OTB's linear temporal interpolation (see Section 2.2.3) was used as a baseline approach for comparison, labeled as *Baseline* in the following sections. As the probabilistic models can only predict quantized intensities, the interpolated values were also projected onto the nearest quantization centroid.

4.1.3 Quality Measures for the Evaluation

Mean absolute error (MAE) and *Error* (ERR) are computed for the n clouded pixels to measure the quantitative prediction quality. MAE here denotes the mean absolute intensity error per band and pixel:

$$\text{MAE} = \frac{1}{d \cdot n} \sum_{i=1}^{n} \sum_{k=1}^{d} |\boldsymbol{I}_k(t_i, x_i, y_i) - \hat{\boldsymbol{I}}_k(t_i, x_i, y_i)| \tag{4.1}$$

(t_i, x_i, y_i) denotes the location of the i-th sample, and $\boldsymbol{I}_k$ (as well as $\hat{\boldsymbol{I}}_k$) describe the k-th band intensity in the reference (or prediction) data entry. The MAE is also used to compute the quantization error MAE_Q by using the quantized intensities $\tilde{\boldsymbol{I}}$ instead of $\hat{\boldsymbol{I}}$. Obviously MAE_Q is contained within MAE, and the remaining prediction error is denoted as MAE_P, such that $\text{MAE} = \text{MAE}_Q + \text{MAE}_P$. ERR denotes the empirical probability that the predicted value is not equal to the quantized originally observed intensity (i.e. $ERR = 1 - \text{Accuracy}$):

$$\text{ERR} = \frac{1}{n} \sum_{i=1}^{n} \begin{cases} 0 & \text{if } \hat{\boldsymbol{I}}_k(t_i, x_i, y_i) = \tilde{\boldsymbol{I}}_k(t_i, x_i, y_i) \\ 1 & \text{else} \end{cases} \tag{4.2}$$

Dealing with THEIA L2A data results in $d = 10$ bands, while the number of evaluated pixels is $n = 165 \cdot 6584 = 1086360$ for testing, $n = 70 \cdot 6584 = 460880$ for validation and $n = 15535050$ for the quantization experiments (number of artificially clouded pixels and number of unclouded intensities in the whole image series).
All experiments in this thesis were run eight times, as the placement of artificial clouds, the sampling of training data, and the approximation of probabilistic inference result in highly non-deterministic behavior. Accordingly all measured qualities are averaged over all eight runs. The results which cannot be easily merged (e.g. gap-filled images or convergence computations) are all taken from the first run. In some experiments we also evaluate the *standard deviation* (SD) of the quality measures over all runs, i.e. $\text{SD} = \sqrt{\frac{1}{8}\sum_i^8 (z_i - \bar{z})^2}$, with z_i denoting a quality measure result from the i-th run, and $\bar{z}$ denoting the averaged quality measure over all runs.

4.2 Experimental Gap Filling with Probabilistic Models

The first experiments focus on the suggested probabilistic gap filling approach, using Markov models with dependency structures introduced earlier (see Section 3.1.2 for more details). We first explore the impact of quantization, prior regularization, and other hyperparameters on the model quality with the help of validation clouds. Based on those findings the quantitative and visual performance of trained models on the test clouds is evaluated.

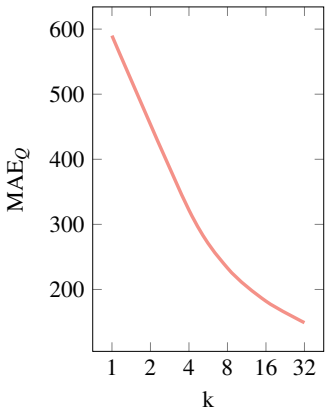
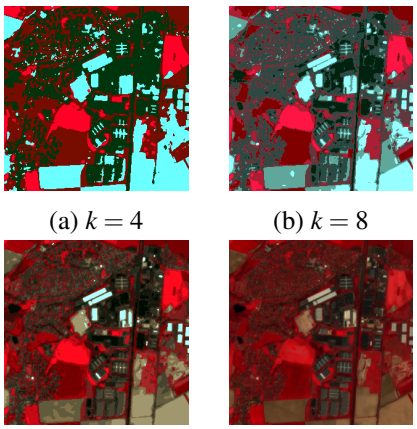


Figure 4.2: MAE$_Q$ for k-means quantized satellite data, with varying number of centroids k (SD $\pm$6.7).



(a) $k = 4$ (b) $k = 8$

(c) $k = 16$ (d) Original

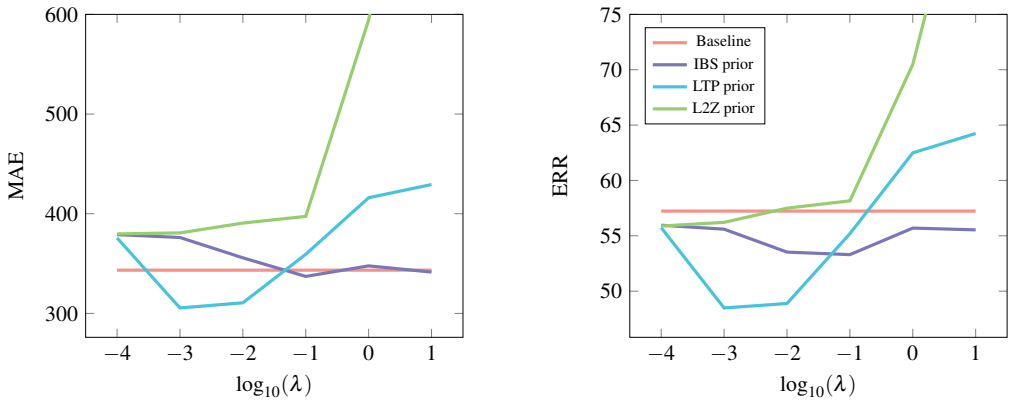
Figure 4.3: Visual k-means quantization results of an image subregion (d) with $k = 4$ (a), $k = 8$ (b) and $k = 16$ (c).

4.2.1 Results of Complexity Reduction with Quantization

The very first hyperparameter in the routine is the number of quantization centroids k, which regulates the state space size per node in the model (and thus the number of parameters per edge, i.e. the model complexity, as described in Section 3.1.3). It can actually be evaluated without artificial clouds, because the error is computed from the quantized and originally unclouded intensities. Quantizations and the resulting MAE$_Q$ were computed with different values of k, the error can be seen in Figure 4.2, while optical quantization results are depicted in Figure 4.3.
One can see that the MAE$_Q$ and visual level of detail converge relatively fast. Choosing the value for k is a balancing act of computation runtime versus quality of the trained model, $k = 16$ was chosen for all following experiments. Moreover a higher k requires a larger amount of training data for obtaining a robust estimation of the average sufficient statistics.

4.2.2 Benefits of Temporal Prior Regularization

The artificial validation clouds allow to measure the model quality, and hence the impact of changing hyperparameters for training. Finding good values for them is necessary, as the quality of predictions can drastically change with different configurations. We chose initial values for all hyperparameters, and then later explored the effects of changing them one by one.
The number of training samples n affects the sufficiency of the estimated average statistics $\tilde{\mu}$, it was set to $n = 100000$. As explained in Section 3.1.4, the training needs to be run in an EM fashion, as it has to deal with missing data. Q, the number of outer EM iterations, was set to $Q = 8$ in the experiments. I and ρ regulate the stop criteria for optimization, the number of iterations without improvement was set to $I = 100$ and the relative stop criteria for improvement was initially set to $\rho = 0.1$. Last but not least parameter regularization based on prior knowledge can be applied, it is the first hyperparameter that we evaluate in detail.



(a) MAE (SD max ±13.2) with different priors

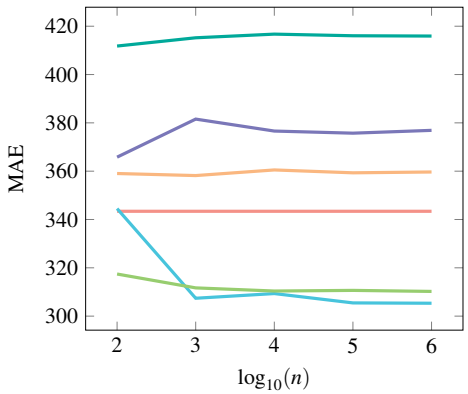
(b) ERR (SD max ±1.4) with different priors

Figure 4.4: MAE (a) and ERR (b) of models with varying temporal regularization priors and values for λ.

