FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Deformation Monitoring Using Artificial Intelligence Techniques

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der Technischen Wissenschaften

by

B.Sc.(Hons) Miloš Miljanović
Registration Number 0527983

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: O.Univ.-Prof. Dipl Ing. Dr Thomas Eiter
Second advisor: A.O.Univ.-Prof. Dipl Ing. Dr Uwe Egly

The dissertation has been reviewed by:

O.Univ.-Prof. Dipl Ing. Dr Thomas Eiter	Univ.-Prof. Dipl Ing. Dr Toša Ninkov

Vienna, 6th July, 2015

Miloš Miljanović

Erklärung zur Verfassung der Arbeit

B.Sc.(Hons) Miloš Miljanović
Puchsbaumgasse 49/16, 1100 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 6. Juli 2015

Miloš Miljanović

Acknowledgements

I would like to thank and express my deepest respect to my advisor Prof. Thomas Eiter for the chance to work with him and to learn from him. I am especially grateful for his patience, support and having time for my work. Further I would like to thank Prof. Uwe Egly for having time for my work and for giving me a chance to learn from him.

I also want to thank all my colleagues Alexander Reiterer, Martin Lehmann, Haider Ali, Gerhard Paar, Heribert Kahmen for friendly working environment.

I would also like to express my gratitude to Prof. Tosa Ninkov for providing me with data for research.

Last but not least I am grateful to my parents and to my sister for supporting me in everything I do.

This work has been partially supported by the project "Multi-Sensor Deformation Measurement System Supported by Knowledge Based and Cognitive Vision Techniques", which was a cooperation between TU Wien and Joanneum Research Graz and has been supported by the Austrian Science Fund (FWF) grant P18286.

Danksagung

Ich würde gerne meinem Betreuer Herrn Prof. Thomas Eiter meinen tiefsten Respekt ausdrücken und ihm für die Möglichkeit danken, mit ihm gearbeitet au haben und von ihm lernen zu können. Ich bin besonders dankbar für seine Geduld, Unterstützung und seine Zeit, die er der Betreuung meiner Arbeit gewidmet hat. Weiters möchte ich mich bei Herrn Prof. Uwe Egly herzlich bedanken, dass auch er Zeit für meine Arbeit gefunden hat und mir ebenefals die Möglichkeit gegeben hat, von ihm zu lernen.

Meinen Kollegen Alexander Reiterer, Martin Lehmann, Haider Ali und Gerhard Paar sowie Prof. Heribert Kahmen möchte ich für eine freundliche Arbeitsumgebung danken.

Ich möchte auch Herrn Prof. Tosa Ninkov dafür danken, dass er mir wichtige Daten für meine Forschungen zur Verfügung gestellt hat.

Zu guter Letzt bedanke ich mich vielmals bin meinen Eltern und meiner Schwester für ihre Unterstützung in allem was ich mache.

Diese Arbeit wurde zum Teil durch das Projekt "Multi-Sensor Deformation Measurement System Supported by Knowledge Based and Cognitive Vision Techniquesünterstützt, das in Kooperation zwischen der TU Wien und Joanneum Research Graz durchgeführt und vom Fonds für die wissenschaftliche Forschung (FWF) finanziert worden ist.

Abstract

Deformation Monitoring is a common practice used by engineers to monitor and track the development of civil engineering objects by means of geodetic measurements in order to prevent hazardous behaviour that can happen during their construction. Measured objects are most commonly residential buildings, dams, bridges, tunnels, but also areas that are affected by earthquakes and landslide movements. In all cases, the geodetic engineers are applying different methodologies that have been developed several decades ago, e.g. Finite Element Method (FEM). With the emergence of information technology, the usage of such methodologies has become more accurate and easier to implement.

The work in this thesis will try to further investigate the applications of computer science and artificial intelligence in engineering geodesy, where the objects of interest will be residential buildings. The thesis consists of two major parts. In the first part, various tools and methods for videometric analysis will be implemented and tested using a benchmark of images taken from the Vienna and Zürich public domain, which contain snapshots of facades of buildings. The images will be used to create a series of simulated images using a warping tool to represent possible deformations. The image processing techniques described in this thesis will be used to isolate the Regions of Interest (ROI), which are small enclosed areas with homogenous deformations using a proposed object segmentation algorithm. The outlier points which are left after the matching process between different epochs of the observed deformation will be identified using a variation of the Randomized Sample Consensus Algorithm (RANSAC) algorithm that does not use a fundamental matrix. Finally, the observed points will then be processed by a simple rule-based expert system, and classified either as a potentially hazardous deformation, or a normal (stagnant) deformation (by comparing the translation, and or rotation vectors along the x, y, and z axes).

The second part is concerned with forecasting future behavior of deformations using Artificial Neural Networks (ANN). Two different architectures are used, the Finite Impulse Response (FIR) ANN which are well known in the field of time series prediction, and the recurrent version of this ANN, the Recurrent FIR (RFIR). The data set that will be used to train the networks has been gathered from a tunneling project Prokop by civil engineering company Energoprojekt in Belgrade, which has had an impact on several buildings in the vicinity of the construction site. The deformation analysis has been performed using the Modified Karlsruhe Method (MKM), where the main points of focus were around the abutment diaphragms around the tunnel, but also dozens of points per residential building. Furthermore, the statistical analysis of the measurements using the

MKM will identify the points which do not deviate from the allowed lack-of-fit, as well as points which were destroyed during the construction. The neural networks will be trained using the data provided from 'Prokop' to predict future values of horizontal and vertical coordinates. The points which will be forecasted are all found in residential buildings, which have been measured from various stations at the construction site. The results show that both the FIR and RFIR networks can be used as a viable tool for predicting deformation measurements.

Kurzfassung

Deformation Monitoring ist eine gewöhnliche Praxis von Ingenieuren, um die Entwicklung von Tiefbau-Objekten durch geodätische Messungen zu überwachen und verfolgen. Damit soll verhindert werden, dass während der Errichtung ein gefährliches Verhalten auftreten kann. Die gemessenen Objekte sind am häufigsten Wohngebäude, Dämme, Brücken, Tunnel, aber auch Bereiche, die durch Erdbeben und Erdrutschbewegungen betroffen sind. In allen Fällen wenden die Geodäten unterschiedliche Methodologien an, die vor einigen Jahrzehnten entwickelt worden sind, zum Beispiel die Finite-Elemente-Methode (FEM). Mit dem Einzug der Informationstechnologie wurde die Verwendung von solchen Methodologien genauer und einfacher zu implementieren.

Die Arbeit in dieser Dissertation wird versuchen, die Anwendungen der Computerwissenschaft und Methoden der künstlichen Intelligenz in der Ingenieurgeodäsie weiter zu beleuchten, wobei im Fokus des Interesses Wohngebäude stehen. Die Dissertation besteht aus zwei Hauptteilen. In ersten Teil werden unterschiedliche videometrische Analysentechniken unter Verwendung einer Benchmarkserie von Bildern implementiert und getestet, die aus den Wien- und Zürich-Datenbanken stammen, welche Momentaufnahmen von Gebäudefassaden im öffentlichen Bereich enthalten. Die Bilder werden auch verwendet, um mit einem speziellen Tool eine Serie von Bildern mit künstlichen Deformationen zu erzeugen. Die Techniken zur Bildverarbeitung, die in dieser Dissertation beschrieben werden, dienen dazu, sogenannte Regionen von Interesse (regions of interest, ROI), die kleine geschlossene Räume mit homogenen Deformationen sind, mit einem vorgeschlagenen Objekt-Segmentierungsalgorithmus zu isolieren. Die Ausreisserpunkte, die nach dem Anpassungsprozess zwischen unterschiedlichen Epochen der beobachteten Deformation gelassen werden, werden durch eine Variante des Randomisierten Probe Consensus-Algorithmus (RANSAC) identifiziert, die im Gegensatz zum traditionellen Algorithmus keine fundamentale Matrix nutzt. Schlussendlich werden dann die beobachteten Punkte mithilfe eines einfachen regelbasierten Expertensystems verarbeitet und sodann entweder als eine möglich gefährliche Deformation oder als eine normale (stehende) Deformation (durch Vergleich der Übersetzung, und/oder der Drehvektoren entlang der X-, Y- und Z-Achse) klassifiziert.

Der zweite Teil der Dissertation beschäftigt sich mit der Vorhersage des zukünftigen Verhaltens der Deformationen durch künstliche neuronale Netze (artificial neural networks, ANN). Zwei unterschiedliche Architekturen werden dabei genutzt: das Endliche Impulsantwort (FIR) ANN, das im Bereich von Zeitreihenvorhersagen sehr bekannt ist, und dessen rekurrente Version, das Endliche Impulsantwort (RFIR) ANN. Um konkrete

Netze zu trainieren, wird ein Datenbestand verwendet, der im Rahmen des Tunnel-Projekts “Prokop“ durch das Tiefbauunternehmen Energoprojekt in Belgrad gesammelt worden ist; dieses Projekt hat eine Auswirkung auf einige Gebäude in der Umgebung der Ausgrabungsstätte gehabt. Die Deformationsanalyse wird durch die Modifizierte Karlsruher Methode (MKM) ausgeführt, wobei die Hauptpunkte entlang der Angrenzungsdiaphragmen um den Tunnel, aber auch Dutzende von Punkten pro Wohngebäude vorlagen. Weiters wird die statistische Analyse der Messungen durch die MKM Punkte identifizieren, die von der erlaubten Toleranz nicht abweichen, genauso wie Punkte, die während des Baues zerstört wurden. Die neuronalen Netze werden unter Verwendung der Prokop-Daten trainiert, um die zukünftigen Werte der horizontalen und der vertikalen Koordinaten vorherzusagen. Die Prognosepunkte sind alle in den Wohngebäuden, die von verschiedenen Stationen auf der Baustelle aus vermessen worden sind. Die Resultate zeigen, dass sowohl das FIR als auch das RFIR als Tool zur Vorhersage von Deformationen brauchbar ist.

Contents

List of Figures

List of Tables

List of Algorithms

CHAPTER 1

Introduction

A great deal of effort has been put into eliciting knowledge and reasoning strategies from structural engineering experts for the purpose of developing a computer model of their expertise in order to assist engineers in their decision-making processes. Historically this has been done by the means of artificial targets, where the entire work has been done manually. But with the emergence of information technology, engineers were able to implement the processes they have been mapping throughout the years. In the process of residential building construction, different degrees of deformation will be present. Deformation within limits is normal, but if it exceeds the prescribed limit may cause deformation disasters.

In this dissertation, a technique used to monitor structural movements present in residential buildings and to forecast the behaviour of the deformations is described. There are currently several deformation monitoring systems [63] [11] [6], but very few deal with monitoring deformations in residential buildings, and even few try to apply artificial intelligence methods to improve the results. Most common structures that are of interest to the engineers and that are being measured by geodesists for the purpose of deformation monitoring include power plants, bridges, and residential buildings.

Residential buildings are always undergoing a deformation, either due to traffic, rise in temperature, or high precipitation for example, but these are negligable. The most common reason for deformation present in residential buildings is induced by applied force from tunelling [16]. Tunelling in cities is either done because of the construction of an underground subway station, parking lots, or even by installing basements. The deformation that will be assessed in this thesis will be based on the data gathered from a tunelling project in Belgrade which resulted in the construction of a subway station 'Prokop'. These types of deformations are known in advance that they will occur, which is why geodetic engineers are being hired to monitor the deformation, and make sure that no hazard occurs to the surrounding environment, especially to the tenants in the buildings. The deformation movements should not exceed several *mm* in order to avoid any lawsuits. On the other hand, deformations present in high rise buildings can exceed several *cm* [57], mainly due to the different materials that are used to construct them.

The method will attempt to utilise image processing techniques, knowledge based systems, and Artificial Neural Networks (ANN) to the problem of deformation monitoring. Residential buildings in this sense are regarded as objects that are moving through time, and even though the movements that are being tracked are not noticeable by the human eye, but are by the usage of sensors, lasers, and cameras [55]. In order to track objects through time, specific points called interest operators must be chosen whose position is well defined in image space and preserved over time. The interest operators [29] [25] are points in the image taken by the camera, and are due to their nature typically around corners of windows, which make them useful for allocating the coordinates of windows present in facades. The glazing percentage, or the ratio of windows to the overall facade is of particular use in urban energy modelling [38].

Another deformation that will be observed in this thesis will be simulated, and tracked in time by an algorithm that will take as input a set of images, and output the 'structural' movement for the purpose of tracking, and monitoring, and no attempt will be made to predict future movements. The work here will entail trying to estimate the coordinates of windows based on the interest operators, followed by elimination of all outliers found in different epochs, and using a knowledge base to form regions of interest (ROI) based on the remaining points.

The forecasting of deformation movements will be accomplished by the usage of Finite Impulse Response (FIR) ANN, which uses temporal backpropagation as the training algorithm. The data used to train them will be based on real actual scenario, gathered from a tunelling project.

1.1 Motivation and Problem

In an interdisciplinary research project entitled: 'Multi-Sensor Deformation Measurement System Supported by Knowledge-Based and Cognitive Vision Techniques', a new kind of image-based measuring framework was developed that combines the usage of Image-Assisted Total Stations (IATS) and Terrestrial Laser Scanning Techniques (TLS) [56]. The system comprises of several sub-systems, covering the fields of image processing, deformation analysis/interpretation, and measurement of points in 3D. The techniques employed by the system are all covered by artificial intelligence, and the main purpose of the project was to see to what extent they can improve the results.

The work presented in this thesis further tries to investigate what other kind of methods in this area can help solve the problem of deformation monitoring, in particular videometric analysis of information gathered from a series of images of facades in order to track the structural behaviour of the observed object, and forecasting future values of deformation measurements using artificial neural networks trained to predict future values of time series. Traditionally the geodetic measurements are monitored and predicted using the FEM which entails creating a model of the deformation, and having expert knowledge in structural engineering. Thus, it would be interesting to see whether the

ANN approach, which requires very little knowledge in the expert area, can be used as a forecasting tool for the problem of deformation monitoring.

In the research project, the system was never tested on a real life scenario, due to the lack of data. It is particularly difficult to gather videometric data of deformations, as well as any historical data. In this thesis, the historical data from 'Prokop' has been used to train ANN to predict future measurement values, which was monitored by geodetic engineers using the Modified Karlsruhe Method (MKM).

1.2 State of the Art

GOCA software is well known and used in the field of deformation monitoring. The software package includes different mathematical models used in deformation analysis. It is being used extensively, among others by the Austrian company Vermessung Angst for deformation monitoring of historical buildings, Vienna and worldwide. Users are able to access live data online at any point, and their systems also alarm the user when a potentially hazardous movement is occurring. Forecasting of future values of horizontal and vertical movements is achieved with Kalman Filtering. Measurements are performed using the theodolites from Leica systems, which are also deployed by the company Energoprojekt. The GOCA system consists of GPS sensors and communication units set up in the monitoring area as well as software for communication and deformation analysis. [33]

Global Positioning Systems (GPS) is currently the only functional Global Satellite Navigation System (GSNS). GPS consists of 24 satellites orbiting around Earth, that are sending radio signals on Earth's surface. The GPS receivers based on these signals can determine their correct location (ellipsoid height, geodetic width and height) at any given time or place, under any weather conditions. GPS satellites cover the entire Earth, and measuring the distance between satellites it is possible to determine the position of any point on Earth with an accuracy of 10m.

There are many other deformation monitoring systems, but most are used for assessing the conditions of hydro plants, mines, or bridges. Other fields of interest include monitoring landslide movements [6] and effects of earthquakes. Landslide movements may cause considerable damage on the engineering constructions such as buildings, roads, or dams. The buildings affected by landslides are mostly located in rural areas however, and not in the city. In this paper, a knowledge base for the causes of landslides is described, where influence factors are identified together with the consequences which they have on the surrounding environment. The influencing factors in landslide movements include vegetation, granual material, soil saturation, leaning trees are then further being analyzed and assessed as risk factors by interviewing experts in the field. The resulting knowledge base is a questionnaire which users can query for potential hazardous landslide movements, and falls into the category of deformation interpretation, as no actual measurement is being done on the site. Also, the knowledge base cannot be applied to residential buildings because the risk factors are different, as well as the surrounding environment.

Additional works include monitoring of tunelling works, which is the most common (and hazardous) cause for deformations present in buildings [16]. The monitoring is done by assessing numerical values of Certainty Factors, allowing representation of and reasoning from the uncertain expert knowledge or domain. Automated monitoring of tunelling works has been applied in real practice on projects in Budapest, Thesalloniki, and Athens [17]. The paper focuses on the use of IT infrastructure to accurately and efficiently measure the deformations, as well as to alarm the user when a potentially hazardous event can occur. This entails installing thousands of geological sensors in the project area, including borehole inclinometers, extensometers, and tilt meters. The advancement of the tunnel, building data, geological and hydrogeological data are being imported into the database in a controlled manner. The system itself serves as an automated reporting device for engineers, which are also required to acknolwedge that they have received the data, and that the measurements are correct, i.e. that the measurements errors are close to minimum.

The Hough transform [18] is used to identify rectangular shaped objects found in image, but it has not been applied to the problem of identifying coordinates of windows in images. Ali et al [5] devises a machine learning approach that utilizes Haar feature model in conjunction with a Gentle Adaboost driven cascaded decision tree by using the set of images from Vienna [2] and Zürich [1] domain to output respective coordinates of windows located within the picture. Cech et al [14] use a segmentation technique that offers a structure model similar to the traditional Markov Random Fields.

Artificial Neural Networks are being accepted as an alternative method in forecasting, because of their advantage in the ability of approximating any linear on non-linear function. In [26], a comparative analysis between a Generalized Regression Neural Network (GRNN), Radial Basis Function (RBF) and Backpropagation Network (BP) is given, for the purpose of predicting different degrees of deformation present in high-rise buildings. Other advantage is that ANNs are a completely data driven method with few assumptions about underlying methods, and instead have the capability to identify the underlying functional relationship among the data. Neural networks simulate a highly interconnected parallel compuational structure with simple individual units that modify the signals that pass through the network.

The GRNN is based on non-linear regression theory, and consists of four layers: input layer, pattern layer, summation layer and the output layer. Let X denote the input vector, Y the output, P the transfer function of the pattern layer, namely:

$$P = exp[-\frac{(X-X_i)^T(X-X_i)}{2\sigma^2}], i = 1, .., n \tag{1.1}$$

Where σ is the smoothing parameter. The summation neurons include two kinds of neurons which can be expressed as $S_D = \sum_{i=1}^{n} P_i$ and $S_N = \sum_{i=1} ny_{ij}P_i$, the expected input $\hat{Y} = \frac{S_N}{S_D}$. Both the number of neurons of hidden layer and output layer are identical with the number of the input sample vectors. When the neurons of hidden layer are enough, the network can approach a smooth function by optional precision. The GRNN had significantly outperformed the BP and RBF function in the problem of prediction deformation values, because of their simple network structure, faster learning speed and function approximation capability. The disadvantage of the other methods are their complex structure, and the necessity to experience trial methods.

In [21], another model of a GRNN network is used to predict the mine slope surface deformations by considering various meteorological factors. The network outperformed BP, and Multiple Layer Regression (MLR) models.

1.3 Contributions

There are several primary contributions in this thesis:

- **Histogram Mean Filtering (HMF)** - is an image processing algorithm which takes images with facades in them as input, and outputs coordinates of windows, i.e. a bounding box. The main advantage of this algorithm is that it uses lightweight methodology and requires little constraints for optimal performance. Additionally, while [5] and [14] both provide good results, the same can be achieved with HMF which has faster completion time.

- **Method for outlier elimination** - After the interest points have been identified by the object segmentation algorithm, the points identified need to be checked for consistency and all outliers need to be removed. A variation of the RANSAC [44] and the SIFT [64] algorithm is described, which does not use a fundamental matrix to find the outliers. The method fully exploits the information gathered from random sampling to eliminate the outliers.

- **Algorithm for merging an ensemble of points** A simplistic algorithm used to merge the interest points into Regions of Interest (ROI) after the consistency checks have been applied, and outliers had been removed is described and implemented. The ROIs can be used by a Knowledge Based System to give further analysis of the deformation measurement process.

- **Application of FIR Neural Networks to the monitoring problem** FIR Neural networks are well known for their application in time series prediction. Artificial Neural Networks are successfully used by the geodetic engineers to minimize measurement errors [62]. Deformation monitoring is a field in engineering geodesy which is almost exclusively done using the Finite Element Method (FEM), or its variations. The FEM method requires the engineers to track the object that is being monitored from the start of its life-cycle, whereas the Artificial Neural Networks take a series of measurements, or epochs and simply output the forecasted or predicted values.

The videometric analysis of the deformations in this thesis represents a novel way of tracking and interpeting the behaviour of the deformations, and the ANNs deployed show that the artificial intelligence techniques can be used as a viable tool for forecasting deformation measurements.

1.4 Thesis Outline

In this chapter, the problem of deformation monitoring is briefly described.

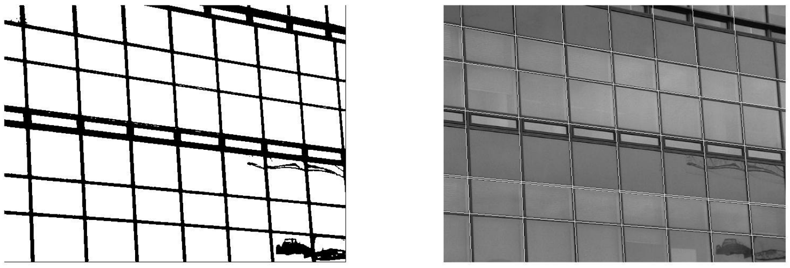


Figure 1.1: Example of Thresholded Image (left) and Final Segmentation (right)

Chapter 2 Different image processing techniques are described, and focus is given to the Histogram Mean Filtering, an algorithm specifically developed for the purpose of identifying coordinates of windows in facades, bounding them in a box. Interest operators, such as Harris and Foerstner are used to first identify the corners of windows, and the algorithm is then applied to fit the bounding box around the window. These coordinates are of special interest to geodetic engineers when they are performing measurement analysis of points in residential buildings, because the points are very robust and easy to measure. An example of the image containing a facade, being processed by the HMF algorithm can be found in Figure 1.1.

Chapter 3 Objects of interest (in this case windows in facade pictures) are observed over time, and the identified points by the algorithm described in Chapter 2 may become misaligned. A variation of the RANdomized SAmple Concensus (RANSAC) algorithm is used to identify these outliers, and perform consistency checks to ensure that only the interest points are being monitored throughout this process. The advantage of this algorithm is that it does not use the fundamental matrix like the RANSAC algorithm, and just like the HMF Algorithm uses lightweight methodology. The method works by comparing vectors within a group, calculating their median and comparing each vector within the selected group.

Chapter 4 After the outliers have been identified, and only the interest points remain throughout the process of monitoring, the next step is to merge the points into Regions of Interest (ROI). A specific algorithm for this problem is described, which takes the (x, y) coordinates of the points of interest, together with the direction of the movement and find the local movement of the cluster of points. The merging algorithm can be used to forward this information to a knowledge based system which identifies potential hazardous movements. Due to the lack of real raw data, artificial examples had been used to conduct the experiments.

Chapter 5 This part of the thesis deals with the structural engineering aspect of deformation monitoring. Every residential building, or structural engineering object for that matter is always under influence of some sort of deformation. There are numerous factors which influence a building to deform, starting from sudden changes in temperature, the mechanics of soil underneath the building, vibrations from the traffic, or tunneling works. The most known reasons for deformation are described, together with the enforcement methods are described which can be used to implement a knowledge based system for deformation interpretation.

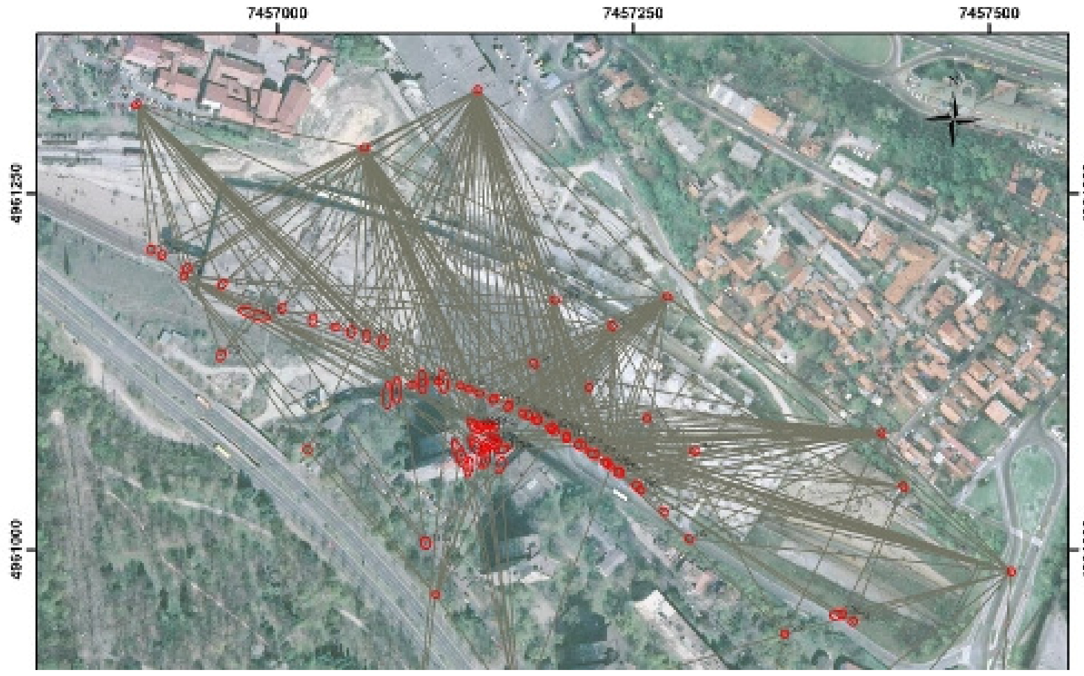


Figure 1.2: Snippet of the Geodetic Control Network, residential buildings included

Chapter 6 The only real data which has been available for study has been gathered from an engineering project which involved tunneling works in Belgrade, and had lasted for several years. A special purpose geodetic network, a variation of the Karlsruhe Method is described in this chapter, together with the measurement findings of the project. A Geodetic Control Network (preview of the network with points from residential buildings can be seen in Figure 1.2) has been reconstructed, and the measurements had been exported into a database which served as training data for the artificial neural networks used in this thesis.

Chapter 7 Different methods used for the problem of time series prediction are being described in this chapter, together with the artificial neural networks. The data gathered from the Prokop project has been used to train a FIR ANN. The network is then used to predict future horizontal and vertical movements. An example of a predicted value of a point in residential building can be seen in Figure 1.3. Horizontal and vertical measurements of the points found in residential buildings can be found in Section 7.12.3, the predictions done by the FIR network can be found in 7.12.4.

Chapter 8 summarises the results achieved in this thesis.

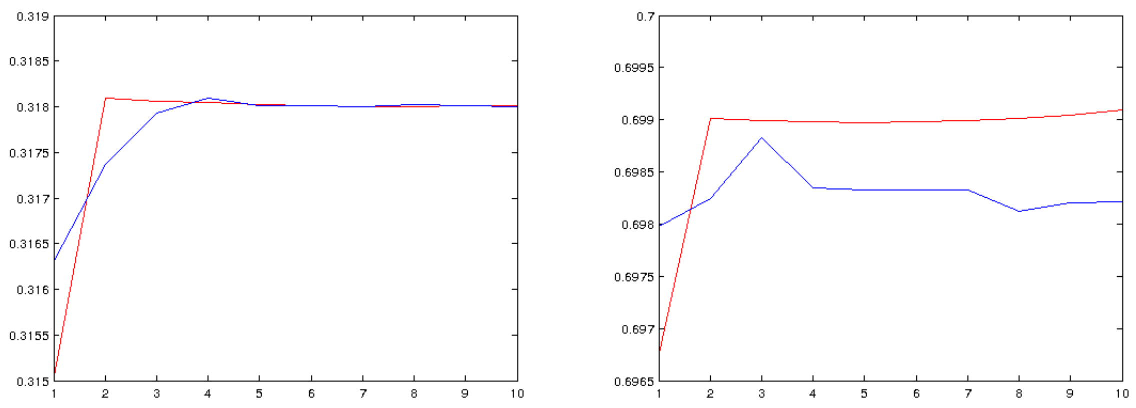


Figure 1.3: Predicted (x,y) values of point 112a, red actual/blue predicted

Appendices A full image of the geodetic control network can be found in Appendix A.1.

1.5 Programming Languages Used

The image processing techniques described in Chapter 2 have been implemented using MATLAB, and can be found in Appendix B. The algorithm for merging interest points into Regions of Interest has been written in CLIPS and can be found in Appendix C. Finally the FIR and RFIR has been implemented using the C++ programming language, and the source code can be found in Appendix E.

1.6 Published Work

The majority of the work presented in this thesis has been previously published in various journals, and proceedings. The idea and concepts behind the thesis have been researched as part of a FWF project, and findings were published in Journal of Applied Geodesy [56] [55]. Computer vision and image processing techniques together with the ANNs described in the thesis have been published in the Indian Journal of Computer Science and Engineering [48] [49].

CHAPTER 2

Object Segmentation

The windows of a building are considered as objects of great interest, since they provide information on the texture of the building to the user. In this work, focus will be given on the facades of buildings. The goal is a method which gives us accurate window coordinates in the image of a facade. We develop such a method which uses an engineering approach rather than a machine learning approach. The argument is that such a method is more reliable and efficient than the machine learning approach. Furthermore, the method we develop may also provide the user with a more general description of objects which can be found in the image.

Motivation Detection of windows in facades and buildings can be of great use in monitoring deformations and rigid body movements. Traditionally, the deformations have been monitored by measuring reflectors attached to the buildings. However, novel measurement setups avoid such reflectors (e.g., for esthetical reasons), and measurement has to be done on image-based data alone, as possible with modern video-theodolite. As for deformation monitoring, the coordinates of windows in a time series of images are identified as reference points for assessing whether, and which, deformations are happening. Currently, the engineer has to determine the window coordinates at each epoch manually. The idea is an automated detection using interest operators (IOPs), most notably Harris [29], and Förstner [25], which single out points with special properties from the image. In order to effectively apply this approach, one needs an implemented method for automatic detection of windows from an image respectively a series of images.

Problem Description We consider the basic problem where the method is given a single image of a facade. The outcome of the method should be the (exact) coordinates of windows found in the image. The windows in the image are not necessarily rectangular, but also might have other shapes due to the perspective or different shape of windows. The method should be robust with respect to different lightning conditions, and also with respect to shadows as met in practice.

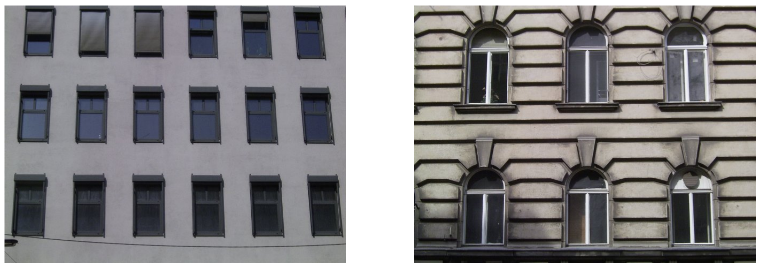


Figure 2.1: Examples of images containing windows

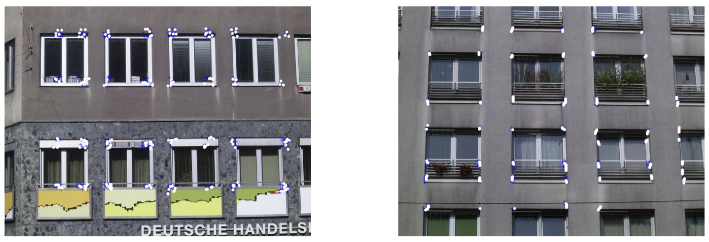


Figure 2.2: Examples of Förstner and Harris interest operators

Current Methods In a paper by Haider et al [5], an algorithm for window detection is presented. The algorithm uses machine learning techniques, to find the coordinates of the window. Other than this method, algorithms have been designed which try to capture rectangular-shaped objects in an image, using the Hough Transform [18]. Although this method is very effective, it is not suitable for window detection, since it captures all objects in an image which resemble to rectangles. Applied to the image of a facade, far too many objects will be classified as windows. The method can be used to find other objects of interest in the image besides windows (like balconies, bars, etc). Still, an algorithm which can filter windows from other objects is needed, and which is also applicable under the conditions mentioned above.

Contribution Our main contribution is a method which takes, as input, an image of a facade, and provides, as output, for each of the windows in the image a bounding box. The method combines several image preprocessing algorithms, but also employs two new algorithms, which have been specifically developed. The first performs thresholding of a facade image, and the second segmentation of objects in a facade image and window separation from other objects. Furthermore, we present two numeric measures for the quality of windows detection, which are motivated by the application, and evaluate an implementation of our method against these measures.

2.1 Preliminaries

The images which we use here are gathered from the Vienna [2] and Zürich [1] public domain databases, which are available on the web and provide us with a very diverse set of images for testing. The pictures in the database give a clear view of the facade, but not of the building.

2.1.1 Assumptions

One of the assumptions is that the picture should not be taken at a small angle (less than 30°). Another important aspect is that the user should use a high quality instrument, such as a video-theodolite, in order to deal with the problems encountered with depth of focus, saturation, motion blur, and lens distortion. Images which are not taken in this manner do not provide the user with good information. However, these assumption apply in practice, even with the traditional methods, and thus are no limitation.

2.1.2 Problem Constraints

The main constraint which is set in this paper is that facades should be clearly visible, and not covered with excessive objects. In particular, the images do not contain additional objects, such as cars or trees (which are also hindering manual measurement). The performance of our method depends on the quality of the image. If additional objects are present, manual (as currently done) or (semi-)automated segmentation has to be done. On the other hand, shadows do not cause problems, because of the thresholding algorithm incorporated in our method.

2.2 Method

This section describes our method, and the image processing techniques which are used by it. The method works in four steps. In the first step, the facade in the image is classified into one of several types. After that, appropriate image preprocessing algorithms, are applied. In the third step, thresholding techniques are applied, and in the fourth and final step the image is segmented using Histogram Median Filtering (HMF).

2.2.1 Classification

In image processing, it is quite common to use simple statistical descriptions of images and subimages. The notion of a statistics is connected to the concept of probability distribution, generally the distribution of signal amplitudes. For a given region which could conceivably be a window, the probability distribution function of the brightness in that image is defined.

The algorithms need to have prior knowledge as to what kind of facade they are segmenting. There are three different types of images in the database: classical, modern, and ordinary. The probability distribution function of the brightness has been used to differentiate between three types of facades. This function behaves differently when dealing with different types of facades (modern facades have a very low value, classical have a high value due to the texture, and the rest are classified as ordinary facades).

2.2.2 Preprocessing

Before the image of a facade can be segmented, it needs to be preprocessed, so that the noise can be discarded. The choice of preprocessing algorithms depends on the type of facade. For example, normal buildings need very few preprocessing algorithms to be applied, such as normalization and equalization, whereas classical buildings need contrast stretching. The description of these algorithms can be found below. Modern buildings require unsharp masking, because the windows are too bright.

Contrast Stretching Frequently an image is scanned in such a way that the resulting brightness values do not make full use of the available dynamic range [52]. By stretching the histogram over the available dynamic range we attempt to correct this situation. If the image is intended to go from brightness 0 to brightness $2^B - 1$ then one generally maps the 0 percent value to the value 0 and the maximum to the value $2^B - 1$. The appropriate transformation is given by:

$$b[m,n] = (2^B - 1)\frac{a[m,n] - min}{max - min} \tag{2.1}$$

This formula can be sensitive to outliers and a less sensitive and more general version is given by:

$$b[m,n] = \begin{cases} 0, & a[m,n] \leq p_{low} \\ (2^B - 1)\frac{a[m,n] - p_{low}}{p_{high} - p_{low}}, & p_{low} < a[m,n] < p_{high} \\ (2^B - 1), & p_{high} \leq a[m,n] \end{cases} \tag{2.2}$$

where p_{low} and p_{high} are dynamic ranges of the percentages in which the contrast of the picture is to be stretched, usually set to 10% and 90% respectively.

Unsharp Masking A well-known technique from photography to improve the visual quality of an image is unsharp masking, i.e., to enhance the edges of the image. This means isolating the edges of an image, amplifying them, and then adding them back into the image. This leads immediately to the technique:

$$a'[m,n] = a[m,n] - (k \times \Delta^2 a[m,n]) \tag{2.3}$$

where the term k is the amplifying term, and $k > 0$.

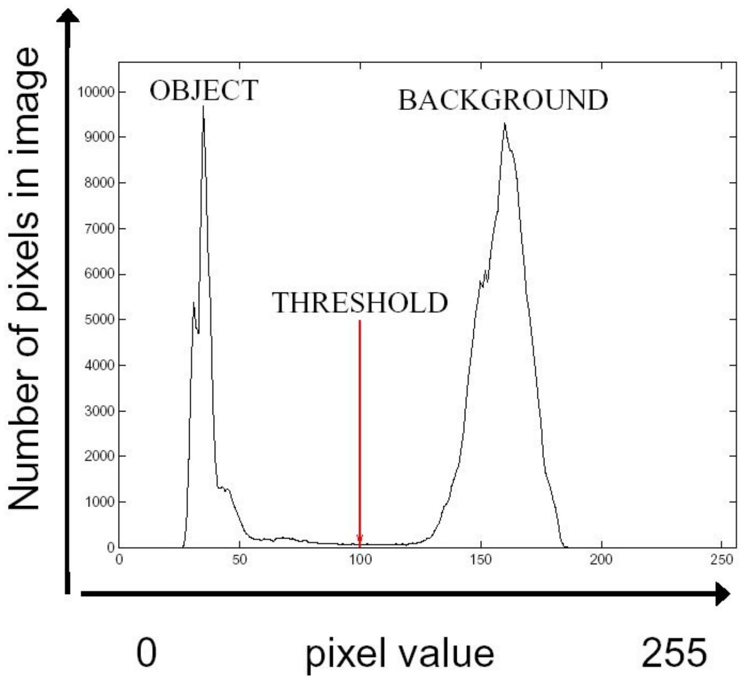


Figure 2.3: Selection of a threshold

2.2.3 Thresholding

Once the image has been preprocessed, thresholding can begin. In order to separate an object from the background, in this case, a window from the facade, an appropriate threshold needs to be chosen. When this is done, bimodal distribution is exploited. Distributions are broad, and may overlap. Possible problems include shadows because they are dark and may be classified as an object. Figure 3 shows an example of how the threshold should be chosen, it is located between the object and the background. In order to get rid of the shadows, which are sometimes included as part of the object, the histogram has to be smoothed.

The process of thresholding is the most important because the windows need to be preserved, and excessive objects have to be removed. Several thresholding algorithms from the literature have been tried (such as isodata, and background symmetry algorithms [52]), but the results have not been robust. Therefore, we have developed a special algorithm for thresholding, described below.