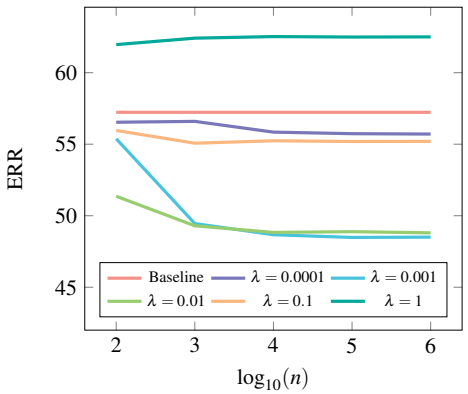
All of the suggested models have temporal edges, which is why we first inspect the impact of temporal regularization. The different priors suggested in Section 3.1.5 were tested with varying values for λ, Figure 4.4 depicts the resulting errors on the Chain model (which only has temporal edges).
It is obvious that the LTP edge prior outperforms the L2Z and IBS priors with a good λ. However it is interesting that the LTP prior performs worse than the others when it is weighted too strong. Apparently a good prior enables our gap filling method to beat the baseline in terms of quality. Results could probably be further improved by choosing λ between 0.001 and 0.0001.

4.2.3 Impacts of Other Hyperparameters

More Chain models with the LTP edge prior and varying λ were trained, with different values for Q, n, and ρ. Figure 4.5 shows that even with few training samples (e.g. $n = 100$) the Chain model can still outperform the baseline method. However a greater weighting of the edge prior regularization can be beneficial, if only a small amount of data ($n \leq 100$) is available for training. Also note that the error seems to converge for all curves, but λ slightly affects the point of convergence. The low impact of n on the quality is probably also rooted in the modest complexity of the Chain model, using more data also allows to train more complex models (e.g. by increasing k).
Effects of changing the value for Q can be seen in Figure 4.6. It shows that the benefit of iteratively rerunning the optimization is lower for training with a very large or small λ. This makes sense as the model is not allowed to adapt to the re-estimated average sufficient statistics (high λ), or is allowed to behave too randomly (small λ). Obviously only few iterations of EM training are needed for a converging error, however more iterations can still improve the results.
Experiments were run with different values for ρ, the quantitative results are displayed in Figure 4.7. It is interesting that the early optimization stopping with $\rho = 1$ leads to a way worse performance with a small regularization weight. It also shows that a lower value for ρ actually further decreases the error. However one has to keep in mind

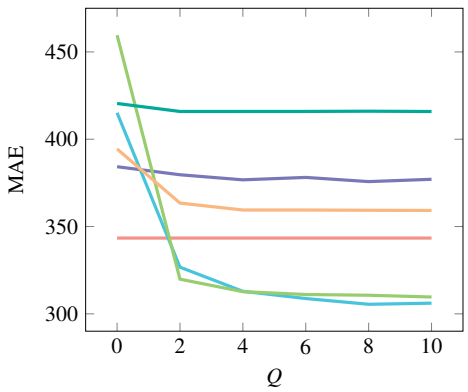


(a) MAE (SD max $\pm$10.6) with varying amount of training data

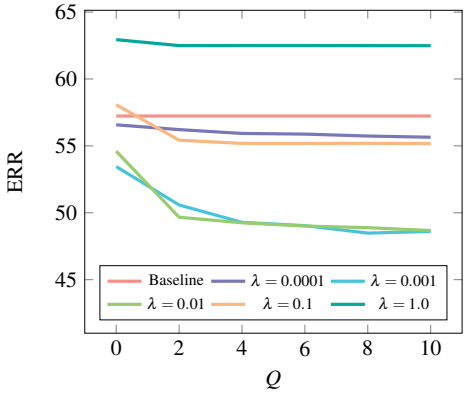


(b) ERR (SD max $\pm$1.3) with varying amount of training data

Figure 4.5: MAE (a) and ERR (b) of models with varying number of training samples n, trained with the LTP prior and different values for λ.

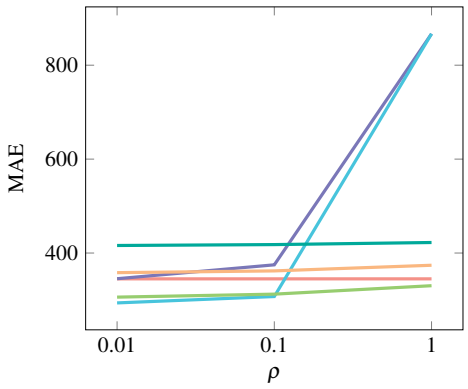


(a) MAE (SD max $\pm$11.1) with varying number of EM Iterations

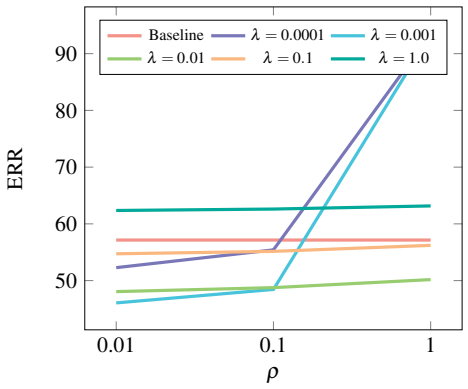


(b) ERR SD max $\pm$1.5) with varying number of EM Iterations

Figure 4.6: MAE (a) and ERR (b) of models with varying outer EM iterations Q, trained with the LTP prior and different values for λ.

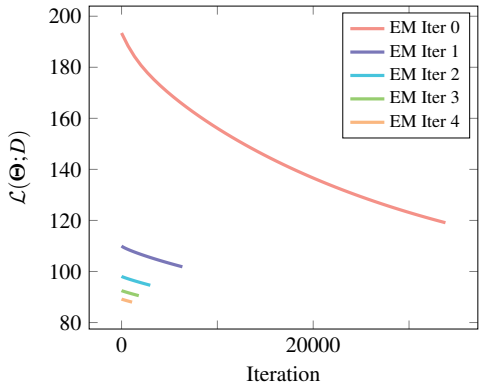


(a) MAE (SD max ±13.4) with varying stop criteria ρ

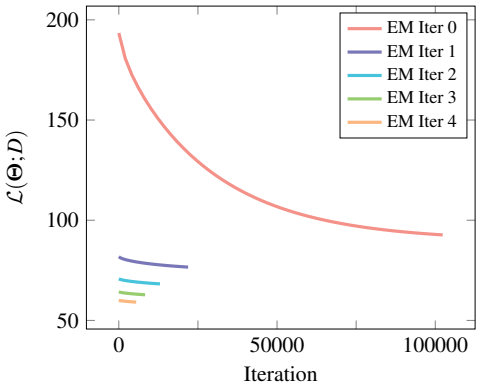


(b) ERR (SD max ±1.8) with varying stop criteria ρ

Figure 4.7: MAE (a) and ERR (b) of models with varying relative optimization termination criteria ρ, trained with the LTP prior and different values for λ.



(a) Convergence with $\rho = 0.1$



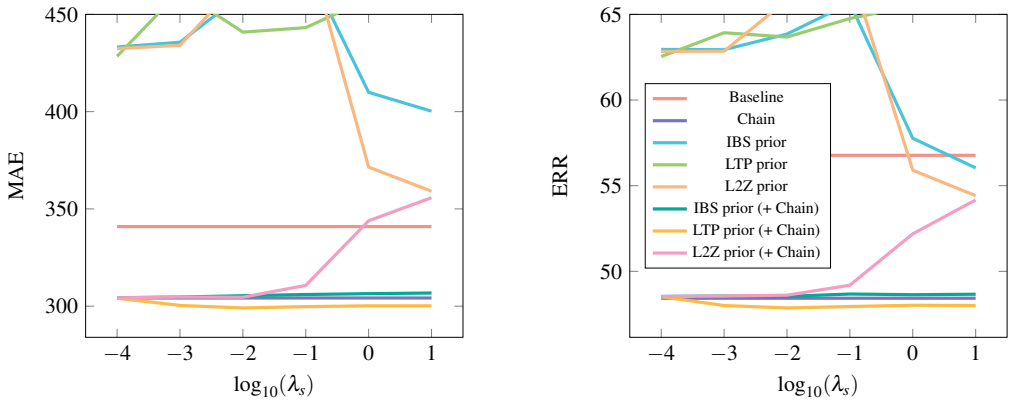
(b) Convergence with $\rho = 0.01$

Figure 4.8: Converging graphs of the objective function $\mathcal{L}(\Theta;D)$ during the training of the Chain model in different EM iterations with $\rho = 0.1$ (a) and $\rho = 0.01$ (b).

that early stopping makes models more general, and can thus alleviate overfitting the training data. Resulting convergence plots for the first EM iterations (*Iter*) are shown in Figure 4.8. Obviously the optimization is stopped way earlier with a lower ρ, however the error does not decrease drastically, and the more general model might even perform better on unknown data.
We conclude that $Q = 8$ and $n = 100000$ were well chosen, they are therefore fixed in all upcoming experiments without further tests. In the test experiments ρ was set to 0.01, to further improve the prediction quality.

4.2.4 Adding Spatial Prior Regularization

Considering local spatial information might help to further increase the model quality. However it requires a more complex graphical dependency structure, which also increases the complexity of probabilistic inference.
A lot of hyperparameters have already been discussed, and we here assume that they



(a) MAE (SD max ±11.3) with different priors

(b) ERR (SD max ±1.6) with different priors

Figure 4.9: MAE (a) and ERR (b) of models with varying spatial regularization priors and values for λ_s. The temporal edges are regularized with the LTP prior and $\lambda_t = 0.01$. The (+ Chain) results were achieved by first predicting the spatial neighborhood pixels with the Chain model.

will affect more complex models in a similar way. However the impact of spatial priors needs to be evaluated, as it does not make sense to regularize spatial edges with the temporal LTP prior (keep in mind that the LTP prior is different for spatial and temporal edges, as described in Section 3.1.5). Therefore Cross models were trained with the differently weighted LTP, L2Z, and IBS priors for spatial edges. The temporal edges were regularized with the LTP prior and $\lambda_t = 0.001$, as determined above.
In the initial experiments all of the trained Cross models performed significantly worse than the Chain model. The cause for that is probably rooted in the nature of clouds, which usually cover large regions. Accordingly the spatial neighborhood is most often unobserved for clouded pixels, and random behavior of the neighbors during sampling deteriorates the prediction (already mentioned in Section 3.1.2).
This problem can be overcome by first predicting the neighborhood of a clouded pixel with the Chain model, and then predicting values for the centered pixel with the Cross model. Figure 4.9 depicts the quantitative results for these experiments, *(+ Chain)* plots indicate that a neighborhood prediction with the Chain model was computed first. It shows that the Cross model can indeed have a higher quality than the Chain, when the spatial neighbors are observed (or predicted first).
Apparently the LTP prior also works best for spatial regularization, it provides better results than the other priors for all tested values of λ_s. Here regularization weighted with $\lambda_s = 0.01$ delivers the best results, however the choice of λ_s does not affect the quantitative quality much.
Another option for solving the problem of unobserved neighbors might be to use a T-Cross model. The newly added temporal edges could then also be regularized with the LTP prior, as mentioned above.

4.2.5 Quantitative Results of Probabilistic Gap Filling

After finding good hyperparameters the test experiments were run. Accordingly the quality measures are computed for the test clouds, the clouded validation data is now

Table 4.1: Quantitative results obtained with different models.

Method	MAE (± SD)	ERR (± SD)	Prediction time	Training time
Baseline	339.79 (± 12.3)	57.25% (± 1.6)	60*s*	-
Chain	288.32 (± 6.4)	46.55% (± 1.3)	3600*s*	10800*s*
Cross (+ Chain)	284.10 (± 6.4)	45.98% (± 1.3)	10800*s*	43200*s*

observed. $n = 100000$ samples were used for training a Chain and Cross model with the spatial and temporal LTP prior ($\lambda_t = 0.001$, $\lambda_s = 0.01$), and $Q = 8$ EM iterations (each one stopping after $I = 100$ iterations with improvement lower than $\rho = 0.01$).
Table 4.1 depicts the average MAEs, ERRs, SDs, approximated computational runtime of probabilistic gap filling, as well as the respective results of the baseline method. One can see that both Chain and Cross model by far outperform the baseline in terms of quality. They also appear to be more robust over all experiments, as the resulting SD is lower. This is interesting as the test cloud placement is the only random factor in the baseline gap filling routine, and apparently has a high impact on the interpolation quality. It also shows that spatial information can possibly increase the model quality, even with clouded data, however it also requires a significantly higher amount of training and prediction runtime, and the neighbors need to be predicted first.
On the other hand the computational runtime is obviously much higher than with the baseline method. This is expected, as the complexity of probabilistic inference is enormous compared to a simple linear interpolation. To make the best out of the time investment, it might make sense to train a model for gap filling in different regions, based on a larger and more diverse amount of training data. The nature of exponential family models give the advantage that a higher number of training samples does not really increase the amount of time needed for training, as n only affects solving the conditional MAP problem and computing the estimated average sufficient statistics $\tilde{\boldsymbol{\mu}}$.
Figure 4.10 depicts the quality for each acquisition date where artificial test clouds were added (see Figure 4.1 for information on cloud placement). It shows that the probabilistic prediction is hardly ever worse than the baseline, and excels on certain dates. Domain experts reckon that these dates feature non-linear developments (e.g. due to growing and harvesting seasons of specific crops), which are captured better by the probabilistic approach. One can also see that the Cross prediction error is slightly lower on some dates, but in general both models perform pretty similarly.
As described earlier the MAEs of both baseline and model prediction consist of quantization error (which is equal for both methods) and prediction error, Figure 4.11 depicts the proportion of both errors. One can see that the MAE_Q dominates the MAE_P, but improving the prediction (for example with more complex models) is still desirable, as a higher value for k could be chosen to reduce the quantization error.
Figure 4.12 depicts how the objective function converges during the first EM iterations. Obviously most of the optimization is achieved in the first EM iteration, although the ones following apparently still slightly fine-tune the model (and according to Figure 4.6, this also further increases the quality). It also shows how the expectation steps in the EM training decrease the loss value (due to the sampling of more likely data). The values for I and ρ seem to be chosen well, as the optimization in each iteration seems to converge. Moreover, the EM training also appears to converge, the gaps between the curves (i.e. the improvement of re-estimating the missing data) decreases from iteration to iteration.

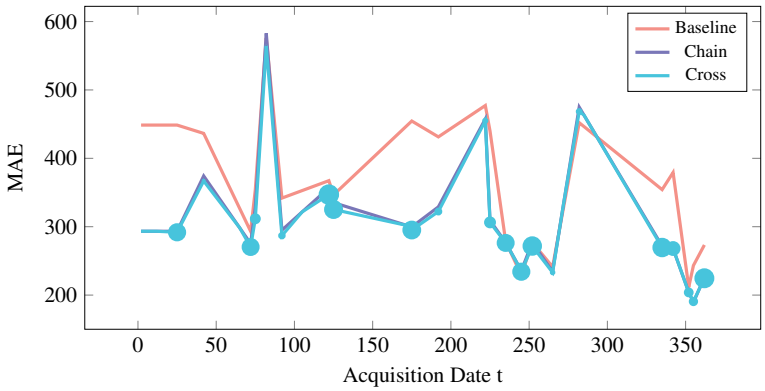


Figure 4.10: Error for artificially clouded test pixels on each separate acquisition date, with mark sizes being proportional to the number of added test clouds (for some dates no clouds were added as shown in Figure 4.1).