Local Adaptive Thresholding 4 (LAT-4) This algorithm has been specifically designed for window detection. First, the horizontal and vertical histograms are calculated. The 1D Convolution [52] is applied to smooth the histogram. After that, the image is partitioned according to the position of local minima. These partitions are then used

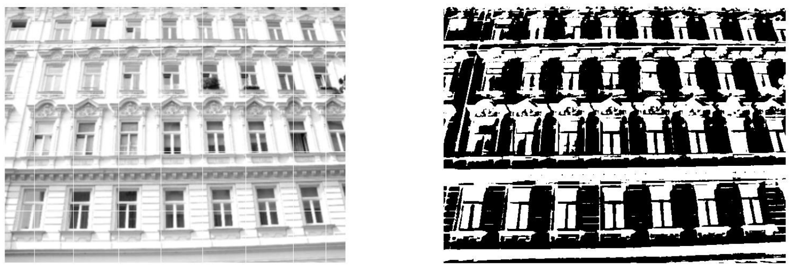


Figure 2.4: Image before and after applying LAT-4 (white lines in the first image represent local minima)

as sub-images, which are later on divided into four parts. Therefore, four different thresholds are used for each subimage, or in total $4 \times N_{min}$, where N_{min} is the number of minima. This algorithm has been chosen, as it gives also good results when the light is not uniformly distributed in the image.

2.2.4 Segmentation

Once the image has been thresholded, the analysis of the histograms comes into place. The main core of the method for window detection is Histogram Mean Filtering (HMF) developed specifically for the detection of windows in facades.

2.2.5 Histogram Median Filtering

Windows in facades follow a particular pattern, and they have similar pixel values (due to their similarity). After thresholding, the user is left with a black and white image $(0, 1)$ which holds the information on the objects present in the building. The next step is to calculate horizontal and vertical histograms H_h and H_v, [55] respectively, given by

$$H_h = \sum_{i=1}^{W} I_t(h, i), h = 1, \ldots, H \tag{2.4}$$

$$H_v = \sum_{i=1}^{H} I_t(i, v), v = 1, \ldots, W \tag{2.5}$$

where W, H are the width and height of an image, respectively, and I_t is the thresholded image. Windows are the most numerous objects to be found in the facade (depicted as white pixels in $I_t(x, y)$) and are aligned in rows and columns i.e., they form a pattern. The values of the two histograms will have high volume when processing an area with windows in them. Therefore, after the calculation of the two histograms, the values which are greater than the medians M_h and M_v will hold the position of the window along the horizontal and vertical axes. After filtering the histogram values, the user is left with a series of numbers in the range of $1, \ldots, W$ and $1, \ldots, H$ which actually represent values along the h and v axes that hold window coordinates.

2.2.6 Histograms at an Angle

The limitation of Histogram Median Filtering (HMF) is that it can only work when the windows are aligned at a near 90° angle. To overcome this limitation, histograms are taken at an angle α, and new histograms $H_h(\alpha)$ and $H_v(\alpha)$ are calculated respectively. The new algorithm HMF(α) works exactly the same as HMF, except that it does not know the exact value of α. The algorithm starts by trying with an angle 10°, finishing at 100°, with a step of 10°. The results obtained in this way are suitable in many cases (see results for a further discussion). The quality can be improved by iterating with a step of 1°, which incurs high computation time, or by dynamic step adjustment, which decreases the step if the matching gets close.

Figure 2.5 (right) shows lines that actually represent the histograms which were taken at an angle. The algorithm [56] stores this information and finds the best fitting line which will capture the window by comparing histogram values (the best one is minimal in this case). The final segmentation can be seen in Figure 2.6 (right).

The idea behind angled histograms is that they will capture much more information than the "regular", histograms, because they can capture more information in the thresholded image. HMF works first by calculating regular histograms. If the results are poor, then the histograms will be taken at an angle. The values are stored, and then later on compared to see which one has captured most information.

2.3 Evaluation

In order to measure the quality of an algorithm for window detection numerically, different criteria may be considered, depending on a particular application. Three natural criteria for assessing the similarity between the captured window W and the target window T (i.e., the manually determined ground truth) are structure (i.e., shape), the area extent, and the location in the image. The desire is to have the captured window include the real window, in particular when doing the segmentation of the image.

2.3.1 Structure Similarity

There are various ways to measure similarity between two structures [64], but there is no precise methodology for determining similarity between two rectangles (windows). Rectangles can be observed as polygons, but this is not the most reliable way. Only a degree of similarity is possible to measure using existing methods, which are in fact only classification algorithms as they can tell the user whether two structures are either very similar or completely different.

A naive approach for measuring structure similarity would be to use the width/height ratio of the target and real window. In order to measure a degree of similarity of two windows, it is appropriate to use distances between corners. For the purposes of our

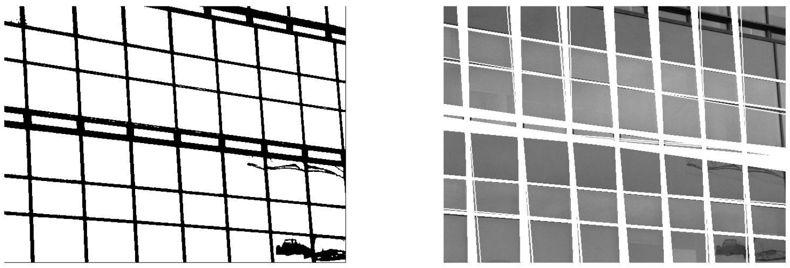


Figure 2.5: Thresholded image (left) and histogram “candidates” for capturing the window (right)

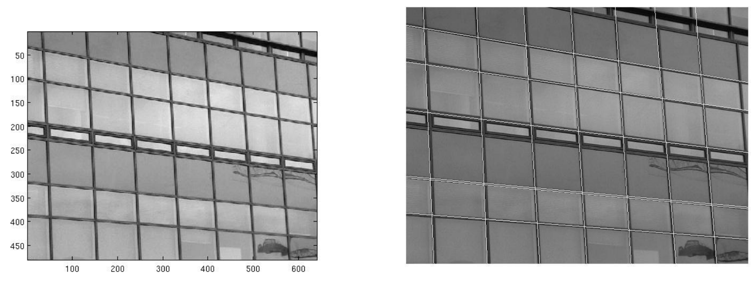


Figure 2.6: Original image (left) and final segmentation (right)

application, the similarity of shapes is measured by comparing the distances between the coordinates of windows, i.e. their corners.

2.3.2 Area Extent

A natural measure for the similarity of the area extents of the captured and the target window is the ratio between their common and total extents, i.e., the fraction of overlap. We refer to this as *Area Coverage (AC)*, which is formally given by

$$AC = \frac{1}{n} \times \sum_{i=1}^{n} \frac{A(T_i) \cap A(W_i)}{A(T_i) \cup A(W_i)} \tag{2.6}$$

where n is the number of windows, and $A(W_i)$ respectively $A(T_i)$ is the area of the captured window W_i respectively the target window T_i.

In particular if it holds that $T_i \subseteq W_i$, i.e. the ground truth is always contained in the captured window, for all $i = 1, \ldots, n$ (as it occurred in our experiments), then the above expression simplifies to

$$AC = \frac{1}{n} \times \sum_{i=1}^{n} \frac{A(T_i)}{A(W_i)} \tag{2.7}$$

which measure coverage. The quality of the result AC is actually the mean of the ratios between the bounding box and the window size. It represents a good quality measurement for the detection of windows, as it provides the user with information on how well the windows have been bounded by the given coordinates from the output. The goal is to have AC as close to 100% as possible. Indeed, when applying the Harris [29] or Förstner [25] operator on the image to single out the window corners precisely, the captured windows W_i (which are used to filter the results of the Harris or Förstner operator), should cover only the target windows T_i. If AC is close to 100%, no other objects will fit into the captured windows W_i.

2.3.3 Location in the image

For measuring the similarity of location between the captured and the target windows, we use *Center of Mass (CoM)*. For each captured window W_i and target window T_i, the center of mass $CoM(W_i)$ resp. $CoM(T_i)$ is evaluated. Then, in order to assess how much W_i resembles T_i, the vector from $CoM(T_i)$ to $CoM(W_i)$ is calculated to see in which cardinal direction the captured window W_i is moving. This CoM vector is calculated for all windows W_i and T_i in all images, in order to produce results that can provide information on how well the location of the captured window fits with reality.

2.3.4 Distance between Corners

For measuring shape similarity, distance between corners (d_c) is used, which is another way to measure similarity between two structures. This measurement indicates how the target and the real window are aligned, and it can cover structure similarity very well. The desire is to minimise the distance. Formally, d_c is calculated by averaging distances between corners in one picture. This measurement methodology is used to determine by what margin the real window fits into the captured. The results of d_c should correlate with the results of AC under the assumption that there is a bounding box present, and can also be used as an alternative to area extent, by normalizing this value with the real height and weight of the window. However, the same method does not correlate with the center of mass, because the distances cannot provide the information on the alignment of the center of masses of the real and target window. Nevertheless, this method represents a good addition when doing the evaluation, as not only does it cover shape similarity, but also area extent.

2.4 Results

We have tested an implementation of our method on a database containing 500 images in total selected from the Vienna and Zürich public domain. The database is full of a

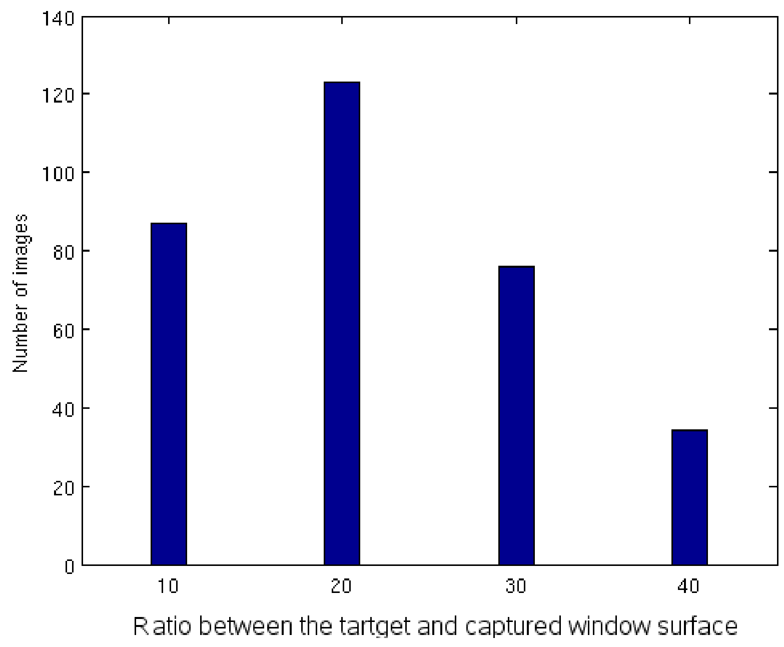


Figure 2.7: Performance of the algorithm

diverse set of images, with classical, modern, and ordinary facades, taken from different perspectives. The images used for the experiments had a resolution of 640 × 480 pixels, and the windows had an average of 50 × 100 pixels.

On visual inspection, the method yields good results. In all cases it singles out reliably bounding boxes for the windows in the images. These bounding boxes properly contain the windows. When determining the coordinates of the bounding box, the desire is to have a small margin of three pixels present, for the purposes of our application. The small margin has to be present in order to properly calculate the Foerstner and Harris operators.

Area Coverage (*AC*) The performance results of our algorithm with respect to area coverage (*AC*) are displayed in Figure 2.7. As mentioned above, in total 500 images have been used for testing, and the algorithm always returned true bounding boxes of the target windows.

Out of the 500 images, the algorithm classified 127 images with an *AC* value for target and captured windows of at least 90% (mean of 92.67%), 197 with a ratio between 80% and 90% (mean of 84.8%), and 138 images with a ratio between 70% and 80% (mean 77.4%). The remaining 78 images had a ratio between 60% and 70%.

Center of Mass (CoM) The dislocation of the CoM had been evaluated with respect to the window width. Out of 500 images, 107 images had the CoM displaced for 5% of the window width, 218 images had the center of mass displaced for 10%, and 88 images had the CoM displaced for 15%, and finally 87 images had a displacement of 20%.

The results have shown that the algorithm can find the exact coordinates of the windows with high accuracy with respect to AC, d_c and CoM. The algorithm will always capture every window in an image. Furthermore, 31% of the images have been segmented with very high accuracy $AC > 90\%$, whereas the images with $70\% < AC < 90\%$ have given very good segmentations of the facade.

With respect to subsequent detection of window corners by using Harris and Förstner operators, an AC quality of less than 70% for an image in the example database did not lead to good results. Even if the windows were captured, the corners of the bounding boxes were too far away from the window corners since the bounding boxes were too big. For the remaining images (87.85%), the result was good.

The experimental results for CoM have shown that in worst case scenarios, the center of mass of the target window is moving upwards, or north. This occurs in particular in facades of classic style. In such facades, other objects are found just above a window. When the algorithm is trying to fit a box around the window, the extra objects intervene in the calculation of the target coordinates. The results were less satisfactory in images where the windows were at an angle.

Distance between Corners The results have shown that in 158 images, dc was between 5 and 15 pixels, whereas in 257 images, the distance was between 15 and 25. The remaining 85 images had the distance over 25 pixels. Furthermore, if one was to measure shape similarity for each rectangular side, that is, to average distances between each of the two corners, and to exclude the images with classical windows, the results would improve greatly, as this distance would never be above 20 pixels. Thus, the results have shown that there is high similarity between the captured and real window in most pictures. For the purposes of our application, the extra distance between the corners of actual and real window is negligable, as in a lot of cases, it is desired to have some extra coverage of the real window, when calculating the Foerstner and Harris operators, in order to get the maximum number of interest points. The results of this methodology have shown that this extra coverage does not produce a lot of unnecessary points, and that all interest points are covered.

2.4.1 Comparison Study

Haider et al [5] describes an algorithm for window detection which uses machine learning techniques. The algorithm uses images from Vienna and ZuBud public domains. Evaluation is carried out using positive true and false based samples, in the frame of the pattern based detector. Furthermore, the algorithm is evaluated using Single Window (SW), and Window Region of Interest (WROI). For SW evaluation, positive true samples are counted if one detection rectangle would be found inside a mask rectangle or at the maximum, would overlap it only by a few pixels in each direction. For WROI evaluation, positive true samples are counted whenever the mask rectangle is covered by a certain percentage. The algorithm has performed well using these testing methodologies, but it is notable to say that our algorithm would yield 100 % on WROI and SW if applied, due to the nature of our algorithm. However, the algorithm from Haider et al works better when there are additional objects present in the picture. The Hough transform can be used to detect rectangular shapes in an image, but these rectangles would have to be filtered to locate the real coordinates of the window. This method can be used as an addition to our algorithm, but the results would not improve significantly.

2.5 Single Point Detection

The feature points are detected at sub-pixel accuracy using a corner detector. Using the detected object structure and the assigned ROIs, individual interest points (IPs) need to be detected. For the task of highest precision point detection both sensor units can be used (IATS and TLS). In the following we will focus our description on the point detection by the means of IATS. The internal camera of the IATS captures images covering the extracted ROIs (the internal camera is used instead of the WA-camera due to the larger image scale and in consequence the higher accuracy of image point detection). The internal camera of the IATS captures images covering the extracted ROIs (the internal camera is used instead of the WA-camera due to the larger image scale and in consequence the higher accuracy of image point detection). As a first step image pre-processing can be performed to transfer the image into a form which is better suited for the subsequent automated image point detection. On such processed images, single interest points (IPs) can be detected by interest operators (IOPs). Among the large set of available IOPs (e.g. Forstner 1987, Harris 1988, Moravec 1997) none is suitable for all IP types.

2.6 Correspondence Analysis

The points can be matched from one image to the next by choosing matches which have the highest cross-correlation of image intensity for regions surrounding the points. This is currently supported by a robust matching technique, we are using the HFVM algorithms. Matching the points in image space and transforming them into object space enables the measurement of corresponding points in different epochs.

CHAPTER 3

Outlier Elimination

Due to misalignments or moving object in the scene, some of the correspondences may be incorrect. A RAndomized SAmple Concensus (RANSAC) [24] approach is used: take small groups of feature points each time and see if others move consistently with them; if not the others may be outliers. RANSAC is a simple, yet powerful technique that is commonly applied to the task of estimating the parameters of a model, using data which may be contaminated by outliers. RANSAC estimates a global relation to that fits the data, while simultaneously classifying the data into inliers (points consistent with the relation) and outlier (points not consistent with the relation). RANSAC operates in a hypothesize-and-verify framework: a minimal subset of the input data points is randomly selected and model parameters are estimated from the subset. The model is then evaluated on the entire dataset and its support (the number of data points consistent with the model) is determined. This hypothesize-and-verify loop is repeated until the probability of finding a model with better support than the current best model falls below a predefined threshold (typically between 1 and 5 percent). RANSAC can often correct solution even for higher levels of contamination; however, the number of samplers required to do so increases exponentially, and the associated computational cost is substantial. Another alternative to this problem would be to use the sequential correspondence algorithm, as an alternative for the RANSAC algorithm. Both algorithms are described below.

3.1 From Feature-Points to Motion Parameters

Once the set of corresponding points between two images is obtained, the information can be used to determine the motion parameters. Since each point-correspondence gives us two equations of constraints âĂŤ one for the horizontal component, and one for the vertical âĂŤ four point-correspondences to solve for the eight parameters of the perspective motion model are required. Inserting any four correspondences $p_i \leftrightarrow \hat{p}_i$ with

$p_i = (x_i, y_i, 1)$ and $\hat{p_i} = (\hat{x_i}, \hat{y_i}, 1)$ into the inhomogeneous formulation and multiplying with the denominator results in a linear equation system.

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1\hat{x_1} & -y_1\hat{y_1} \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1\hat{x_1} & -y_1\hat{y_1} \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2\hat{x_2} & -y_2\hat{y_2} \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2\hat{x_2} & -y_2\hat{y_2} \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3\hat{x_3} & -y_3\hat{y_3} \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3\hat{x_3} & -y_3\hat{y_3} \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_1\hat{x_4} & -y_1\hat{y_4} \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_1\hat{x_4} & -y_1\hat{y_4} \end{bmatrix} \begin{pmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \end{pmatrix} = \begin{pmatrix} \hat{x_1} \\ \hat{y_1} \\ \hat{x_2} \\ \hat{y_2} \\ \hat{x_3} \\ \hat{y_3} \\ \hat{x_4} \\ \hat{y_4} \end{pmatrix} \tag{3.1}$$

which can be easily solved for the model parameters h_{ik}. While four cor- respondences are enough to solve for the motion parameters, although many more correspondences are available. However, these are contami- nated with outliers and inaccuracies in the feature positions. Apart from errors in the computation of the point-correspondences, correspondences that are part of foreground object motion as outliers are considered.

3.2 RANSAC Algorithm

In this section, the RANSAC algorithm for the special case of estimating the parameters of a two-dimensional perspective motion model is described. The set of correspondences, which are used as algorithm input are denoted by $C = p_i \leftrightarrow p_i$, whereas the Euclidean distance between two points p_i and $\hat{p_k}$ as $d(p_i, \hat{p_k})$. The RANSAC algorithm can then be described with the following steps.

1. Draw a subset S of size $|S| = 4$ from C. Four correspondences are required to solve for the eight free parameters of the motion model.

2. Compute the parameters h_{jk} of the motion model H from the correspondences in S using the linear system in Equation 3.1.

3. Determine the set of inliers $I = p_i \leftrightarrow \hat{p_i} \in C | d(\hat{p_i}, Hp_i) < \epsilon$ which is the set of correspondences that comply with the motion model. In other words, this means that we use the current set of parameters to transform the features from the first image into the second and compare this with the measured positions. If the distance is low, then the pair of points is assumed to comply with the motion model, and it is selected as an inlier.

4. Repeat Steps 1–3 several times (N) and choose the set of inliers for which $|I|$ is largest.

5. Perform a least-squares approximation of the motion parameters with the set of inliers. The solution is the result of the RANSAC algorithm.

The RANSAC algorithm has two parameters that have to be chosen initially: the number of draws N and the inlier threshold ϵ. A good value for the inlier threshold can be obtained from the evaluation of the feature-point detector. The more accurate it can locate the features, the smaller can be chosen. [59] The number of found correspondences by increasing ϵ saturates very quickly. Hence, we have chosen a small value around 1.5 for ϵ, but the right selection of is not critical. If it is chosen too low, some correct correspondences will be sorted out as outliers, but usually the set of inliers is still large enough to estimate accurate model parameters. If it is chosen too high, some outlier data will be included, but since these outliers cannot differ much from the inliers (their error is below ϵ), their influence in the least-squares approximation will be limited.

3.3 Consistency Checking and Outlier Removal

A new method has been developed for the elimination of outliers found within the matching process of two or more images. The method does not use the fundamental matrix like in RANSAC, or SIFT, and requires little calculation. The algorithm works by comparing vectors within a group, calculating their median, and comparing it with each vector within the selected group.

The idea is to remove the outliers using a recurrent filter. For each group of the vectors, which lie at the assigned area (a specified window), the average length of vector and slope angle is calculated. All of vectors, whose length and angle differ more than to the assigned magnitude of deviation from the average - are moved away, and the entire procedure is repeated with the more rigid values of the maximum permissible deviation from the average. This procedure works with at least three vectors being present in the group. But if the group contains only one vector, then the algorithm finds the nearest and adds it to this group. If the group contains two vectors, then the angle and the length of difference from one to another is calculated and if it does not exceed the threshold value, then those two vectors are considered to be consistent with the rest. Otherwise the nearest vector is added to the group, and the calculation is performed as for the rest of the vectors, where there are three and more present.

The advantage of this algorithm is that it does not use the fundamental matrix like the RANSAC, or SIFT algorithms. RANSAC and SIFT algorithms are mostly applied in tracking of objects, because the search space is too big. In this problem however, the vectors that are being compared have limited movements as opposed to objects that can be tracked by RANSAC that can go in all sorts of directions.

3.4 Performance of the Outlier Algorithm

The following three examples were created using a warping tool from TU Graz which deforms an image based on the inputed vector. The outlier algorithm has then been applied to identify the outliers. Figures below depict the process of matching points between different epochs, from identifying the interest operators, identifying their matches, and matching feature points. Figures 3.1 3.2 3.3 depict the interest operators between both epochs, Figures 3.4- 3.6 their matches together with their feature points, and Figures 3.7- 3.9 show the matches on both epochs marked with circles in different color, where the image on the left is a the image of the current epoch merged on top of the image of the next, together with their respective matches.

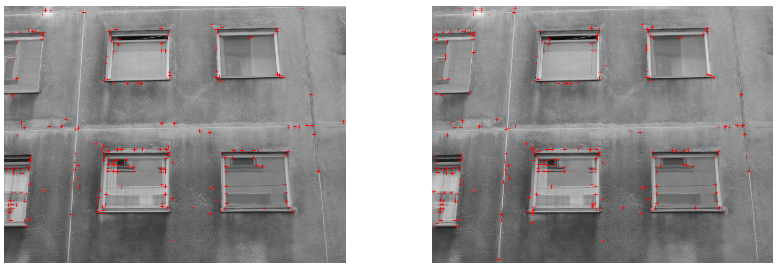


Figure 3.1: Example 1: Interest Operators between two epochs

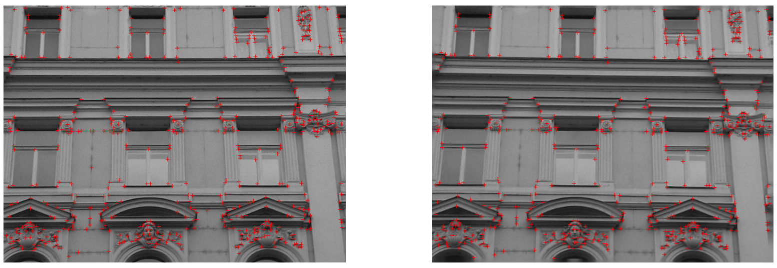


Figure 3.2: Example 2: Interest Operators between two epochs

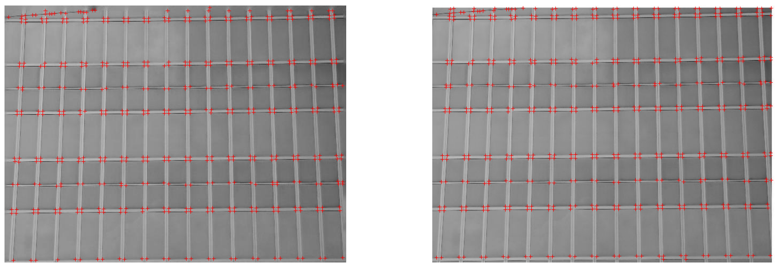


Figure 3.3: Example 3: Interest Operators between two epochs

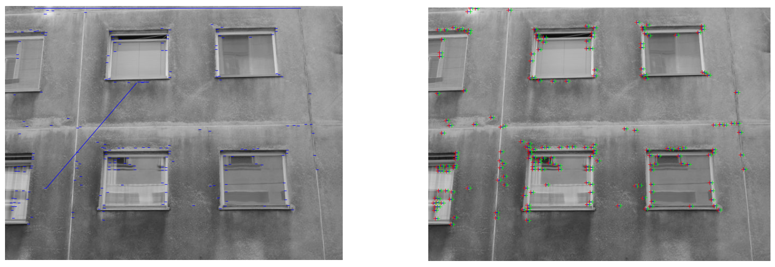


Figure 3.4: Example 1: Matches between two epochs (left), Inlied matched feature points (right)

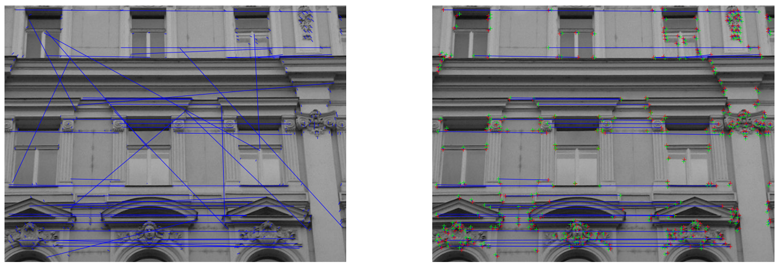


Figure 3.5: Example 2: Matches between two epochs (left), Inlied matched feature points (right)

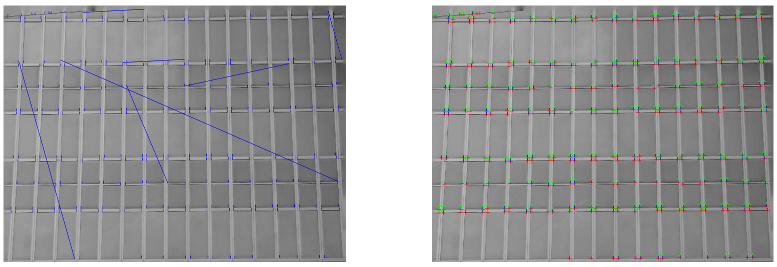


Figure 3.6: Example 3: Matches between two epochs (left), Inlied matched feature points (right)

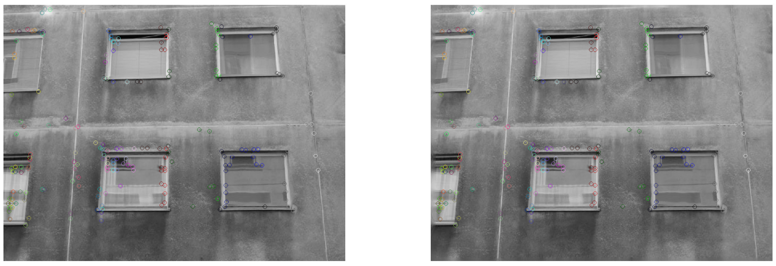


Figure 3.7: Example 1: Matches displayed on both epochs

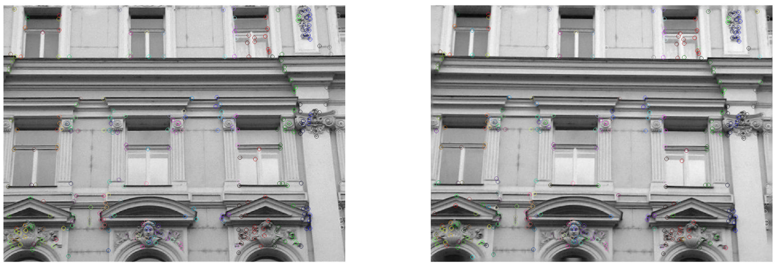


Figure 3.8: Example 2: Matches displayed on both epochs

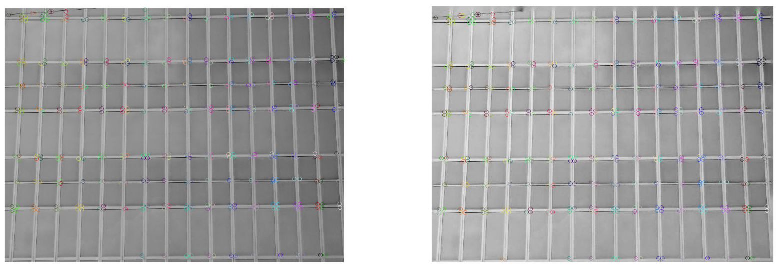


Figure 3.9: Example 3: Matches displayed on both epochs

The blue lines on the left depict the actual movement that takes place within the image which was generated using the warping tool, so in fact it is the vector that has been used to distort the image. In all three examples listed above, the image is distorted by a translation vector, except for the second example which also uses a rotation vector. The results showed that when the translation vector is used as an example of a potential deformation, the algorithm had no problem matching the interest operators between the two epochs (see Figure 3.4 and Figure 3.6. On the other hand when the vector containing a rotation is used, the algorithm performed slightly worse, making it a more difficult 'deformation' to track. The second example has been chosen as a worst case scenario, because it depicts a classical residential building, which has been affected by a 'deformation' with a vector representing a rotation.

Outliers can easily be detected before they are passed on to the knowledge base for interpretation, and the algorithm performs as well as the RANSAC method except for the fact that it does not use a fundamental matrix, making it more cost-effective in terms of computational power. In the listed examples, one can see that there are not that many outliers present between epochs, and that most of the interest operators are easily matched.

This can be explained by the nature of the warping tool, even though it can transform an image over time, it cannot simulate the behaviour of noise between the epochs. Even though the tool can simulate structural movements, it is not able to simulate different levels of lighting that may affect the quality of the image, which have a big impact on the matching success rate between the two epochs.

Had the algorithm been used on a real life scenario, the RANSAC algorithm probably would have yielded better results, but because it was tested on simulated data, both of them detected the outliers efficiently, except that the method described in this section did so with very little effort, which made it a good choice for this particular type of problem. From the results, it can be seen that the algorithm did not perform well in Figure 3.5 (left), where the observed facade falls into the category of more classical (historical) buildings. These particular type of facades have also been a problem for the object segmentation algorithm used in this thesis, because the interest operators tend not to be so focused around the window, but rather on the triangular shapes below the window, as seen in Figure 3.2. Additionally, the rotation vector used in this example made the matching more difficult for the algorithm.

CHAPTER 4

Merging an Ensemble of Points

Once all the outliers have been detected, the user is left with a dataset of points together with their corresponding description of a movement. RANSAC has pruned the data set considerably, and only the points which are consistently moving with each other remain. The next step is to merge this point into a Region of Interest (ROI), and to find the local movement of that particular cluster of points.

This can be done by incorporating a simple merging algorithm. Assuming that we are left with an array of points L, then each point is represented as (x, y, m) where x and y are the coordinates of the point, and m is the direction (movement) of that point, i.e. translation, or rotation. Then we proceed as following:

1. Compare two points (x_1, y_1, m_1) and (x_2, y_2, m_2), and check if they are moving together. Next, check if they are close to each other, by calculating their equidistance from one another. If that is the case, then merge the two points together as (x_1, y_1, x_2, y_2, m) where $m = m1 = m2$. Remove the both points from the list, and insert the newly formed into the list.

2. Compare the remaining points left in the list, together with the ones which were formed in (1) as well. For example, if (x_3, y_3, m) is moving the same as (x_1, y_1, x_2, y_2, m) and (x_3, y_3) is a point which lies close either to (x_1, y_1) or (x_2, y_2), then it should be included in this subset, and the designated points from that subset should be removed. Consequently, similarly as in (1) remove the point which has been compared, and classified.

3. Repeat steps 1–2 until there are no more neighbouring points.

The user is then left with a list L of Regions of Interest which describes in what way a particular ROI is moving, which is then later used for deformation assessment, and consequently interpretation.

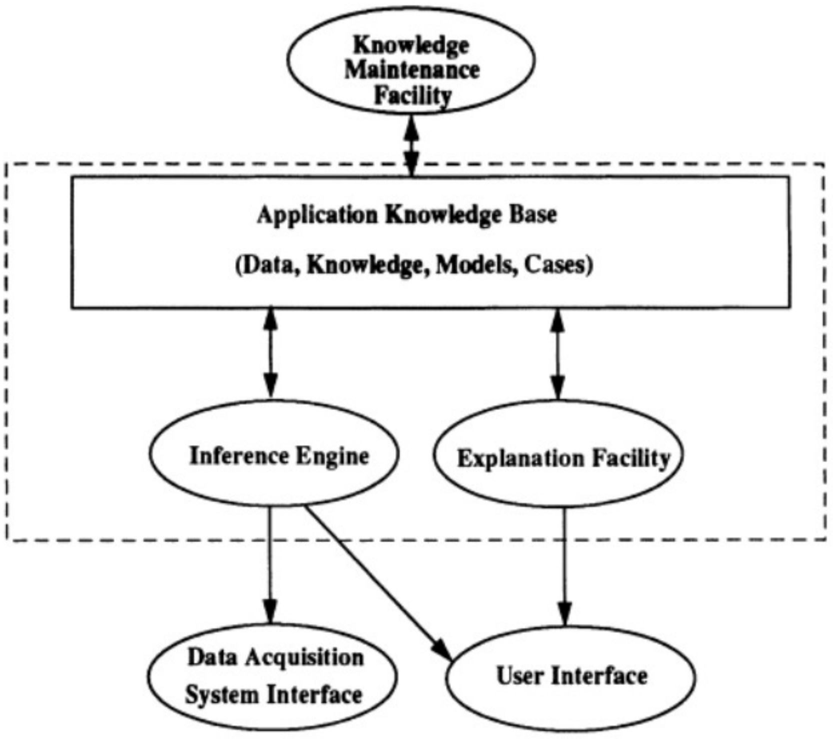

Knowledge Maintenance Facility
Application Knowledge Base
(Data, Knowledge, Models, Cases)
Inference Engine
Explanation Facility
Data Acquisition System Interface
User Interface

Figure 4.1: Structure of an expert system [46]

4.1 Rule-based Systems

Conventional problem-solving algorithms make use of data structures, and reasoning strategies to find solutions. For the difficult problems which are addressed by the expert systems, it may be more useful to employ heuristics, or strategies that often lead to a viable strategy, but that also sometimes fail. Conventional rule-based expert systems use human expert knowledge to solve problems that normally would require human intelligence. Expert knowledge is represented in the form of rules, or as data structure within the computer.

The basic components of an expert system can be found in 4.1. The knowledge base stores all data, rules, and cases used by the expert system. A rule is a conditional statement that links to actions or outcomes. A frame is an alternate approach used to capture and store knowledge, which relates an object or item to various facts or values. Expert systems that are using frames are often referred to as frame-based expert systems. The purpose of the inference engine is to retrieve information and relationship from the knowledge base, and to provide the user with answers, or suggestions.

A rule-based system consists of *if-then* statements, a bunch of facts, and an interpreter controlling the usage of the rules, given the facts. These statements are then used to

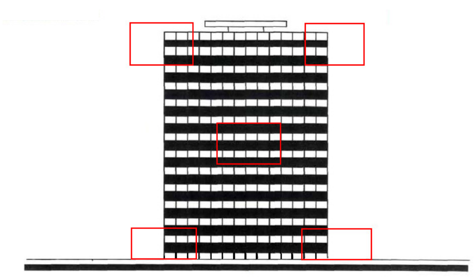


Figure 4.2: Suggested ROIs to be observed by the KBS

formulate the conditional statements that form the entire knowledge base. A single *if-then* rule is in the form of 'if x is A then y is B', where the if-part of the rule is called the antecedent, or premise, while the then-part is called the consequent or conclusion. There are two kinds of inference engines used in rule-based systems: forward chaining, and backward chaining. In a forward chaining system, the initial facts are processed first and are kept being used until a conclusion has been reached, whereas in a backward chaining system the conclusion, or the hypothesis is being processed first along with the related rules that would allow to conclude the hypothesis.

Most expert systems are developed using tools called shells, which are much like the terminal shells found in UNIX distributions, except that they are equipped with an inference mechanism that supports both forward and backward chaining, and require knowledge to be entered according to a specified syntax. CLIPS [3] is an expert system tool that provides the environment for the development of rule, object oriented, or procedural expert systems. CLIPS is written in C for portability and speed and is available on different operating systems.

4.2 Knowledge Base for Regions of Interest

Once the ROIs have been identified and classified, it is time to detect the patterns that may appear in building deformation. There are many deformation patterns possible, and the expert system should be able to detect them all and provide the user with some feedback, such as whether the deformation is plausible, or whether the deformation is potentially hazardous.

Typical deformation patterns may include settlements which can be present either at the center of the building, or at both ends of the building, or at the whole building itself. To detect these deformations, the expert system must take several deformation measurements into account. Table 5.1 in Chapter 5 shows the typical deformation patterns that may appear during the lifecycle of a building.

In geodetic measurements, there are a couple of vulnerable points of the building which are measured. Figure 4.2 suggests which measurements are to be taken into account to provide a general assessment of the deformation. In order to provide the user with

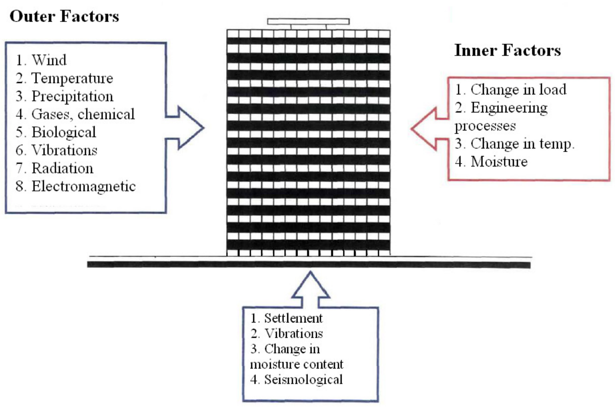


Figure 4.3: Number of factors that influence deformations to take place

the description of the deformation, five different measurements are to take place, two at the top of the building (on both sides), two at the bottom, and one in the middle of the building. The ROIs have been chosen just like in Figure 6.6, Chapter 6, as it is common practice for engineers to perform measurements on three different levels of the observed object.

The middle measurement is to investigate whether there is a settlement present in the center of the building. This may indicate either that the building itself is entirely under settlement, or that only the middle part of the building is sliding down. If however, the other four measurements show any signs of deformation, then the diagnosis will change with respect to the other measurements - i.e. if the left part of the building is sliding together with the central, but the right part of the building does not seem to show any movements whatsoever it means that the building is under settlement at the left side.