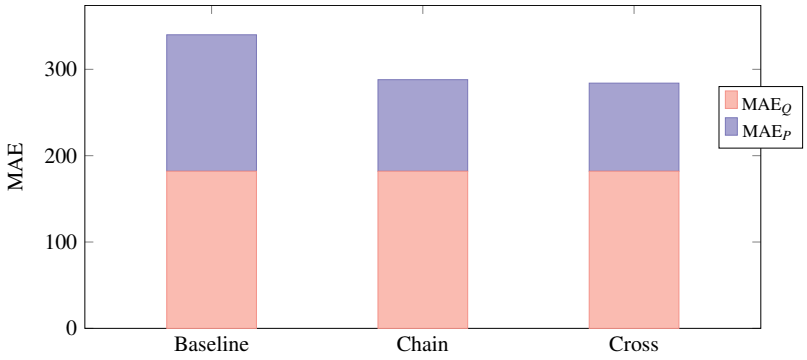
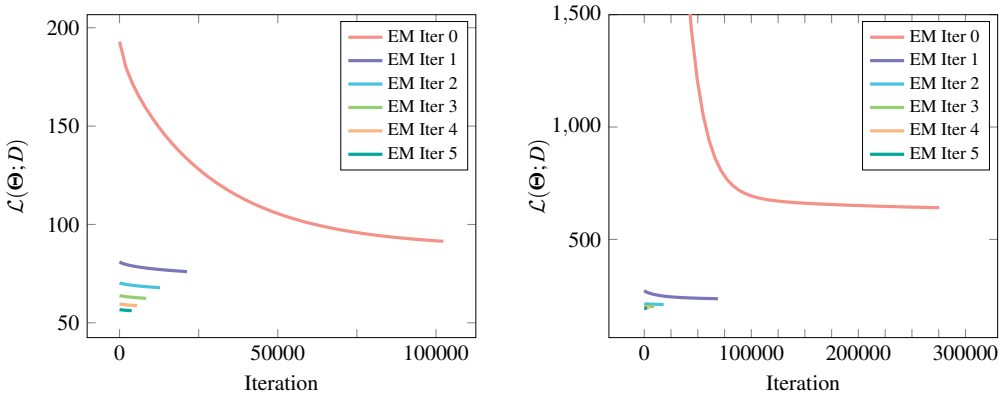


Figure 4.11: Proportion of quantization and prediction error MAE_Q and MAE_P for Chain and Cross model and baseline method.



(a) Convergence of Chain Training

(b) Convergence of Cross Training

Figure 4.12: Converging graphs of the objective function $\mathcal{L}(\Theta; D)$ during the training of the Chain (a) and Cross (b) model in different EM iterations.

(a) Original at $t = 222$ (b) Original at $t = 225$ (c) Original at $t = 235$

(d) Cross Prediction (e) Cross Prediction (f) Cross Prediction

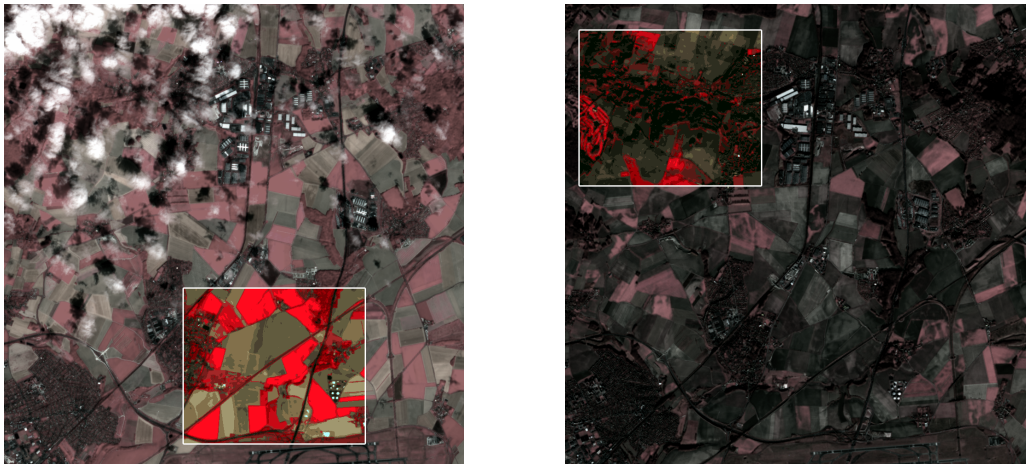
Figure 4.13: Comparison of original data (clouds indicated in white) on consecutive dates (a, b, c) and the corresponding Cross model prediction (d, e, f).

4.2.6 Visual Results of Probabilistic Gap Filling

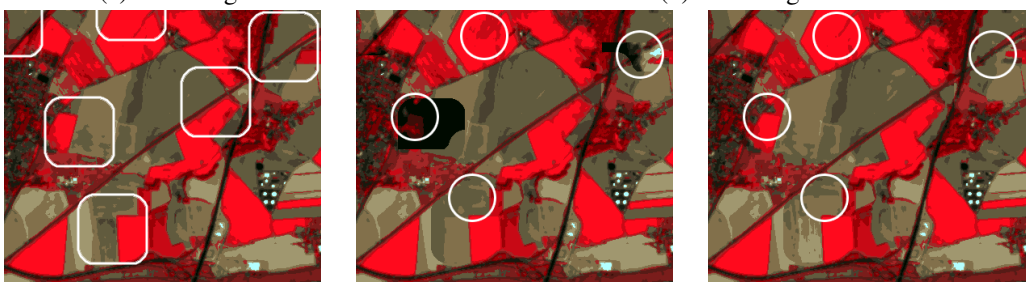
Besides a quantitative evaluation it also makes sense to inspect the prediction quality visually.
The first obvious approach here is to evaluate how the prediction for clouded pixels blends with the unclouded data. Figure 4.13 displays satellite images with clouds on three consecutive dates (cloud masks are outlined in white), and the corresponding Cross model prediction. One can see that the probabilistic gap filling indeed leads to reasonable values. It is also evident that the cloud masks do not capture the cloud outlines perfectly. The artificial clouds allow to compare the original captured reflectances with the model prediction and the baseline method result. This comparison is displayed in Figure 4.14, showing that the Cross model can indeed visually outperform the baseline. This is rooted in globally sampled training data affecting the model prediction, while the baseline method results are only based on the temporal information per pixel. As an example the not perfectly captured cloud shadow at $t = 222$ (Figure 4.13a) obviously deteriorates the linear interpolation for $t = 225$ (Figure 4.14d). The Cross model on the other hand predicts better values, i.e. reflectances that are more likely to be observed.

4.3 Consequences of Compressing Probabilistic Models

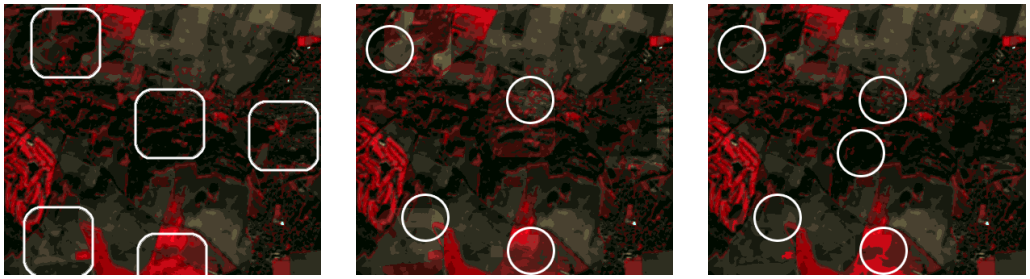
In Section 3.2 PSCS was introduced, a method that can be used for compressing spatio-temporal probabilistic models. Some experiments were run to evaluate the consequences of compression, and explore the usefulness of this novel approach.



(a) Full image at $t = 225$ (b) Full image at $t = 335$

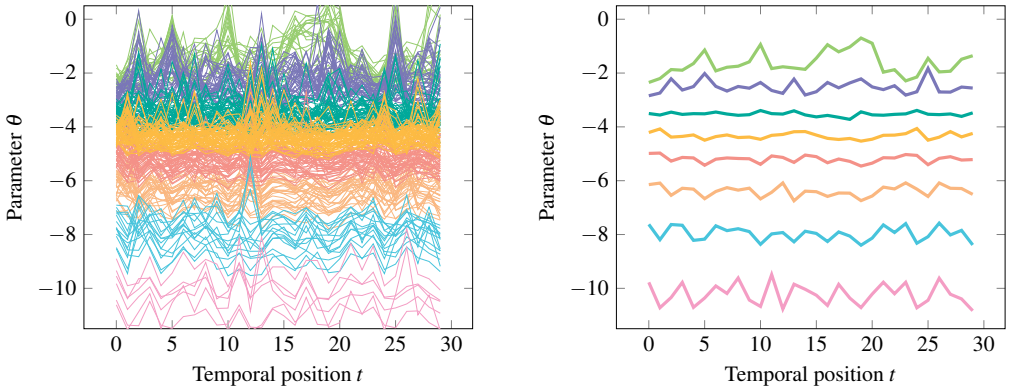


(c) Quantized observations (d) Baseline result (e) Probabilistic result



(f) Quantized observations (g) Baseline result (h) Probabilistic result

Figure 4.14: Comparison of quantized observed intensities (c, f) (clouds indicated in white) with the quantized baseline result (d, g) and Cross model prediction (e, h) on dates $t = 225$ and $t = 335$, region location is shown in (a, b). White circles indicate regions were the probabilistic results beat the Baseline.



(a) Parameter series

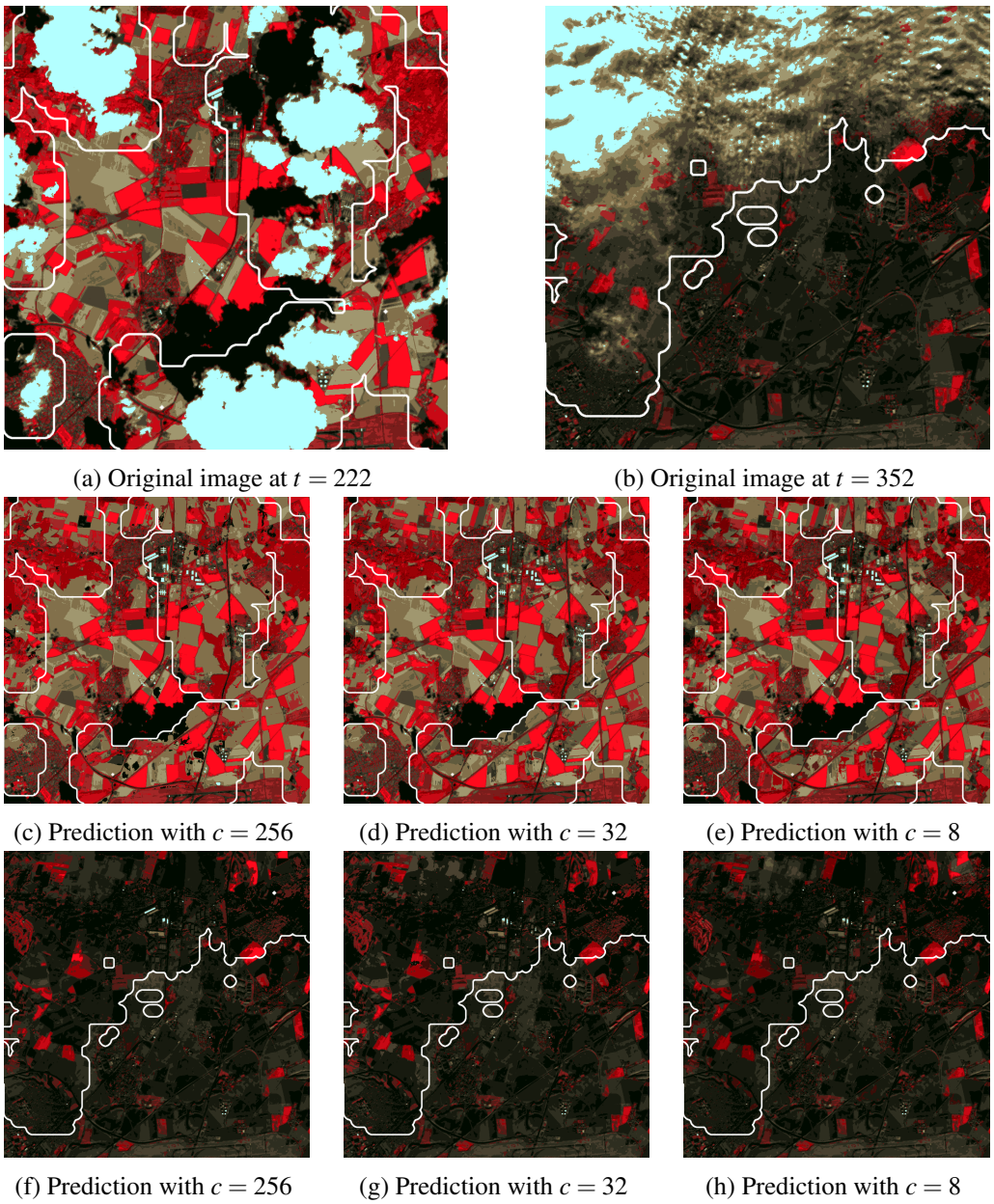
(b) Parameter series centroids

Figure 4.15: Parameter series of a trained Chain model (a), colored corresponding to the associated parameter series centroid (b).

Table 4.2: Quantitative results obtained with differently compressed Chain models.

Complexity	Number of centroids	MAE (± SD)	ERR (± SD)
100%	256	288.32 (± 6.4)	46.55% (± 1.4)
50%	128	290.93 (± 6.1)	47.49% (± 1.3)
23%	32	306.75 (± 6.9)	51.35% (± 1.6)
3%	8	357.34 (± 14.7)	59.62% (± 2.3)

We ran PSCS compression on the previously trained and fairly simple Chain model. It only consists of $T-1$ temporal edges, each one represented by $k^2 = 16 \cdot 16 = 256$ parameters (for the chosen quantization rate $k = 16$). Accordingly the whole model is parametrized by only 256 temporal parameter series.
We compressed the Chain to only consist of $c \in \{8, 32, 128\}$ different parameter series, i.e. a compression to $3\%, 23\%$ and 50% of the initial model complexity. Figure 4.15 depicts the original model parameter series and the $c = 8$ calculated centroids, with colors indicating the associated centroid index that was determined during compression. It clearly shows that some redundancies are present in the parameters, and can be eliminated with compression. However it also opens the question whether more refined quantization methods might lead to better results.
The quality of PSCS compressed models (measured with test clouds) is shown in Table 4.2. As expected MAE and ERR increase with firmer compression (i.e. lower value of c), however a compression to 50% or even 23% of the original complexity results in a model, which still performs reasonable well and beats the baseline quality. One can also see that light compression has hardly any impact on the robustness of our approach, as the SD does not increase much.
Visual prediction results which have been generated with the compressed Chain models can be seen in Figure 4.16. They show that the visual prediction quality is solid, even with a drastically compressed model. The compression results for the Chain look quite promising, especially because the original model does not contain a lot of redundancies. Compression of more complex model could possibly prove to be even more beneficial. More visual results can be found on the accompanying data medium.



(a) Original image at $t = 222$

(b) Original image at $t = 352$

(c) Prediction with $c = 256$

(d) Prediction with $c = 32$

(e) Prediction with $c = 8$

(f) Prediction with $c = 256$

(g) Prediction with $c = 32$

(h) Prediction with $c = 8$

Figure 4.16: Comparison of predictions at $t = 222$ and $t = 352$, which have been generated with PSCS compressed Chain models at different compression rates ($c \in \{8, 32, 256\}$) (clouds indicated in white).

Chapter 5

Conclusion

This thesis explored how probabilistic machine learning techniques can be applied for gap filling in the remote sensing context. Our findings can be beneficial for state-of-the-art projects working with satellite data, or any other kind of spatio-temporal and possibly incomplete data. We here summarize our methods and findings, and afterwards give a short outlook on possible extensions and future work.

5.1 Summary of Novel Methods and Findings

Probabilistic machine learning is well-founded and provided us with robust routines, which we were able to apply for gap filling in remote sensing.
Our core idea is to probabilistically model a subpatch of the dataset, and fill gaps by sliding the model over the whole dataset. The model structure controls which local information affects the prediction, and how complex the model is. A very basic but necessary step is reducing the satellite data complexity, as otherwise discrete probabilistic models would be infeasible for the task. EM training allowed us to construct MRFs based on incomplete datasets, and hence solved a big problem in the application context. Our novel regularization concept proved to be very beneficial during experiments, we found that restricting the model based on prior knowledge helps when large parts of the data are missing. The PSCS compression approach subdivides the model parameters into parameter series, which are then clustered and replaced by their assigned centroid series.
In the end all these methods allowed us to train models, which can quantitatively and visually outperform other gap filling approaches, and at the same time are still feasible to use. The high quality of probabilistically gap filled results is probably rooted in the utilization of globally and locally available data, which might be the biggest benefit of our introduced methodology. We were able to show that considering more local data can actually lower the prediction error. Our compression also led to satisfactory proof-of-concept results, and might be even more advantageous with complexer (and thus possibly more redundant) models.
Moreover the methods are highly adaptable, and therefore can also be applied in other application contexts which deal with incomplete data. We showed that our methods obtain good results on complex remote sensing data, and hopefully the probabilistic approach of gap filling will also lead to new insights in other research ares.