However, any building is a subject to a deformation regardless of the different factors which may influence it to lead it to a deformation (see Figure 4.3). These may include vibrations from the traffic, or change in weather which do not really lead to any serious deformations. Also, in some rural areas in the United Kingdom, the building may even have upwards movements due to the condition of the soil and the climate change. So, if there have been slight movements detected in the building, they may not necessarily mean that the deformation has reached critical state. To completely verify and classify a deformation, the expert system must combine the results of measurements from multiple epochs into one accurate diagnosis.

4.3 Consistency Checks

The system is fed by a series of ROI [3], which hold the information such as the (x, y) coordinates of the ROI, together with their 3D equivalents (two vectors describing the translation (tx, ty, tz) and rotation (α, β, γ):

```
(deftemplate ROI
(slot id (type NUMBER))
(slot WA_x (type NUMBER)) (slot WA_y (type NUMBER))
(slot tx) (slot ty) (slot tz)
(slot alpha) (slot beta) (slot gamma))
```

Before any interpretation can be done, the system needs to check whether the ROIs differ from one another. Some of the consistency checks are rather simple, for example, if one region of interest is moving in one direction, then the adjacent ROIs need to be moving in the same direction, and at the same magnitude. If this is not the case, the system will inform the user about that ROI, so that it can be re-calculated (or ignored).

The only type of translation allowed is the translation along the z-axis. This is because the object being observed cannot move to the left, right, or to and away from the camera. Thus, the main concern for the KBS is to observe the values of the tz parameter. If there is a translation along the z-axis, then it is possible that there is also a translation at the x or y axis as well (which would mean that the ROI is moving diagonally). However, a lonely translation at the x and y axis is not possible.

As for the rotations, a negative rotation cannot be adjacent next to a positive, and vice-versa. A combination of a couple of rotations is possible, but again, if such a rotation is present, then all the adjacent ROIs need to be rotating in the same direction, at a similar magnitude. In total, 40 consistency checks are performed before the interpretation.

4.3.1 Representing Deformations

The KBS deals with a simple, yet important deformation, which is present in building construction. A settlement can occur, if for example, one of the underground stations is being built, or if the soil underneath the building is moist. When this happens, one of the columns will have a negative tz parameter, and the adjacent columns will be rotating left, and right. The first thing the KBS needs to do is locate the position of the ROI which has a negative parameter tz and then check that all the ROIs which are located on the left have a positive rotation, whereas those which are on the right have a negative rotation. The rotations must be of the same origin, rotating along the same axis. If that is the case, then the system will notify the user that a settlement has occured. Needless to say, to detect a settlement the system needs to perform additional consistency checks, such as the compatibility of the rotations with respect to the negative translation along the z-axis. For example, if there is only a negative translation along the z-axis present (without the x and y), then the adjacent columns need to have a rotation along the

```
(defrule check-tz-magnitude
(adjacent (id1 ?id1) (id2 ?id2))
(or (and (or (ROI (id ?id1) (tz "sg")) (ROI (id ?id2) (tz "g")))
     (or (ROI (id ?id1) (tz "k")) (ROI (id ?id2) (tz "sk"))))
(and (or (ROI (id ?id1) (tz "nsg")) (ROI (id ?id2) (tz "ng")))
     (or (ROI (id ?id1) (tz "nk")) (ROI (id ?id2) (tz "nsk")))))
=>
(printout t "Magnitude of translation along the
 z-axis changed drastically from " ?id1 " to " ?id2 crlf)
)
```

Figure 4.4: Rule used for consistency checks

z-axis(γ parameter needs to be posive or negative). On the other hand, a z/x translation (diagonal movement) will have γ/α rotation present. Any other combination will result in a warning.

The system will also locate cracks in the wall, and notify the user. A crack occurs, if there is a column of ROIs with all parameters set to zero, but with deformations present at the adjacent columns. Similar to the settlement problem, the system has to make sure that the adjacent columns have the same type of deformation present on each side.

Another possible deformation is a diagonal settlement, that is, one half of the object (split diagonally) is moving to the left, or right. This is the only case, where translation along the x-axis is allowed. Such a settlement can happen if there is something wrong with the soil near the corner of the building. In such a case, the upper part of the building will move slightly left, or right.

Once the system has performed consistency checks, and located possible settlements, or cracks, it will do a final check on the parameter values. If there has been a considerate change in the magnitude of the translation/rotation parameters, the system will alert the user about a potential hazard which could happen in the near future. An example of such a rule is in 4.4:

4.4 Experiments

The experiments were carried out using a benchmark of 50 images from the Vienna public domain database [2]. Deformations have been generated artificially, due to the lack of real data by using a warping tool which takes as input a list of vectors and outputs a distorted version of the image to simulate a deformation. Examples can be seen in Section 3.4. The vectors, or movements have been chosen randomly from the large set of data gathered from the project 'Prokop', described in Chapter 6 to simulate potential settlements.

Once the ROIs have been calculated, together with their appropriate translation and rotation parameters, the KBS was tested with a series of images. 10 settlements have been simulated using an image processing warping tool from TU Graz, which transforms an image using a vector as an input. The images have been transformed over 10 epochs additionally, by using a vectors representing diagonal and or complete settlement (no translation along the x-axis). Some randomness has also been introduced over the 10 epochs, in order not to simulate a deformation that is completely linear (i.e. a small random percentage is added, or deducted from the actual vector value per epoch). A successful deformation interpretation would entail identifying the direction of the settlement, and the system was able to identify the 'randomized' movements. Additionally, a series of images depicting deformations that would never be possible in real life have been presented to the system, such as rapid movements (in particular negative translations along the x-axis, meaning that the building has risen up to 10m in the air) which rejected them with the consistency checks in place. The system has not been tested in real life, mainly due to the fact that it is hard to keep track of a building with potential deformations by using only digital cameras. Proper setup would include continuous snapshots of the observed building taken periodically over time, from an exact same location. It would have been really interesting to see whether the correct movements have been identified, or to find a threshold for rapid movements so that the consistency checks can filter any measurement errors.

4.4.1 Representing Deformations

Since the warping tool takes as input vectors of movement (in our simulated environment - deformation) to modify the image along the axes, assuming that all the outliers have been isolated, then all movements in ROIs are actually just the same vector movements used by the warping tool to deform the image it takes as the input. The CLIPS representation of a translation along the z-axis, can be found in Figure 4.5, and in this case represents a complete settlement (all ROIs are moving in one direction).

```
(assert (ROI (id 1) (WA_x 100.00) (WA_y 100.00) (tx O) (ty O) (tz ng) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 2) (WA_x 300.00) (WA_y 100.00) (tx O) (ty O) (tz ng) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 3) (WA_x 500.00) (WA_y 100.00) (tx O) (ty O) (tz ng) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 4) (WA_x 700.00) (WA_y 100.00) (tx O) (ty O) (tz ng) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 5) (WA_x 100.00) (WA_y 300.00) (tx O) (ty O) (tz ng) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 6) (WA_x 300.00) (WA_y 300.00) (tx O) (ty O) (tz ng) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 7) (WA_x 500.00) (WA_y 300.00) (tx O) (ty O) (tz ng) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 8) (WA_x 700.00) (WA_y 300.00) (tx O) (ty O) (tz ng) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 9) (WA_x 100.00) (WA_y 500.00) (tx O) (ty O) (tz ng) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 10) (WA_x 300.00) (WA_y 500.00) (tx O) (ty O) (tz ng) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 11) (WA_x 500.00) (WA_y 500.00) (tx O) (ty O) (tz ng) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 12) (WA_x 700.00) (WA_y 500.00) (tx O) (ty O) (tz ng) (alpha O) (beta O) (gamma O)))
```

Figure 4.5: Settlement Along the Z-Axis (Negative Translation)

Diagonal Cracking, or a positive translation along the X-axis can be found in Figure 4.6, where only the ROIs located in one part of the image.

```
(assert (ROI (id 1) (WA_x 100.00) (WA_y 100.00) (tx m) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 2) (WA_x 300.00) (WA_y 100.00) (tx m) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 3) (WA_x 500.00) (WA_y 100.00) (tx m) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 4) (WA_x 700.00) (WA_y 100.00) (tx m) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 5) (WA_x 100.00) (WA_y 300.00) (tx O) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 6) (WA_x 300.00) (WA_y 300.00) (tx O) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 7) (WA_x 500.00) (WA_y 300.00) (tx m) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 8) (WA_x 700.00) (WA_y 300.00) (tx m) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 9) (WA_x 100.00) (WA_y 500.00) (tx O) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 10) (WA_x 300.00) (WA_y 500.00) (tx O) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 11) (WA_x 500.00) (WA_y 500.00) (tx O) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 12) (WA_x 700.00) (WA_y 500.00) (tx m) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
```

Figure 4.6: Diagonal Cracking (Positive Translation along the X-axis)

The knowledge base should also discard any corrupt data which may be a result of any mismatch between epochs. These may include Figure 4.7, where two ROIs (with ids [7, 8]) are the only ones affected, or Figure 4.8 where only the upper part of the image is moving.

```
(assert (ROI (id 1) (WA_x 100.00) (WA_y 100.00) (tx m) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 2) (WA_x 300.00) (WA_y 100.00) (tx m) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 3) (WA_x 500.00) (WA_y 100.00) (tx m) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 4) (WA_x 700.00) (WA_y 100.00) (tx m) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 5) (WA_x 100.00) (WA_y 300.00) (tx O) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 6) (WA_x 300.00) (WA_y 300.00) (tx O) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 7) (WA_x 500.00) (WA_y 300.00) (tx m) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 8) (WA_x 700.00) (WA_y 300.00) (tx m) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 9) (WA_x 100.00) (WA_y 500.00) (tx O) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 10) (WA_x 300.00) (WA_y 500.00) (tx O) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 11) (WA_x 500.00) (WA_y 500.00) (tx O) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 12) (WA_x 700.00) (WA_y 500.00) (tx m) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
```

Figure 4.7: Example 1: Deformation incompatible with the rules in place

```
(assert (ROI (id 1) (WA_x 100.00) (WA_y 100.00) (tx m) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 2) (WA_x 300.00) (WA_y 100.00) (tx m) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 3) (WA_x 500.00) (WA_y 100.00) (tx m) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 4) (WA_x 700.00) (WA_y 100.00) (tx m) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 5) (WA_x 100.00) (WA_y 300.00) (tx O) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 6) (WA_x 300.00) (WA_y 300.00) (tx O) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 7) (WA_x 500.00) (WA_y 300.00) (tx O) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 8) (WA_x 700.00) (WA_y 300.00) (tx O) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 9) (WA_x 100.00) (WA_y 500.00) (tx O) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 10) (WA_x 300.00) (WA_y 500.00) (tx O) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 11) (WA_x 500.00) (WA_y 500.00) (tx O) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
(assert (ROI (id 12) (WA_x 700.00) (WA_y 500.00) (tx O) (ty O) (tz O) (alpha O) (beta O) (gamma O)))
```

Figure 4.8: Example 2: Deformation incompatible with the rules in place

Other examples would include roations along the axes, which would indicate bulging, or tilting. In each of the scenarios that were presented to the knowledge base, the pattern has been detected. Any sudden increments in the translations, or rotations would also trigger an alarm which would indicate any potentially hazardous behaviour. However, due to the nature of the experiments, to make any assumptions would require live data from construction sites (images of facades of residential buildings over time while any deformations are present).

4.5 Comparison to Other Related Work

Chmelina et al. [16] monitored deformations induced by tunneling works. The method entails the usage of Certainty Factors which represent quantitative uncertain reasoning. The factors are used to express the certainty of statements (about the existence of errors in monitoring data) and to further process them to an output certainty (possible error cause). The knowledge based system has been based on certainty logic which uses two operations, a Conjunction $U(x, y) = min(x, y)$ and disjunction $O(x, y)$:

$$O(x,y) = \begin{cases} x + y - x * y & x, y \geq 0 \\ x + y + x * y & x, y < 0 \\ \frac{x+y}{1-min(|x|,|y|)} & x * y < 0 \end{cases} \tag{4.1}$$

In the paper several rules have been described which are being deployed by the expert system to express the certainty of statements, as well as for the possible error cause. The data that has been used in this paper is from a tunneling project 'Brauherr ARGE' with data gathered over a two month period (from September to November) and the settlement that took place can be seen in Figure 4.9. The results could not be compared to the one found in 'Prokop' as the focus is more on the structure of the tunnel, and not residential buildings. Furthermore, the paper discusses on potential applications for the error analysis of displacement data in tunelling, whereas the focus of this thesis is on predicting the behaviour of deformations found in residential buildings.

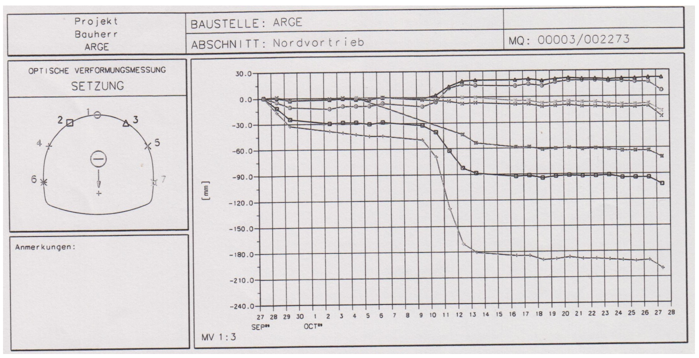


Figure 4.9: Results of the tunneling done in Project Brauherr ARGE [16]

CHAPTER 5

Deformation Interpretation

This chapter deals with the interpretation of deformation found in building construction. In order to find out why the deformation occurred, several other parameters are taken into account, in particular the material used in construction, or the soil underneath the building. Combining this knowledge together with the monitoring of the deformation which has been described in previous chapters, the system is finally able to provide the user with the appropriate description of what has happened with the structure over time.

5.1 Movement types

Every building undergoes a series of movements throughout its lifecycle. Most of the movements are negligable, and are of not much interest to engineers, others however require intervention. In such cases, conservation principles need to be applied, some which will be described in this chapter.

The movements which occur in buildings may be instantaneous, or may happen gradually over time. Normally, instantaneous movements occur in steel or iron beams when loaded [23], whereas gradual movements occur in buildings made out of timber, or masonry materials. Materials made out of timber may be a subject to shrinkage or swelling, whereas masonry materials may be a subject to drying or wetting. The temperature of the surrounding also plays a role, as it can induce thermal movements. Typically, when engineers are performing geodetic measurements, they will pick out a day in the month when the temperature is constant, so that thermal movements would not interfere with the measurements.

The most common types of movements, or deformations found in buildings are:

- Cracking
- Tilt
- Bowing and Bulging

5.1.1 Cracking

Most cracks that occur are not a sign of a serious structural distress, or deformation. They can occur naturally, e.g. cracks found in timber structures are a result of drying shrinkages, and the only remedy is to paint over them. Other cracks require plasterboard to fill in the cracks which is used in modern ceilings, made out of masonry meterials. In these cases, cracks are not a sign of structural deformations, but are a nuisance to the dwellers and residents who are even sometimes requesting a professional opinion to make sure that they are living in a safe environment. Cracking also typically occurs where the cross section is weaker, such as line of windows in a wall. In such cases, the height of the crack does not indicate a serious deformation, it is rather in the case of a crack with high width that an expert opinion is required. Formally, cracking occurs when the total imposed tensile applied strain on a material exceeds its tensile capacity. Every material is subject to strain, and once this strain exceeds the allowed threshold is when the cracks in walls appear. Variations in temperature, and the moisture content of the ground beneath the building also play a role, and typically they are the reason for innoccuous deformations (but still damaging to the aesthetics however). Buildings made out of steel, and reinforced concrete have much higher tensile strain capacities, so they can withstand larger movements without being subject to cracking.

The pattern in which cracking forms can be used to determine what kind of deformation is present in a building [23]. Table 5.1 shows typical patterns found in masonry buildings, and the causes for such deformations. It also classifies these patterns, as potentialy dangerous, or insignificant. A neural network from Haider et al [5] could be trained to detect cracking patterns in masonry buildings, and then an expert system could then classify these patterns as dangerous or negligable. The main challenge would be to gather images of buildings in which cracking has occurred. In addition, services from structural engineers would also be required, as their opinion is needed to identify and give exact information as to what kind of deformation is taking place.

5.1.2 Tilt

Tilt represents all movements which involve rotation of the whole building, or parts of it, such as walls, or columns. This type of movement is present in a lot of buildings, as it is impossible to build a truly vertical building. Best known example is the Leaning Tower of Pisa, and the deformation found in this historical building is present because of the ground strata underneath it, as the variations had caused the building to tilt. Serious tilting is prevented by on-site planning by the engineers who do some thorough examinations of the ground at which the building will be constructed. Tall and narrow buildings are also affected by tilting.

5.1.3 Bowing and Bulging

Typically when tilting occurs, the building as a whole is affected without any so-called out-of-plane [12] movements. The other rotation movement types (as observable by

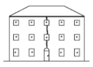
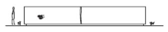
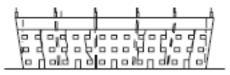
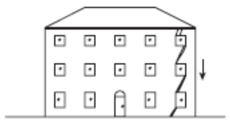
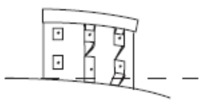
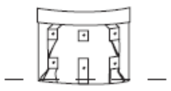
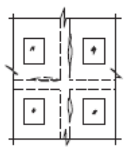
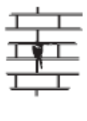
Typical elevation with cracking patterns		Probable cause(s)
	Cracking essentially vertical and of reasonably constant width	Temperature changes and/or during shrinkage; often not significant
	Expansion, then essentially vertical cracking	Irreversible expansion of bricks; followed by expansion in warmer weather and then cracking in cooler weather as wall tries to contract but is restrained from weight of wall resting on its thermally more stable base; often not significant. Can also occur in parapets.
	Progressive outward movement towards ends of terrace, with associated vertical/diagonal cracking	Progressive outward 'shunting' effect, which has been described as the 'bookend' effect; could be or become significant, e.g. requiring stabilization work to ends of terrace
	Diagonal cracking as right end of building settles or subsides (or as remainder heaven upwards)	Ground movements such as differential settlement, or subsidence, could be or become significant
	Cracking (vertical and diagonal) wider at top of building, can result in 'lozening' of openings	Ground movements such as subsidence, differential settlement – building on edge of sinking area; could be or become significant
	Cracking (vertical and diagonal) wider at base of building, can result in 'lozening' of openings	Ground movements such as differential settlement, subsidence – building within sinking area; could be or become significant
	Cracking and disturbance of masonry suggests corrosion of embedded steel frame (shown in broken lines)	Corrosion of embedded steel column, beam, etc; could be or become significant
	Cracking of mortar joints (and staining of surface) suggests corrosion of wall tie or other embedded iron or steel (also occurs in stonework and terra cota)	Corrosion of wall ties, cramps or other embedded metal; could reduce lateral restraint to structural members, and also increasing physical damage to masonry units, and staining over time

Table 5.1: Typical crack patterns in masonry walls [23]

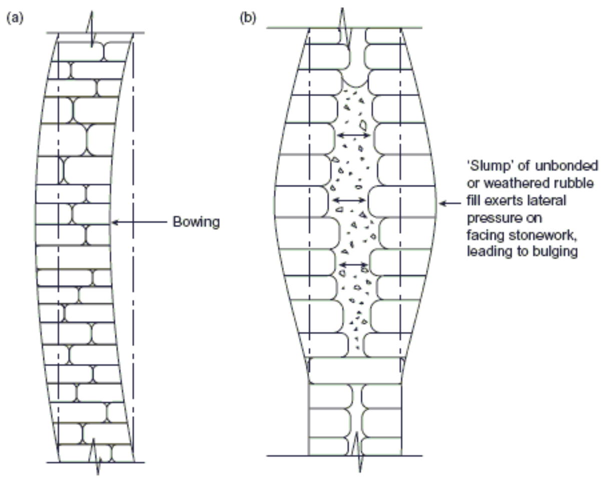

(a)
(b)
Bowing
'Slump' of unbonded or weathered rubble fill exerts lateral pressure on facing stonework, leading to bulging

Figure 5.1: Bowing and bulging of a wall: (a) bowing (b) bulging

theodolites, or other instruments) and that do not involve tilting are called bulging and bowing (see Figure 5.1). These movements are present in parts of the building, such as columns or walls, but not the entire structure, and can be potentially threatening.

Typical causes for bowing included material decay (for timber), sulphate attacks on masonry, or excessive loading on the foundation of the structure. Bulging occurs normally because of the absence of binding mortar, or cavities in walls, and corrosion failure.

5.2 Movement causes

The most common causes of movement in buildings are:

- change of temperature
- change in moisture content
- applied loading
- material decay or deterioration

5.2.1 Change of temperature

Thermal movements can be estimated using the following formula [12]:

$$e = \alpha \times L \times \delta t \tag{5.1}$$

where e is the linear extension (or contraction) of a component of dimension L owing to a rise (or fall) in temperature of δt degrees, and α is the coefficient of linear thermal expansion of the material from which the component is made. Changes in temperature can cause small movements, however they can be a problem for engineers when they are performing on site geodetic measurements, as sudden temperate changes can have an impact on the results of their measurements. Typically temperature changes are not of great concern, and can be classified in the same group as vibrations coming from the traffic. They are of concern however for historical buildings, which require much greater care than regular residential buildings.

5.2.2 Change in moisture content

Some building materials increase in volume when wetted, and return to their original size when dry again. Metals, glass, and certain minerals are dimensionally stable. Timber is the material which is greatly affected by change in moisture. When not properly treated, this can lead to huge structural damage. Masonry materials are not greatly affected by this, and in worst case scenarios can lead to formation of small wall cracks.

5.2.3 Applied loading

Every building consists of structural elements which carry loads that affect the gravitional force. Apart from the foundation of the building these may include joists, beams, girders (that are used to carry loads by bending), columns and walls (carry loads in a downward fashion), hangers and tie rods (tension). In addition, the load from a building is affected every day by its occupants, furniture, or water in tanks. All these movements are occuring every day throughout the life cycle of a building, and do not cause significant damage. A more significant problem in loading would be an addition of an extra wall, or even a cellar in which case monitoring is required.

5.2.4 Material decay

All materials suffer from decay, in particular timber which can have serious impact on the structure, often causing it to collapse. The only remedy for buildings made of timber which suffer from decay is to cut out the parts which have been affected and replace them with supporting bricks. Brick mortar, and masonry materials suffer from sulphate attacks which can cause increase in volume size when rainwater leaches into cement constituents. Sulphate attacks will not lead to the collapse of the building, but require the affected bricks to be replaced with new ones. Changes in sulphate content are monitored for historical building in great extent due to their importance. Concrete materials can also increase in volume size when alkali-aggregate reactions form an expansive gel that can lead to cracking. Alternating changes in temperature that can cause wetting or drying can also create more serious damage, in which case sometimes reconstruction of certain parts of the building is required. Structures made from iron and steel suffer from corrosion which is triggered when water gets in contact with unprotected metal. Normally it occurs in buildings on seaside, and in cities it can be present on roof structures when a lot of precipitation occurs. Corrosion can completely be avoided when stainless steel is used as building materials which is very common nowadays.

5.3 Enforcement methods

In the previous sections, most common deformation patterns and the reasons why they occur have been described. Implementation of a small knowledge base, written in CLIPS can be found in Appendix D, which prompts the users with questions regarding the structural behaviour of the building, including extra questions in case there is any tunneling present, and gives answers to the possible causes of the deformation, based on the information from the previous sections.

The deformation patterns discussed in the previous section can be diagnosed using the measurements that have been acquired by the system. However, in order to complete the diagnosis, the expert system needs to have additional information on the building.

There can be many reasons to a deformation, which may include outer and inner factors, as seen previously in Table 5.1.

A survey has been taken using 300 case reports from the Serbian company Energoprojekt. The typical defects included cracks, spallings, and damages to the layers of concrete caused either by settlements or yielding of the foundation. Table 5.2 shows the enforcement methods that have been taken by the company, together with the type of construction, and the defect that has been detected. The parts of the building which have been investigated include brick walls, columns, panels, monolithic plates and overlapping beams, ribbed slabs and hollow-core plates. In cases of potentially hazardous deformations, these parts have always been involved in remedial works. The main causes of deformations included load bearing, aggressive environment or changes in the soil surrounding the buildings, and in rare cases improper course of action during construction.

It should be noted that the enforcement methods described below only apply to a certain type of buildings, the ones which have been constructed using the same materials, and which have the same structure. A more sophisticated table would include the common causes of deformation in historical buildings, modern, as well as the ones which have been constructed using timber.

Construction Type	Defect		Enforcement Method
	Type	Cause	
Brick wall facing	Damage to the plastered layer	Retting of the plastered layer	Restoration of the facing by grid plastering
Brick wall	Cracks	Settlement	Grid plastering using concrete
Brick wall	Cracks	Settlement	Installation of double-sided metal straps
Brick pier	Cracks	Settlement, increase in load, mechanical damage	Applying ferroconcrete, grid plastering
Brick pier	Cracks	Settlement, increase in load, mechanical damage	Steel Caging
Wall panels	Cracks and spallings	Mechanical damage during construction	Applying concrete on the panels
Columns	Damage to the protective layer of concrete	Increase in load, mechanical damage	Applying metal straps
Monolithic overlapping beams	Damage to the protective layer of concrete	Aggressive environment	Installation of combined joining beams using reinforced steel
Overlapping beams	Cracks, damage to the protective layer of concrete	Aggressive environment, mechanical damage	Restoration of the protective layer of concrete and armature
Ribbed slabs	Damage to the protective layer of concrete	Aggressive environment, mechanical damage	Restoration of the protective layer of concrete and armature
Monolithic plate	Damage to the protective layer of concrete	Aggressive environment	Restoration of the protective layer of concrete and armature
Hollow-core plate	Damage to the construction	Improper course of action during construction	Installation of the pilot hole

Table 5.2: Enforcement methods for different construction types

CHAPTER 6

’Prokop’ Train Station

Train station “Center” is located in the vicinity of “Prokop” next to the northeastern part of Topcider hill in Belgrade, Serbia. Continuation of civil engineering works at the construction of train station “Center” has had negative impact on the stability of abutment diafragms. This has led to the necessity of geodetic measurements of abutment diafragms and residential buildings in the street “Stevan Filipovic”. The buildings have been monitored in horizontal and vertical axis, and in total around forty measurements have been performed. However, this number may vary as some points had been destroyed over period of time. In order to perform the geodetic measurements, the company needs to undergo a series of judicial processes to be able to properly monitor the deformations that are taking place. This has also been the case with the local company, Energoprojekt who had to undergo such procedures.

Figure 6.1: ’Prokop’ Station

6.1 Geodetic Control Network

Geodetic works are permanently present in all phases of space changes which include idea, realization and evaluation. These works are applied for measuring the area for groundwork, transportation of objects to the field, tracking of their construction, measurement of objects for the purpose of managing records, tracking of the behaviour of the object during exploitation, etc.

Designers in the field are securing a permanent stability of the object, however during the construction of complex buildings, some unplanned changes can occur, which can lead to catastrophic damage to the object and its surrounding. For the purpose of preventing negative consequences, it is necessary in certain time intervals to monitor civil engineering works using one of geodetic methods, with a network which can meet the demands of designer documentation and technical specification as per project requirements.

During the process of projecting geodetic control networks, for the purpose of retrieving numerical values and assessing previous accuracy of the network, it is necessary to determine the design of the network and to plan measurements in it. Temporary values of unknown coordinates or the height of points are determined mostly from exsisting maps or from known methods. When planning the measurements, different choices are taken into consideration together with their measurement accuracy methods. When the temporary values of coordinates and/or the height of the points are determined together with the defined plan of measured lengths, as well as their accurate measurement in the network, the matrix of design A and a covariation matrix of measured lengths $K_1 = \sigma_o Q_1$ is given. This way, a functional and stochastic model of indirect equalization of free or closed networks is provided.

The covariation matrix of unknown parameters for closed networks is represented as:

$$K_{\hat{X}} = \sigma_o^2 \cdot Q_{\hat{X}} = \sigma_o^2 \cdot N^{-1} = \sigma_o^2 \cdot (A^T Q_1^{-1} A)^{-1} \tag{6.1}$$

where $\hat{X}$ is a vector of differential increments, σ_o is the standard a priori lack-of-fit, or a priori accuracy given during the process of leveling, N is the matrix of normal equations, Q_1 is the cofactor matrix of measured lengths, and $Q_{\hat{X}}$ is the cofactor matrix of unknown parameters (in this case coordinates). And for free networks:

$$K_{\hat{X}} = \sigma_o^2 \cdot Q_{\hat{X}} = \sigma_o^2 \cdot N^{+} = \sigma_o^2 \cdot (A^T Q_1^{-1} A)^{+} \tag{6.2}$$

where N^+ is the notation for pseudoinversion. In the case of object 'Prokop', the previous value of accuracy was processed with minimal traces of covariation matrix on the stability points of geodetic control network, which have been placed outside the zone of expected deformations. Standard of horizontal and vertical directions, lengths and height differentials in the geodetic networks, as well as measurement plan have secured the targeted projected value of unknown parameters. Average values of standard coordinates and heights as well as elements of standard confidence ellipse (with probability $p = 0.95$) is shown in Table 6.1.

Figure A.1 depicts previous analysis of free geodetic control network on object 'Prokop' with planned measurement directions ($\sigma_\alpha = 1''$) and lengths ($\sigma_D = 1mm + 1ppm$), where after the analysis a homogeneous accuracy of points is preserved.

Accuracy assessment of the geodetic control network	[mm]	Accuracy assessment of observation points	[mm]
Coordinate standards for control network points	$\sigma_Y = 0.69$ $\sigma_X = 0.67$ $\sigma_H = 0.70$	Coordinate standards for observation points	$\sigma_Y = 1.23$ $\sigma_X = 1.30$ $\sigma_H = 0.95$
Absolute confidence ellipses for control network points	$A = 1.77$ $B = 1.55$	Absolute confidence ellipses for observation points	$A = 3.97$ $B = 1.97$

Table 6.1: Previous accuracy assessment of geodetic control network points and points for observation on object 'Prokop'

6.1.1 Hypothesis

Tracking of movements and deformation on every complex object, including object 'Prokop' because of its specificity, is a procedure which demands the realization of a series of very complex processes and rules, as well as unavoidable cooperation from various areas of science. The quality of procedures depends not only on the quality of geodetic measurements, data processing and equalization of data, but also from the universal approach in solving engineering demands. Universality entails the knowledge of: outer and inner influences on the deformation processes, compatibility of methodologies and instruments for analysing the geo-mechanical behaviour of the observed object, procedure of projecting and establishing a special purpose geodetic network, expected accuracy upon realizing control measurements, periodic control measurements, the type of model and methods used in deformation monitoring, as well as methods for analysis and interpretation of measurement results.

Based on the knowledge of the procedure, characteristics of the object and all previous geodetic works (from projecting and establishing a geodetic network to the realization of control measurements that took several years), it is necessary to show in a universal way the procedure of projecting the control network, the measurement plan, and acquisition and processing of data with conventional and modern instruments. Analysis of the results of unknown parameters from these measurements together with the basic and other control measurements will contribute to the findings as well assessment of all future geodetic auscultations of similar objects, and to the improve the approach to deformation monitoring in all civil engineering objects.

Recommendations, based on the analysis and investigations will be used in establishing geodetic network of specific uses, methodology of acquiring data and the choice of deformation model. The results of the previous assessment value, measurement zero or M_0 and realized control measurements will be used for the purpose of predicting movements using artificial neural networks.

6.1.2 Research of conventional methods

With the advances in science and technology different areas of science are showing the need for geodetic networks, which have horizontal and vertical coordinates that enable monitoring of movements and deformations.

Despite enormous literature about measurement techniques and methods, the real measurement analysis becomes feasible only until the emergence of modern computers which are capable of processing large quantity of data. Moreso, the development of the improvement of existing instruments have significantly increased measurement accuracy even creating new applications. Geodetic deformation measurements include all measurements that have the goal of determining elastic, plastic, and elasto-plastic changes in object or parts of Earth surface under the influence of inner or outer forces that result in movements.

The facts that points in the object are moving or the ground between two or more series of measurements, and that the deformation changes are incurred mostly because of uneven movements of points, the object that is being observed is monitored using a series of points, which is encompassed by the measurements performed in object 'Prokop'. Points are interconnected and forming a geodetic control network that is for the purposes of assessing shifts and deformations. The network consists of a basic referenced network of points outside the border of expected deformations and network points at the object [13]. If the movement is correct, the object has moved but has not deformed. The main problem during measurements of movements in a network or deformation analysis is to ensure which points remain stable between multiple series of measurements. That is, conventional deformation analysis is dealing with the problem of determining stable points in the network, or localization of unstable points in the network that is modelled using statistical tests, and testing of certain hypotheses.

6.1.3 Deformation model design

Design, measurements in deformation models have been a topic of many discussions [13] [37]. Based on many years of experience in deformation measurements, Eger (1997, 1993) has given important advices regarding terrestrial deformation measurements. The deformation model needs to be designed in such a way so that it can be used for a long period of time which can range up to 50 years. The network must be complete, undamaged and flexible in case it is required to be expanded in the case of an abnormal behaviour of the object, or in the case of additional construction works. Cooperation between the civil engineer at the field, and the geodetic engineer is crucial and should start as soon as possible.

Geodetic network needs to be designed in such a way to allow application of new technologies in later stages, and are usually designed for long-term. Reference points need to be close to the object, outside the zone of deformation influences, and need to be observed mutually, as well as be available during the whole year. Experiences have shown that, when there are more than two damaged geodetic points in the deformation model, then they are a consequence of settlements or additional works.

According to Eger's [36] observation, four reference points are an absolute minimum. Reference points are usually concrete columns consisting of double walls, with a forced centering system placed on top protected from natural disasters and outer influences that could cause the damaging or even destruction of the point. It is desirable that the concrete columns be supported by fixed umbrellas. At the points where the reference points are exposed to the increased risk from landslides, or avalanches the columns need to be placed in a concrete shelter to protect them. Four dislocated points that are nearby are useful in making sure that the movements in columns are with accordance to the nearby field observations. Unless GPS measurements are planned, it is necessary to ensure that the points are on such a place that the elevation angle is at least 15-20 degrees, and that there is no reflecting surface nearby.

Points at the object need to be distributed in such a way that they represent real information on how the object behaves over time, as well as its surrounding. Object points need to be available for conventional geodetic measurements. This may include length measurements with electro-optic devices, so that points are stabilized and that a suitable reflector or benchmark is placed.

Measurements with motorized total stations, with automated recognition of signals enables that the measurements of reference points and object points equipped with EDM reflectors are processed with computer aided control. If the instrument is permanently placed on objects that protect it from outer influences of weather and from movements, continued measurements are possible. If the communication link is established, remote control is possible. These systems are very expensive because they require sophisticated equipment and maintenance. In some cases however, the usage of such equipment is completely justified.

During one epoch of measurements in the network, it is necessary to stick according to the previously agreed observation plan. It is of great significance for the accuracy of measurements. The procedure itself needs to be designed in such a way that obvious errors could be spotted easily and dealt with quickly. This leads to a demand where some data is processed as early as possible so that errors can be removed efficiently. It is common to use the same gear at the same orientation, reference points, and object points in order to eliminate system errors. Daily backup of all data is a must.

Some aspects are always important, regardless on the type of equipment:

- Length of the instrument, target, reflectors
- Atmospheric parameters (temperature, pressure, humidity)
- Inconsistency of volume errors, eccentricity of instruments
- Avoidance or elimination of other system errors that may contain measurements

6.2 Special purpose geodetic networks

Basic national geodetic networks had been designed using the method of triangulation in 19th century and improved during 20th they were the foundation for geodesy and topographic measurements, but in most cases today they do not meet the demands in terms of accuracy, in particular marking and monitoring the stability of complex objects, as well as monitoring movements and deformations on the surface of Earth.

Because of these reasons, new independent geodetic networks have been developed: special purpose geodetic networks, local geodetic networks, control networks, independent geodetic networks and geodetic networks for objects.

A special purpose geodetic network consists of geodetic points outside the object (basic network) and points on the object (geometric control and deformation analysis) that are connected with terrestrial measurements (direction, angle, azimuth, spatial lengths, height differentials, etc.), satellite, astronomical measurements or a combination of both.

There are three types of networks, based on the dimensions of the coordinate system in which the position of points is defined:

- Vertical networks - one-dimensional model (1D)
- Horizontal networks - two-dimensional model (2D)
- Spatial networks - three-dimensional model (3D)

According to the type of measured lengths that are used for positioning of the points inside the network, the following methods are used for projecting geodetic networks:

- Terrestrial (triangulation, trilateration, triangle trilateration, precise polygonometry, precise elevation)
- Satellite (GPS, GLONASS, GALILEO)
- Combination of the aforementioned methods.

6.2.1 Special purpose geodetic control network models

The main purposes of a special purpose geodetic network are as follows:

- Define mathematical background for spatial location of the object.
- Enable markings of characteristic points, lines and surfaces of civil engineering objects
- Enable geometry control during construction
- Enable monitoring of the object (network is expanded with points outside the range of expected deformations and with points on the object whose movements is characterized by the soil underneath the building)

The main characteristic of these networks is:

Timeliness - network is projected in the phase of construction of preliminary design based on the schedule of of projected objects, durability - projecting the network needs to encompass the whole construction field and serve until the very end of construction.

Length adaptability - according to the object that is built or the part of surface of the Earth whose movements are being monitored (network in most objects is developed according levels, and at tall objects by floors)

Accuracy - standard lack-of-fit of the position of points of basic networks needs to be negligible with respect to the targeted accuracy, or other geometric demands of the object, by the principle of negligibility $1/5 < \sigma < 1/3$ where σ is the standard lack-of-fit accuracy.

Shape adaptability - according to the object that is being constructed by: possibility of stabilization of points, type of planned measurements, standard projections, observation plan and accuracy of measurements, and with regard to the characteristics of the object, configuration of the field and targeted accuracy.

Independence - characteristic points, lines, and surfaces of the object and a group of points in geodetic network positioned on a small area where high accuracy is wanted are required to be in the same local coordination system.

Adaptability - included in the basic network (horizontal and vertical), mainly with objects that are on higher ground where it is necessary to connect a series of smaller objects (hydroenergy systems, water flow regulation, etc.)

Homogeneity - all points of the same order are in the process of leveling simultaneously.

The choice of the model of the network depends on the demands of the object that is being built, satisfying certain constraints such as for marking benchmarks, determining with high accuracy the movements and deformations of built objects and natural phenomena. Based on these criterion, applications of geodesy include:

- civil engineering (industrial buildings, pillars, sylos, stadiums, tall buildings)
- traffic (railways, highways, bridges, tunnels, airports, and underground stations)
- energy (dams, pipelines, transmission lines, mines)

Other applications even include natural objects, that is natural forms of the surface of the Earth such as volcanoes, landslides, or faulting.

Special purpose geodetic networks consists of a series of points of same order that are mutually connected in figures, at which their shape, or model is adapting to the object and configuration of the field at which it was placed. The type of network and measurement also influences the model, which can lead to the choice between horizontal, vertical, or GPS special purpose geodetic networks.

Geodetic control measurement needs to be adapted to the object itself, for the purpose of efficient monitoring.

Angles and lengths of slopes in special purpose geodetic networks depend on the size of the object and the space which is necessary for the construction, configuration of the field, and the methods of measurement. At trilateration and triangulation networks, special attention needs to be given to the value of closed angles which need to be approximately equal.

GPS geodetic network of points is not dependent on shape and geometry, so the positioning instability is minimal. The points are positioned where they are needed, that

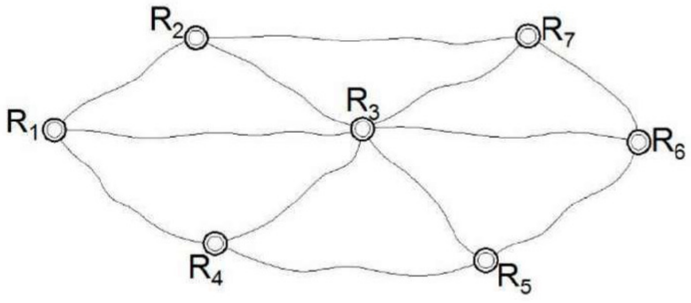


Figure 6.2: Vertical geodetic network model

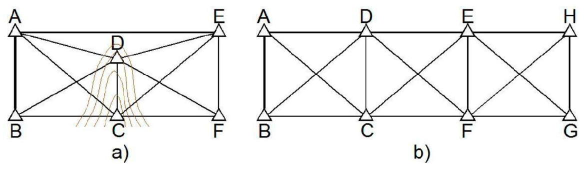


Figure 6.3: a) twofold geodetic tetragon b) chain of geodetic tetragons

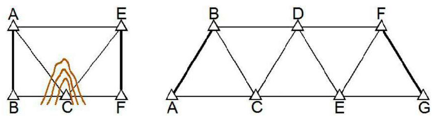


Figure 6.4: Chain of triangles

is close to the object. The shape of the network is determined mainly by the desired accuracy, number of available receivers, and the shape of the GPS network determines the measurement plan.

For the purposes of determining movements on the object 'Prokop', the deformation analysis 'Modified Karlsruhe Method' is used [31] [50].

Method consists of independent leveling of previous and current epoch and their mutual leveling. Deformation analysis is conducted using statistical testing of linear hypothesis. All epochs of geodetic measurements are leveled with the method of indirect leveling based on the method of Least Squares.

Linear functional model to be used is:

$$V = A \cdot \hat{X} + f \tag{6.3}$$

where $\hat{x}$ is the vector of differential increments, and f is the vector of free members.

Whereas the stochastic model is:

$$K_1 = \sigma_0^2 \cdot Q_1 \tag{6.4}$$

Mutual leveling is performed under the assumption:

- basic points are matched, or stable in both epochs.
- same ratio is present in both epochs.

Before the second phase, that is before mutual leveling it is necessary to determine whether the mentioned demands are met. In order to check the stability, geological and geophysical research is conducted. The ratio is determined by transforming the coordination points in the current to the next epoch. Because stable points are defined in this process, the whole group of stable points is adapted. Further research requires verification of their stability and to exclude from the group the points that have been confirmed unstable so that consistency is preserved.

Homogeneous accuracy of observation in both epochs is determined just like in the Pelzer method [53] - using Fisher test. If it is concluded that both observations are not of homogeneous accuracy, then after mutual leveling special attention is given to the weight at certain observations. Numerical procedures in the Karlsruhe method are applied in multiple phases that enable identification of stable points and according to them movements of unstable points.

The 'Modified Karlsruhe' method consists of leveling of M_0 using the least squares method by minimizing the part of the trace which corresponds to the assumed stable points. Equalization of control measurement series with the least squares method by minimizing the parts of traces corresponding to stable points of the deformation model entails that coordinates in M_0 are being used as approximate coordinates of the control series.

Control of stability points is performed using a Helmert transformation. When unstable points are discovered, it is necessary to perform new equalization using the same method described above. Afterwards, relative error ellipses are calculated using coordinates from the M_0 and control series of measurement.

Increments of approximate (leveled) coordinates from the process of equalization of spatial movements are calculated as:

$$\begin{aligned} dy &= Y_k - Y_0, \\ dx &= X_k - X_0 \end{aligned} \tag{6.5}$$

Standard components of spatial movements are approximately equal to the square root of the sum of squares of the coordinates in the M_0 and control series of measurement.

The hypothesis of statistical testing can be geometrically interpreted as:

$$\hat{d}_i \cdot Q_{\hat{d}_i}^{-1} \cdot \hat{d}_i \leq 2\hat{\sigma}^2 F_{1-\alpha,2,f} \tag{6.6}$$

where F represents the value of Fischer distribution using probability $1-\alpha$ at 2 and f degrees of freedom.

If 6.6, by substituting $\leq$ with $=$ then the expression represents the equation for the ellipse which is identical to the relative ellipse error between points T_{1i} and T_{2i}, increased by a factor of $\sqrt{2 \cdot F_{1-\alpha,2,f}}$. Semi-major axis is determined by:

$$A_j = \hat{\sigma_o}^2 \cdot 2 \cdot F_{1-\alpha,2,f} \cdot \lambda_j, j = 1,2 \tag{6.7}$$

where λ_j are the values given by the matrix $Q_{\hat{d}_2}$.

$$\begin{aligned} \lambda_1 &= \frac{1}{2}(q_{xxd} + qyyd + k), \\ \lambda_2 &= \frac{1}{2}(q_{xxd} + qyyd - k), \\ k &= \sqrt{(q_{xxd} - q_{yyd})^2 + 4q_{yxd}^2} \end{aligned} \tag{6.8}$$

The direction θ of semi-major axis A according to the positive coordinate axis is:

$$tg2\theta = \frac{2q_{xyd}}{q_{xxd} - q_{yyd}} \tag{6.9}$$

Relative semi-axis ellipses are multiplied by a factor of $\sqrt{2 \cdot F_{1-\alpha,2,f}}$, so that the surface of the ellipse error corresponds to the confidence area with a probability of $1 - \alpha$.

Deformation analysis using relative error ellipses can be easily represented graphically. The direction vectors of points are drawn on the same sketch together with the enlarged relative error ellipse. The vector of movement d_i goes from point T_{1i} to T_{2i}, and in point T_{2i} locates the center of the ellipse. If the point T_{1i} is outside the surface of the ellipse then the zero hypothesis is excluded, that is with certain probability deformations in point T_{1i} are being defined.

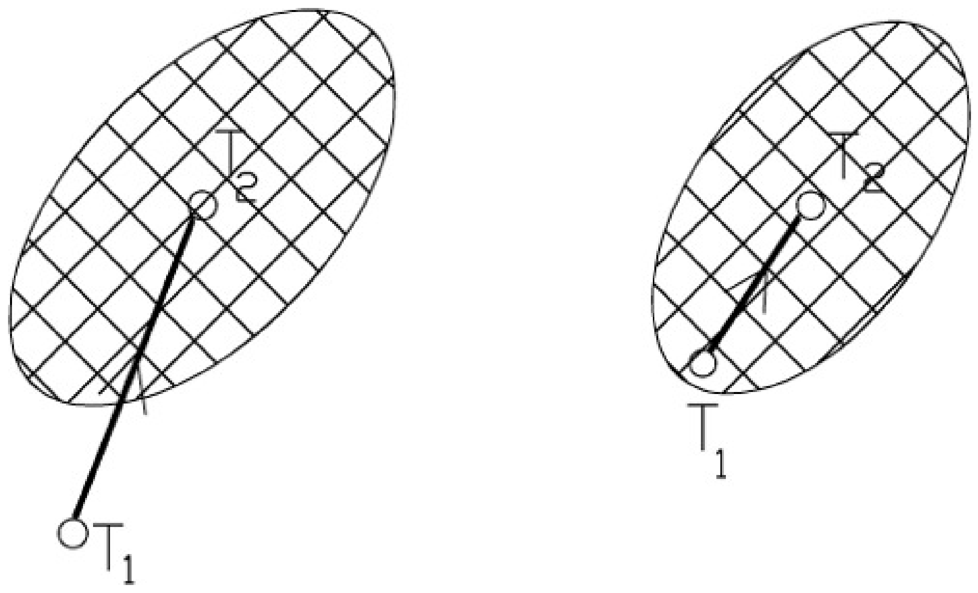


Figure 6.5: Deformation analysis using relative error ellipses; a) unstable point b) stable point

6.3 Measurements

When the measurements took place, the temperature was between $+4\,°C$ and $+16\,°C$, which is considered good as the thermal movements are minimal at these temperatures.

The following objects have been taken into consideration during field investigations:

1. Abutment diaphragms A-B, C-D, and E-F have all been observed in vertical and horizontal axis in order to monitor the deformations that are taking place. These objects are considered the best source of information in deformation monitoring in the case of underground construction works.

2. Residential building in street “Stevan Filipovic 28” has been observed in horizontal and vertical axis, but not using the same points. Horizontal deformations have been observed on three horizontal levels, whereas vertical on two levels, see Figure 6.6.

3. The other residential buildings in the same street listed above are only observed on vertical axis.

4. Overpass in the street “Oktobarska revolucija” - deformations are only monitored on vertical axis, same like in the case of residential buildings.

The data to be used for forecasting deformation movements will be focused on the residential building in “Stevan Filipovic 28”, because it has undergone extensive geodetic measurements. The main reason is that the tunnel which is connecting the two underground stations is located exactly underneath this building.

Over the course of time, point 13 had been damaged, but was still a valuable source of information as it was used as the centre point. Point 2a and 5a were not observed in horizontal axis because they were obstructed by the construction materials and huts.

Horizontal measurements have been performed using the elevation network which has been designed by the engineers and experts in geodesy. Angled measurements have been conducted by observing the 4 gyres in the elevation network and 3 gyres for the points on objects using the instrument WILD T20005 with forced centralization of the instrument and traverse tables for visualization.

Length measurements have been performed with distomat DI-20 and GPS where the statistical method of differential positioning has been used. Two GPS receivers of type Dimension from company Ashtec have been used, which are single-frequency with 12 channel and enable the user to receive signals from 12 NAVSTAR GPS satellites from 28 satellites launched at the height of 20200 km. 24 are active and are circling in 6 orbital planes with the inclination of $55\,°$, and time of rotation of 12h. The remaining 4 satellites are in reserve.

The best time intervals for measurements have been determined with the module ’Multi-site Mission planning’ [68] from Prism software package of the company Ashtec, that provides full support for all activities related to GPS measurements. This module based on the geographic coordinates of the network and the date of the observation provides the user with the number of available satellites in form of a diagram. The time

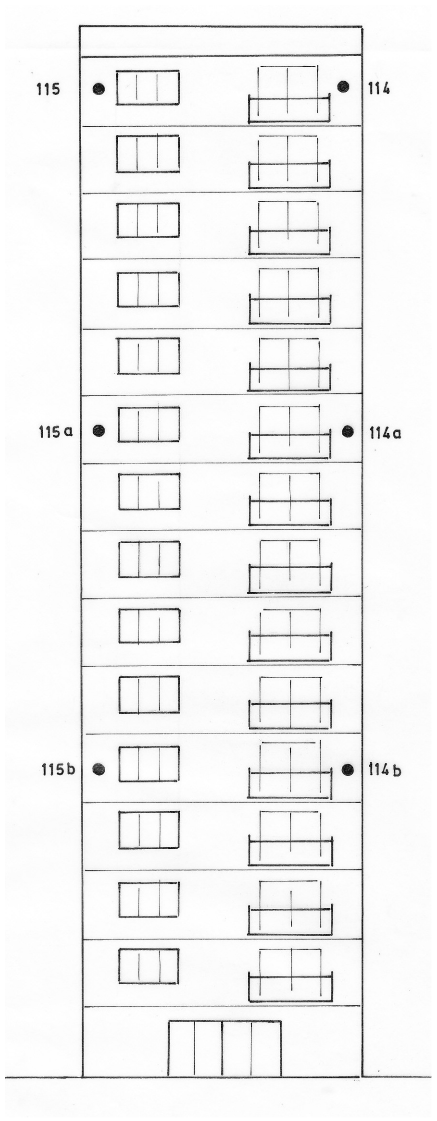


Figure 6.6: Observed points in residential buildings

of observation of one vector was 1.5h, at which point 20 seconds were registered from all satellites. When measuring the lengths meteorological parameters were used, along with temperature and pressure.

For the purpose of determining total vertical movements, an elevation network was used with the method of precise leveling. Measurements were conducted with level WILD NA-2 with an optical plate and WILD invar rods. At the field site, control measurements were conducted in order to minimize errors and provide accurate measurements targeted by the project of geodetic measurements.

For controlling angled measurements, triangle closure in the elevation network has been tested. Permitted, or targeted lack-of-fit was $\pm 5in$, directions between gyres $4in$, and between minus and plus sight $\pm 5in$. For level measurement control, polygon closure under criteria for precise leveling was used. Permitted lack-of-fit for one level of the network is

$$\delta s = \pm 0.4\sqrt{2n}\, mm \tag{6.10}$$

where n is the number of stations in one direction. Permitted lack-of-fit for polygon closure is:

$$\delta p = \pm 0.2\sqrt{2np}\, mm \tag{6.11}$$

where np is the nubmer of polygon stations in one direction. All field measurements that entered the numerical processing have satisfied the conditions above in terms of accuracy.

6.4 Data processing

After field investigation control measurements, in vertical and horizontal plane, calculation of definite directions and lengths from all observed gyres has been performed. When processing conventional length measurements, appropriate adjustments based on atmospheric influence, and altitude have been applied (the project was done at the altitude of $97m$). For GPS measurements, simultaneous registered data in receivers at the final points of the measured vector were grouped in three files. The data was transferred and processed using the software package PROCESS for each individual vector. For measured vectors the same adjustments like in length measurements need to be applied.

As a result of GPS measurements, based on the least squares method, length of the vector was calculated, including average squared error, spatial components of the vector together with the least squared errors.

When the errors were not in the allowed threshold, the vectors were re-processed interactively by the system. In this variant the software enables the user to omit the results from certain satellites and time intervals that are not aligned with the others, as well as the possibility of changing the referred satellite. The built-in software comes with the modules necessary modules to deal with the result variance.

When the results have been processed, they are then used by the software to perform the leveling of the elevation network, which is done by the method of group leveling with the condition that the least squared errors are minimal.

The necessary steps needed to perform the leveling are: error equations, inversion and control of inverted matrices for accuracy measurements, calculation of definite coordinates, calculation of mean error unit as well as the mean errors of units which are respectively:

$$\begin{aligned} m_o &= \pm 0.72, \\ my_max &= \pm 1.00\, mm, \\ mx_max &= \pm 1.20\, mm \end{aligned} \tag{6.12}$$

Point XX has been taken from polar coordinates because barracks of construction workers had been built nearby.

With the processed coordinates, the stability of the elevation network has been determined using the Helmert transformation. For stable points, elevation network has been leveled together with the points for observation on abutment diaphragms and residential buildings. Leveling has been performed based on the groups the points belong to. The exact information that is being fed to the software is:

- id and name of stability points
- id and name of points that determine movements
- measured values, directions, and lengths
- spatial coordinates of the points in elevation network together with the approximate coordinates of the observed points

Mean error unit that characterizes accuracy of the performed measures is:

$$mo = \pm 1.0\, sec \tag{6.13}$$

Mean error for determining coordinates for most observed points have the value around $\pm 1mm$. Maximum value by T axis is $\pm 2.3mm$, and by R $\pm 2.8mm$, excluding points 1 and 1a whose mean error can go up to 3.6 mm. The description of axes T and R can be found in the below paragraph.

Because of the works being done at the construction site, deposition of materials etc. it is not always possible at any given time to get an accurate reading of any point. This is especially reflected at the diaphragm under the residential building for observed points that in their index have an 'a'. This has lead the engineers to create additional points for observation (1E to 4E), in order to provide more accuracy.

Elevation network with the observed points has been leveled in the local coordination system which uses radial axis $R(x)$ that matches the axis of the metro line (973 and 989), with positive mark towards the metro platform, and tangent axis $T(Y)$ perpendicular to R, with a positive mark towards the tunnel 'Dedinje'.

The leveling of the elevation network and the calculations of height differences has been done using a benchmark (GN 236) placed on the portal of the 'Dedinje' tunnel. Processing of this data using the software has been done using:

- id and name of stability benchmark with value
- id and name of the stability points that determine movements with their approximate values
- measured height differences
- weight $p = 1/n$ where n is the number of stations in one direction

In this case, the elevation networks has also been leveled by grouping all the points and using the least squares method. Mean error unit which characterized the accuracy of performed measures in this case is $mo = \pm 0.1mm$ per point. With the averaging method (which is in fact Helmert transformation by vertical axis) stable benchmarks are being identified. The movements for these benchmark points have been marked as $0mm$, as they were identified as stable points. The mean error for all vertical movements for all observed points has value of $0.3mm$.

Movements in horizontal and vertical plane for abutment diaphragms and residential buildings have been determined based on measurement zero, or the first recorded measurement M_0. When analysing total movements of observed points for these objects including the overpass with pillars, movements in horizontal planes R and T and vertical plane are being taken into consideration. The accuracy of the movements played an important role in the final report of the measurements, as the ones that are deviating from the allowed threshold had been marked as errors.

6.5 Horizontal plane measurement results

(a) Abutment diaphragms C-D Total radial movements at the final elevation (114.5m) are relatively of small magnitude direction towards platforms, maximum was $+6.5mm$ (point 14). Benchmark 1 has movement of +7.4, but slightly higher determination error ($\pm 3.6mm$). Total radial movements for benchmarks of diaphragms with index 'a' with elevations of 106.60 to 110.7m, as well as observed points at the final elevation of diaphragms have the same movement direction and similar values, and maximum is at point 12a at $+7.2mm$. Points from 1a to 6a have average determination errors of $+3.4mm$, and that their movements are around $+7.2mm$. Tangent movements for all observed points are mostly in the range of the double average determination error and are around $+2.0mm$. A few points are outside the range (1a and 18) but their mean error is slightly higher. At this diaphragm point 17 had been destroyed, and point 13 damaged. Increment of horizontal radial movements with respect to the previous measurements were relatively low. For points on elevation $114.5m$ the highest was point 14 with value ($3.3m$). For points with index 'a' the highest was point 9a with value $+4.6mm$. Increment of tangent movements (towards the platform) are of low value and were in the range of $-2.5mm$ (point 1) to $+2.3mm$.

(b) Abutment diaphragm E-F Main characteristic of horizontal movements for this diaphragm was that the movements were rather equal and in the range of $1.9mm$ (point 24) to 6.8 (point 22). Tangent movements for all observed points are of low and equal values of negative direction of up to $-1.4mm$ (point 26). Two points 22 and 29 have slightly higher movement of up to $-3.0mm$. Point 21 was destroyed in the process. Increment of horizontal radial movements with respect to previous movements have the direction towards the rails, and maximum was at $+3.0mm$ (point 25), excluding the final points of the diaphragm 19 and 29 where the increment was sligthly higher, up to $+5.9mm$. Increment of tangent movements in the same period were of low values (up to $-1.3mm$ excluding point 29 (-3.3mm).

(c) Abutment diaphragms A-B Movements at this diaphragm had been approximated using point 32 whose movements were in the range of accuracy of their determination and two points 31 and 32a that had movements on both axes $-3.0mm$ and $+2.8mm$.

(d) Residential building 28 As mentioned before, this object was observed at three different levels (fourth, ninth, and fourteenth floor). The main characteristic of radial movements was that all observed points had a positive direction, that is movements towards the abutment diaphragms of varying intensity:

- for observed points on 14th floor, movements are up to $5.2mm$ (point 105)
- for observed points on 9th floor, movements are up to $4.5mm$ (point 112a)
- for observed points on 4th floor movements are up to $4.4mm$ (point 111b)

Movements of all other observed points are of equal value and are of decreasing value from 14th to 4th floor. Tangent movements are also of equal values and of negative direction, opposite of 'Dedinje' tunnel, and maximum was at $-6.2mm$ (point 107). The value of tangent movements is decreasing from 14th to 4th floor. Increment of horizontal radial movements with respect to previous measurements were in the interval of $-2.8mm$ (point 113) to $+3.2mm$ (point 108). Most increments are in the range of determination accuracy or double mean error. Increment of horizontal tangent movements with respect to previous measurements are in the direction of the Dedinje tunnel, and are equally distributed for all observed points in the range of $-4.0mm$. A few points have slightly worse mean determination error and are outside the range of these values (106-109) and their increments are up to $-8.4mm$.

6.6 Vertical plane measurement results

(a) Abutment diaphragm C-D This diaphragm is observed on two levels and the final elevation at $114.0m$, and the second group of benchmarks is installed depending on the site conditions at elevations of 110.7 and 106.6m. The observed points at this diaphragm indicate settlement, and maximum value has point 15 of $-5.0mm$. For points in the second level that have an index 'a' highest movement has point 15a in the range of $-4.7mm$. It is evident that the movements are increasing from point 1 to point 15 where they reach peak. The same ascertainment can be made for the benchmarks at the diaphragm, points 1a-16a, that are positioned at elevation 106.6 to 110.7m, except that the movements are relatively low. Increment of vertical movements with respect to previous measurements are in the direction of the settlement and the maximum is present at point 15a and in the range of $-2.6mm$. The beginning of diaphragm from street 'Oktobarska revolucija' has very low heaving of up to $+1.2mm$.

(b) Abutment diaphragm E-F For all observed points the measurements are in the range of double determination error and in the direction of heaving, and maximum is at $1.1mm$ (point 27). Increment of vertical observation comparing with previous values are in the range of allowed threshold.

(c) Abutment diaphragm A-B For observed points the measurements are in the border of the accuracy threshold. Increments of vertical movements with respect to previous measurements have a trend of low heaving $+1.1mm$ (point 32a).

(d) Residential objects in the street 'Stevan Filipovic' In this street, buildings 28-39 are being observed. For all of these buildings the movements for observed points are in the range of $-0.8mm$ (point 28) to $+1.2mm$ (points 29 and 33). Increments of vertical movements when comparing them to previous have the direction of heaving which is in the range of $1.0mm$.

(e) Overpass with pillars in 'Bulevar mira' All observations are in the range of $-0.3mm$ (point 41) to $+1.5mm$ (point 35). Increments of vertical movements have very slight heaving of up to ($1.0mm$).

6.7 Conclusion

Several points have been destroyed (17, 21, 13, 30). Points 3a and 4a have not been observed in horizontal plane because of stocking of building materials in the viccinity. Points 1a, 2a, 6a, and 7a had not been measured in vertical plane because of the same reason. Point 39 (located at the pillar of overpass) had been mechanically damaged. New points have been added to the elevation network, because a fence and barracks had been built and hence obstructed the sight of points (direction XVI-VI). New points (1E-4E) of temporary character (used in some measurements) have been added so that the cohesion of the network would remain intact. In these measurements, columns I and VI had been left out but this has not had an impact on the permanence of the coordinated system. In the elevation network there were no changes compared to the projected state. The main characteristic of total horizontal movements for observed points of diaphragm was that the movements on radial component were slightly higher than of the previous and that their direction stayed the same towards railway station 'Centar'. Tangent movement for observed points have usually been in the range of allowed threshold, or smaller than the double mean error. For building in street 'Stevan Filipovic' radial movements are in the value of $+5.2mm$ towards the railway station 'Centar'. Tangent movements are of equal values, and maximum was at $-6.2mm$ opposite of 'Dedinje' tunnel. Vertical movements in the direction of the settlement shows that the diaphragm C-D is of maximum peak at -5.0 at the final elevation. For all other observed objects, movements are at the border of determination accuracy. When interpreting the results, one has to take into account the difficulty of observing certain points due to works at the construction site, or even degree of realization of the project at that point in time.

CHAPTER 7

Time Series Prediction

7.1 The Task

Let X_t denote time series, or a sequence of data $x_1, x_2, ..., x_N$ which are successive points measured in time intervals. The objective is to forecast future values of x_{N+h}. The integer h is called the horizon, or lead time. *A forecasting method* is a procedure for computing forecasts from present and past values. As such it may simply be an algorithmic rule and need not depend on an underlying probability model. Alternatively it may arise from identifying a particular model for the given data and finding optimal forecasts conditional on that model. Forecasting methods can be classified into three types: [15]

- **Judgemental forecasts** based on subjective judgement, intuition, and any other relevant information. Most famous is the *Delphi technique* [41] which aims to find a consensus of opinion for a group of experts, based on a series of questionnaires.
- **Univariate methods** where forecasts depend only on present and past values of the single series being forecasted, possibly augmented by a function of time such as a linear trend.
- **Multivariate methods** where forecasts of a given variable depend, at least partly, on values of one or more additional time series variables, called *predictor* or *explanatory variables*. Multivariate forecasts may depend on a multivariate model involving more than one equation if the variables are jointly dependent.

A forecasting method could also combine more than one of the above approaches. For example, when univariate or multivariate forecasts are adjusted subjectively to account for external information which is difficult to express formally in a mathematical model.

7.2 Existing models for time series prediction

There are three types of linear models:

- Box-Jenkins [61], also known as ARIMA (Autoregressive Integrated Moving Average) and its variations, such as moving average (MA), autoregression (AR), and autoregressive moving average (ARMA) describe future data as linear combination of historical data and some random processes.
- State-space models [58] represent inputs as a linear combination of a set of space vectors that evolve over time according to some linear equations. The phrase “state space” derives from a class of models developed by control engineers for systems that vary through time.
- Exponential smoothening [9] models smoothed data S(t) as a function of raw data R(t) by $S(t+1) = AR(t) + (1-A)S(t)$

7.2.1 Autoregression

A time series X_t is said to be an autoregressive process of order p, if it is a weighted linear sum of the past p values plus a random shock such that:

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + ... + \phi_p X_{t-p} + Z_t \tag{7.1}$$

where Z_t is a purely random process, and $\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - ... - \phi_p B^p$ is a polynomial in B of order p.

7.2.2 Moving Average

A time series X_t is said to be an moving average process of order q if it is a weighted linear sum of the last q random shocks such that:

$$X_t = Z_t + \theta_1 Z_{t-1} + ... + \theta_q Z_{t-q} \tag{7.2}$$

where Z_t is a purely random process, and $\theta(B) = 1 + \theta_1 B + ... + \theta_q B^q$ is a polynomial of order q.

7.2.3 Nonlinear Models

The time series literature has traditionally concentrated on linear methods and models, because of mathematical and practical convenience. Despite their simplicity, linear methods often work well and may well provide an adequate approximation for the task at hand, even when attention is restricted to univariate methods.

There are, however many time series which exhibit features that cannot be explained by a linear model. For example, the famous time series showing average monthly sunspot numbers exhibits cyclic behaviour with a period of approximately 11 years, but in such a way that the series increases at a faster rate than it decreases. Another example are the economic time series, which tend to behave differently when the economy is moving into recession rather than when coming out of recession. Many financial time series however, show periods of stability, followed by unstable periods with high volatility. Sometimes financial data series will look stationary in the mean, but non-stationary in variance. Behavior like this cannot be explained with a linear model, and so non-linear models are usually needed to describe data where variance changes over time.

Nonlinear Models can be classified into models with predefined nonlinearity assumptions and general models. Pre-defined nonlinearity methods are not effective for modelling time series with unknown nonlinear behaviour. Machine learning [10] can handle nonlinear time series because it learns a model without nonlinearity assumptions. Specific methods include static learning (such as k-nearest neighbours), reinforcement learning (Q-learning), and supervised learning (decision trees, ANNs).

7.3 Artificial Neural Networks

A neural network (NN) can be thought of as a system connecting a set of inputs to a set of outputs in a nonlinear way. In a time series context the output could be the value of a time series to be forecasted and the inputs could be lagged values of the series. The connections between inputs and outputs are typically made via one or more hidden layers of neurons or nodes. The structure of an NN is called *architecture* usually. Choosing the architecture includes determining the number of layers, the number of neurons in each layer, and how the inputs, hidden layers and output(s) are connected.

The strength of each connection between neurons is measured by a parameter called weight. There may be a large number of parameters to estimate. A numerical value is calculated for each neuron at each time period t, as follows. Let $y_{i,t}$ denote the value of the ith input at time t. For each neuron, a weighted linear sum of inputs $v_i(t) = \sum_j w_{ij} y_{ij}(t)$ is calculated. Next, an activation function is selected, for transforming the values of the weighted linear sum of inputs into a final value for the neuron. A commonly used function is the logistic function, $z = \frac{1}{1+e^(-v)}$ which gives values in range $0, 1$. The logistic function should not be used at the output stage in time series forecasting unless the data are suitably scaled to lie in the interval $(0, 1)$. Otherwise, forecasts will be of the wrong order of magnitude.

A constant input unit is connected to every neuron in hidden layer and also to the output. This is known as the *bias* in NN terminology, or an intercept term in statistics. Essentially the biases are replaced by weights which measure the strength of each connection from the unit input and so become part of the overall set of weights.

7.4 Computing the forecast

For an ANN model with one hidden level of H neurons, the general prediction equation for computing a forecast of x_t (output) using past observations $x_{t-j_1}, ..., x_{t-j_k}$ as inputs may be written in the form [15]:

$$\hat{x_t} = \phi_o(w_{co} + \sum_{h=1}^{H} w_{ho}\phi_h(w_{ch} + \sum_{i=1}^{k} w_{ih}x_{t-j_i})) \quad (7.3)$$

where w_{ch} denotes weights for the connections between the constant input and the hidden neurons, for $h = 1..H$ and w_{co} denotes the weight of the direct connection between the constant input and the output. The weights w_{ih} and w_{ho} denote the weights for the other connections between the inputs and the hidden neurons and between the neurons and the output, respectively. The two functions ϕ_h and ϕ_o denote the activation functions used at the hidden layer and at the output.

The weights to be used in the ANN model are estimated from the data by minimizing the sum of squares of the within sample one step ahead forecast errors, namely $S = \sum_t (\hat{x_{t-1}}(1) - x_t)^2$, over a suitable portion of the data.

In practice, the data is divided into three sections, to fit the ANN model by the first section, called the training set which is used to adjust the weights of the network, a validation set used to minimize overfitting, and the test set to hold back the part of the data, so as to get an independent check on predictions. The test set is kept in reserve so that genuine out-of sample forecasts can be made and compared with the actual observations.

The number of parameters in an ANN model is typically much larger than in traditional time series models. For a single layer ANN model, it is given by $p = H \times (k+2) + 1$ where k is the number of input variables, and H is the number of hidden neurons. The large number of parameters means there is a possibility that model-fitting will overtrain the data and produce a spuriously good fit which does not lead to better forecasts. This motivates the use of regularization where the error function is modified to include a penalty term which prefers small parameter values.

7.5 Architectures used

Neural network must contain memory in order to process temporal information. There are two basic ways to build memory into neural networks [45]. The first is to introduce *time delays* in the network and to adjust their parameters during the learning phase. The second way is to introduce *positive feedback*, that is making the network recurrent. This project will concentrate on two architectures: finite impulse response (FIR) and recurrent neural networks.

FIR neural network uses the unfold-in-time static approach, and is a functional equivalent of the time delay neural network (TDNN). They do not have feedback connections between units. TDNN provide simple forms of dynamics by buffering lagged input variables at the input layer and/or lagged hidden unit outputs at the hidden layer. FIR network is a feedforward network whose static connection weights between units are replaced by an FIR linear filter that can be modeled with tapped delay lines. After applying the unfold-in-time technique to a FIR, all delays will be removed by expanding the network into a large equivalent static structure. The standard backpropagation algorithm [60] is then applied for training. Formally, time delays are identical to time windows and can thus be viewed as autoregressive models.

Recurrent Neural Networks (RNN) have feedback connections. They address the temporal relationship of inputs by maintaining internal states that have memory. Real time recurrent back-propagation and backpropagation through time are two popular algorithms for training RNNs [40], which can be difficult due to the feedback connections.

7.5.1 Finite Impulse Response (FIR) Neural Networks

This chapter introduces the first network architecture that will be used for forecasting future values of nonlinear dynamic systems described in Section 7.2. Section 7.5.2 describes the linear systems (filters) which are used as a substitution to normal feedforward network's weights. Sections 7.5.3 and 7.5.4 describe the training algorithm used by this network, called temporal backpropagation.

7.5.2 Linear Systems

It is possible that P, the process whose output is trying to get predicted is governed by linear dynamics [30]. The study of linear systems is the domain of Digital Signal Processing (DSP).

DSP is concerned with linear, translation-invariant (LTI) operations on data systems. Those operations are implemented by filters. The analysis and design of filters effectively forms the core of this field.

Filters operate on an input sequence u_t, producing an output sequence x_t. They are typically described in terms of their frequency response, i.e. low pass, high-pass, and band-stop.

There are two basic filter architectures, known as the Finite Impulse Response (FIR) filter and the Infinite Impulse Response (IIR) filter.

FIR filters are characterized by $q + 1$ coefficients:

$$x[t] = \sum_{i=0}^{q} \beta_i u[t-i] \tag{7.4}$$

These filters implement the convolution of the input signal [7] with a given coefficient vector β_i. The input in these filters $u[i]$ is the impulse function, and the output of the filter is inherently stable, as it is a sum of a finite number finite multiples of the input values.

IIR filters are characterized by p coefficients.

$$x[t] = \sum_{i=1}^{p} \alpha_i x[t-i] + u[t] \tag{7.5}$$

The input $u[i]$ contributes directly to $x[i]$ at time t, but, crucially, $x[t]$ is otherwise a weighted sum of its own past samples [34]. Because both the input signal and vector α_i are finite in duration, the response asymptotically decays to zero. Once on of the $x[i]$ is non-zero, it will make non-zero contributions to future values of $x[t]$ infinitely often.

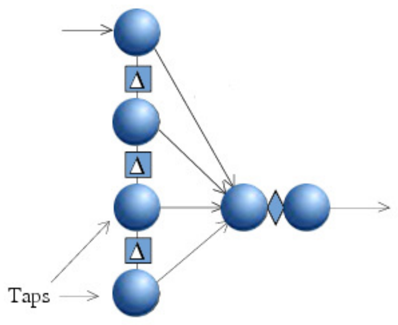


Figure 7.1: Finite Impulse Response Filter

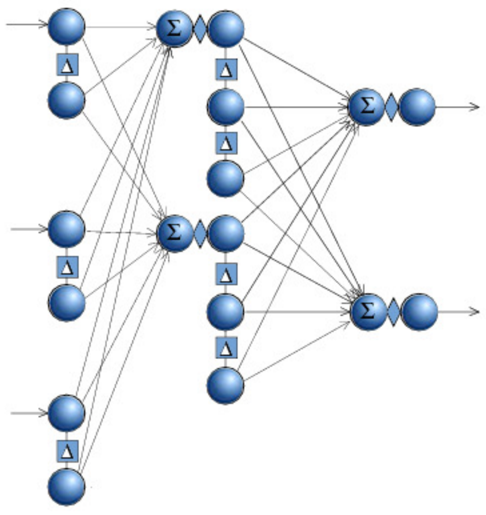


Figure 7.2: Multilayer feedforward network equivalent

7.5.3 FIR filters in ANNs

In Finite impulse response (FIR) Neural networks, each neuron is extended to be able to process temporal features by replacing synapse weights by finite impulse response filters. A general structure of this filter is shown in Figure 7.1. A multilayer feedforward network [42] is then built using these neurons as shown in Figure 7.2. The network input layer consists of FIR filters feeding the data into the neurons in hidden layer. Output of a layer may only connect to the first tap of a node in next layer. The network may have one or several hidden layers. Output layer consists of neurons which receive their inputs from the preceding hidden layer.

At each time increment, one new value is fed to input filters, and output neuron produces one scalar value. In effect this structure has the same functional properties as the Time Delayed Neural Networks (TDNN). However, the FIR neural network is interpreted as a vectoral and temporal extension of the Multilayer Perceptron (MLP). This interpretation leads to the temporal backpropagation algorithm.

7.5.4 Temporal Backpropagation

The basic backpropagation algorithm assumes that the neural network is a combinational circuit, providing an output for a given input. However, many applications suitable for adaptive learning have inherently temporal structures. Every time-delay neural network can be represented as a standard static network simply by duplicating nodes and removing delays. The resultant net is much larger, contains a large number of weight duplications (or triplications), and is not fully interconnected. The process of creating the static equivalent can be thought of as 'unfolding' the network. Once the network is unfolded, the backpropagation algorithm can be applied directly to solve the static network.

Creating a static equivalent of a finite impulse response neural network consists of two stages. In the first stage, all the nodes in the static network are counted. [20] Portions of the static equivalent network are a 'virtual' network of nodes and weights representing past states of the physical network. Counting the total number of virtual nodes is a simple procedure, starting at the last (output) layer and working backwards to the input layer. The output layer of the static network contains the same number of nodes as the output layer of the temporal network. Because this layer has no delay taps, the next layer has no non-physical nodes, and the number of virtual nodes of the static equivalent is equal to the number of filters in that layer times the number of placeholders in each filter. For each layer back to the input, the number of total virtual nodes is a cumulative sum of the number of virtual nodes in that layer plus the number of virtual nodes calculated for the previous layer minus one (because the first placeholder in each filter accepts and propagates its input without delay to the next layer). Mathematically, the notation for total virtual nodes at a layer is:

$$T_l = \begin{cases} 1 & l = L \\ T_{l+1} + T_l - 1, & 1 \leq l \leq L \end{cases} \tag{7.6}$$

where T_l is the physical number of taps per filter, and l denotes the number of layers. The next stage is to actually unfold the network. The first step is to copy down the output nodes, then to copy down all the placeholders of the next layer back, and make each one into a node by prepending a processing element to it [28]. The result is a partial network shown in Figure 7.3. So far, the training algorithm is simply a standard backpropagation without modifications, except that the hidden-layer nodes are referenced with three, instead of two variables. Therefore the corresponding weights are calculated as:

$$\Delta w_{Lyt} = -\eta \delta_{Li1} y(L-1)jt, \qquad 1 \leq j \leq I_{(L-1)}, 1 \leq t \leq T_{(L-1)}, 1 \leq i \leq I_i \tag{7.7}$$

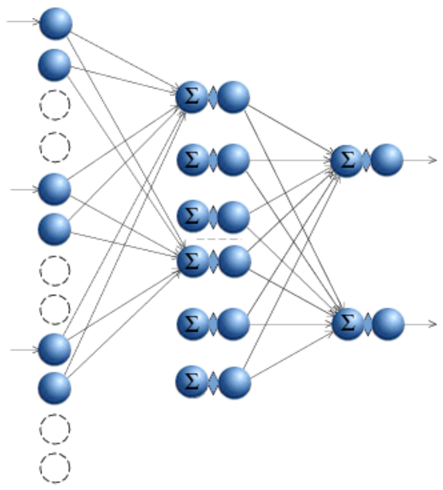


Figure 7.3: Partial static neural network in the unfolding process

$$\delta_{Li1} = -2(d_i - y_{Li1})\frac{d}{dx}(y_{Li1}), \qquad 1 \leq i \leq I_L \tag{7.8}$$

where 1 is used as a subscript to denote that there is only one tap $s = 1$ associated with the output layer L. The network still has the exact physical layout of the temporal net, but without the delay units.The next step is to copy the first layer and second layer weights downward, overlapping placeholders when necessary, until the number of inputs in the first layer equals the number of accumulated inputs calculated.