5.2 Possibilities of Extension and Future Work

The introduced methodology also provides a lot of room for improvement and expansion, which is why we want to shortly address some possible enhancements.
For a start more work could go into finding better graphical dependency structures. Besides choosing it arbitrarily, one could also determine a good structure in an algorithmic way (as shortly mentioned in Section 2.3.1). Running several different structures against each other (or collectively, similar to the discussed Cross + Chain predictions) might lead to even better results (it reminds of the idea of machine learning ensembles). It would also be possible to model the spectral data dimension, allowing spectro-spatio-temporal gap filling, as encouraged by Shen et al. [65]. Obviously the suitability of different dependency structures is always related to the specific data and nature of gaps.
More work could also go into finding better parameter priors or refining the regularization. The empirical state transition probability worked great, but maybe a local empirical estimate (e.g. computed with a sliding window) could further improve results. Our PSCS compression approach results in parameter series centroids, which could be used as prior values during training of new models. One could also extend the prior regularization to consider different priors, for example by randomly choosing from a set of priors during training (thus combining different prior approaches).
The parameter series cluster sharing approach could be extended to already consider parameter similarities during training. It might be possible to reparametrize the model accordingly (similar to Piatkowski, Lee, and Morik's STRF reparametrization [58]), such that only centroid values are used and changed during optimization. This special form of regularization (or parameter tying) would already decrease the model complexity during training. It might lead to faster convergence, and result in a model which is less prone to overfitting. Moreover other time series similarity measures might improve results for compressing or regularizing a model. As an example *dynamic time warping* (DTW) is a popular time series distance measure, which could also be applied for averaging the parameters [56].
Another interesting idea would be to slightly change the application goal. With a trained model it might be possible to enhance cloud masks, as unlikely data in the dataset, such as not perfectly captured clouds, could be identified. Gap filled results could possibly also be improved, for example by applying smoothing, utilizing information from the quantization process, or taking the probability of predicted values into consideration.
Lastly all fundamental subroutines, such as the optimization with GD, the LBP approximation for inference, or the Gibbs sampling approach for solving the MAP problem, could be refined or substituted by other methods. The adaptability and extendability is definitely a benefit of the approach, especially because even the initially chosen (and pretty basic) routines led to outstanding results. Chances are good that some of those enhancements will be tackled by the authors in near future.
We hope that our findings contribute to the state-of-the-art computer science and remote sensing research. Missing data is a widespread problem, and maybe our probabilistic gap filling approach can be beneficial in future projects working with satellite data, or other scenarios. With the high popularity of probabilistic models chances are good that people can also benefit from our compression approach. The proof-of-concept experiments led to many promising results, and hopefully forthcoming research can build on our suggested methods.

Bibliography

[1] Alekh Agarwal, Sahand N. Negahban, and Martin J. Wainwright. "Fast global convergence rates of gradient methods for high-dimensional statistical recovery". In: *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada.* 2010, pp. 37–45.

[2] Christophe Andrieu et al. "An Introduction to MCMC for Machine Learning". In: *Machine Learning* 50.1-2 (2003), pp. 5–43. DOI: 10.1023/A:1020281327116.

[3] David Arthur and Sergei Vassilvitskii. "k-means++: the advantages of careful seeding". In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*. 2007, pp. 1027–1035.

[4] Célia A. Zorzo Barcelos and Marcos Aurélio Batista. "Image Inpainting and Denoising by Nonlinear Partial Differential Equations". In: *16th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2003), 12-15 October 2003, Sao Carlos, Brazil*. 2003, pp. 287–293. DOI: 10.1109/SIBGRA.2003.1241021.

[5] Pieter S.A. Beck et al. "Improved monitoring of vegetation dynamics at very high latitudes: A new method using MODIS NDVI". In: *Remote Sensing of Environment* 100.3 (2006), pp. 321–334. ISSN: 0034-4257. DOI: 10.1016/j.rse.2005.10.021.

[6] Andrei A. Bulatov and Martin Grohe. "The complexity of partition functions". In: *Theor. Comput. Sci.* 348.2-3 (2005), pp. 148–186. DOI: 10.1016/j.tcs.2005.09.011.

[7] James B Campbell and Randolph H Wynne. *Introduction to Remote Sensing*. Guilford Publications, 2011. ISBN: 9781609181765.

[8] Jin Chen et al. "A simple and effective method for filling gaps in Landsat ETM+ SLC-off images". In: *Remote Sensing of Environment* 115.4 (2011), pp. 1053–1064. ISSN: 0034-4257. DOI: 10.1016/j.rse.2010.12.010.

[9] Qing Cheng et al. "Cloud removal for remotely sensed images by similar pixel replacement guided with a spatio-temporal MRF model". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 92 (2014), pp. 54–68. ISSN: 0924-2716. DOI: https://doi.org/10.1016/j.isprsjprs.2014.02.015.

[10] Qing Cheng et al. "Inpainting for Remotely Sensed Images With a Multichannel Nonlocal Total Variation Model". In: *IEEE Trans. Geoscience and Remote Sensing* 52.1 (2014), pp. 175–187. DOI: 10.1109/TGRS.2012.2237521.

[11] C. K. Chow and C. N. Liu. "Approximating discrete probability distributions with dependence trees". In: *IEEE Trans. Information Theory* 14.3 (1968), pp. 462–467. DOI: 10.1109/TIT.1968.1054142.

[12] J Cihlar. "Remote sensing of global change: an opportunity for Canada". In: *Proceedings of 11th Canadian Symposium on Remote Sensing*. 1987, pp. 39–48.

[13] Jan G. P. W. Clevers and Anatoly A. Gitelson. "Remote estimation of crop and grass chlorophyll and nitrogen content using red-edge bands on Sentinel-2 and -3". In: *Int. J. Applied Earth Observation and Geoinformation* 23 (2013), pp. 344–351. DOI: 10.1016/j.jag.2012.10.008.

[14] Antonio Criminisi, Patrick Pérez, and Kentaro Toyama. "Region filling and object removal by exemplar-based image inpainting". In: *IEEE Trans. Image Processing* 13.9 (2004), pp. 1200–1212. DOI: 10.1109/TIP.2004.833105.

[15] Arthur Dempster, Natalie Laird, and Donald B. Rubin. "Maximum Likelihood From Incomplete Data Via The EM algorithm". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 39 (Jan. 1977), pp. 1–38.

[16] M. Drusch et al. "Sentinel-2: ESA's Optical High-Resolution Mission for GMES Operational Services". In: *Remote Sensing of Environment* 120 (2012). The Sentinel Missions - New Opportunities for Science, pp. 25–36. ISSN: 0034-4257. DOI: 10.1016/j.rse.2011.11.026.

[17] Yun Du et al. "Water bodies' mapping from Sentinel-2 imagery with modified normalized difference water index at 10-m spatial resolution produced by sharpening the SWIR band". In: *Remote Sensing* 8.4 (2016), p. 354. DOI: 10.3390/rs8040354.

[18] Menachem Fromer and Amir Globerson. "An LP View of the M-best MAP problem". In: *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada.* 2009, pp. 567–575.

[19] Stuart Geman and Donald Geman. "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 6.6 (1984), pp. 721–741. DOI: 10.1109/TPAMI.1984.4767596.

[20] Allen Gersho and Robert M Gray. *Vector Quantization and Signal Compression*. The Springer International Series in Engineering and Computer Science. Springer US, 2012. ISBN: 9780792391814.

[21] Robert M Gray. "Vector quantization". In: *IEEE ASSP Magazine* 1.2 (Apr. 1984), pp. 4–29. ISSN: 0740-7467. DOI: 10.1109/MASSP.1984.1162229.

[22] Jack Grove. "Do great minds think alike? The THE/Lindau Nobel Laureates Survey". In: *Times Higher Education. August* 31 (2017).

[23] Philip George Guest. *Numerical Methods of Curve Fitting*. Cambridge University Press, 2012. ISBN: 9781107646957.

[24] Olivier Hagolle et al. "A multi-temporal method for cloud detection, applied to FORMOSAT-2, VENμS, LANDSAT and SENTINEL-2 images". In: *Remote Sensing of Environment* 114.8 (2010), pp. 1747–1755. DOI: 10.3390/rs70302668.

[25] Olivier Hagolle et al. *MAJA Algorithm Theoretical Basis Document*. Dec. 2017. DOI: 10.5281/zenodo.1209633.