The final step involves refolding the network by interpreting its elements in the temporal notation. For any weight w_{2ijt} the hidden layer elements to which it connects are $Y_{2in}, 1 \leq n \leq T_2$, where T_2 is the number of filter taps in the hidden layer [28]. A single weight is attached to multiple elements because the weight has been copied several times, and this formula keeps track of all copies. Even though the weight connects only to the x summation element physically in the temporal network, it connects to each of the taps virtually, acting through the summation process. Each one of the hidden-layer nodes in the virtual network is shown as having its own summation element. The resulting network is illustrated in Figure 7.4.

The inputs to which the same weight connects are defined by:

$$y_{1j(t+m-1)}, \qquad 1 \leq m \leq T_2 \tag{7.9}$$

For a given node y_{lis}, the associated weights attached to it in the direction of output are given by:

$$y_{lis} \rightarrow w_{(l+1)kim}, \qquad max(1, s - T_{l+1}) \leq m \leq min(s, T_l) \tag{7.10}$$

where $\rightarrow$ means that y and w are connected.

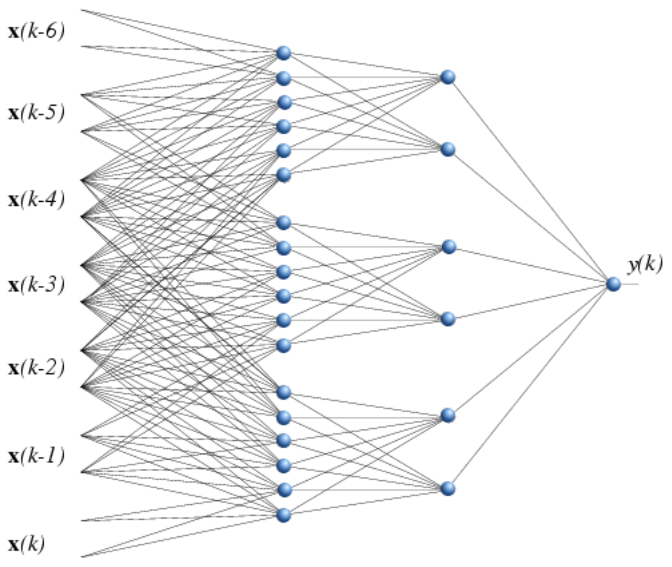


Figure 7.4: Static equivalent of neural network from figure

Finally, for a given y_{lis} and a given weight attached to it $w_{(l+1)kim}$, the associated node on the other end of weight is given by:

$$y_{lis}, w_{(l+1)kim} \rightarrow y(l+1)k(s-m+1) \tag{7.11}$$

These equations define relationship between the connectivity of a temporal network and its static equivalent regardless of the training algorithm used.

At this point, we make some changes in the standard backpropagation algorithm. The change in weight between the current and next iteration is given by:

$$\Delta w_{lijt} = -H \sum_{n=1}^{T_j} \Delta_{liny_{(l-1)j(i+n-1)}} \tag{7.12}$$

where $1 \le l \le L, 1 \le i \le I_j, 1 \le j \le I_{j-1}, 1 \le t \le T_{j-1}$ and the backpropagated error term needed to complete this equation is:

$$\Delta_{lin} = \begin{cases} -2(d_i - y_{Li1})\frac{d}{dx}y_{Li1} & (1) \\ \frac{d}{dx}y_{lin}\sum_{k=1}^{I_{j+1}}\sum_{m=max(1,n-T_{j+1}+1)}^{min(n,T_j)} \Delta_{(l+1)k(n-m+1)}w_{(j+1)kim} & (2) \end{cases} \tag{7.13}$$

where $l = L, n = 1$, $1 \le i \le I_j$ in (1), and $1 < l < L$, $1 \le i \le I_j$, $1 \le n \le T_l$ in (2).

In these equations, L represents the total number of network layers, I_j the number of filters per layer, T_l the number of taps per filter in a given layer, the value y_{Li1} is the output per output node, and d_i is the desired output per output node. The basic feedforward relationship is given by

$$y_{lis} = f(x_{lis}), \qquad 1 < l \leq L, 1 \leq i \leq I_l, 1 \leq s \leq T_l \tag{7.14}$$

where f() is a limited function, such as sigmoid, or tanh.

7.6 Implementation

The unfortunate point about temporal propagation algorithm is that components of the virtual network appear in the training algorithm. Fortunately, it is possible to remove virtual weights from the algorithm but the virtual placeholders remain and must be accounted for in every network realization. Each virtual placeholder is a time-dependent value and so must have an associated delay element. For the temporal propagation algorithm, each δ is associated with a node in the virtual network, and also requires memory for storage, but that is a small price to pay for the benefit of compression of the entire network. The total amount of computation performed is slightly less than standard backpropagation algorithm, because the static equivalent of the temporal network is not fully interconnected.

Given the original static network, it is difficult to make a guess as to a number of hidden layers and hidden layer units needed to optimally solve the problem [32]. The number of layers in the network is equal to the hierarchy of features we wish to detect (for example stock market trends). The number of filters is equal to the number of features, and the number of taps per filter is equal to the time span of the feature divided by the delay time of the tap delays. The total number of virtual taps at the input should the total time length of an input pattern group, even though the actual number of physical input taps is quite small.

The FIR network designed in this project has 2 hidden layers. The training algorithm is given in pseudocode and can be found in 7.1. The other two algorithms are procedures which are called within TrainFIR.

Algorithm 7.1: FIR Training Algorithm

```
Input: NumberofVectors, NumberOfEpochs, Input
Output: Target
1 for $Epoch \leftarrow 0$ to $NumberOfEpochs$ do
2     for $i \leftarrow 0$ to $NumberOfVectors$ do
3         $Layer1pass[i]$ $Layer2pass[i]$ $Layer2backprop[i]$ $Layer1backprop[i]$
          $Update[i]$
4     end
5     $MSE[Epoch]$ $ResetDelta$
6 end
```

Algorithm 7.2: Calculate the output using weights and taps

```
Input: NumHidden,NumInput,BiasVectors
Output: Future values of the time series
for $i \leftarrow 0$ to $NumHidden$ do
    $Out+ = BiasVectors[0][i]$; for $j \leftarrow 0$ to $Taps[0]$ do
        $temp = index - j$ for $k \leftarrow 0$ to $NumInput$ do
            $Out = Input[Weight_k] * Weights[0][Taps_{k,j}]$
        end
        $HiddenOutput[i][temp] = tanh(Out)$
    end
end
```

Passing values to the second layer is similar to 7.2. The only difference between Layer2pass and Layer1pass is that indexing of matrices will change (when addressing taps, and delays). The same applies to Layer2backprop, and Layer1 backprop.

Algorithm 7.3: Forwarding the values to the first tap layer

```
Input: NumOutput,Taps,Weights
Output: HiddenDelta
for $i \leftarrow 0$ to $NumHidden$ do
    $Delta = 0$; for $j \leftarrow 0$ to $Taps[1]$ do
        $temp = index - j$ for $k \leftarrow 0$ to $NumOutput$ do
            $Delta+ = NetDelta[k][temp] * Weights[1][Taps_{k,i}]$
        end
        $temp = index - Taps[1] + 1$
        $HiddenDelta[i] = Delta * tanh(HiddenOutput[i][temp])$
    end
end
```

7.7 Recurrent Neural Networks

A recurrent net is a neural network with feedback (closed loop connections) [67]. The examples include Bidirectional Associative Memory (BAM) [43], Hopfield, Boltzmann machine [4], and recurrent backpropagation nets. The architectures range from fully interconnected to partially connected nets, including multilayer feedfoward networks with distinct input and output layers. Fully connected networks do not have distinct input layers of nodes, and each node has input from all other nodes. Feedback to the node itself is impossible.

Two fundamental ways can be used to add feedback into feedforward multilayer neural networks. Elman introduced feedback from the hidden layer to the context portion of the input layer [27]. This approach pays more attention to the sequence of input variables. Jordan recurrent neural networks [35] use feedback from the output layer to the context nodes of the input layer and give more emphasis to the sequence of output values.

Another popular architecture of these type of ANNs are Dynamic Recurrent Neural Networks (DRNN) [19], which are a class of fully recurrent neural networks obtained by modelling synapses as autoregressive filters. These networks approximate underlying law governing the time series by a system of nonlinear differential equations of internal variables [39]. Therefore, they provide a history-sensitive forecasts without having to be explicitly fed with external memory. The model is trained by a local and recursive error backpropagation algorithm called temporal-recurrent backpropagation. The efficiency of the procedure benefits from the exponential decay of gradient-terms backpropagated through the adjoint network.

7.7.1 Learning in Recurrent Neural Networks

Hebbian learning and gradient ascent learning are key concepts upon which neural network techniques have been based. While backpropagation is relatively simple to implement, several problem can occur in its use in practical applications [47], including the difficulty of avoiding entrapment in local minima. The added complexity of the dynamical processing in recurrent neural networks from the time delayed updating of the input data requires more complex algorithms for representing learning.

Neural networks with recurrent connections and dynamical processing elements are finding increasing applications in diverse areas [69]. While feedforward networks have been recognized to perform excellent pattern recognition even with complex nonlinear decision surfaces, they are limited to processing stationary patterns (invariant with time).

7.8 RNN Architectures

There are two ways to include feedback connections in neural networks[49]: activation feedback, and output feedback. These schemes are related to state space representation of neural networks. The output of a neuron in a network using activation feedback is:

$$\begin{aligned} v(k) &= \sum_{i=0}^{M} w_{uj}(k)u(k-i) + \sum_{j=1}^{N} w_{v,j}(k)v(k-j), \\ y(k) &= \Phi(v(k)) \end{aligned} \tag{7.15}$$

The transfer function of a neuron in a network using the output feedback scheme can be expressed as:

$$\begin{aligned} v(k) &= \sum_{i=0}^{M} w_{uj}(k)u(k-i) + \sum_{j=1}^{N} w_{y,j}(k)v(k-j), \\ y(k) &= \Phi(v(k)) \end{aligned} \tag{7.16}$$

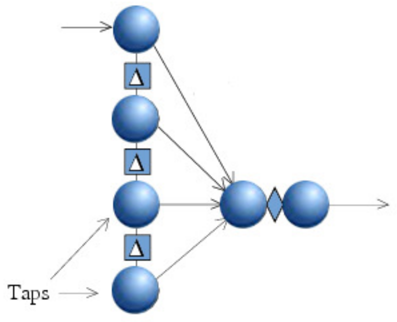


Figure 7.5: Example of an Elman recurrent network

7.8.1 The Elman Network

The Elman network was introduced by Elman in 1990 [22]. In this network a set of context units are introduced which are extra input values whose activation values are fed back from the hidden units. This network architecture is similar to the Jordan network except that inputs are fed back from hidden units, instead of output units, and the values are fed back to context units instead of state units. Figure 7.5 shows an example of an Elman recurrent network.

A three-layer network is used, with the addition of a set of "context units" in the input layer. There are connections from the middle (hidden) layer to these context units fixed with a weight of one. At each time step, the input is propagated in a standard feed-forward fashion [39], and then a learning rule (usually back-propagation) is applied. The fixed back connections result in the context units always maintaining a copy of the previous values of the hidden units (since they propagate over the connections before the learning rule is applied). Thus the network can maintain a sort of state, allowing it to perform such tasks as sequence-prediction that are beyond the power of a standard multi-layer perceptron.

7.8.2 Fully recurrent networks

Fully recurrent networks, as their name suggests, provide two-way connections between all processors in the neural network. A subset of the units is designated as the input processors, and they are assigned or clamped to the specified input values [8]. The data then flows to all adjacent connected units and circulates back and forth until the activation of the units stabilizes. Figure 7.6 shows the input units feeding into both the hidden units (if any) and the output units. The activations of the hidden and output units then are recomputed until the neural network stabilizes. At this point, the output values can be read from the output layer of processing units.

Fully recurrent networks are complex, dynamical systems, and they exhibit all of the power and instability associated with limit cycles and chaotic behavior of such systems. [8]

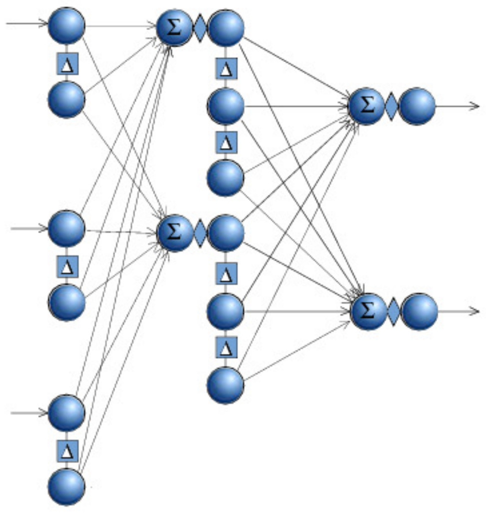


Figure 7.6: Structure of fully recurrent neural networks

Unlike feed-forward network variants, which have a deterministic time to produce an output value (based on the time for the data to flow through the network), fully recurrent networks can take an in-determinate amount of time.

By placing some constraints on the connection weights, we can ensure that the network will enter a stable state. The connections between units must be symmetrical. Fully recurrent networks are used primarily for optimization problems and as associative memories. A nice attribute with optimization problems is that depending on the time available, one can choose to get the recurrent network's current answer or wait a longer time for it to settle into a better one [8]. This behavior is similar to the performance of people in certain tasks. The training algorithm which will be used to train these networks for this project is backpropagation through time, which will be described in the next section.

7.9 Backpropagation through time

The backpropagation through time algorithm for training recurrent neural networks can be derived the temporal operation of the network into a multilayer feedforward network, which will grow at every time step t. All recurrent weights can be duplicated spatially for an arbitrary number of time steps τ. Consequently, each node which sends activation along a recurrent connection has at least τ number of copies as well.

These networks may contain any number of feedback loops in their connectivity graph. The only restriction in this implementation is that there may be no connections between input units. The gradients of the weights in the recurrent network [66] are approximated using an feedforward network with a fixed number of layers. By unfolding the network at

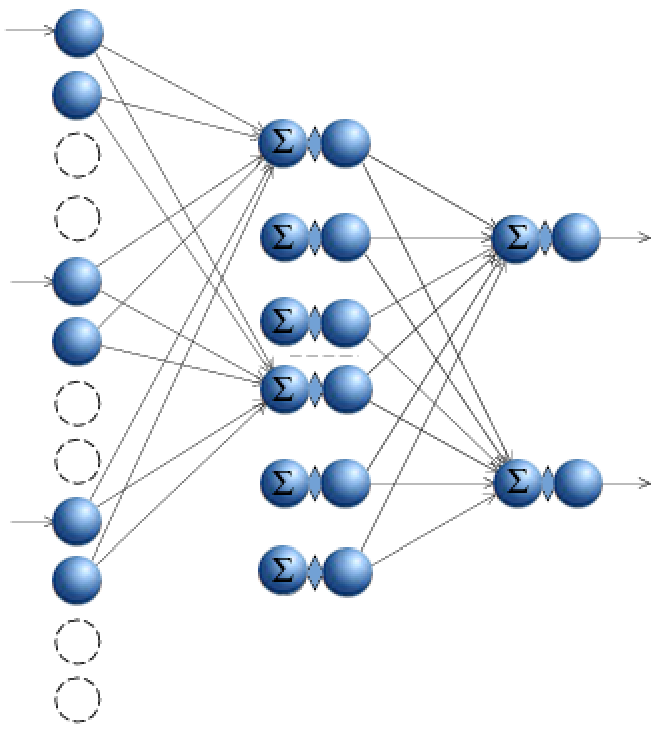


Figure 7.7: Unfolding the network using BPTT

time steps $1, .., T$ the recurrent network gets transformed to a feedforward neural network with T stages of computation. At each time step t an external output $x(t)$ is fed into the network and the outputs of all computing units are recorded. Initial values of output units are assumed to be zero at $t = 0$ but the external input can be different than zero. Figure 7.7 shows a diagram of the unfolded network.

At each time step, vectors consisted of units outputs, and derivatives of the activation function are stored. The error of the network can be measured after each time step if a sequence of values is to be produced, or just after the final step T if only the final input is of importance. In the unfolding process, the transformed network is indistinguishable from the original network from the viewpoint of the result it produces. [49]

There are three versions of backpropagation through time:

- **BPTT**: Backpropagation through time with online-update. The gradient for each weight is summed over backstep copies between successive layers and the weights are adapted using the formula for backpropagation with momentum term after each pattern. The momentum term uses the weight change during the previous pattern.

- **BBPTT**: Batch backpropagation through time. The gradient for each weight is calculated for each pattern as in BPTT and then averaged over the whole training set. The momentum term uses update information closer to the true gradient than in BPTT.

- **QPTT**: Quickprop through time. The gradient in quickprop through time is calculated as in BBPTT, but the weights are adapted using the substantially more efficient quickprop-update rule.

A recurrent network has to start processing a sequence of patterns with defined activations. All activities in the network may be set to zero by applying an input pattern containing only zero values. If such all-zero patterns are part of normal input patterns, an extra input unit has to be added for reset control. If this reset unit is set to 1, the network is in the free running mode. If the reset unit and all normal input units are set to 0, all activations in the network are set to 0 and all stored activations are cleared as well.

Gradient δ is calculated as follows [30]:

$$\delta_k(k) = \begin{cases} \Phi'(v_j(k))e_j(k), & k = k_1 \\ \Phi'(v_j(k))[e_j(k) + \sum_{i \in A} w_{i,j}\delta_i(k-1)], & k_0 < k < k_1 \end{cases} \tag{7.17}$$

Weight changes are computed as in standard BP:

$$\frac{\delta E(t)}{\delta w_{ij}} = \sum_{\tau=t0+1}^{t} \delta_i(\tau)x_j(\tau - 1) \tag{7.18}$$

7.10 Recurrent FIR Neural Networks

Although variety of ANNs are used for time series prediction, there was no consensus on the best architecture to use. Horne and Giles [65] concluded that "recurrent networks usually did better than TDNNs except on the finite memory machine problem". On the other hand Hallas and Dorffner [28] stated that "recurrent networks do not seem to be able to do prediction under the given conditions" and a "simple feedforward network significantly performs best for most nonlinear time series". However, many agree that the best architecture is problem dependent and that efficiency of the learning algorithm is more important than the network model used.

Not a lot of research has been done in time series modelling using RFIRNN [49]. In RFIR, each feedback link has a non-zero delay, and the link between any two nodes has an explicit memory modelled by a multi-tap FIR filter for storing history information, example is shown in Figure 7.8. The advantage of this architecture is that it can store more historical information than both RNN and FIR. The concept of RFIR can be generalized easily to existing recurrent ANN, and nonlinear autoregressive networks. These networks use a violation-guided backpropagation algorithm.

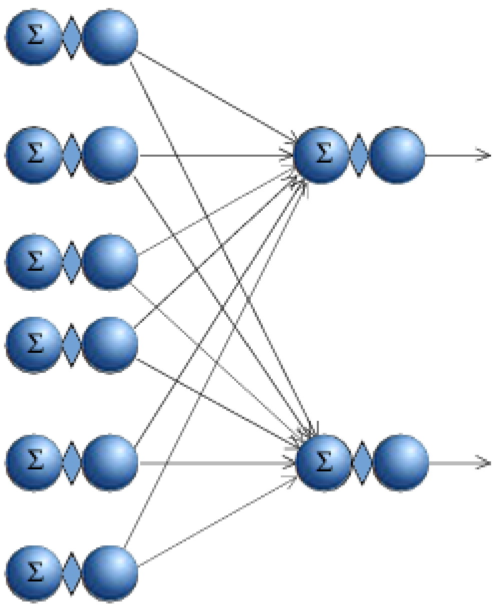


Figure 7.8: Example of a three layer Recurrent FIR neural network

7.11 Generalized Epochwise Backpropagation Through Time

Generalized Epochwise Backpropagation Through Time is used to train RFIR networks [54], as the errors propagate back in time domain. Standard epochwise backpropagation through time is used to train normal recurrent neural networks [67]. For a L-layer RFIR, a bias node is included in each layer except the output layer. The bias node is indexed as Node $1, N(l)$ as regular nodes from 2 to $N(l) + 1$, with recurrent nodes starting from $N(l) + 2$. The number of taps for an FIR filter connecting nodes between layer l and layer $l + 1$ is denoted by $T(l)$ with $T(l) + 1$ coefficients. $w^l_{i,j}(m)$ denotes the weight for the mth coefficient of the FIR filter that connects the i^{th} node in layer $l + 1$ and the j^{th} node in layer l. The activation function for layer l is denoted by $\phi_l(x)$. The derivative for $y = \phi^l(x)$ is given by:

$$\psi^l(y) = \begin{cases} \phi_l'(x) = \alpha(1 - y^2) & \text{if } \phi(x) \text{ is hyperbolic} \\ \phi_l'(x) = 1 & \text{if } \phi(x) \text{ is linear} \end{cases} \tag{7.19}$$

The output value of a node indexed as i in layer l at time t is denoted by $s^l_i(t)$ and $s^l_i(t) = \psi(a^l_i(t))$ for $l \geq 2$ with $a^l_i(t)$ being the input to the i^{th} node in the l^{th} layer:

$$a_i^l(t) = w_{i,1}^{l-1} + \sum_{j=2}^{N(l-1)+1} \sum_{m=1}^{T(l-1)+1} w_{i,j}^{l-1}(m) s_j^{l-1}(t+1-m) + \sum_{j=N(l-1)+2}^{N(l-1)+N(k)+1} \sum_{m=1}^{T(k)+1} w_{i,j}^{l-1} w_{i,j}^{l-1}(m) s_j^{l-1}(t-m) \tag{7.20}$$

where k is the layer feeding back to layer $l-1$. When there is no recurrent node in layer $l-1$ the last part in 7.20 is omitted. In the Generalized epochwise Back-propagation through time, the output error for node k at time t is given by:

$$e_k(t) = o_k(t) - d_k(t) \tag{7.21}$$

The energy function over interval $[t_0, t_1]$ is:

$$\epsilon_{av}(t_0, t_1) = \frac{1}{t_1 - t_0 + 1} \sum_{t=t_0}^{t_1} \sum_{k=1}^{N(L)} e_k(t) \tag{7.22}$$

The gradient of the energy function 7.22 over $w_{i,j}^t(m)$ can be expressed as follows, which represents the 'cost' function over state space (as opposed to weight space):

$$\frac{\partial \epsilon_{av}(t_0, t_1)}{\partial w_{i,j}^l(m)} = \frac{1}{n} \sum_{t=t_0}^{t_1} \vartheta_i^l(t) s_j^l(t-m) \tag{7.23}$$

where $n = t_1 - t_0 + 1$. The local gradient $\delta_i^l(t)$ is computed through backpropagation based on the values of t and L as follows:

$$\delta_i^l(t) = 2e_i(t)\phi^{l+1}(s_{i+1}^{l+1}(t)) \tag{7.24}$$

for $t = t_1$ and $l = L - 1$,

$$\delta_i^l(t) = (2e_i(t) + \sum_{k=1}^{N}(l+1) \sum_{m \le t_1, m=1}^{T(l')+1} w_{k,i+1+N(l)}^l(m) \delta_k^{l'}(t+m+1))\phi^{l+1}(s_{i+1}^{l+1}(t)) \tag{7.25}$$

for $t < t_1$ and $l = L - 1$,

$$\delta_i^l(t) = (\sum_{k=1}^{N(l+2)} \sum_{m \le t_1, m=1}^{T(l+1)=1} w_{k,i+1}^{l+1}(m) \delta_k^{l+1}(t+m))\phi^{l+1}(s_{i+1}^{l+1}(t)) \tag{7.26}$$

for $t = t_1$ and $l < L - 1$, and

$$\delta_i^l(t) = (\sum_{k=1}^{N(l+2)} \sum m \le t_1, m = 1^{T(l+1)+1} w_{k,i+1}^{l+1}(m)\delta_k^{l+1}(t+m) + \\ = + \sum_{k=1}^{N(l+1)} \sum_{m \le t_1, m=1}^{T(l')+1} w_{k,i+1+N(l)}^{l}(m)\delta_k^{l'}(t+m+1))\phi^{l+1}(s_{i+1}^{l+1}(t)) \tag{7.27}$$

for $t < t_1$ and $l < L-1$. In all cases, $1 \le l \le L-1$, and index l' is related to the recurrent node in a way that the recurrent node in layer l is fed by the regular node in layer $l'+1$. When there is no recurrent node in layer l then the term involving $\delta_k^{l'}$ is discarded.

The weights can now be updated along the negative gradient direction by means of:

$$\Delta w_{i,j}^l(m) = -\eta \frac{\partial \epsilon_{av}(t_0, t_1)}{\partial w_{i,j}^l(m)} \tag{7.28}$$

where η is the learning rate.

7.12 Forecasting Horizontal and Vertical Movements using FIR and RFIR

As an alternative to empirical prediction methods, methods based on influential functions and methods based on mechanical model, artificial neural networks can be used for the surface subsidence prediction.

The training and testing of neural network is based on available data from the project 'Prokop'. Input variables represent extraction parameters and coordinates from the local coordination system of the points of interest, while the output variables represent future values. After the neural network has been successfully trained, its performance is tested on a separate testing set. Finally, the surface subsidence trough above the projected excavation is predicted by the trained neural network. The applicability of artificial neural networks for the prediction of surface subsidence was verified in different subsidence models and proved on actual excavated levels and in leveled data on surface profile points on several buildings near the 'Prokop' train station in Belgrade, which was under influence of tunelling due to the construction of an underground station.

The number of output neurons is set to 10, as the desired outcome is to predict the last ten values in the time series. The number of input neurons depends on the length of the time series, which is always $L-10$, where L is the length. In both architectures used, FIR and RFIR, three layers had been used with hidden neurons depending again on the length of the series. For the FIR ANN, the number of hidden neurons per layer was usually the quarter of the size of L, whereas in the RFIR nearly as half. The number of taps per layer were chosen as $(4, 2)$ for FIR and $(7, 3)$ for RFIR network. The optimum value for the learning rate was $\eta = 0.1$.

7.12.1 Prediction of Points in the Local Coordination System

The local coordination system mapped a couple of kilometers around the vicinity of the station, and thus the values of (x, y) were projected in meters. In order to get the best results, these values had to be leveled, and preprocessed. To do this, only the movements in millimeters were predicted, i.e. $(x - \lfloor x \rfloor)$ for horizontal, and $(y - \lfloor y \rfloor)$ for vertical movements, due to the fact that deformations never exceed one full meter. Measured values for all points found in residential buildings can be found in Appendix 7.12.3.

The time series consist of a total of up to 50 measurements. Some points have been measured less often because normally the measurements occur on the day after the construction work is being done. The points which have undergone most measurements involved the ones around the abutment diaphragms, which are actually the main reason why the company was hired to perform geodetic measurements.

The points used to measure the behaviour of deformations in residential buildings have the index between 100 and 120. The points with the index value less than or equal to 32 were used to measure the deformation found in a small building where the equipment was stored, which was also very close to the entrance of the tunnel.

7.12.2 Comparative analysis of FIR and RFIR Neural Networks

FIR and RFIR have been used for deformation prediction based on the previously used data set 'Prokop'. Previously, the data was tested by using a rule-based expert system and results obtained were promising and useful. As the basic focus of this research is to predict the behaviour of deformations in buildings, so machine learning based approaches have been explored and tested.

FIR and RFIR Neural Networks have been used for the prediction of deformation from the available data. The input variables represent extraction parameters and coordinates from the local coordination system of the points of interest. The output variables represent future values. After the successful training of neural network, its performance has been tested on a separate testing data.

There are three types of data sets for neural networks: the validation data, training data and the testing data. The validation data is for the validation of the neural network architecture, the training data is for learning the patterns and testing data is for the testing accuracy of the classification. The trained neural network successfully predicted the surface subsidence trough above the projected excavation. The results have been validated and proved to be better as compared to the empirical prediction methods. The measured values for all points found in residential buildings are presented in Tables 7.3- 7.16.

The predicted values of the horizontal and vertical points are given in Tables 7.21-7.48. The results show that even though the forecasted values could not match the actual values at all times, the trend of movement had been successfully predicted in most cases, using both the FIR and RFIR.

The results of the measurements show that the movements are typically of lowest value at the base of the building and highest at the top of the building as proved by the

Tables 7.3- 7.16. When comparing the performance of FIR and RFIR, the proposed RFIR outperformed FIR in forecasting the actual values in different experiments. The mean absolute percentage error for the predicted horizontal measured points using the RFIR was 3.1%, and vertical 5.3%, whereas for the FIR, this was 5.4%, and 7.7%. In some instances, the mean error was less than 1%, specifically for points where no vertical or horizontal movement was recorded (there was no deformation present, either because the point was too far from the construction site or because of the nature of the settlement). In order to increase the accuracy of these two networks, more data gathered from deformation monitoring is required in order to increase the size of the training data set and make more accurate assessments. Both ANNs would yield better results when used on long term engineering projects where measurements are taken in frequent time intervals.

The predicted results obtained from both the FIR and RFIR ANN for the horizontal and vertical measurements can be found in Tables 7.21- 7.48, for all points from (x14,y14)–(x114b,y114b) (20 total). The predicted deformations are the same as that obtained from the empirical methods. The plots depicting the mean and standard deviation for horizontal and vertical points can be found in Figures 7.9 and 7.10 respectively, and for (X, Y) in 7.11. The plots of absolute and relative errors can be found in Figures 7.12- 7.15.

The deformations in high rise building are caused by many external factors and cannot be compared to residential medium size buildings. In the data from 'Prokop' station, the deformation was entirely induced by man, and the geodetic engineers had to make sure that the structural movements were reduced to a minimum, making them much tougher to predict, as one cannot easily predict the engineer's mistake during the process of tunneling.

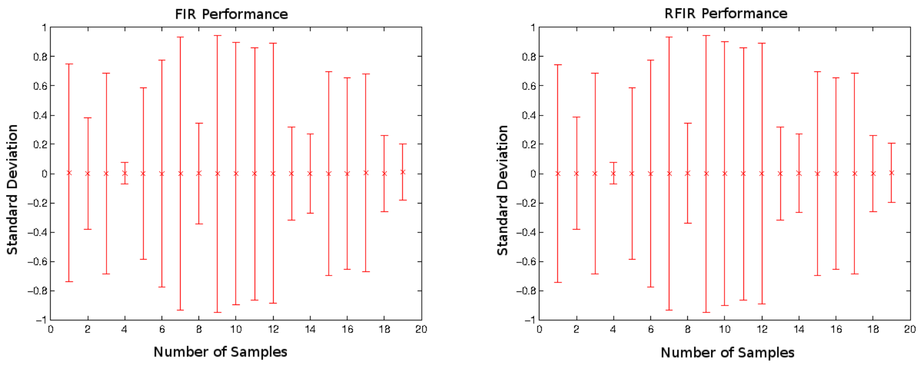


Figure 7.9: Mean and standard deviation for horizontal points (X) for FIR (left) and RFIR (right)

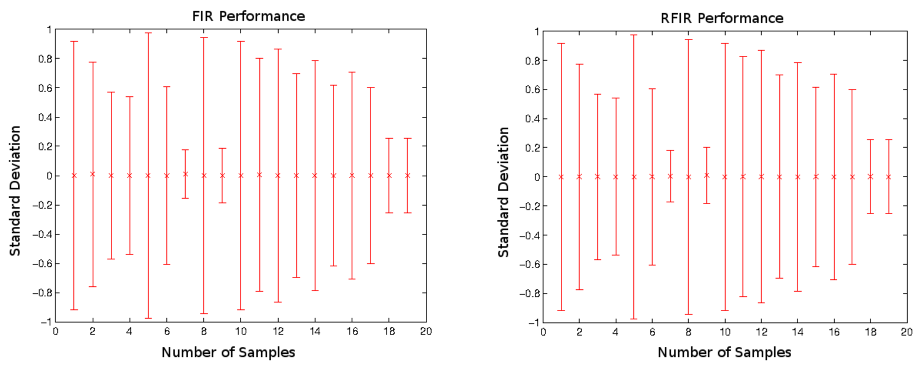


Figure 7.10: Mean and standard deviation for vertical points (Y) for FIR (left) and RFIR (right)

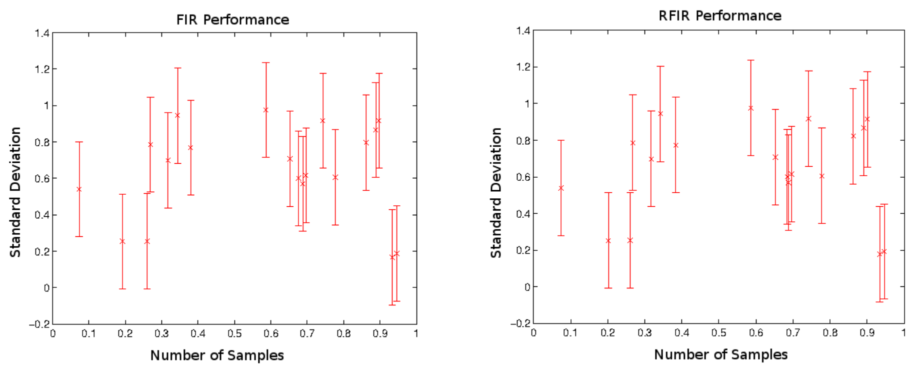


Figure 7.11: Mean and standard deviation for (X,Y) for FIR (left) and RFIR (right)

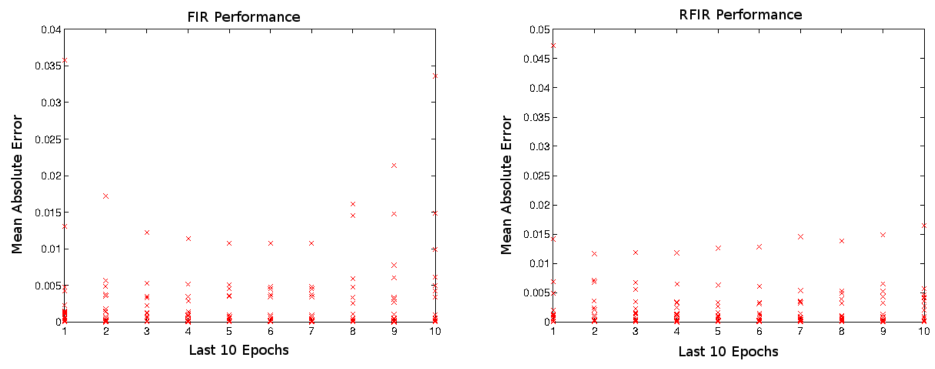
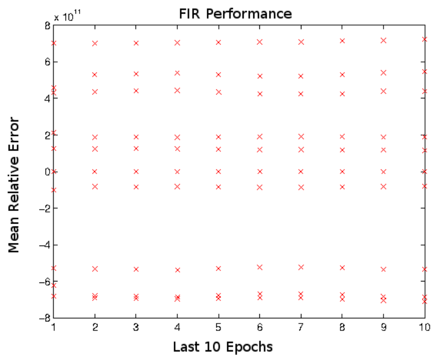


Figure 7.12: Absolute error for horizontal points FIR (left) and RFIR (right)



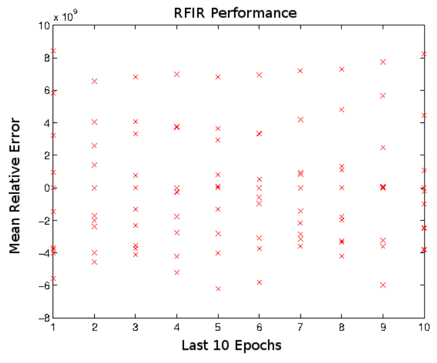
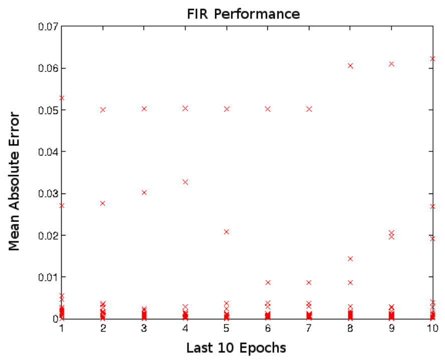


Figure 7.13: Relative error for horizontal points FIR (left) and RFIR (right)



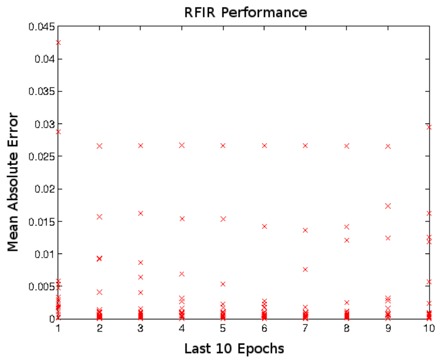
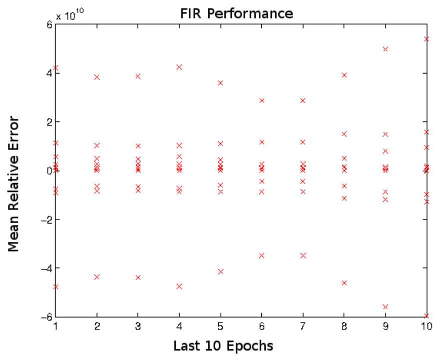


Figure 7.14: Absolute error for vertical points FIR (left) and RFIR (right)



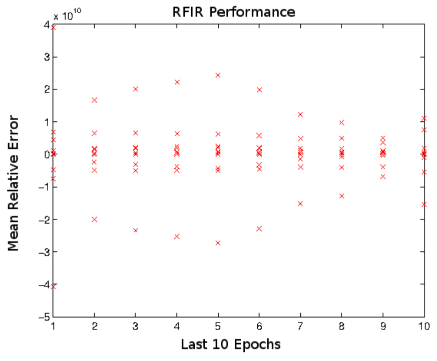


Figure 7.15: Relative error for vertical points FIR (left) and RFIR (right)

7.12.3 Actual Values of Measured Points

PointID/Epoch	**1**	**2**	**3**	**4**	**5**
x14	0.789224	0.729091	0.729064	0.729040	0.729024
x22	0.377875	0.380933	0.380889	0.380841	0.380793
x24	0.686123	0.688992	0.688963	0.688941	0.688924
x29	0.074574	0.075533	0.075488	0.075442	0.075393
x32	0.586475	0.586534	0.586489	0.586441	0.586392

Table 7.1: x14-x32 actual values, epochs 1-5

PointID/Epoch	**6**	**7**	**8**	**9**	**10**
x14	0.729011	0.729005	0.729003	0.729005	0.729011
x22	0.380742	0.380694	0.380646	0.380602	0.380560
x24	0.688911	0.688904	0.688903	0.688906	0.688911
x29	0.075343	0.075294	0.075246	0.075201	0.075160
x32	0.586342	0.586293	0.586247	0.586202	0.586161

Table 7.2: x14-x32 actual values, epochs 6-10

PointID/Epoch	**1**	**2**	**3**	**4**	**5**
x105	0.778524	0.777492	0.777463	0.777441	0.777423
x110	0.933923	0.934191	0.934164	0.934140	0.934124
x110a	0.340924	0.342491	0.342464	0.342440	0.342424
x110b	0.945824	0.946092	0.946063	0.946041	0.946023

Table 7.3: x105-x110b actual values, epochs 1-5

PointID/Epoch	**6**	**7**	**8**	**9**	**10**
x105	0.777412	0.777404	0.777403	0.777406	0.777412
x110	0.934111	0.934105	0.934102	0.934105	0.934111
x110a	0.342411	0.342405	0.342402	0.342405	0.342411
x110b	0.946012	0.946004	0.946003	0.946006	0.946012

Table 7.4: x105-x110b actual values, epochs 6-10

PointID/Epoch	1	2	3	4	5
x111	0.901923	0.900692	0.900663	0.900641	0.900623
x111a	0.862124	0.862492	0.862463	0.862441	0.862423
x111b	0.890575	0.893791	0.893764	0.893740	0.893723
x112a	0.315024	0.318092	0.318064	0.318041	0.318024
x112b	0.268723	0.269291	0.269263	0.269240	0.269223

Table 7.5: x111-x112b actual values, epochs 1-5

PointID/Epoch	6	7	8	9	10
x111	0.900612	0.900604	0.900603	0.900606	0.900612
x111a	0.862412	0.862404	0.862403	0.862406	0.862412
x111b	0.893712	0.893705	0.893702	0.893705	0.893712
x112a	0.318011	0.318005	0.318003	0.318006	0.318011
x112b	0.269212	0.269204	0.269202	0.269205	0.269212

Table 7.6: x111-x112b actual values, epochs 6-10

PointID/Epoch	1	2	3	4	5
x113	0.696124	0.696692	0.696664	0.696641	0.696624
x113a	0.651324	0.652992	0.652963	0.652941	0.652923
x113b	0.699324	0.678291	0.678264	0.678240	0.678224
x114a	0.261024	0.260892	0.260863	0.260841	0.260823
x114b	0.201624	0.200292	0.200264	0.200240	0.200224

Table 7.7: x113-x114b actual values, epochs 1-5

PointID/Epoch	6	7	8	9	10
x113	0.696611	0.696605	0.696603	0.696606	0.696611
x113a	0.652911	0.652904	0.652903	0.652906	0.652911
x113b	0.678211	0.678205	0.678203	0.678205	0.678211
x114a	0.260812	0.260804	0.260803	0.260806	0.260812
x114b	0.200211	0.200205	0.200203	0.200206	0.200211

Table 7.8: x113-x114b actual values, epochs 6-10

PointID/Epoch	1	2	3	4	5
y14	0.914936	0.917513	0.917494	0.917482	0.917476
y22	0.730648	0.788496	0.788448	0.788402	0.788355
y24	0.567936	0.569513	0.569494	0.569482	0.569476
y29	0.543448	0.536297	0.536249	0.536201	0.536156
y32	0.971948	0.976097	0.976049	0.976001	0.975956

Table 7.9: y14-y32 actual values, epochs 1-5

PointID/Epoch	**6**	**7**	**8**	**9**	**10**
y14	0.917478	0.917491	0.917513	0.917546	0.917589
y22	0.788312	0.788271	0.788233	0.788198	0.788166
y24	0.569478	0.569491	0.569513	0.569546	0.569589
y29	0.536113	0.536072	0.536034	0.535997	0.535965
y32	0.975913	0.975872	0.975834	0.975797	0.975765

Table 7.10: y14-y32 actual values, epochs 6-10

PointID/Epoch	**1**	**2**	**3**	**4**	**5**
y105	0.606937	0.603913	0.603895	0.603881	0.603875
y110	0.177336	0.172913	0.172894	0.172882	0.172876
y110a	0.947036	0.943312	0.943294	0.943282	0.943275
y110b	0.183436	0.189813	0.189795	0.189781	0.189775

Table 7.11: y105-y110b actual values, epochs 1-5

PointID/Epoch	**6**	**7**	**8**	**9**	**10**
y105	0.603879	0.603891	0.603913	0.603945	0.603989
y110	0.172878	0.172891	0.172913	0.172946	0.172989
y110a	0.943278	0.943290	0.943312	0.943346	0.943390
y110b	0.189779	0.189791	0.189813	0.189845	0.189889

Table 7.12: y105-y110b actual values, epochs 6-10

PointID/Epoch	**1**	**2**	**3**	**4**	**5**
y111	0.918637	0.914213	0.914194	0.914182	0.914176
y111a	0.852436	0.849613	0.849594	0.849582	0.849576
y111b	0.867847	0.866412	0.866394	0.866382	0.866375
y112a	0.696737	0.699012	0.698995	0.698981	0.698975
y112b	0.785036	0.785912	0.785895	0.785882	0.785875

Table 7.13: y111-y112b actual values, epochs 1-5

PointID/Epoch	**6**	**7**	**8**	**9**	**10**
y111	0.914178	0.914190	0.914213	0.914246	0.914290
y111a	0.849578	0.849591	0.849613	0.849646	0.849689
y111b	0.866378	0.866390	0.866412	0.866446	0.866490
y112a	0.698979	0.698990	0.699012	0.699045	0.699090
y112b	0.785878	0.785890	0.785912	0.785945	0.785990

Table 7.14: y111-y112b actual values, epochs 6-10

PointID/Epoch	**1**	**2**	**3**	**4**	**5**
y113	0.614236	0.616513	0.616495	0.616481	0.616475
y113a	0.704737	0.707212	0.707195	0.707182	0.707175
y113b	0.600336	0.599813	0.599795	0.599781	0.599775
y114a	0.252636	0.254612	0.254595	0.254581	0.254575
y114b	0.252636	0.252913	0.252894	0.252882	0.252876

Table 7.15: y113-y114b actual values, epochs 1-5

PointID/Epoch	**6**	**7**	**8**	**9**	**10**
y113	0.616479	0.616491	0.616513	0.616545	0.616589
y113a	0.707179	0.707190	0.707212	0.707245	0.707290
y113b	0.599779	0.599791	0.599813	0.599845	0.599889
y114a	0.254579	0.254591	0.254612	0.254645	0.254689
y114b	0.252878	0.252891	0.252913	0.252946	0.252990

Table 7.16: y113-y114b actual values, epochs 6-10

7.12.4 FIR and RFIR Prediction Results

PointID/Epoch	1	2	3	4	5
x14	0.753445	0.746285	0.741321	0.740431	0.739763
x22	0.379035	0.379134	0.379721	0.379922	0.380190
x24	0.687665	0.687572	0.687660	0.687844	0.687973
x29	0.070349	0.071795	0.073257	0.074600	0.075343
x32	0.586432	0.586423	0.586383	0.586376	0.586372

Table 7.17: FIR: x14-x32 predicted values, epochs 1-5

PointID/Epoch	6	7	8	9	10
x14	0.739759	0.739756	0.743565	0.743761	0.743834
x22	0.380546	0.380522	0.380498	0.380211	0.380288
x24	0.687970	0.687968	0.687853	0.687877	0.687893
x29	0.075475	0.075107	0.074741	0.072474	0.070913
x32	0.586349	0.586312	0.586274	0.586277	0.586269

Table 7.18: FIR: x14-x32 predicted values, epochs 6-10

PointID/Epoch	1	2	3	4	5
x14	0.742009	0.740726	0.740884	0.740845	0.741629
x22	0.382738	0.382986	0.383157	0.383252	0.383431
x24	0.688098	0.688088	0.688075	0.688068	0.688055
x29	0.073207	0.073433	0.073840	0.074143	0.074329
x32	0.586385	0.586382	0.586376	0.586373	0.586370

Table 7.19: RFIR: x14-x32 predicted values, epochs 1-5

PointID/Epoch	6	7	8	9	10
x14	0.741824	0.743560	0.742788	0.743874	0.745495
x22	0.383850	0.384164	0.384551	0.384892	0.385214
x24	0.688043	0.688029	0.688010	0.687997	0.687991
x29	0.074782	0.074831	0.074940	0.075279	0.072430
x32	0.586366	0.586361	0.586352	0.586341	0.586341

Table 7.20: RFIR: x14-x32 predicted values, epochs 6-10

PointID/Epoch	**1**	**2**	**3**	**4**	**5**
x105	0.777962	0.777702	0.777474	0.777435	0.777382
x110	0.933963	0.933979	0.934024	0.934048	0.934050
x110a	0.341509	0.342817	0.343274	0.343269	0.343142
x110b	0.946100	0.946085	0.946101	0.946114	0.946114

Table 7.21: FIR: x105-x110b predicted values, epochs 1-5

PointID/Epoch	**6**	**7**	**8**	**9**	**10**
x105	0.777493	0.777490	0.777491	0.777544	0.777560
x110	0.934040	0.934034	0.933991	0.933995	0.933993
x110a	0.343121	0.343111	0.344938	0.345566	0.347413
x110b	0.946112	0.946110	0.946118	0.946119	0.946119

Table 7.22: FIR: x105-x110b predicted values, epochs 6-10

PointID/Epoch	**1**	**2**	**3**	**4**	**5**
x105	0.777746	0.777741	0.777725	0.777727	0.777719
x110	0.934022	0.934021	0.934021	0.934022	0.934024
x110a	0.339766	0.340096	0.341084	0.342041	0.342590
x110b	0.946031	0.946030	0.946029	0.946027	0.946026

Table 7.23: RFIR: x105-x110b predicted values, epochs 1-5

PointID/Epoch	**6**	**7**	**8**	**9**	**10**
x105	0.777706	0.777704	0.777690	0.777686	0.777702
x110	0.934024	0.934023	0.934026	0.934028	0.934028
x110a	0.342654	0.343122	0.343484	0.343754	0.344479
x110b	0.946024	0.946026	0.946026	0.946028	0.946029

Table 7.24: RFIR: x105-x110b predicted values, epochs 6-10

PointID/Epoch	**1**	**2**	**3**	**4**	**5**
x111	0.897133	0.897123	0.897134	0.897214	0.897156
x111a	0.862320	0.862344	0.862378	0.862381	0.862366
x111b	0.888298	0.888178	0.888508	0.888579	0.888642
x112a	0.316305	0.317372	0.317932	0.318092	0.318017
x112b	0.269407	0.269491	0.269428	0.269379	0.269383

Table 7.25: FIR: x111-x112b predicted values, epochs 1-5

PointID/Epoch	**6**	**7**	**8**	**9**	**10**
x111	0.897157	0.897157	0.897260	0.897254	0.897263
x111a	0.862363	0.862362	0.862355	0.862352	0.862351
x111b	0.888915	0.888911	0.888907	0.887663	0.887617
x112a	0.318007	0.318001	0.318029	0.318010	0.318005
x112b	0.269383	0.269384	0.269433	0.269440	0.269451

Table 7.26: FIR: x111-x112b predicted values, epochs 6-10

PointID/Epoch	**1**	**2**	**3**	**4**	**5**
x111	0.900812	0.900807	0.900799	0.900790	0.900787
x111a	0.862338	0.862339	0.862340	0.862339	0.862341
x111b	0.890175	0.890215	0.890290	0.890313	0.890353
x112a	0.315521	0.316017	0.316426	0.316863	0.317353
x112b	0.268337	0.268093	0.267791	0.267867	0.267596

Table 7.27: RFIR: x111-x112b predicted values, epochs 1-5

PointID/Epoch	**6**	**7**	**8**	**9**	**10**
x111	0.900783	0.900778	0.900770	0.900763	0.900768
x111a	0.862340	0.862341	0.862342	0.862340	0.862340
x111b	0.890399	0.890453	0.890568	0.890510	0.890167
x112a	0.317804	0.318372	0.318617	0.318413	0.318249
x112b	0.267759	0.267830	0.268236	0.268824	0.264928

Table 7.28: RFIR: x111-x112b predicted values, epochs 6-10

PointID/Epoch	**1**	**2**	**3**	**4**	**5**
x113	0.696556	0.696694	0.696650	0.696625	0.696644
x113a	0.652362	0.652464	0.652400	0.652523	0.652467
x113b	0.686277	0.683150	0.681556	0.676848	0.673705
x114a	0.260907	0.260818	0.260734	0.260672	0.260637
x114b	0.203007	0.201688	0.199035	0.197377	0.196608

Table 7.29: FIR: x113-x114b predicted values, epochs 1-5

PointID/Epoch	**6**	**7**	**8**	**9**	**10**
x113	0.696642	0.696641	0.696650	0.696648	0.696646
x113a	0.652478	0.652478	0.652477	0.652449	0.652453
x113b	0.673691	0.673682	0.672299	0.670434	0.668270
x114a	0.260609	0.260594	0.260586	0.260314	0.260228
x114b	0.196518	0.196457	0.184122	0.178825	0.166587

Table 7.30: FIR: x113-x114b predicted values, epochs 6-10

PointID/Epoch	**1**	**2**	**3**	**4**	**5**
x113	0.696520	0.696519	0.696535	0.696547	0.696551
x113a	0.652594	0.652591	0.652592	0.652594	0.652594
x113b	0.685157	0.685149	0.685044	0.684717	0.684515
x114a	0.261142	0.261035	0.260915	0.260792	0.260667
x114b	0.208490	0.207432	0.205874	0.203526	0.201243

Table 7.31: RFIR: x113-x114b predicted values, epochs 1-5

PointID/Epoch	**6**	**7**	**8**	**9**	**10**
x113	0.696550	0.696551	0.696549	0.696546	0.696540
x113a	0.652593	0.652595	0.652594	0.652594	0.652594
x113b	0.684270	0.683561	0.683525	0.683406	0.683953
x114a	0.260565	0.260435	0.260307	0.260186	0.260169
x114b	0.198971	0.196620	0.195317	0.193700	0.204378

Table 7.32: RFIR: x113-x114b predicted values, epochs 6-10

PointID/Epoch	**1**	**2**	**3**	**4**	**5**
y14	0.917043	0.916964	0.916559	0.916278	0.916229
y22	0.757800	0.760840	0.758149	0.755645	0.767567
y24	0.569074	0.569096	0.568932	0.568939	0.568976
y29	0.538858	0.539048	0.538749	0.539099	0.538661
y32	0.974393	0.974360	0.974641	0.974668	0.974745

Table 7.33: FIR: y14-y32 predicted values, epochs 1-5

PointID/Epoch	**6**	**7**	**8**	**9**	**10**
y14	0.916231	0.916232	0.916536	0.916572	0.916603
y22	0.779604	0.779587	0.779570	0.767572	0.769018
y24	0.568976	0.568975	0.569021	0.569015	0.569013
y29	0.539060	0.539063	0.539066	0.538905	0.538846
y32	0.974758	0.974753	0.974749	0.974629	0.974629

Table 7.34: FIR: y14-y32 predicted values, epochs 6-10

PointID/Epoch	**1**	**2**	**3**	**4**	**5**
y14	0.916874	0.916868	0.916873	0.916871	0.916875
y22	0.773121	0.772798	0.772194	0.772993	0.772995
y24	0.569057	0.569113	0.569150	0.569143	0.569174
y29	0.537631	0.537715	0.537796	0.537800	0.537887
y32	0.974962	0.974964	0.974965	0.974969	0.974967

Table 7.35: RFIR: y14-y32 predicted values, epochs 1-5

PointID/Epoch	**6**	**7**	**8**	**9**	**10**
y14	0.916875	0.916874	0.916868	0.916873	0.916869
y22	0.774116	0.774626	0.774065	0.775766	0.775587
y24	0.569180	0.569184	0.569234	0.569258	0.569207
y29	0.537852	0.537872	0.538548	0.538790	0.538346
y32	0.974971	0.974967	0.974968	0.974974	0.974969

Table 7.36: RFIR: y14-y32 predicted values, epochs 6-10

PointID/Epoch	**1**	**2**	**3**	**4**	**5**
y105	0.605369	0.605413	0.605202	0.605098	0.604869
y110	0.182871	0.176617	0.172301	0.169931	0.169092
y110a	0.944556	0.944517	0.944515	0.944452	0.944388
y110b	0.185031	0.186466	0.187756	0.188202	0.188840

Table 7.37: FIR: y105-y110b predicted values, epochs 1-5

PointID/Epoch	**6**	**7**	**8**	**9**	**10**
y105	0.604623	0.604621	0.604620	0.604896	0.604866
y110	0.169092	0.169124	0.158549	0.153316	0.146046
y110a	0.944388	0.944388	0.944456	0.944455	0.944454
y110b	0.188837	0.188838	0.187875	0.188111	0.188316

Table 7.38: FIR: y105-y110b predicted values, epochs 6-10

PointID/Epoch	**1**	**2**	**3**	**4**	**5**
y105	0.604805	0.604817	0.604776	0.604722	0.604696
y110	0.182655	0.182138	0.181590	0.179789	0.178263
y110a	0.944258	0.944253	0.944243	0.944241	0.944230
y110b	0.178709	0.180469	0.183350	0.186616	0.189301

Table 7.39: RFIR: y105-y110b predicted values, epochs 1-5

PointID/Epoch	**6**	**7**	**8**	**9**	**10**
y105	0.604644	0.604606	0.604611	0.604584	0.604647
y110	0.175618	0.172858	0.170388	0.169782	0.184834
y110a	0.944214	0.944206	0.944195	0.944182	0.944198
y110b	0.192079	0.197378	0.201930	0.207221	0.206082

Table 7.40: RFIR: y105-y110b predicted values, epochs 6-10

PointID/Epoch	**1**	**2**	**3**	**4**	**5**
y111	0.915823	0.915844	0.915680	0.915471	0.915379
y111a	0.799528	0.799564	0.799290	0.799196	0.799374
y111b	0.866994	0.866938	0.866786	0.866734	0.866721
y112a	0.697977	0.698243	0.698830	0.698347	0.698325
y112b	0.786308	0.786090	0.785818	0.785677	0.785610

Table 7.41: FIR: y111-y112b predicted values, epochs 1-5

PointID/Epoch	**6**	**7**	**8**	**9**	**10**
y111	0.915378	0.915377	0.915495	0.915478	0.915467
y111a	0.799376	0.799381	0.789094	0.788661	0.787387
y111b	0.866706	0.866707	0.866710	0.866779	0.866781
y112a	0.698327	0.698331	0.698118	0.698209	0.698215
y112b	0.785611	0.785602	0.785822	0.785878	0.785856

Table 7.42: FIR: y111-y112b predicted values, epochs 6-10

PointID/Epoch	**1**	**2**	**3**	**4**	**5**
y111	0.915279	0.915281	0.915272	0.915259	0.915249
y111a	0.823680	0.823014	0.822970	0.822852	0.822897
y111b	0.866773	0.866774	0.866772	0.866771	0.866771
y112a	0.698501	0.698482	0.698489	0.698488	0.698464
y112b	0.785810	0.785813	0.785809	0.785816	0.785818

Table 7.43: RFIR: y111-y112b predicted values, epochs 1-5

PointID/Epoch	**6**	**7**	**8**	**9**	**10**
y111	0.915255	0.915251	0.915246	0.915239	0.915248
y111a	0.822922	0.822927	0.823037	0.823110	0.820170
y111b	0.866773	0.866776	0.866780	0.866783	0.866783
y112a	0.698465	0.698501	0.698498	0.698518	0.698514
y112b	0.785829	0.785821	0.785820	0.785817	0.785822

Table 7.44: RFIR: y111-y112b predicted values, epochs 6-10

PointID/Epoch	1	2	3	4	5
y113	0.615920	0.615961	0.615654	0.615498	0.615607
y113a	0.705838	0.705929	0.705615	0.706196	0.707000
y113b	0.600332	0.600270	0.600129	0.600036	0.600031
y114a	0.253340	0.253675	0.253993	0.254029	0.254449
y114b	0.252665	0.252876	0.253037	0.253204	0.253263

Table 7.45: FIR: y113-y114b predicted values, epochs 1-5

PointID/Epoch	6	7	8	9	10
y113	0.615614	0.615622	0.615527	0.615483	0.615456
y113a	0.706897	0.706892	0.706892	0.706603	0.706810
y113b	0.600026	0.600025	0.600156	0.600133	0.600129
y114a	0.254243	0.254242	0.254245	0.254106	0.254202
y114b	0.253267	0.253307	0.254477	0.255568	0.256986

Table 7.46: FIR: y113-y114b predicted values, epochs 6-10

PointID/Epoch	1	2	3	4	5
y113	0.615936	0.615967	0.615988	0.615943	0.615861
y113a	0.706707	0.706711	0.706711	0.706666	0.706675
y113b	0.600130	0.600115	0.600112	0.600092	0.600094
y114a	0.250702	0.250516	0.250583	0.251923	0.252283
y114b	0.252770	0.252738	0.252795	0.252778	0.252771

Table 7.47: RFIR: y113-y114b predicted values, epochs 1-5

PointID/Epoch	6	7	8	9	10
y113	0.615812	0.615805	0.615884	0.615827	0.615857
y113a	0.706605	0.706597	0.706601	0.706562	0.706600
y113b	0.600101	0.600124	0.600142	0.600152	0.600151
y114a	0.252799	0.254430	0.255793	0.256319	0.248967
y114b	0.252780	0.252928	0.253041	0.253043	0.252978

Table 7.48: RFIR: y113-y114b predicted values, epochs 6-10

7.13 Comparison to Related Results

In [26], a GNRR network outperformed all other networks in deformation monitoring of high-rise buildings. The network was trained using both the Gradient Descent (GD), and the Levenberg-Marquardt algorithms (LM) and the results can be seen in Figure 7.16. In another paper [51], a simple Multi-Layer Perceptron (MLP) network was trained using a variety of algorithms including Levenberg-Marquardt backpropagation, resilient backpropagation, scaled conjugate gradient backpropagation, and GD with adaptive learning rate backpropagation. The resilient backgropagation algorithm outperformed all the others with a mean absolute error of just 1%, just like the GNRR network in [26]. In both papers, the historical data covered measurements which took part over 7 years or more where the points were measured more frequently, particularly in [51] where the object of interest was a Byzantine Church "Magalie Panagia", which has been characterised as a heritage site by UNESCO. The church is undergoing settlements because of unfavourable ground, composed mainly of clay, silt, and peat, and is most likely continued to be monitored due to its historical significance.

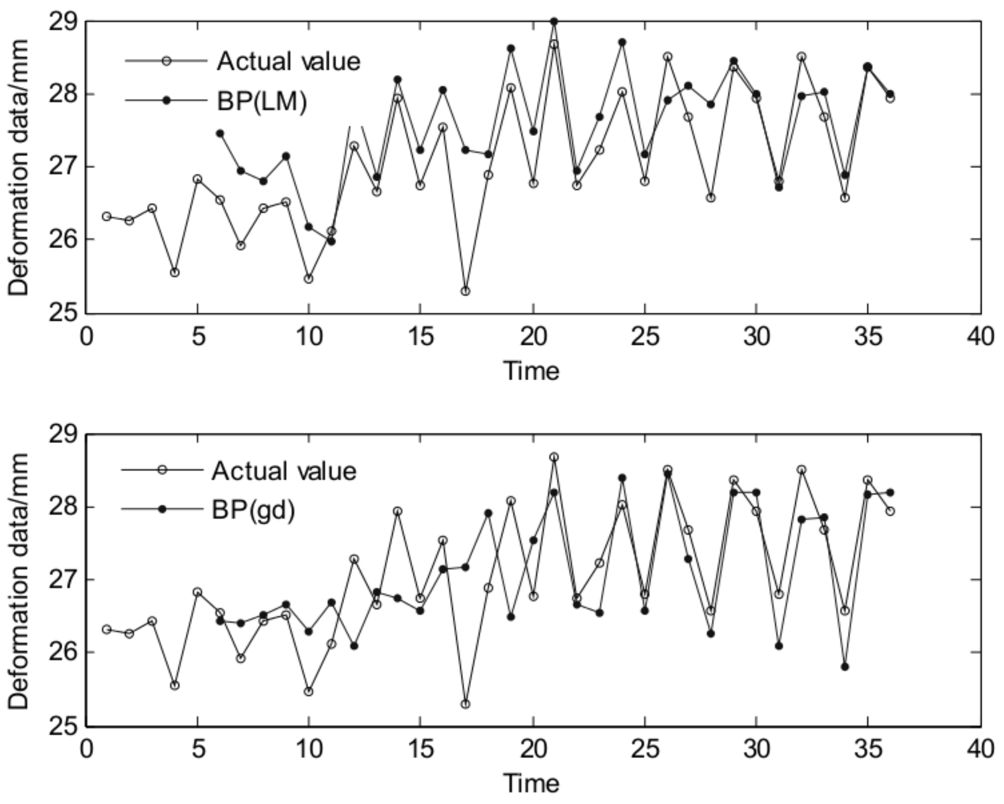


Figure 7.16: Comparison deformation cure between BP(LM) and BP(GD) [26]

CHAPTER 8

Conclusions and Future Work

8.1 Summary

In this thesis, methods and techniques for monitoring of deformations present in residential buildings are proposed. The methods consist of image processing, computer vision, knowledge based systems, and artificial neural networks, which differ from conventional methods such as the Finite Element Method. The computer vision techniques described in this thesis were not able to be fully tested in real life, due to the lack of data. The image processing techniques perform well under certain conditions, and assumptions, such as the lighting and the angle at which the picture had been taken. The merging algorithm together with the variation of the RANSAC algorithm deal well with inconsistent points. And finally the predictions made by the FIR neural networks represent a novel way of monitoring deformations. The results could have been better, if more historical data had been gathered. The method also presents a new use for neural networks in geodesy, other than their ability to minimize measurement errors and perform quality checks.

8.2 Results

Image processing technique used to locate windows in facades and fit a bounding box has been developed and implemented in Chapter 2. A variant of RANSAC together with a merging algorithm which represents deformations in a rule based system had also been developed in Chapters 3, and 4. All implementations have been done using lightweight methodology. FIR ANN which are widely known for their forecasting power, had been used for the purpose of predicting data gathered from geodetic measurements in project 'Prokop' in Chapter 7. Forecasted values have shown that the FIR neural networks were able to predict the trend of the movement with a high success rate. The results have shown that the movements are of highest value at the top of the building, and lowest at the base.

8.3 Future Work

The techniques for videometric analysis described in this thesis had never been put to use in real life, due to the fact that it was hard finding a database of images of facades which had been taken on a monthly or daily basis. And even if such images could have been collected, due to the fact that the movements are measured in millimeters, their application and usage would have been minimal. Because the movements are of highest value at the top of the residential building, the best case scenario to test the algorithm would involve a high-rise residential building. Results found in [57] show that the Burj Khalifa tower is undergoing highest recorded deformations (measured in cm, and not mm). For experimental use, the easiest way to gather the data would be by gathering them from twin towers, by placing two separate cameras on both objects thus covering most regions of interest from each side. The results from these experiments would then be passed on to the FIR or RFIR neural network to measure the behaviour of deformations that are being present. Other usage would include monitoring of historical buildings, like the “Magalie Panagia” [51] that are of great historical significance and that will always have the need for deformation monitoring.

Bibliography

[1] `http://www.vision.ee.ethz.ch/showroom/zubud/index.en.html`, April 2003.

[2] `http://info.tuwien.ac.at/ingeo/research/FdbV/index.htm`, November 2006.

[3] `http://clipsrules.sourceforge.net/`, June 2015.

[4] H. Ackley, E. Hinton, and J. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

[5] H. Ali, C. Seifert, N. Jindal, L. Paletta, and G. Paar. Window detection in facades. In *14th International Conference on Image Analysis and Processing (ICIAP 2007), 10-14 September 2007, Modena, Italy*, pages 837–842, 2007.

[6] T. Vicovac andr U. Egly, T. Eiter, and D. Rieke-Zapp. Knowledge-based geo-risk assessment for an intelligent measurement system. In *Artificial Intelligence in Theory and Practice III - Third IFIP TC 12 International Conference on Artificial Intelligence, IFIP AI 2010, Held as Part of WCC 2010, Brisbane, Australia, September 20-23, 2010. Proceedings*, pages 215–224, 2010.

[7] D.L. Bailey. and D. Thompson. Developing neural-network applications. In *AI Expert*, volume 5, pages 34–41. Miller Freeman, Inc., San Francisco, CA, USA, June 1990.

[8] J.P. Bigus. *Data Mining with Neural Networks: Solving Business Problems from Application Development to Decision Support.* McGraw-Hill, Inc., Hightstown, NJ, USA, 1996.

[9] B. Billah, M.L. King, R.D. Snyder, and A.B. Koehler. Exponential smoothing model selection for forecasting. *International Journal of Forecasting*, 22(2):239 – 247, March 2006.

[10] G. Bontempi, S.B. Taieb, and Y. Le Borgne. Machine learning strategies for time series forecasting. ULB Institutional Repository 2013/167761, ULB – Universite Libre de Bruxelles, 2013.

[11] F. Bozzano, I. Cipriani, P. Mazzanti, and A. Prestininzi. Displacement patterns of a landslide affected by human activities: insights from ground-based insar monitoring. In *Natural Hazards*, volume 59, pages 1377–1396. Springer Netherlands, 2011.

[12] M. Bussell. *Structures & Construction in Historic Building Conservation: Effects of Induced Movement*, pages 111–139. Blackwell Publishing Ltd, 2008.

[13] W.F. Caspary. *Concepts of Network and Deformation Analysis.* Monograph. School of Surveying, University of New South Wales, 1988.

[14] J. Cech and R. Sára. Windowpane detection based on maximum aposteriori probability labeling. In *Image Analysis - From Theory to Applications. Proceedings of IWCIA 2008 Special Track on Applications, Buffalo, NY, USA, April 7-9, 2008.*, pages 3–11, 2008.

[15] C. Chatfield. *The analysis of time series: an introduction.* CRC Press, Florida, US, 6th edition, 2004.

[16] K. Chmelina, H. Kahmen, T. Eiter, and U. Egly. Heuristische Echtzeit-Fehlererkennung bei Deformationsmessungen während des Tunnelvortriebs. *Zeitschrift fuer Vermessungswesen (ZfV)*, 128(5):333–340, 2003.

[17] K. Chmelina and K. Rabensteiner. Improvement of the safety and profitability of tunnel drives through the use of automated measurement and alarm systems – examples in practice / Verbesserung der Sicherheit und Wirtschaftlichkeit von Tunnelvortrieben durch den Einsatz automatisierter Mess- und Alarmsysteme – Ausfuehrungsbeispiele. In *Geomechanik Tunnelbau*, volume 3, pages 215–224, 2010.

[18] C.R.Jung and R. Schramm. Rectangle detection based on a windowed hough transform. In *XVII Brazilian Symposium on Computer Graphics and Image Processing, (SIBGRAPI 2004) 17-20 October 2004, Curitiba, PR, Brazil*, pages 113–120. IEEE Computer Society, 2004.

[19] J.P. Draye, D. Pavisic, G. Cheron, and G. Libert. Dynamic recurrent neural networks: a dynamical analysis. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 26(5):692–706, 1996.

[20] R. Drossu and Z. Obradovic. Rapid design of neural networks for time series prediction. *IEEE Comput. Sci. Eng.*, 3(2):78–89, June 1996.

[21] S. Du, J. Zhang, Z. Deng, and J. Li. A neural network based intelligent method for mine slope surface deformation prediction considering the meteorological factors. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, 12(4):2882–2889, 2014.

[22] J.L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[23] M.B. Feilden. *Conservation of Historic Buildings.* Technical studies in the arts, archeology and architecture. Architectural Press, 2003.

[24] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[25] W. Förstner and E. Guelch. A fast operator for detection and precise location of distinct points, corners and centres of circular features. In *ISPRS Intercommission Workshop, Interlaken*, pages 281–305, 1987.

[26] C. Gao, X. Cui, and X. Hong. Study on the applications of neural networks for processing deformation monitoring data. In *Applied Mechanics and Materials*, volume 501-504, pages 2149–2153, 2014.

[27] C. Grosan and A. Abraham. *Intelligent Systems: A Modern Approach.* Springer Publishing Company, Incorporated, 1st edition, 2011.

[28] M. Hallas and G. Dorffner. A comparative study on feedforward and recurrent neural networks in time series prediction using gradient descent learning. In R. Trappl, editor, *Cybernetics and Systems 98, Proceedings of 14th European Meeting on Cybernetics and Systems Research, Vienna*, pages 644–647, 1998.

[29] C. Harris and M. Stephens. A combined corner and edge detection. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[30] Simon Haykin. *Neural Networks: A Comprehensive Foundation.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 1998.

[31] B. Heck. Geometrische Analyse und Interpretation von Deformationen Geodätischer Netze. In W. Welsch, editor, *Deformationsanalysen*, volume 9, pages 299–238. Neubiberg : Hochschule der Bundeswehr München, April 1983.

[32] Bill G. Horne and C. Lee Giles. An experimental comparison of recurrent neural networks. In *Advances in Neural Information Processing Systems 7*, pages 697–704. MIT Press, 1995.

[33] R. Jaeger, S. Kaelber, and M. Oswald. GNSS/LPS based online control and alarm system (GOCA) - mathematical models and technical realization of a system for natural and geotechnical deformation monitoring and hazard prevention. In S. Fernando and G.J. AntonioJ, editors, *Geodetic Deformation Monitoring: From Geophysical to Engineering Roles*, volume 131 of *International Association of Geodesy Symposia*, pages 293–303. Springer Berlin Heidelberg, 2006.

[34] J.O.Katz. Developing neural network forecasts for trading. *Stocks & Commodities*, 10(4):160–168, 1992.

[35] M. I. Jordan. Serial order: A parallel distributed processing approach. Technical Report ICS Report 8604, Institute for Cognitive Science, University of California, San Diego, 1986.

[36] Egger K. Terrestrial measurement of deformation. In *Swiss National Committee on Large Dams – Working Group on Dam Monitoring, The Geodetic and Photogrammetric Deformation Measurement of Dams, published on the occasion of the 19th ICOLD Congress, Florence, Italy*, pages 15–23, 1997.

[37] Egger K. and Keller. W. New instruments and their applications for geodetic deformation measurements on dams. In *Transactions, 12th International Congress on Large Dams, Mexico, Reprinted by Kern & Co AG (now Leica-Geosystems), Aarau, Switzerland*, 1987.

[38] J. Kämpf and D. Robinson. Optimisation of urban energy demand using an evolutionary algorithm. In *Eleventh International IBPSA Conference, Glasgow, Sctotland*, pages 668–673, 2009.

[39] H. Kantz and T. Schreiber. *Nonlinear Time Series Analysis.* Cambridge nonlinear science series. Cambridge University Press, 2nd edition, 2004.

[40] Nikola K. Kasabov. *Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering.* MIT Press, Cambridge, MA, USA, 1st edition, 1996.

[41] Karlo Kauko and Peter Palmroos. The Delphi method in forecasting financial markets— An experimental study. *International Journal of Forecasting*, 30(2):313–327, 2014.

[42] T. Koskela, M. Lehtokangas, J. Saarinen, and K. Kaski. Time series prediction with multilayer perceptron, FIR and elman neural networks. In *Proceedings of the World Congress on Neural Networks*, pages 491–496. Press, 1996.

[43] B. Kosko. Bidirectional associative memories. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):49–60, 1988.

[44] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 1150–, Washington, DC, USA, 1999. IEEE Computer Society.

[45] D. P. Mandic and J. Chambers. *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability.* John Wiley & Sons, Inc. New York, NY, USA, August 2001.

[46] J.R. McDonald. *Intelligent knowledge based systems in electrical power engineering.* Springer US, 1997.

[47] L. Medsker and L.C. Jain. *Recurrent Neural Networks: Design and Applications.* International Series on Computational Intelligence. Taylor & Francis, 1999.

[48] M. Miljanovic, T. Eiter, and U. Egly. Detection of windows in facades using image processing algorithms. *Indian Journal of Computer Science and Engineering*, 3(4):539–547, 2010.

[49] M.Miljanovic. Comparative analysis of recurrent and finite impulse response neural networks in time series prediction. *Indian Journal of Computer Science and Engineering*, 3(1):180–191, March 2012.

[50] T. Ninkov. Deformaciona analiza i njena praktična primena (deformation analysis and its practical use). In *Geodetski List*, volume 39, pages 167–178, 1985.

[51] G. Pantazis and E.G. Alevizakou. The use of artificial neural networks in predicting vertical displacements of structures. *International Journal of Applied Science and Technology*, 3(5), 2013.

[52] J.R. Parker. *Algorithms for Image Processing and Computer Vision.* Wiley computer publishing. John Wiley & Sons, 1996.

[53] H. Pelzer. Zur analyse geodätischer deformationsmessungen. *Deutsche Geodätische Kommission, München*, C(164), 1971.

[54] M. Qian. *Neural Network Learning for Time-Series Predictions Using Constrained Formulations.* PhD thesis, University of Illinois at Urbana-Champaign, 2005.

[55] A. Reiterer, M. Lehmann, M.Miljanovic, Haider Ali, G. Paar, Uwe Egly, T. Eiter, and H. Kahmen. Deformation monitoring using a new kind of optical 3D measurement system: Components and perspectives. In 13th *FIG Symposium on Deformation Measurement and Analysis, 4th IAG Symposium on Geodesy for Geotehnical and Structual Engineering, LNEC, Lisbon*, volume 3, pages 12–15, May 2008.

[56] A. Reiterer, M.Lehmann, M.Miljanovic, Haider Ali, Gerhard Paar, Uwe Egly, T. Eiter, and H.Kahmen. A 3D optical deformation measurement system supported by knowledge based and learning techniques. In *Journal of Applied Geodesy*, pages 1–13, 2009.

[57] G. Russo, V. Abagnara, H. Poulos, and J. Small. Re-assessment of foundation settlements for the Burj Khalifa, Dubai. *Acta Geotechnica*, 8(1):3–15, 2013.

[58] Tep Sastri. A State Space Modeling Approach for Time Series Forecasting. *Management Science*, 31(11):1451–1470, November 1985.

[59] R. Szeliski. *Computer Vision: Algorithms and Applications.* Texts in Computer Science. Springer, 2010.

[60] Georg Thimm, Perry Moerland, and Emile Fiesler. The interchangeability of learning rate and gain in backpropagation neural networks. *Neural Computation*, 8(2):451–460, 1996.

[61] Helmut Thome. A Box-Jenkins Approach to Modeling Outliers in Time Series Analysis. *Sociological Methods & Research*, 23(4):442–478, May 1995.

[62] A. Tierra, R. Dalazoana, and S. De Freitas. Using an artificial neural network to improve the transformation of coordinates between classical geodetic reference frames. *Computers and Geosciences*, 34(3):181–189, 2008.

[63] J. van Cranenbroeck. Continuous beam deflection monitoring using precise inclinometers. In *Strategic Integration of Surveying Services*. FIG Working Week, Hong Kong SAR, China, May 2007.