[26] J. M. Hammersley and P. E. Clifford. "Markov random fields on finite graphs and lattices". In: Unpublished manuscript (1971).

[27] Russell C. Hardie, Kenneth J. Barnard, and Ernest E. Armstrong. "Joint MAP registration and high-resolution image estimation using a sequence of undersampled images". In: *IEEE Trans. Image Processing* 6.12 (1997), pp. 1621–1633. DOI: 10.1109/83.650116.

[28] Kaiming He and Jian Sun. "Statistics of Patch Offsets for Image Completion". In: *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part II*. 2012, pp. 16–29. DOI: 10.1007/978-3-642-33709-3_2.

[29] Eileen H. Helmer and Bonnie Ruefenacht. "Cloud-free satellite image mosaics with regression trees and histogram matching". In: *Photogrammetric Engineering & Remote Sensing* 71.9 (2005), pp. 1079–1089.

[30] Zhexue Huang. "Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values". In: *Data Min. Knowl. Discov.* 2.3 (1998), pp. 283–304. DOI: 10.1023/A:1009769707641.

[31] Per Jönsson and Lars Eklundh. "Seasonality extraction by function fitting to time-series of satellite sensor data". In: *IEEE Trans. Geoscience and Remote Sensing* 40.8 (2002), pp. 1824–1832. DOI: 10.1109/TGRS.2002.802519.

[32] Sivasathivel Kandasamy et al. "A comparison of methods for smoothing and gap filling time series of remote sensing observations–application to MODIS LAI products". In: *Biogeosciences* 10.6 (2013), pp. 4055–4071. DOI: 10.5194/bg-10-4055-2013.

[33] Steven M. Kay. *Fundamentals of Statistical Signal Processing: Estimation Theory*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993. ISBN: 0-13-345711-7.

[34] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models - Principles and Techniques*. MIT Press, 2009. ISBN: 978-0-262-01319-2.

[35] Nikos Komodakis. "Image Completion Using Global Optimization". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), 17-22 June 2006, New York, NY, USA*. 2006, pp. 442–452. DOI: 10.1109/CVPR.2006.141.

[36] Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. "Factor graphs and the sum-product algorithm". In: *IEEE Trans. Information Theory* 47.2 (2001), pp. 498–519. DOI: 10.1109/18.910572.

[37] Bassam Abdel Latif et al. "Preprocessing of Low-Resolution Time Series Contaminated by Clouds and Shadows". In: *IEEE Trans. Geoscience and Remote Sensing* 46.7 (2008), pp. 2083–2096. DOI: 10.1109/TGRS.2008.916473.

[38] Min Li, Soo Chin Liew, and Leong Keong Kwoh. "Producing cloud free and cloud-shadow free mosaic from cloudy IKONOS images". In: *Geoscience and Remote Sensing Symposium, 2003. IGARSS'03. Proceedings. 2003 IEEE International*. Vol. 6. Ieee. 2003, pp. 3946–3948. DOI: 10.1109/IGARSS.2003.1295323.

[39] Xinghua Li et al. "Recovering Quantitative Remote Sensing Products Contaminated by Thick Clouds and Shadows Using Multitemporal Dictionary Learning". In: *IEEE Trans. Geoscience and Remote Sensing* 52.11 (2014), pp. 7086–7098. DOI: 10.1109/TGRS.2014.2307354.

[40] Thomas Liebig et al. "Predictive Trip Planning - Smart Routing in Smart Cities". In: *Proceedings of the Workshops of the EDBT/ICDT 2014 Joint Conference (EDBT/ICDT 2014), Athens, Greece, March 28, 2014.* 2014, pp. 331–338.

[41] Stuart P. Lloyd. "Least squares quantization in PCM". In: *IEEE Trans. Information Theory* 28.2 (1982), pp. 129–136. DOI: 10.1109/TIT.1982.1056489.

[42] Luca Lorenzi, Farid Melgani, and Grégoire Mercier. "Missing-Area Reconstruction in Multispectral Images Under a Compressive Sensing Perspective". In: *IEEE Trans. Geoscience and Remote Sensing* 51.7-1 (2013), pp. 3998–4008. DOI: 10.1109/TGRS.2012.2227329.

[43] J. B. Macqueen. "Some Methods for classification and analysis of multivariate observations". In: *Procedings of the Fifth Berkeley Symposium on Math, Statistics, and Probability*. Vol. 1. University of California Press, 1967, pp. 281–297.

[44] Erik. Meijering. "A chronology of interpolation: from ancient astronomy to modern signal and image processing". In: *Proceedings of the IEEE* 90.3 (Mar. 2002), pp. 319–342. ISSN: 0018-9219. DOI: 10.1109/5.993400.

[45] Roi Méndez-Rial, María Calvino-Cancela, and Julio Martín-Herrero. "Anisotropic Inpainting of the Hypercube". In: *IEEE Geosci. Remote Sensing Lett.* 9.2 (2012), pp. 214–218. DOI: 10.1109/LGRS.2011.2164050.

[46] Sergio Rodrigues de Morais and Alex Aussem. "A Novel Scalable and Data Efficient Feature Subset Selection Algorithm". In: *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML/PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part II*. 2008, pp. 298–312. DOI: 10.1007/978-3-540-87481-2_20.

[47] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*. Vol. 87. Springer Science & Business Media, 2013. ISBN: 9781441988539.

[48] Yurii E Nesterov. "A method for solving the convex programming problem with convergence rate O (1/kˆ 2)". In: *Dokl. Akad. Nauk SSSR*. Vol. 269. 1983, pp. 543–547.

[49] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2000. ISBN: 9780387987934.

[50] Manuel M. Oliveira et al. "Fast Digital Image Inpainting". In: *Proceedings of the IASTED International Conference on Visualization, Imaging and Image Processing (VIIP 2001), Marbella, Spain, September 3-5, 2001*. 2001, pp. 261–266.

[51] Margaret A Oliver and Richard Webster. "Kriging: a method of interpolation for geographical information systems". In: *International Journal of Geographical Information Science* 4.3 (1990), pp. 313–332. DOI: 10.1080/02693799008941549.

[52] Judea Pearl. *Probabilistic reasoning in intelligent systems - networks of plausible inference*. Morgan Kaufmann series in representation and reasoning. Morgan Kaufmann, 2014.

[53] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[54] François Petitjean and Geoffrey I. Webb. "Scalable Learning of Graphical Models". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. 2016, pp. 2131–2132. DOI: 10.1145/2939672.2945382.

[55] François Petitjean and Geoffrey I. Webb. "Scaling log-linear analysis to datasets with thousands of variables". In: *Proceedings of the 2015 SIAM International Conference on Data Mining, Vancouver, BC, Canada, April 30 - May 2, 2015*. 2015, pp. 469–477. DOI: 10.1137/1.9781611974010.53.

[56] François Petitjean et al. "Dynamic Time Warping Averaging of Time Series Allows Faster and More Accurate Classification". In: *2014 IEEE International Conference on Data Mining, ICDM 2014, Shenzhen, China, December 14-17, 2014*. 2014, pp. 470–479. DOI: 10.1109/ICDM.2014.27.

[57] Nico Piatkowski. "Exponential families on resource-constrained systems". PhD thesis. Technical University of Dortmund, Germany, 2018.

[58] Nico Piatkowski, Sangkyun Lee, and Katharina Morik. "Spatio-temporal random fields: compressible representation and distributed estimation". In: *Machine Learning* 93.1 (2013), pp. 115–139. DOI: 10.1007/s10994-013-5399-7.

[59] H Rahman and G Dedieu. "SMAC: A simplified method for the atmospheric correction of satellite measurements in the solar spectrum". In: *International Journal of Remote Sensing - INT J REMOTE SENS* 15 (Jan. 1994), pp. 123–143. DOI: 10.1080/01431169408954055.

[60] Jacques Rivoirard. *Introduction to Disjunctive Kriging and Non-linear Geostatistics*. Spatial information systems. Clarendon Press, 1994. ISBN: 9780198741800.

[61] G.J. Roerink, Massimo Menenti, and Wout Verhoef. "Reconstructing cloudfree NDVI composites using Fourier analysis of time series. International Journal of Remote Sensing". In: *Int. J. remote Sens. 21 (2000), 9: 1911-1917* 21 (June 2000). DOI: 10.1080/014311600209814.

[62] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *CoRR* abs/1609.04747 (2016). arXiv: 1609.04747.