[64] R.C. Veltkamp. Shape matching: Similarity measures and algorithms. In *2001 International Conference on Shape Modeling and Applications (SMI 2001), 7-11 May 2001, Genoa, Italy*, pages 188–199. IEEE Computer Society, 2001.

[65] B. Wah and M. Qian. Constraint-based neural network learning for time series predictions. In *Intelligent Technologies for Information Analysis*, pages 409–431. Springer Berlin Heidelberg, 2004.

[66] E. Wan. An efficient Algorithm for Finite Impulse Response Neural Networks. In *1990 Connectionist Models Summer School*, pages 131–140. Morgan Kaufmann, San Mateo, CA, USA, 1990.

[67] Ronald J. Williams and David Zipser. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation*, 1(2):270–280, 1989.

[68] `ftp://ftp.ashtech.com/UtilitySoftware/MissionPlanning/Manuals/600071Rev.DMulti-siteMissionPlanning.pdf`, May 2006.

[69] Pablo Zegers and Malur K. Sundareshan. Periodic motions, mapping ordered sequences, and training of dynamic neural networks to generate continuous and discontinuous trajectories. In *International Joint Conference on Neural Networks (IJCNN) (3)*, pages 9–14, 2000.

APPENDIX A

Geodetic Control Network

Figure A.1 shows the full geodetic controll network, with all the measured points in the local coordination system including ones found in residential buildings, around the tunnel, as well as abutment diaphragms which were the main points of interest in this project.

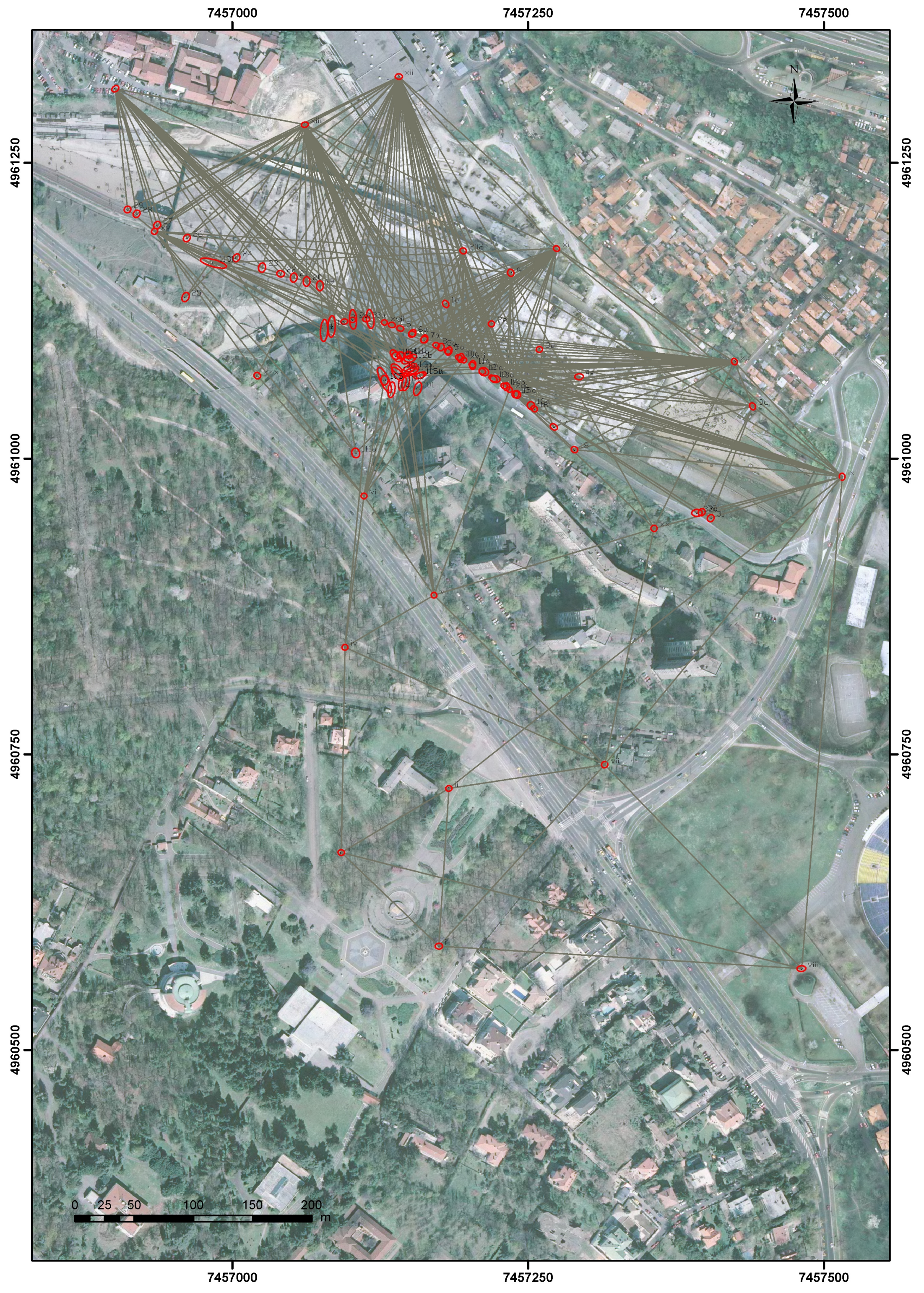


Figure A.1: Geodetic Control Network used in "Prokop"

APPENDIX B

Histogram Mean Filtering

```
function newimage = bwconv(name)
[Hh, Vv] = hist_hv(name);
[HL_min, HL_max, VL_min, VL_max] = convol1d(Hh,Vv);

theimage = imread(name);
orig = theimage;

gaussianFilter = fspecial('gaussian', [10,10], 1);
theimage = imfilter(theimage, gaussianFilter, 'symmetric', 'conv');

newimage = zeros(size(theimage,1),size(theimage,2));

HL_n = HL_min(:,2);
HL_x = HL_max(:,2);
HL = [];
for i = 1 : length(HL_n)
    HL = [HL, round((HL_n(i)+HL_x(i))./2)];
end
HL = [HL, HL_x(length(HL_x))];

HL = HL_n;
HL = [1;HL];

VL_n = VL_min(:,2);
VL_x = VL_max(:,2);
VL = [];
for i = 1 : length(VL_n)
    VL = [VL, round((VL_n(i)+VL_x(i))./2)];
end
VL = [VL, VL_x(length(VL_x))];

HL = HL_n;
```

```
VL = [1,VL];


%Filter out small local minimums
HL_temp = HL(1);
for i = 2 : length(HL)
    if (HL(i)-HL(i-1)>50) HL_temp = [HL_temp, HL(i)]; end
end
HL = [1,HL_temp, size(theimage,1)];

VL_temp = VL(1);
for i = 2 : length(VL)
    if (VL(i)-VL(i-1)>50) VL_temp = [VL_temp, VL(i)]; end
end
VL = [1,VL_temp, size(theimage,2)];


MeanSq = [];
for i = 1 : length(VL)-1
    for j = 1 : length(HL)-1

        subimage = theimage(HL(j):HL(j+1),VL(i):VL(i+1));

        j_mid = round((HL(j)+HL(j+1))./2);
        i_mid = round((VL(i)+VL(i+1))./2);
        subimage1 = theimage(HL(j):j_mid, VL(i):i_mid);
        subimage2 = theimage(HL(j):j_mid, i_mid:VL(i+1));
        subimage3 = theimage(j_mid:HL(j+1), VL(i):i_mid);
        subimage4 = theimage(j_mid:HL(j+1), i_mid:VL(i+1));
        M = mean(mean(subimage));

        M1 = (mean(mean(subimage1))+M)./2-12;
        M2 = (mean(mean(subimage2))+M)./2-12;
        M3 = (mean(mean(subimage3))+M)./2-12;
        M4 = (mean(mean(subimage4))+M)./2-12;

        for x = VL(i) : VL(i+1)
            for y = HL(j) : HL(j+1)
                if (x<i_mid && y<j_mid && theimage(y,x)<M1)
                newimage(y,x)=1; end
                if (x<i_mid && y<j_mid && theimage(y,x)>=M1)
                newimage(y,x)=0; end
                if (x>=i_mid && y<j_mid && theimage(y,x)<M2)
                newimage(y,x)=1; end
                if (x>=i_mid && y<j_mid && theimage(y,x)>=M2)
                newimage(y,x)=0; end
                if (x<i_mid && y>=j_mid && theimage(y,x)<M3)
                newimage(y,x)=1; end
                if (x<i_mid && y>=j_mid && theimage(y,x)>=M3)
                newimage(y,x)=0; end
                if (x>=i_mid && y>=j_mid && theimage(y,x)<M4)
                newimage(y,x)=1; end
                if (x>=i_mid && y>=j_mid && theimage(y,x)>=M4)
                newimage(y,x)=0; end

            end
        end

    end
end


SE = strel('square',30);        % structuring element
SE1 = strel('disk',15);
newimage = imdilate(newimage,SE1);
newimage = imerode(newimage,SE);       % remove spurs


function [Hh, Vv] = hist_hv(name)

%loaded = importdata(name,'jpg');
%theimage = myrgb2gray(loaded);


theimage = imread(name);

%gaussianFilter = fspecial('gaussian', [3,3], 1);
%theimage = imfilter(theimage, gaussianFilter, 'symmetric', 'conv');

%theimage = myim2bw(theimage,0.4);
%imshow(theimage);
%calculate horizontal histogram
[W,H] = size(theimage);
Hh = []; Vv = [];
for i = 1 : W
    Hh = [Hh, sum(theimage(i,:))];
end
%calculate vertical histogram
for i = 1 : H
    Vv = [Vv, sum(theimage(:, i))];
end



%%% Histogram Mean Filtering
%%% Algorithm for window detection
%%% developed by Milos Miljanovic

function ROI = HMF(name)

origimage = imread(name);
theimage = bwconv(name);

[W,H] = size(theimage);
Hh = [];
Vv = [];
```

```
%calculate horizontal histogram
for i = 1 : W
    Hh = [Hh, sum(theimage(i,:))];
end
%calculate vertical histogram
for i = 1 : H
    Vv = [Vv, sum(theimage(:, i))];
end

Ph = [];
Kh = [];
meanVv = mean(Vv);
for i = 1 : length(Vv)
    if (Vv(i)<meanVv) Ph = [Ph, i]; end
end

meanHh = mean(Hh);
for j = 1 : length(Hh)
    if (Hh(j)<meanHh) Kh = [Kh, j]; end
end

ROI = winplot(Ph,Kh,origimage);


function newimage = imnorm(theimage)

%loaded = importdata(name,'jpg');
%theimage = myrgb2gray(loaded);

hist_im = sum(theimage);
hist_peak = hist_im./size(theimage,1);

%a = 100;
%b = 110;
a = 0;
b = 255;
c = 0.05*max(hist_peak);
d = 0.95*max(hist_peak);
newimage = theimage;
for i = 1 : size(theimage,1)
    for j = 1 : size(theimage,2)
        newimage(i,j) = (theimage(i,j)-c) * ((b-a)/(d-c))+a;
    end
end

%imshow(newimage);


function y = unsharp(name)

imrgb = imread(name);
im = im2double(imrgb);
```

```
g = fspecial('gaussian',25,4);
%g = fspecial('log',25,4);
imblur = conv2(im,g,'same');
%imagesc([im imblur])
%imagesc([im im+.4*(im-imblur)])
y = im+.4*(im-imblur);

imshow(y);


function ROI = winplot(Ph,Kh,theimage)
    if (Ph(1)==1) Ph_x = [1];
    else Ph_x = [1, Ph(1)]; end

    if (Kh(1)==1) Kh_y = [1];
    else Kh_y = [1, Kh(1)]; end

    for i = 2 : length(Ph)
        if (Ph(i)-Ph(i-1)>5) Ph_x = [Ph_x, Ph(i), Ph(i-1)]; end
    end
    for j = 2 : length(Kh)
        if (Kh(j)-Kh(j-1)>5) Kh_y = [Kh_y, Kh(j), Kh(j-1)]; end
    end

    Ph_x = sort([Ph_x, Ph(length(Ph)),size(theimage,2)]);
    Kh_y = sort([Kh_y, Kh(length(Kh)),size(theimage,1)]);

    newimg = theimage;
    for i = 1 : length(Ph_x)
        for j = 1 : size(newimg,1)
            %if (i>1 & Ph_x(i)-Ph(i-1)>10)
                newimg(j,Ph_x(i))=255;
            %end
        end
    end
    for i = 1 : length(Kh_y)
        for j = 1 : size(newimg,2)
%             if (i>1 & Kh(i)-Kh(i-1)>10)
               newimg(Kh_y(i),j)=255;
%             end
        end
    end

    ROI = [];
    for i = 2 : length(Ph_x)-1
        for j = 2 : length(Kh_y)-1
            ROI = [ROI; Ph_x(i)-10, Kh_y(j)-10, Ph_x(i)+10, Kh_y(j)+10];
        end
    end

%%%%% VISUALISATION
%    imshow(newimg);
%    hold on;
```

```
%
%    ROI = [];
%    for i = 2 : length(Ph_x)-1
%        for j = 2 : length(Kh_y)-1
%            plot(Ph_x(i)-10,Kh_y(j)-10,'--rs');
%            plot(Ph_x(i)+10,Kh_y(j)+10,'--rs');
%
%            hold on;
%            ROI = [ROI; Ph_x(i)-10, Kh_y(j)-10, Ph_x(i)+10, Kh_y(j)+10];
%        end
%    end
```

APPENDIX C

Merging Points of Interest into ROI

```
; ROI Variable is defined by the following parameters:
; id: id number of the ROI
; WA_x: Wide Angle x coordinate of the ROI
; WA_y: Wide Angle y coordinate of the ROI
; tx: translation around the x-axis
; ty: translation around the y-axis
; tz: translation around the z-axis
; alpha: rotation around the x-axis
; beta: rotation around the y-axis
; gamma: rotation around the z-axis

(deftemplate ROI
    (slot id (type NUMBER))
    (slot WA_x (type NUMBER))
    (slot WA_y (type NUMBER))
    (slot tx)
    (slot ty)
    (slot tz)
    (slot alpha)
    (slot beta)
    (slot gamma)
)

; This is a template used to describe which ROIs are adjacent
; to each other: id1 represents the ROI to the left, whereas id2
; represents ROI to the right
(deftemplate adjacent
    (slot id1)
    (slot id2)
)
```

```
; This template is used to signify whether there is a deformation
; present at ROI with an id ?id. The field slot is used to store the
; appropriate deformation
(deftemplate present
    (slot id)
    (slot field)
)

; This template is used to signify whether there is no deformation at
; ROI with an id ?id. The field slot is used to store the appropriate
; deformation
(deftemplate absent
    (slot id)
    (slot field)
)

; This template is used to signify whether a deformation at ROI with
; an id ?id is positive. The field slot is used to store the
; appropriate deformation
(deftemplate positive
    (slot id)
    (slot field)
)

; This template is used to signify whether a deformation at ROI with an
; id ?id is negative. The field slot is used to store the appropriate
; deformation
(deftemplate negative
    (slot id)
    (slot field)
)

; This function will return TRUE if the coordinates of ROI id1 (x1,y1)
; and ROI id2 (x2,y2) are close to each other
(deffunction adj (?id1 ?id2 ?x1 ?y1 ?x2 ?y2)
(and (!= ?id1 ?id2)
(<= (- ?x2 ?x1) 100)
(>= (- ?x2 ?x1) 0)
(<= (- ?y2 ?y1) 100)
(>= (- ?y2 ?y1) 0))
)

; This function will return true, if the fuzzy value which is forwarded
; to the KBS by the system is negative
(deffunction neg (?a)
(or (= (str-compare ?a "nsg") 0)
    (= (str-compare ?a "ng") 0)
    (= (str-compare ?a "nm") 0)
    (= (str-compare ?a "nk") 0)
    (= (str-compare ?a "nsk") 0))
)
```

```
; This function will return true, if the fuzzy value which is forwarded
; to the KBS by the system is positive
(deffunction pos (?a)
(or (= (str-compare ?a "sk") 0)
    (= (str-compare ?a "k") 0)
    (= (str-compare ?a "m") 0)
    (= (str-compare ?a "g") 0)
    (= (str-compare ?a "sg") 0))
)

; ----------------------------------------------------------------------
; The following rules will generate instances of facts which will be later
; used by the consistency check rules. These facts are used to simplify
; constant checks such as whether to see there is a deformation present,
; or if two ROIs are adjacent to each other. These facts are crucial when
; determing if there are discrepancies in the calculation of the
; deformation parameters tx, ty, tz, alpha, beta, gamma
; ----------------------------------------------------------------------


; This rule will generate instances of facts which describe which ROIs
; are adjacent to each other
(defrule find-adjacent
(ROI (id ?id1) (WA_x ?WA_x1) (WA_y ?WA_y1))
(ROI (id ?id2) (WA_x ?WA_x2) (WA_y ?WA_y2))
=>
(if (adj ?id1 ?id2 ?WA_x1 ?WA_y1 ?WA_x2 ?WA_y2)
  then
  (assert (adjacent (id1 ?id1) (id2 ?id2))))
)

; The following rules will generate instances of facts which tell the user
; which deformations are present or absent in the ROIs
(defrule present-tx
(ROI (id ?id) (tx ?tx))
=>
(if (!= (str-compare ?tx "null") 0)
   then
   (assert (present (id ?id) (field "tx")))
   else
   (assert (absent (id ?id) (field "tx"))))
)

(defrule present-ty
(ROI (id ?id) (ty ?ty))
=>
(if (!= (str-compare ?ty "null") 0)
   then
   (assert (present (id ?id) (field "ty")))
   else
   (assert (absent (id ?id) (field "ty"))))
)
```

```
(defrule present-tz
(ROI (id ?id) (tz ?tz))
=>
(if (!= (str-compare ?tz "null") 0)
   then
   (assert (present (id ?id) (field "tz")))
   else
   (assert (absent (id ?id) (field "tz"))))
)

(defrule present-alpha
(ROI (id ?id) (alpha ?alpha))
=>
(if (!= (str-compare ?alpha "null") 0)
   then
   (assert (present (id ?id) (field "alpha")))
   else
   (assert (absent (id ?id) (field "alpha"))))
)

(defrule present-beta
(ROI (id ?id) (tx ?beta))
=>
(if (!= (str-compare ?beta "null") 0)
   then
   (assert (present (id ?id) (field "beta")))
   else
   (assert (absent (id ?id) (field "beta"))))
)

(defrule present-gamma
(ROI (id ?id) (gamma ?gamma))
=>
(if (!= (str-compare ?gamma "null") 0)
   then
   (assert (present (id ?id) (field "gamma")))
   else
   (assert (absent (id ?id) (field "gamma"))))
)

; The following rules will generate instances of facts which tell the
; user which deformation parameters are positive or negative
(defrule sign-tx
(ROI (id ?id) (tx ?tx))
=>
(if (and (pos ?tx) (!= (str-compare ?tx "null") 0))
   then
   (assert (positive (id ?id) (field "tx")))
   else
   (assert (positive (id ?id) (field "tx"))))
)

(defrule sign-ty
(ROI (id ?id) (ty ?ty))
=>
(if (and (pos ?ty) (!= (str-compare ?ty "null") 0))
   then
   (assert (positive (id ?id) (field "ty")))
   else
   (assert (positive (id ?id) (field "ty"))))
)

(defrule sign-tz
(ROI (id ?id) (tz ?tz))
=>
(if (and (pos ?tz) (!= (str-compare ?tz "null") 0))
   then
   (assert (positive (id ?id) (field "tz")))
   else
   (assert (positive (id ?id) (field "tz"))))
)

(defrule sign-alpha
(ROI (id ?id) (alpha ?alpha))
=>
(if (and (pos ?alpha) (!= (str-compare ?alpha "null") 0))
   then
   (assert (positive (id ?id) (field "alpha")))
   else
   (assert (positive (id ?id) (field "alpha"))))
)

(defrule sign-beta
(ROI (id ?id) (beta ?beta))
=>
(if (and (pos ?beta) (!= (str-compare ?beta "null") 0))
   then
   (assert (positive (id ?id) (field "beta")))
   else
   (assert (positive (id ?id) (field "beta"))))
)

(defrule sign-gamma
(ROI (id ?id) (gamma ?gamma))
=>
(if (and (pos ?gamma) (!= (str-compare ?gamma "null") 0))
   then
   (assert (positive (id ?id) (field "gamma")))
   else
   (assert (positive (id ?id) (field "gamma"))))
)

; ---------------------------------------------------------------------
; ACTUAL RULES used by the consistency checks
; ---------------------------------------------------------------------
; This rule will warn the user if there is ONLY a translation along
; the y-axis The reason why this deformation is not allowed is because
; we  assume that the object we are observing is not moving forward or
; backward. Therefore, this rule will be executed only if ty is present,
; and tz is absent
(defrule incompatible-ty
(absent (id ?id) (field "tz"))
(present (id ?id) (field "ty"))
=>
(printout t "Translation along the y-axis present at " ?id crlf)
)

; This rule will warn the user if there is ONLY a translation along the
; x-axis The reason why this deformation is not allowed is because we
; assume that the object we are observing is not moving left or right.
; Therefore, this rule will be executed only if tx is present, and tz
; is absent
(defrule incompatible-tx
(absent (id ?id) (field "tx"))
(present (id ?id) (field "tx"))
=>
(printout t "Translation along the y-axis present at " ?id crlf)
)

; This rule warns the user if there is a lonely translation (tz) at the
; corner of the building. The ROI in the corner is bordered with three
; other ROIs. If these three ROIs are adjacent to the first ROI, and if
; the first ROI has a translation, whereas the others do not, this rule
; will be executed.
(defrule alone-translation-corner
(adjacent (id1 ?id1) (id2 ?id2))
(adjacent (id1 ?id1) (id2 ?id3))
(present (id ?id1) (field "tz"))
(absent (id ?id2) (field "tz"))
(absent (id ?id3) (field "tz"))
=>
(assert (alone-tz-corner ?id1))
)

; This rule works if there is a lonely rotation (alpha) at the corner of the
; building. Works similarly as the previous rule
(defrule alone-alpha-rotation-corner
(adjacent (id1 ?id1) (id2 ?id2))
(adjacent (id1 ?id1) (id2 ?id3))
(present (id ?id1) (field "alpha"))
(absent (id ?id2) (field "alpha"))
(absent (id ?id3) (field "alpha"))
=>
(assert (alone-alpha-corner ?id1))
)

; This rule works if there is a lonely rotation (beta) at the corner of the
; building. Works similarly as the previous rule
```

```
(defrule alone-beta-rotation-corner
(adjacent (id1 ?id1) (id2 ?id2))
(adjacent (id1 ?id1) (id2 ?id3))
(present (id ?id1) (field "beta"))
(absent (id ?id2) (field "beta"))
(absent (id ?id3) (field "beta"))
=>
(assert (alone-beta-corner ?id1))
)

; This rule works if there is a lonely rotation (gamma) at the corner
; of the building. Works similarly as the previous rule

(defrule alone-gamma-rotation-corner
(adjacent (id1 ?id1) (id2 ?id2))
(adjacent (id1 ?id1) (id2 ?id3))
(present (id ?id1) (field "gamma"))
(absent (id ?id2) (field "gamma"))
(absent (id ?id3) (field "gamma"))
=>
(assert (alone-gamma-corner ?id1))
)

; This rule warns the user if there is a lonely translation (tz) in a
; ROI which  is located either in the middle of the image, or at the
; edge of the picture (but not corner) Necessary because this rule is
; different than the corner-rule, as there are more possible
; combinations of ROIs that are adjacent to the observed ROI with a
; tz deformation

(defrule alone-translation
(adjacent (id1 ?id1) (id2 ?id2))
(adjacent (id1 ?id1) (id2 ?id3))
(adjacent (id1 ?id1) (id2 ?id4))
(adjacent (id1 ?id1) (id2 ?id5))
(adjacent (id1 ?id1) (id2 ?id6))
(present (id ?id1) (field "tz"))
(absent (id ?id2) (field "tz"))
(absent (id ?id3) (field "tz"))
(absent (id ?id4) (field "tz"))
(absent (id ?id5) (field "tz"))
(absent (id ?id6) (field "tz"))
=>
(assert (alone-translation ?id1))
)

; This rule warns the user if there is a lonelz rotation (alpha) in a ROI
; located either in the middle of the image, or at the edge of the picture.
; Works similarly as the previous rule

(defrule alone-x-rotation
(adjacent (id1 ?id1) (id2 ?id2))
```

```
(adjacent (id1 ?id1) (id2 ?id3))
(adjacent (id1 ?id1) (id2 ?id4))
(adjacent (id1 ?id1) (id2 ?id5))
(adjacent (id1 ?id1) (id2 ?id6))
(present (id ?id1) (field "alpha"))
(absent (id ?id2) (field "alpha"))
(absent (id ?id3) (field "alpha"))
(absent (id ?id4) (field "alpha"))
(absent (id ?id5) (field "alpha"))
(absent (id ?id6) (field "alpha"))
=>
(assert (alone-alpha ?id1))
)

; This rule warns the user if there is a lonelz rotation (beta) in a ROI
; located either in the middle of the image, or at the edge of the picture.
; Works similarly as the previous rule

(defrule alone-y-rotation
(adjacent (id1 ?id1) (id2 ?id2))
(adjacent (id1 ?id1) (id2 ?id3))
(adjacent (id1 ?id1) (id2 ?id4))
(adjacent (id1 ?id1) (id2 ?id5))
(adjacent (id1 ?id1) (id2 ?id6))
(present (id ?id1) (field "beta"))
(absent (id ?id2) (field "beta"))
(absent (id ?id3) (field "beta"))
(absent (id ?id4) (field "beta"))
(absent (id ?id5) (field "beta"))
(absent (id ?id6) (field "beta"))
=>
(assert (alone-beta ?id1))
)

; This rule warns the user if there is a lonely rotation (gamma) in a ROI
; located either in the middle of the image, or at the edge of the picture.
; Works similarly as the previous rule


(defrule alone-z-rotation
(adjacent (id1 ?id1) (id2 ?id2))
(adjacent (id1 ?id1) (id2 ?id3))
(adjacent (id1 ?id1) (id2 ?id4))
(adjacent (id1 ?id1) (id2 ?id5))
(adjacent (id1 ?id1) (id2 ?id6))
(present (id ?id1) (field "gamma"))
(absent (id ?id2) (field "gamma"))
(absent (id ?id3) (field "gamma"))
(absent (id ?id4) (field "gamma"))
(absent (id ?id5) (field "gamma"))
(absent (id ?id6) (field "gamma"))
=>
(assert (alone-beta ?id1))
```

```
)

; The following rules are consistency checks for the tz parameters
; Translation along the z-axis is the only allowed translation in the system.
; If tz is present in ROI #1, then the only other deformations which are
; allowed are rotations. However, these rotations need to be compatible by
; the negative tz deformation (sliding down, or settlement). The ROI located
; to the left of the tz deformation has to be a positive rotation (to the
; right, whereas the ROI located to the right has to be a negative rotation.
; Therefore in the following rules, we check whether these consistency checks
; have been violated.

(defrule incompatible-gammatz
(adjacent (id1 ?id1) (id2 ?id2))
(negative (id ?id1) (field "gamma"))
(negative (id ?id2) (field "tz"))
=>
(printout t "Incompatible deformations found (gamma, tz) at " ?id1 " and "
?id2 crlf)
)

(defrule incompatible-betatzty
(adjacent (id1 ?id1) (id2 ?id2))
(negative (id ?id1) (field "beta"))
(negative (id ?id2) (field "tz"))
(negative (id ?id2) (field "ty"))
=>
(printout t "Incompatible deformations found (beta,tz,ty) at " ?id1 " and "
?id2 crlf)
)

(defrule incompatible-alphatztx
(adjacent (id1 ?id1) (id2 ?id2))
(negative (id ?id1) (field "alpha"))
(negative (id ?id2) (field "tz"))
(negative (id ?id2) (field "tx"))
=>
(printout t "Incompatible deformations found (alpha,tz,tx) at " ?id1 " and "
?id2 crlf))

(defrule incompatible-tzgamma
(adjacent (id1 ?id1) (id2 ?id2))
(negative (id ?id1) (field "tz"))
(positive (id ?id2) (field "gamma"))
=>
(printout t "Incompatible deformations found (tz,gamma) at " ?id1 " and "
?id2 crlf))

(defrule incompatible-tzbetaty
(adjacent (id1 ?id1) (id2 ?id2))
(negative (id ?id1) (field "tz"))
(positive (id ?id2) (field "beta"))
(negative (id ?id2) (field "ty"))
=>
(printout t "Incompatible deformations found (tz,beta,ty) at " ?id1 " and "
?id2 crlf))

(defrule incompatible-tzalphatx
(adjacent (id1 ?id1) (id2 ?id2))
(negative (id ?id1) (field "alpha"))
(negative (id ?id2) (field "tz"))
(negative (id ?id2) (field "tx"))
=>
(printout t "Incompatible deformations found (alpha,tz,tx) at " ?id1 " and "
?id2 crlf))

; The following rules will check whether two adjacent ROIs have the
; exact same rotations, which are however rotation in different directions.
; i.e. Rotation to the left is never allowed to be next to a Rotation to the
; right, and vice-versa. This is done for all rotations (alpha, beta, gamma).

(defrule different-alpha
(adjacent (id1 ?id1) (id2 ?id2))
(or (and (positive (id ?id1) (field "alpha")) (negative (id ?id1)
    (field "alpha")))
    (and (negative (id ?id1) (field "alpha")) (positive (id ?id1)
    (field "alpha"))))
=>
(printout t "Two different rotations found along the x-axis at " ?id1 " and "
?id2 crlf))

(defrule different-beta
(adjacent (id1 ?id1) (id2 ?id2))
(or (and (positive (id ?id1) (field "beta")) (negative (id ?id1)
    (field "beta")))
    (and (negative (id ?id1) (field "beta")) (positive (id ?id1)
    (field "beta"))))
=>
(printout t "Two different rotations found along the y-axis at " ?id1 " and "
?id2 crlf))

(defrule different-gamma
(adjacent (id1 ?id1) (id2 ?id2))
(or (and (positive (id ?id1) (field "gamma")) (negative (id ?id1)
    (field "gamma")))
    (and (negative (id ?id1) (field "gamma")) (positive (id ?id1)
    (field "gamma"))))
=>
(printout t "Two different rotations found along the z-axis at " ?id1 " and "
?id2 crlf)
)

; The following rules will check whether the magnitude of a particular
; deformation is growing, or shrinking at a reasonable level. For example, if
; a deformation is very big, or big in one ROI, and then very small, or small
; in a ROI which is next to this one, then a warning should be issued.
```

```
(defrule check-tx-magnitude
(adjacent (id1 ?id1) (id2 ?id2))
(or (and (or (ROI (id ?id1) (tx "sg")) (ROI (id ?id2) (tx "g")))
     (or (ROI (id ?id1) (tx "k")) (ROI (id ?id2) (tx "sk"))))
(and (or (ROI (id ?id1) (tx "nsg")) (ROI (id ?id2) (tx "ng")))
     (or (ROI (id ?id1) (tx "nk")) (ROI (id ?id2) (tx "nsk")))))
=>
(printout t "Magnitude of translation along the x-axis changed drastically
from " ?id1 " to " ?id2 crlf))

(defrule check-ty-magnitude
(adjacent (id1 ?id1) (id2 ?id2))
(or (and (or (ROI (id ?id1) (ty "sg")) (ROI (id ?id2) (ty "g")))
     (or (ROI (id ?id1) (ty "k")) (ROI (id ?id2) (ty "sk"))))
(and (or (ROI (id ?id1) (ty "nsg")) (ROI (id ?id2) (ty "ng")))
     (or (ROI (id ?id1) (ty "nk")) (ROI (id ?id2) (ty "nsk")))))
=>
(printout t "Magnitude of translation along the y-axis changed drastically
from " ?id1 " to " ?id2 crlf))

(defrule check-tz-magnitude
(adjacent (id1 ?id1) (id2 ?id2))
(or (and (or (ROI (id ?id1) (tz "sg")) (ROI (id ?id2) (tz "g")))
     (or (ROI (id ?id1) (tz "k")) (ROI (id ?id2) (tz "sk"))))
(and (or (ROI (id ?id1) (tz "nsg")) (ROI (id ?id2) (tz "ng")))
     (or (ROI (id ?id1) (tz "nk")) (ROI (id ?id2) (tz "nsk")))))
=>
(printout t "Magnitude of translation along the z-axis changed drastically
from " ?id1 " to " ?id2 crlf))

(defrule check-alpha-magnitude
(adjacent (id1 ?id1) (id2 ?id2))
(or (and (or (ROI (id ?id1) (alpha "sg")) (ROI (id ?id2) (alpha "g")))
     (or (ROI (id ?id1) (alpha "k")) (ROI (id ?id2) (alpha "sk"))))
(and (or (ROI (id ?id1) (alpha "nsg")) (ROI (id ?id2) (alpha "ng")))
     (or (ROI (id ?id1) (alpha "nk")) (ROI (id ?id2) (alpha "nsk")))))
=>
(printout t "Magnitude of rotation along the x-axis changed drastically from
" ?id1 " to " ?id2 crlf))

(defrule check-beta-magnitude
(adjacent (id1 ?id1) (id2 ?id2))
(or (and (or (ROI (id ?id1) (beta "sg")) (ROI (id ?id2) (beta "g")))
     (or (ROI (id ?id1) (beta "k")) (ROI (id ?id2) (beta "sk"))))
(and (or (ROI (id ?id1) (beta "nsg")) (ROI (id ?id2) (beta "ng")))
     (or (ROI (id ?id1) (beta "nk")) (ROI (id ?id2) (beta "nsk")))))
=>
(printout t "Magnitude of rotation along the y-axis changed drastically from
" ?id1 " to " ?id2 crlf))

(defrule check-gamma-magnitude
(adjacent (id1 ?id1) (id2 ?id2))
(or (and (or (ROI (id ?id1) (gamma "sg")) (ROI (id ?id2) (gamma "g")))
     (or (ROI (id ?id1) (gamma "k")) (ROI (id ?id2) (gamma "sk"))))
(and (or (ROI (id ?id1) (gamma "nsg")) (ROI (id ?id2) (gamma "ng")))
     (or (ROI (id ?id1) (gamma "nk")) (ROI (id ?id2) (gamma "nsk")))))
=>
(printout t "Magnitude of rotation along the z-axis changed drastically from
" ?id1 " to " ?id2 crlf))
```