[63] Abraham Savitzky and Marcel JE Golay. "Smoothing and Differentiation of Data by Simplified Least Squares Procedures". In: *Analytical chemistry* 36 (July 1964), pp. 1627–1639. DOI: 10.1021/ac60214a047.

[64] Robert A Schowengerdt. *Remote Sensing, Models, and Methods for Image Processing*. Academic Press, 1997. ISBN: 9780126289817.

[65] Huanfeng Shen et al. “Missing Information Reconstruction of Remote Sensing Data: A Technical Review”. In: *IEEE Geoscience and Remote Sensing Magazine* 3 (Sept. 2015), pp. 61–85. DOI: 10.1109/MGRS.2015.2441912.

[66] Frank Soong et al. “Report: A Vector Quantization Approach to Speaker Recognition”. In: vol. 66. May 1985, pp. 387–390. DOI: 10.1109/ICASSP.1985.1168412.

[67] M.L. Stein. *Interpolation of Spatial Data: Some Theory for Kriging*. Springer Series in Statistics. Springer New York, 2012. ISBN: 9780387986296.

[68] J Storey et al. “Landsat 7 scan line corrector-off gap filled product development”. In: *Proceedings of Pecora 16 Global Priorities in Land Remote Sensing* (Jan. 2005), pp. 23–27.

[69] R. Sundberg. *Maximum Likelihood Theory and Applications for Distributions Generated when Observing a Function an Exponential Family Variable*. Dissociation. Institute of Mathematical Statistics, Stockholm University. Institute of Mathematical Statics, Stockholm University, 1972. ISBN: 9789197001601.

[70] Rolf Sundberg. “Maximum Likelihood Theory for Incomplete Data from an Exponential Family”. In: *Scandinavian Journal of Statistics* 1.2 (1974), pp. 49–58. ISSN: 03036898, 14679469.

[71] Chang Wei Tan, Geoffrey I. Webb, and François Petitjean. “Indexing and classifying gigabytes of time series under time warping”. In: *Proceedings of the 2017 SIAM International Conference on Data Mining, Houston, Texas, USA, April 27-29, 2017*. 2017, pp. 282–290. DOI: 10.1137/1.9781611974973.32.

[72] Ghada Trabelsi et al. “Dynamic MMHC: A Local Search Algorithm for Dynamic Bayesian Network Structure Learning”. In: *Advances in Intelligent Data Analysis XII - 12th International Symposium, IDA 2013, London, UK, October 17-19, 2013. Proceedings*. 2013, pp. 392–403. DOI: 10.1007/978-3-642-41398-8_34.

[73] Silvia Valero, Charlotte Pelletier, and Marina Bertolino. “Patch-based reconstruction of high resolution satellite image time series with missing values using spatial, spectral and temporal similarities”. In: *2016 IEEE International Geoscience and Remote Sensing Symposium, IGARSS 2016, Beijing, China, July 10-15, 2016*. 2016, pp. 2308–2311. DOI: 10.1109/IGARSS.2016.7729596.

[74] Freek Van der Meer. “Remote-sensing image analysis and geostatistics”. In: *International Journal of Remote Sensing - INT J REMOTE SENS* 33 (Sept. 2012), pp. 5644–5676. DOI: 10.1080/01431161.2012.666363.

[75] Aleixandre Verger, Frédéric Baret, and Marie Weiss. “A multisensor fusion approach to improve LAI time series”. In: *Remote Sensing of Environment* 115.10 (2011), pp. 2460–2470. ISSN: 0034-4257. DOI: 10.1016/j.rse.2011.05.006.

[76] N Viovy, O Arino, and AS Belward. “The Best Index Slope Extraction (BISE): A method for reducing noise in NDVI time-series”. In: *International Journal of Remote Sensing* 13.8 (1992), pp. 1585–1590. DOI: 10.1080/01431169208904212.

[77] Kiri Wagstaff et al. “Constrained K-means Clustering with Background Knowledge”. In: *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*. 2001, pp. 577–584.

[78] Grace Wahba. *Spline models for observational data*. Vol. 59. Siam, 1990. DOI: 10.1137/1033124.

[79] Martin J. Wainwright, Tommi S. Jaakkola, and Alan S. Willsky. “A new class of upper bounds on the log partition function”. In: *IEEE Trans. Information Theory* 51.7 (2005), pp. 2313–2335. DOI: 10.1109/TIT.2005.850091.

[80] Martin J. Wainwright, Tommi S. Jaakkola, and Alan S. Willsky. “Tree-based reparameterization framework for analysis of sum-product and related algorithms”. In: *IEEE Trans. Information Theory* 49.5 (2003), pp. 1120–1146. DOI: 10.1109/TIT.2003.810642.

[81] Martin J. Wainwright and Michael I. Jordan. “Graphical Models, Exponential Families, and Variational Inference”. In: *Foundations and Trends in Machine Learning* 1.1-2 (2008), pp. 1–305. DOI: 10.1561/2200000001.

[82] Lingli Wang et al. “A new method for retrieving band 6 of aqua MODIS”. In: *IEEE Geosci. Remote Sensing Lett.* 3.2 (2006), pp. 267–270. DOI: 10.1109/LGRS.2006.869966.

[83] Yair Weiss, Chen Yanover, and Talya Meltzer. “MAP Estimation, Linear Programming and Belief Propagation with Convex Free Energies”. In: *CoRR* abs/1206.5286 (2012). arXiv: 1206.5286.

[84] Jun Yang et al. “The role of satellite remote sensing in climate change studies”. In: *Nature Climate Change* 3 (Sept. 2013), pp. 875–883. DOI: 10.1038/nclimate2033.

[85] Chen Yanover, Talya Meltzer, and Yair Weiss. “Linear Programming Relaxations and Belief Propagation - An Empirical Study”. In: *Journal of Machine Learning Research* 7 (2006), pp. 1887–1907.

[86] Chen Yanover and Yair Weiss. “Finding the M most probable configurations using loopy belief propagation”. In: *Advances in neural information processing systems*. 2004, pp. 289–296.

[87] Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. “On Early Stopping in Gradient Descent Learning”. In: *Constructive Approximation* 26 (Aug. 2007), pp. 289–315. DOI: 10.1007/s00365-006-0663-2.

[88] Chao Yu et al. “Kriging interpolation method and its application in retrieval of MODIS aerosol optical depth”. In: *Proceedings - 2011 19th International Conference on Geoinformatics, Geoinformatics 2011* (June 2011), pp. 1–6. DOI: 10.1109/GeoInformatics.2011.5981052.

[89] Fei Yuan et al. “Land cover classification and change analysis of the Twin Cities (Minnesota) Metropolitan Area by multitemporal Landsat remote sensing”. In: *Remote sensing of Environment* 98.2-3 (2005), pp. 317–328.

[90] Chao Zeng, Huanfeng Shen, and Liangpei Zhang. "Recovering missing pixels for Landsat ETM+ SLC-off imagery using multi-temporal regression analysis and a regularization method". In: *Remote Sensing of Environment* 131 (2013), pp. 182–194. ISSN: 0034-4257. DOI: https://doi.org/10.1016/j.rse.2012.12.012.

[91] Chuanrong Zhang, Weidong Li, and David Travis. "Gaps-fill of SLC-off Landsat ETM+ satellite image using a geostatistical approach". In: *International Journal of Remote Sensing* 28 (Jan. 2007), pp. 5103–5122. DOI: 10.1080/01431160701250416.

[92] Chuanrong Zhang, Weidong Li, and David Travis. "Restoration of clouded pixels in multispectral remotely sensed imagery with cokriging". In: *International Journal of Remote Sensing* 30 (May 2009), pp. 2173–2195. DOI: 10.1080/01431160802549294.

[93] Qiang Zhang et al. "Missing Data Reconstruction in Remote Sensing Image With a Unified Spatial-Temporal-Spectral Deep Convolutional Neural Network". In: *IEEE Trans. Geoscience and Remote Sensing* 56.8 (2018), pp. 4274–4288. DOI: 10.1109/TGRS.2018.2810208.

[94] Ping Zhu, L.W. Zhang, and K Liew. "Geometrically nonlinear thermomechanical analysis of moderately thick functionally graded plates using a local PetrovC-Galerkin approach with moving Kriging interpolation". In: *Composite Structures* 107 (Jan. 2014), pp. 298–314. DOI: 10.1016/j.compstruct.2013.08.001.