APPENDIX D

Deformation Interpretation

```
(defglobal ?*VI* = 0)
(defglobal ?*St_VI* = 0)
(defglobal ?*Lt_VI* = 0)
(defglobal ?*L1* = 0)
(defglobal ?*L2* = 0)
(defglobal ?*x* = 0)
(defglobal ?*D* = 0)

(deffunction ask-question (?allowed-values)
   (bind ?answer (read))
   (if (lexemep ?answer)
       then (bind ?answer (lowcase ?answer)))
   (while (not (member ?answer ?allowed-values)) do
      (bind ?answer (read))
      (if (lexemep ?answer)
          then (bind ?answer (lowcase ?answer))))
   ?answer)


; Structural Behaviour of the Building

;Analysis of the horizontal elements
(defrule horizontal-structural-elements
    (declare (salience 10))
    =>
    (printout t "What is the material used in the horizontal
    structural elements?" crlf)
    (printout t "a) Wood Structure" crlf)
    (printout t "b) Reinforced structure" crlf)
    (printout t "c) Mixed Structure" crlf)
    (bind ?response (ask-question a b c))
    (if (eq ?response a)
    then (bind ?*VI* (+ ?*VI* 6)))
    (if (eq ?response c)
    then (bind ?*VI* (+ ?*VI* 3)))
)

;Analysis of the vertical elements
(defrule vertical-structural-elements
    (declare (salience 20))
    =>
    (printout t "What is the material used in the vertical
    structural elements?" crlf)
    (printout t "a) Masonry elements" crlf)
    (printout t "b) Steel elements" crlf)
    (printout t "c) Reinforced concrete elements" crlf)
    (printout t "d) Mixed elements" crlf)
    (bind ?response (ask-question a b c d))

    (if (eq ?response a)
    then (bind ?*VI* (+ ?*VI* 6)))
    (if (eq ?response c)
    then (bind ?*VI* (+ ?*VI* 3)))
    (if (eq ?response d)
    then (bind ?*VI* (+ ?*VI* 4)))
)

;Foundations - source of information
(defrule foundations-info
    (declare (salience 30))
    =>
    (printout t "What is the source of information on the
    foundation of the building like?" crlf)
    (printout t "a) Direct (drawings, contractor)" crlf)
    (printout t "b) Indirect (property owner, inhabitants, for
    similarity with known structures, assessed)" crlf)
    (bind ?response (ask-question a b))
    (if (eq ?response b)
    then (bind ?*VI* (+ ?*VI* 4)))
)

;Refurbishments
(defrule refurbishments-type
    (declare (salience 40))
    =>
    (printout t "Is there any type of refurbishment present?" crlf)
    (printout t "a) Unknown" crlf)
    (printout t "b) Increasing opening in the facade
    (or bearing walls)" crlf)
    (printout t "c) Modifications maintaining the
    construction method" crlf)
    (printout t "d) Modifications improving the
    construction method" crlf)
    (printout t "e) Consolidation (bearing structure or
    foundations)" crlf)
    (printout t "f) Adding floors" crlf)
    (printout t "g) Small interior works" crlf)
    (bind ?response (ask-question a b c d e f g))
    (if (eq ?response a)
    then (bind ?*VI* (+ ?*VI* 2)))
    (if (eq ?response b)
    then (bind ?*VI* (+ ?*VI* 6)))
    (if (eq ?response d)
    then (bind ?*VI* (+ ?*VI* 3)))
    (if (eq ?response e)
    then (bind ?*VI* (+ ?*VI* 5)))
    (if (eq ?response f)
    then (bind ?*VI* (+ ?*VI* 4)))
    (if (neq ?response a)
    then (printout t "What is the state of refurbishment works" crlf)
         (printout t "a) Done or in progress" crlf)
         (printout t "b) Designed" crlf)
         (bind ?response2 (ask-question a b))
         (if (eq ?response2 a) then (bind ?*St_VI* (+ ?*St_VI* 1)))
         (if (eq ?response2 b) then (bind ?*Lt_VI* (+ ?*Lt_VI* 1))))
)

;Basement
(defrule basement-presence
    (declare (salience 50))
    =>
    (printout t "Are there any basement levels present?" crlf)
    (printout t "No" crlf)
    (printout t "Yes" crlf)
    (bind ?response (ask-question a b))
    (if (eq ?response b)
    then (bind ?*VI* (+ ?*VI* 3)))
)

;Orientation and position of the building
(defrule orientation-position
    (declare (salience 60))
    =>
    (printout t "Enter the average dimension in the direction
    parallel to the tunnel alignment (L1): ")
    (bind ?*L1* read)
    (printout t crlf)
    (printout t "Enter the average dimension in the direction
    perpendicular to the tunnel alignment (L2): ")
    (bind ?*L2* read)
    (printout t crlf)
    (printout t "Enter the distance of the building from the tunnel
    axis (x): ")
    (bind ?*x* read)
    (printout t crlf)
    (printout t "Enter the dimensions of the tunnel diameter (D): ")
    (bind ?*D* read)
    (printout t crlf)

    ; Orientation
    (if (< (/ ?*L1* ?*L2*) 0.5)
    then (bind ?*St_VI* (+ ?*St_VI* 5))
         (bind ?*Lt_VI* (+ ?*Lt_VI* 10)))
    (if (and (< (/ ?*L1* ?*L2*) 2) (> (/ ?*L1* ?*L2*) 0.5))
    then (bind ?*St_VI* (+ ?*St_VI* 6))
         (bind ?*Lt_VI* (+ ?*Lt_VI* 6)))
    (if (> (/ ?*L1* ?*L2*) 2)
    then (bind ?*St_VI* (+ ?*St_VI* 10))
         (bind ?*Lt_VI* (+ ?*Lt_VI* 5)))

    ; Group effect of buildings
    (printout t "Is the building isolated or a part of a group?" crlf)
    (printout t "a) Isolated building" crlf)
    (printout t "b) Grouped buildings parallel to the tunnel axis" crlf)
    (printout t "c) Group buildings perpendicular to the tunnel axis" crlf)
    (bind ?response (ask-question a b c))
    (if (eq ?response a)
    then (if (and (< ?*L1* (* 2 ?*D*)) (< ?*L2* (* 2 ?*D*)))
         then (bind ?*VI* (+ ?*VI* 15)))
         (if (and (> ?*L1* (* 2 ?*D*)) (> ?*L2* (* 2 ?*D*)))
         then (bind ?*VI* (+ ?*VI* 5)))
         (if (and (< ?*L1* (* 2 ?*D*)) (> ?*L2* (* 2 ?*D*)))
         then (bind ?*VI* (+ ?*VI* 10)))
         (if (and (> ?*L1* (* 2 ?*D*)) (< ?*L2* (* 2 ?*D*)))
         then (bind ?*VI* (+ ?*VI* 10))))
    (if (eq ?response b)
    then (bind ?*Lt_VI* (+ ?*Lt_VI* 7)))
    (if (eq ?response c)
    then (bind ?*St_VI* (+ ?*St_VI* 7)))

    ;Position (relative to tunnel) factor
    (if (< (/ ?*x* ?*D*) 1)
    then (bind ?*Lt_VI* (* ?*Lt_VI* 1))
         (bind ?*Lt_VI* (* ?*St_VI* 1)))
    (if (and (> (/ ?*x* ?*D*) 1) (< (/ ?*x* ?*D*) 3))
    then (bind ?*Lt_VI* (* ?*Lt_VI* 0.5))
         (bind ?*Lt_VI* (* ?*St_VI* 0.5)))
    (if (> (/ ?*x* ?*D*) 3)
    then (bind ?*Lt_VI* (* ?*Lt_VI* 0))
         (bind ?*Lt_VI* (* ?*St_VI* 0)))

)

;Use of the building
(defrule building-usage
    (declare (salience 70))
    =>
    (printout t "What kind of building is it?" crlf)
    (printout t "a) Highly sensitive building (hospital,
    historical/classical building)" crlf)
    (printout t "b) Medium sensitive building (modern, newly built)" crlf)
    (printout t "c) Low sensitive building (abandonded building)" crlf)
    (bind ?response (ask-question a b c))
    (if (eq ?response a)
    then (bind ?*VI* (+ ?*VI* 10)))
    (if (eq ?response b)
    then (bind ?*VI* (+ ?*VI* 5)))
)

;Aesthetic features of the building
(defrule aesthetic-features
    (declare (salience 80))
    =>
    ;Historic/artistic heritage
    (printout t "Does the building have historic/artistic heritage?" crlf)
    (printout t "a) No" crlf)
    (printout t "b) Yes" crlf)
    (bind ?response (ask-question a b))
    (if (eq ?response b)
    then (bind ?*VI* (+ ?*VI* 12)))

    ;Internal not bearing walls
    (printout t "What kind of material is used in the internal walls?" crlf)
    (printout t "a) Wood" crlf)
    (printout t "b) Bricks" crlf)
    (printout t "c) Cartongesso" crlf)
    (printout t "d) Alluminium and glass" crlf)
    (bind ?response (ask-question a b c d))
    (if (eq ?response a)
    then (bind ?*VI* (+ ?*VI* 1)))
    (if (eq ?response b)
    then (bind ?*VI* (+ ?*VI* 4)))
    (if (eq ?response c)
    then (bind ?*VI* (+ ?*VI* 3)))
    (if (eq ?response d)
    then (bind ?*VI* (+ ?*VI* 2)))

    ;External finishes
    (printout t "What kind of external finishes does the
    building have?" crlf)
    (printout t "a) Artistic tailing" crlf)
    (printout t "b) Ordinary tailing" crlf)
    (printout t "c) Plaster" crlf)
    (printout t "d) Other" crlf)
    (bind ?response (ask-question a b c d))
    (if (eq ?response a)
    then (bind ?*VI* (+ ?*VI* 4)))
    (if (eq ?response b)
    then (bind ?*VI* (+ ?*VI* 3)))
    (if (eq ?response c)
    then (bind ?*VI* (+ ?*VI* 2)))
    (if (eq ?response d)
    then (bind ?*VI* (+ ?*VI* 1)))
```

```
)

;State of the building
(defrule building-state
    (declare (salience 90))
    =>
    ;Visual conditions
    (printout t "General visual conditions" crlf)
    (printout t "a) Good" crlf)
    (printout t "b) Medium" crlf)
    (printout t "c) Bad" crlf)
    (bind ?response (ask-question a b c))
    (if (eq ?response b)
    then (bind ?*VI* (+ ?*VI* 4)))
    (if (eq ?response c)
    then (bind ?*VI* (+ ?*VI* 8)))

    ;Settlement signals
    (printout t "Are there any signals of settlements in the
    surrounding area" crlf)
    (printout t "a) Yes" crlf)
    (printout t "b) No" crlf)
    (bind ?response (ask-question a b))
    (if (eq ?response a)
    then (bind ?*VI* (+ ?*VI* 4)))

    ;Cracks
    (printout t "Are there any visible cracks on the building?" crlf)
    (printout t "a) Major cracks and extensive patterns" crlf)
    (printout t "b) Cracks and some patterns" crlf)
    (printout t "c) Isolated minor cracks" crlf)
    (printout t "d) None" crlf)
    (bind ?response (ask-question a b c d))
    (if (eq ?response a)
    then (bind ?*VI* (+ ?*VI* 8)))
    (if (eq ?response b)
    then (bind ?*VI* (+ ?*VI* 5)))
    (if (eq ?response c)
    then (bind ?*VI* (+ ?*VI* 3)))

)
```

APPENDIX E

FIR Artificial Neural Network

```
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include "fireng.h"

#define NUM_IN      (m_Nodes[0])
#define NUM_OUT     (m_Nodes[2])
#define NUM_HID     (m_Nodes[1])


void fir2layer::RandomizeWeights()
{
   Deallocate();

   m_WeightSize[0] = m_Nodes[0] * m_Nodes[1] * m_Taps[0];
   m_Weights[0] = new double[m_WeightSize[0]];
   m_LastWeights[0] = new double[m_WeightSize[0]];

   m_WeightSize[1] = m_Nodes[1] * m_Nodes[2] * m_Taps[1];
   m_Weights[1] = new double[m_WeightSize[1]];
   m_LastWeights[1] = new double[m_WeightSize[1]];

   m_Biases[0] = new double[m_Nodes[1]];
   m_LastBiases[0] = new double[m_Nodes[1]];
   m_Biases[1] = new double[m_Nodes[2]];
   m_LastBiases[1] = new double[m_Nodes[2]];

   assert(m_Weights[0] != NULL);
   assert(m_LastWeights[0] != NULL);
   assert(m_Weights[1] != NULL);
   assert(m_LastWeights[1] != NULL);
   assert(m_Biases[0] != NULL);
   assert(m_LastBiases[0] != NULL);
   assert(m_Biases[1] != NULL);
   assert(m_LastBiases[1] != NULL);

   RandomizeArray(m_Weights[0], m_WeightSize[0]);
   RandomizeArray(m_Weights[1], m_WeightSize[1]);
   RandomizeArray(m_Biases[0], m_Nodes[1]);
   RandomizeArray(m_Biases[1], m_Nodes[2]);

   ZeroArray(m_LastWeights[0], m_WeightSize[0]);
   ZeroArray(m_LastWeights[1], m_WeightSize[1]);
   ZeroArray(m_LastBiases[0], m_Nodes[1]);
   ZeroArray(m_LastBiases[1], m_Nodes[2]);
}

void fir2layer::Train(int NumVectors, double* Input, double* Target)
{
   int    Epoch;
   int    i, j, tmp;
   double RateDecrease;

   m_NumInputVectors = NumVectors;
   m_InputVectors = Input;
   m_TargetVectors = Target;

   if (m_Taps[1] > 1)
      m_Mask = (1 << ((int)(log((double)(m_Taps[1] - 1)) /
      0.69314718) + 1)) - 1;
   else m_Mask = 0;

   m_HiddenOutput = new double*[NUM_HID]; assert(m_HiddenOutput != NULL);
   for (i = 0; i < NUM_HID; i++)
      m_HiddenOutput[i] = new double[m_Mask + 1];
   m_NetOutput = new double[NUM_OUT]; assert(m_NetOutput != NULL);
   m_HiddenDelta = new double[NUM_HID]; assert(m_HiddenDelta != NULL);
   m_NetDelta = new double*[NUM_OUT]; assert(m_NetDelta != NULL);
   for (i = 0; i < NUM_OUT; i++)
      m_NetDelta[i] = new double[m_Mask + 1];
   m_MSE = new double[NUM_OUT * m_NumEpochs]; assert(m_MSE != NULL);
   m_CSE = new double[NUM_OUT]; assert(m_CSE != NULL);

   RateDecrease = m_LearningRate * 0.90 / (double)m_NumEpochs;

   for (Epoch = 0; Epoch < m_NumEpochs; Epoch++)
   {
      for (i = m_Taps[0]-1; i<m_Taps[0]+m_Taps[1]-2; i++) FwdPassLayer1(i);

      for (; i < m_Taps[0] + 2 * m_Taps[1] - 3; i++)
      {
         FwdPassLayer1(i);
         FwdPassLayer2(i);
         BackPropLayer2(i);
         UpdateWeights(i);
      }

      for (; i < m_NumInputVectors; i++)
      {
         FwdPassLayer1(i);
         FwdPassLayer2(i);
         BackPropLayer2(i);
         BackPropLayer1(i);
         UpdateWeights(i);
      }

      for (; i < m_NumInputVectors + m_Taps[1] - 1; i++)
      {
         for (tmp = i & m_Mask, j = 0; j < NUM_OUT; j++)
       m_NetDelta[j][tmp] = 0.0;

         BackPropLayer1(i);
         UpdateWeights(i);
      }

      CalcMSE(Epoch);
      ResetDeltas();
      m_EpochsTrained++;

      if (m_Decrease)
         m_LearningRate -= RateDecrease;
   }

   for (i = 0; i < NUM_HID; i++)
      delete [] m_HiddenOutput[i];
   for (i = 0; i < NUM_OUT; i++)
      delete [] m_NetDelta[i];
   delete [] m_HiddenOutput;
   delete [] m_NetOutput;
   delete [] m_HiddenDelta;
   delete [] m_NetDelta;
   delete [] m_MSE;
   delete [] m_CSE;
}

void fir2layer::FwdPassLayer1(int Ind)
{
   int    i, j, k, temp, temp1;
   int    MatrixSize = NUM_HID * NUM_IN;
   double Out;
   temp1 = Ind & m_Mask;
   for (i = 0; i < NUM_HID; i++)
   {
      Out = m_Biases[0][i];
      for (j = 0; j < m_Taps[0]; j++)
      {
         temp = Ind - j;
         for (k = 0; k < NUM_IN; k++)
            Out += m_InputVectors[MATRIX_WEIGHT(k, temp, NUM_IN)] *
               m_Weights[0][MATRIX_TAP(i, k, j, NUM_HID, MatrixSize)];
      }
      m_HiddenOutput[i][temp1] = HID_FN(Out);
   }
}

void fir2layer::FwdPassLayer2(int Ind)
{
   int    i,j,k, temp;
   double Out;
   int    MatrixSize = NUM_OUT * NUM_HID;
   for (i = 0; i < NUM_OUT; i++)
   {
      Out = m_Biases[1][i];
      for (j = 0; j < m_Taps[1]; j++)
      {
         temp = (Ind - j) & m_Mask;
         for (k = 0; k < NUM_HID; k++)
            Out += m_HiddenOutput[k][temp] *
               m_Weights[1][MATRIX_TAP(i, k, j, NUM_OUT, MatrixSize)];
      }
      m_NetOutput[i] = OUT_FN(Out);
   }
}

void fir2layer::BackPropLayer1(int Ind)
{
   int    i, j, k, temp;
   int    MatrixSize = NUM_OUT * NUM_HID;
   double Delta;

   for (i = 0; i < NUM_HID; i++)
   {
      Delta = 0.0;
      for (j = 0; j < m_Taps[1]; j++)
      {
         temp = (Ind - j) & m_Mask;
         for (k = 0; k < NUM_OUT; k++)
            Delta += m_NetDelta[k][temp] *
        m_Weights[1][MATRIX_TAP(k, i, m_Taps[1] - j - 1,
        NUM_OUT, MatrixSize)];
      }
      temp = (Ind - m_Taps[1] + 1) & m_Mask;
      m_HiddenDelta[i] = Delta * D_HID_FN(m_HiddenOutput[i][temp]);
   }
}

void fir2layer::BackPropLayer2(int Ind)
{
   int    i, temp;
   double Error;
   temp = Ind & m_Mask;

   for (i = 0; i < NUM_OUT; i++)
   {
      Error = m_TargetVectors[MATRIX_WEIGHT(i, Ind, NUM_OUT)] -
      m_NetOutput[i];
      m_CSE[i] += Error * Error;
      m_NetDelta[i][temp] = Error;
   }
}

void fir2layer::UpdateWeights(int Ind)
{
   int    i, j, k, temp;
   int    Matrix = NUM_OUT * NUM_HID;
   double Delta, Evol;

   for (i = 0; i < NUM_OUT; i++)
   {
      Delta = m_LearningRate * m_NetDelta[i][Ind & m_Mask];
      for (j = 0; j < m_Taps[1]; j++)
      {
         temp = (Ind - j) & m_Mask;
         for (k = 0; k < NUM_HID; k++)
         {
            Evol = Delta * m_HiddenOutput[k][temp] + m_Momentum *
            m_LastWeights[1][MATRIX_TAP(i, k, j, NUM_OUT, Matrix)];

            m_Weights[1][MATRIX_TAP(i, k, j, NUM_OUT, Matrix)] += Evol;
            m_LastWeights[1][MATRIX_TAP(i, k, j, NUM_OUT, Matrix)] = Evol;
         }
      }
      Evol = Delta + m_Momentum * m_LastBiases[1][i];
      m_Biases[1][i] += Evol;
      m_LastBiases[1][i] = Evol;
   }

   Matrix = NUM_IN * NUM_HID;

   for (i = 0; i < NUM_HID; i++)
   {
      Delta = m_LearningRate * m_HiddenDelta[i];
      for (j = 0; j < m_Taps[0]; j++)
      {
         temp = Ind - j - m_Taps[1] + 1;
         for (k = 0; k < NUM_IN; k++)
         {
            Evol = Delta * m_InputVectors[MATRIX_WEIGHT(k, temp, NUM_IN)]
            + m_Momentum * m_LastWeights[0][MATRIX_TAP(i, k, j, NUM_HID,
     Matrix)];
            m_Weights[0][MATRIX_TAP(i, k, j, NUM_HID, Matrix)] += Evol;
            m_LastWeights[0][MATRIX_TAP(i, k, j, NUM_HID, Matrix)] = Evol;
         }
      }
      Evol = Delta + m_Momentum * m_LastBiases[0][i];
      m_Biases[0][i] += Evol;
      m_LastBiases[0][i] = Evol;
   }
}

void fir2layer::CalcMSE(int Epoch)
{
   int i;

   for (i = 0; i < NUM_OUT; i++)
   {
      m_MSE[MATRIX_WEIGHT(i, Epoch, NUM_OUT)] =
         m_CSE[i] / (double)(m_NumInputVectors - m_Taps[1] - m_Taps[0] + 2);
      m_CSE[i] = 0.0;
   }
}

void fir2layer::ResetDeltas()
{
   int i, t;

   for (i = 0; i < NUM_OUT; i++)
   {
      for (t = m_Taps[1]; t < m_Mask + 1; t++)
         m_NetDelta[i][t] = 0.0;
   }
}

void fir2layer::ForwardPass()
{
   int     i;
   double* OrigOutput;


   if (m_Taps[1] > 1)
      m_Mask = (1 << ((int)(log((double)(m_Taps[1] - 1)) / 0.69314718)
      + 1)) - 1;
   else
      m_Mask = 0;

   m_HiddenOutput = new double*[NUM_HID];
   assert(m_HiddenOutput != NULL);
   for (i = 0; i < NUM_HID; i++)
      m_HiddenOutput[i] = new double[m_Mask + 1];

   for (i = m_Taps[0] - 1; i < m_Taps[0] + m_Taps[1] - 2; i++)
      FwdPassLayer1(i);    // Prime the second layer of taps

   OrigOutput = m_NetOutput;
   m_NetOutput += i * NUM_OUT;   // Skip past primer values
   // Go over all remaining input vectors
   for (; i < m_NumInputVectors; i++)
   {
      FwdPassLayer1(i);
      FwdPassLayer2(i);    // Output goes into m_NetOutput
      m_NetOutput += NUM_OUT;
   }

   // There should be (NumInputVectors - WindowSize) output points

   m_NetOutput = OrigOutput;

   for (i = 0; i < NUM_HID; i++)
      delete [] m_HiddenOutput[i];
   delete [] m_HiddenOutput;
}



#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <assert.h>
#include "fireng.h"

fir::fir()
{
   m_Weights[0] = m_Weights[1] = m_Weights[2] = NULL;
   m_Biases[0] = m_Biases[1] = m_Biases[2] = NULL;
   m_LastWeights[0] = m_LastWeights[1] = m_LastWeights[2] = NULL;
   m_LastBiases[0] = m_LastBiases[1] = m_LastBiases[2] = NULL;
   m_EpochsTrained = 0;
}

fir& fir::operator=(fir& network)
{
   int i;

   for (i = 0; i < 4; i++)
      m_Nodes[i] = network.m_Nodes[i];

   for (i = 0; i < 3; i++)
      m_Taps[i] = network.m_Taps[i];

   return *this;
}

void fir::Deallocate()
{
   // Massive memory deallocation
  for (int i = 0; i <= 2; i++)
    {
      if (m_Weights[i] != NULL) delete [] m_Weights[i];
      if (m_Biases[i] != NULL) delete [] m_Biases[i];
      if (m_LastWeights[i] != NULL) delete [] m_LastWeights[i];
      if (m_LastBiases[i] != NULL) delete [] m_LastBiases[i];
    }
}

void RandomizeArray(double* Array, int n) // n = length of the array
{
   while (n-- > 0)
      Array[n] = (double)rand() / (double)RAND_MAX;
}

void ZeroArray(double* Array, int n) // n = length of the array
{
   while (n-- > 0) Array[n] = 0.0;
}


void fir::Predict(int Start, int End, int NumVectors,
   double* Input, double* Output)
{
  double* Prediction;
  double* PredEnd;
  int     i, MaxEnd, Dim = m_Nodes[0];

  assert(NumVectors > WindowSize());

  printf("Prediction for time interval [%i, %i]\n", Start, End);

  if (End < Start)
    {
      i = Start;
      Start = End;
      End = i;
    }

  assert(End >= 0); assert(Start >= 0);

  if (End >= NumVectors) MaxEnd = End;
  else MaxEnd = NumVectors - 1;

  Prediction = new double[(MaxEnd + 1 + WindowSize()) * Dim];
  // Temporary memory allocation to hold the prediction

  for (i = 0; i <= MaxEnd * Dim; i++) Prediction[i] = 0.0;

  m_NumInputVectors = NumVectors;
  m_InputVectors = Input; // recreation of original input series
  m_NetOutput = &Prediction[Dim]; // skip first vector



  ForwardPass();
  // ***** Iterative Prediction ***** //

  PredEnd = &Prediction[NumVectors * Dim];

  for (i = 0; i < MaxEnd - NumVectors + 1; i++)
    {
      m_InputVectors = &PredEnd[(i - WindowSize()-1) * Dim];
      m_NumInputVectors = WindowSize() + 1;
      m_NetOutput = &PredEnd[(i - WindowSize()) * Dim];
      ForwardPass();
    }

  PredEnd = &Prediction[Start * Dim];
  for (i = 0; i < (End - Start + 1) * Dim; i++)
    Output[i] = PredEnd[i];

  delete [] Prediction;

  return;
}

int fir::Load(FILE* InFile)
{
   int i, j;
   fscanf(InFile, "%i ", &m_EpochsTrained);
   for (i = 0; i < 4; i++) fscanf(InFile, "%i ", &m_Nodes[i]);
   for (i = 0; i < 3; i++) fscanf(InFile, "%i ", &m_Taps[i]);
   for (i = 0; i < 3; i++)
   {
      fscanf(InFile, "%i", &m_WeightSize[i]);
      if (m_WeightSize[i] == 0)
      {
         m_Weights[i] = NULL;
         m_LastWeights[i] = NULL;
      } else
      {
         m_Weights[i] = new double[m_WeightSize[i]];
         m_LastWeights[i] = new double[m_WeightSize[i]];
         assert(m_Weights[i] != NULL);
         assert(m_LastWeights[i] != NULL);
         for (j = 0; j < m_WeightSize[i]; j++)
            fscanf(InFile, "%lf ", &m_Weights[i][j]);
         ZeroArray(m_LastWeights[i], m_WeightSize[i]);
         m_Biases[i] = new double[m_Nodes[i + 1]];
         m_LastBiases[i] = new double[m_Nodes[i + 1]];
         assert(m_Biases[i] != NULL);
         assert(m_LastBiases[i] != NULL);
         for (j = 0; j < m_Nodes[i + 1]; j++)
            fscanf(InFile, "%lf ", &m_Biases[i][j]);
         ZeroArray(m_LastBiases[i], m_Nodes[i + 1]);
      }
   }
   return 0;
}

int fir::Save(FILE* OutFile)
{
   int i, j;
   fprintf(OutFile, "%i\n", m_EpochsTrained);
   for (i = 0; i < 4; i++) fprintf(OutFile, "%i ", m_Nodes[i]);
   fprintf(OutFile, "\n");
   for (i = 0; i < 3; i++) fprintf(OutFile, "%i ", m_Taps[i]);
   fprintf(OutFile, "\n");
   for (i = 0; i < 3; i++)
   {
      if (i == 2 && m_Weights[i] == NULL) fprintf(OutFile, "0\n");
      else
      {
         fprintf(OutFile, "%i\n", m_WeightSize[i]);
         for (j = 0; j < m_WeightSize[i]; j++)
            fprintf(OutFile, "%g ", m_Weights[i][j]);
         fprintf(OutFile, "\n");

         for (j = 0; j < m_Nodes[i + 1]; j++)
            fprintf(OutFile, "%g ", m_Biases[i][j]);
         fprintf(OutFile, "\n");
      }
   }

   return 0;
}


#ifndef __FIR_H__

#include <stdio.h>

// Transfer function macros
#define HID_FN(x)    (tanh(x))      // Hidden layer transfer function
#define OUT_FN(x)    (x)            // Output layer transfer function
#define D_HID_FN(x)  (1.0 - (x*x)) // Derivative of HID_FN w.r.t. x

// Macros for indexing into weight matrices
#define COL_ROW(i, j, ROWS) (i + (j) * (ROWS))
#define COL_ROW_TAP(i, j, tap, ROWS, ROWSXCOLS) (i + (j) * (ROWS)
+ (tap) * (ROWSXCOLS))

class fir   // A Finite Impulse Response neural network
{
public:

   fir();
   ~fir() { Deallocate(); }
   fir& operator=(fir& net);

   int  Load(FILE* InFile);
   int  Save(FILE* OutFile);

   void SetNodes(int Layer, int NumNodes) {
      m_Nodes[Layer] = NumNodes;    // # of nodes in a layer
   }

   void SetTaps(int Layer, int NumTaps) {
      m_Taps[Layer] = NumTaps;   // # of time taps in a layer
   }

   void SetTrainingParams(int Epochs, double Rate, int Freq, double
   Momentum, int Decrease) {
      m_NumEpochs = Epochs;
      m_LearningRate = Rate;
      m_DisplayFreq = Freq;      // Display status every X epochs
      m_MomentumRate = Momentum;
      m_DecreaseRate = Decrease; // Linearly decrease learning rate?
   }

   virtual void RandomizeWeights() {}
   virtual int  WindowSize() { return 0; }   // Size of "primer" window
   virtual void Train(int NumVectors, double* InputVectors, double*
   TargetVectors) {}

   void Predict(int StartTime, int EndTime, int NumVectors, double*
   InputVectors, double* OutputVectors);

protected:

   void     Deallocate();

   //
   // State
   //

   int      m_Nodes[4];          // # of nodes in input/hidden/output layers
   int      m_Taps[3];          // # of taps in input/hidden layers

   double*  m_Weights[3];        // Intra-layer weight matrices
   int      m_WeightSize[3];     // # elements in each weight matrix
   double*  m_Biases[3];         // Bias vectors for each layer

   double*  m_LastWeights[3];    // Last weight matrix for momentum learning
   double*  m_LastBiases[3];     // Last bias matrix for momentum learning

   //
   // Training
   //

   int      m_NumEpochs;         // # of epochs to train for
   int      m_EpochsTrained;     // Total epochs this net has trained
   double   m_LearningRate;      // Weight adjustment learning rate
   int      m_DisplayFreq;       // Display status every X epochs
   double   m_MomentumRate;      // Momentum weighting constant
   int      m_DecreaseRate;      // Flag to decrease rate during training

   double*  m_InputVectors;      // Input vectors to train on
   double*  m_TargetVectors;     // Desired output vectors
   int      m_NumInputVectors;  // Total # of input vectors

   int      m_Mask;              // Circular index
   double*  m_MSE;               // Mean squared error per epoch
   double*  m_CSE;               // Cumulative squared error
   double*  m_NetOutput;         // Network outputs
   double** m_NetDelta;          // Network error
   double** m_HiddenOutput;      // Hidden layer outputs
   double*  m_HiddenDelta;       // Hidden layer error

   virtual void ForwardPass() {} // Implemented by specific FIR classes
};

class fir2layer : public fir    // Special case: 2-layer FIR
{
public:

   virtual void RandomizeWeights();

   virtual int WindowSize() {
      return (m_Taps[0] + m_Taps[1] - 2);
   }

   virtual void Train(int NumVectors, double* InputVectors, double*
   TargetVectors);

protected:

   void FwdPassLayer1(int Index);
   void FwdPassLayer2(int Index);
   void BackPropLayer1(int Index);
   void BackPropLayer2(int Index);
   void UpdateWeights(int Index);
   void CalcMSE(int Epoch);
   void ResetDeltas();

   virtual void ForwardPass();
};


void RandomizeArray(double* Array, int Length);
void ZeroArray(double* Array, int Length);

#define __FIR_H__
#endif
```

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include <ctype.h>
#include "timeseries.h"
#include "fireng.h"

const char* Usage =
   "Usage: firnet -f filename [switches...]\n"
   "\n"
   "Switches:\n"
   "    -l layers       # of layers in the FIR network: 2 (default) or 3\n"
   "    -n n1 [n2]      # of nodes in each hidden layer\n"
   "    -t n1 n2 [n3]   # of taps in each layer\n"
   "    -p start end    Range of time series values to predict\n"
   "    -e epochs       # of epochs to train for\n"
   "    -c              Force training (if disabled by loading state)\n"
   "    -r rate         Learning rate\n"
   "    -m alpha        Use momentum with parameter alpha in [0, 1]\n"
   "    -d              Linearly decrease learning rate";

int main(int argc, char** argv)
{
   TimeSeries T, P;
   char*      FileName = NULL;
   int        Layers = 2;
   int        StartTime = 0;
   int        EndTime = -1;
   int        Epochs = 75;
   int        DispFreq = 5;
   double     Rate = 0.005;
   double     Alpha = 0.0;
   int        i, j;
   int        ForceTrain = 0;
   int        WeightsLoaded = 0;
   int        DecreaseRate = 0;

   srand(time(NULL));    // Seed random number generator

   // Convert all switches to lower case
   for (i = 1; i < argc; i++)
   {
      if (argv[i][0] == '-')
         for (j = 0; j < strlen(argv[i]); j++)
            argv[i][j] = tolower(argv[i][j]);
   }

   // Seek out the -layers switch on the command line First
   for (i = 1; i < argc; i++)
   {
      if (strcmp(argv[i], "-l") == 0)
      {
         if (i + 1 >= argc)
         {
            printf("Expected parameter after %s\n", argv[i]);
            return 1;
         }
         Layers = atoi(argv[++i]);
      }
   }

   assert(Layers == 2 || Layers == 3);

   // Some default network parameters
   T.SetNetworkSize(Layers);
   T.SetHiddenNodes(0, 5);
   T.SetHiddenNodes(1, 4);
   T.SetLayerTaps(0, 4);
   T.SetLayerTaps(1, 3);
   T.SetLayerTaps(2, 2);

   // Parse the command line to change program parameters
   for (i = 1; i < argc; i++)
   {
      // Command line help
      if (strcmp(argv[i], "-h") == 0)
      {
         printf("%s\n", Usage);
         return 1;
      } else
      // # of epochs to train for
      if (strcmp(argv[i], "-e") == 0)
      {
         if (i + 1 >= argc)
         {
            printf("Expected parameter after %s\n", argv[i]);
            return 1;
         }
         Epochs = atoi(argv[++i]);
      } else
      // # of layers in network (2 or 3)
      if (strcmp(argv[i], "-l") == 0)
         Layers = atoi(argv[++i]);
      else
      // # of nodes in each layer -- sequence of ints follows
      if (strcmp(argv[i], "-n") == 0)
      {
         if (i + 1 >= argc)
         {
            printf("Expected parameter after %s\n", argv[i]);
            return 1;
         }
         T.SetHiddenNodes(0, atoi(argv[++i]));
         if (i + 1 < argc && atoi(argv[i + 1]) > 0)
            T.SetHiddenNodes(1, atoi(argv[++i]));
```

```
} else
// # of taps in each layer -- sequence of ints follows
if (strcmp(argv[i], "-t") == 0)
{
   if (i + 2 >= argc)
   {
      printf("Expected parameters after %s\n", argv[i]);
      return 1;
   }
   T.SetLayerTaps(0, atoi(argv[++i]));
   if (i + 1 < argc && atoi(argv[i + 1]) > 0)
   {
      T.SetLayerTaps(1, atoi(argv[++i]));
      if (i + 1 < argc && atoi(argv[i + 1]) > 0)
         T.SetLayerTaps(2, atoi(argv[++i]));
   }
} else
// Range of time values to predict
if (strcmp(argv[i], "-p") == 0)
{
   if (i + 2 >= argc)
   {
      printf("Expected parameters after %s\n", argv[i]);
      return 1;
   }
   StartTime = atoi(argv[++i]);
   EndTime = atoi(argv[++i]);
} else
// Input series file name
if (strcmp(argv[i], "-f") == 0)
{
   if (i + 1 >= argc)
   {
      printf("Expected parameter after %s\n", argv[i]);
      return 1;
   }
   FileName = argv[++i];
} else
// Force training
if (strcmp(argv[i], "-c") == 0)
   ForceTrain = 1;
else
// Learning rate
if (strcmp(argv[i], "-r") == 0)
{
   if (i + 1 >= argc)
   {
      printf("Expected parameter after %s\n", argv[i]);
      return 1;
   }
   Rate = atof(argv[++i]);
} else
// Use momentum
```

```
      if (strcmp(argv[i], "-m") == 0)
      {
         if (i + 1 >= argc)
         {
            printf("Expected parameter after %s\n", argv[i]);
            return 1;
         }
         Alpha = atof(argv[++i]);
      } else
      // Decrease learning rate during training
      if (strcmp(argv[i], "-d") == 0)
         DecreaseRate = 1;
      else
      // Unknown switch
      {
         printf("Unknown switch %s\n", argv[i]);
         return 1;
      }
   }

   if (FileName == NULL)
   {
      printf("%s\n", Usage);
      return 1;
   }

   if (T.LoadSeries(FileName) != 0)
   {
      printf("There was an error reading the input file.\n");
      return 1;
   }

   if (T.LoadNetwork(FileName) == 0)
      WeightsLoaded = 1;
   else
      T.RandomizeNetwork();

   T.SetTrainingParams(Epochs, Rate, DispFreq, Alpha, DecreaseRate);

   if (!WeightsLoaded || ForceTrain)
      T.TrainNetwork();

   // If the prediction range was not set by the user, just do 10%
   if (EndTime == -1)
      EndTime = T.Length() - 1 + T.Length()/10;

   T.Predict(StartTime, EndTime, P);

   P.SaveSeries(FileName);
   T.SaveNetwork(FileName);

   return 0;
}
```

APPENDIX F

RFIR Artificial Neural Network

```
#include "rfir.h"
#include "rfir_util.h"

void rfir(const matrix *Wmask, matrix *W, const
          matrix *DI, const matrix *DT, const matrix *
          EO, double alpha, matrix *CO, matrix *Wd,
          matrix *Z, double *rel_mean_sq, matrix *COL)
{
  int I;
  int N;
  double P;
  matrix *OM;
  boolean_T guard2 = false;
  int cr;
  int k;
  int iv2[2];
  int m;
  int i0;
  int ia;
  int ic;
  int br;
  int ar;
  int ib;
  int i1;
  matrix *Wd_m;
  matrix *dW;
  double SE;
  int l;
  matrix *ET;
  matrix *a;
  matrix *b_DT;
  double d0;
  double t;
```

```
int c;
double S;
double b_i1;
double b;
boolean_T guard1 = false;
double E;
I = DI->size[0];
N = W->size[0];
P = (double)W->size[1] / ((double)W->size[0] + (double)DI->size[0]);
c_matrix_init(&OM, 1);
guard2 = false;
if (EO->size[1] == 1) {
  guard2 = true;
} else {
  cr = DT->size[0];
  if (cr == 1) {
    guard2 = true;
  } else {
    k = EO->size[1];
    iv2[0] = EO->size[0];
    m = EO->size[0];
    i0 = OM->size[0];
    OM->size[0] = iv2[0];
    matrix_cap((matrix__common *)OM, i0, (int)sizeof(double));
    ia = iv2[0];
    for (i0 = 0; i0 < ia; i0++) {
      OM->data[i0] = 0.0;
    }

    if (EO->size[0] == 0) {
    } else {
      cr = 0;
      while ((m > 0) && (cr <= 0)) {
        for (ic = 1; ic <= m; ic++) {
          OM->data[ic - 1] = 0.0;
        }

        cr = m;
      }

      br = 0;
      cr = 0;
      while ((m > 0) && (cr <= 0)) {
        ar = 0;
        i0 = br + k;
        for (ib = br + 1; ib <= i0; ib++) {
          ia = ar;
          for (ic = 0; ic + 1 <= m; ic++) {
            ia++;
            OM->data[ic] += EO->data[ia - 1];
          }

          ar += m;
        }

        br += k;
        cr = m;
      }
    }
  }
}

if (guard2) {
  i0 = OM->size[0];
  OM->size[0] = EO->size[0];
  matrix_cap((matrix__common *)OM, i0, (int)sizeof(double));
  ia = EO->size[0];
  for (i0 = 0; i0 < ia; i0++) {
    OM->data[i0] = 0.0;
    cr = EO->size[1];
    for (i1 = 0; i1 < cr; i1++) {
      OM->data[i0] += EO->data[i0 + EO->size[0] * i1];
    }
  }
}

matrix_init(&Wd_m, 2);
i0 = Wd_m->size[0] * Wd_m->size[1];
Wd_m->size[0] = 1;
matrix_cap((matrix__common *)Wd_m, i0, (int)sizeof(double));
cr = (int)((double)W->size[0] * (double)W->size[0] * P *
((double)W->size[0] + (double)DI->size[0]));
i0 = Wd_m->size[0] * Wd_m->size[1];
Wd_m->size[1] = cr;
matrix_cap((matrix__common *)Wd_m, i0, (int)sizeof(double));
ia = (int)((double)W->size[0] * (double)W->size[0] * P *
((double)W->size[0] + (double)DI->size[0]));
for (i0 = 0; i0 < ia; i0++) {
  Wd_m->data[i0] = 0.0;
}

cr = W->size[0];
i0 = COL->size[0] * COL->size[1];
COL->size[0] = cr;
matrix_cap((matrix__common *)COL, i0, (int)sizeof(double));
i0 = COL->size[0] * COL->size[1];
COL->size[1] = 1;
matrix_cap((matrix__common *)COL, i0, (int)sizeof(double));
cr = W->size[0];
for (i0 = 0; i0 < cr; i0++) {
  COL->data[i0] = 0.0;
}

//  CO=Z(1:P:P*N,1);
for (i0 = 0; i0 < 2; i0++) {
  iv2[i0] = W->size[i0];
```

```
}

matrix_init(&dW, 2);
i0 = dW->size[0] * dW->size[1];
dW->size[0] = iv2[0];
matrix_cap((matrix__common *)dW, i0, (int)sizeof(double));
i0 = dW->size[0] * dW->size[1];
dW->size[1] = iv2[1];
matrix_cap((matrix__common *)dW, i0, (int)sizeof(double));
ia = iv2[0] * iv2[1];
for (i0 = 0; i0 < ia; i0++) {
  dW->data[i0] = 0.0;
}

SE = 0.0;
l = 0;
c_matrix_init(&ET, 1);
matrix_init(&a, 2);
c_matrix_init(&b_DT, 1);
while (l <= DT->size[1] - 1) {
  d0 = P * ((double)N + (double)I);
  for (cr = 0; cr < (int)-(2.0 + (-1.0 - d0)); cr++) {
    t = d0 + -(double)cr;
    Z->data[(int)t - 1] = Z->data[(int)(t - 1.0) - 1];
  }

  d0 = P * ((double)N + (double)I);
  if ((P == 0.0) || (((P > 0.0) && (1.0 > d0)) || ((0.0 >
    P) && (d0 > 1.0))))
  {
    i0 = 1;
  } else {
    i0 = (int)P;
  }

  ia = DI->size[0] - 1;
  cr = CO->size[1];
  for (i1 = 0; i1 < cr; i1++) {
    br = CO->size[0];
    for (ar = 0; ar < br; ar++) {
      Z->data[i0 * ar] = CO->data[ar + CO->size[0] * i1];
    }
  }

  for (i1 = 0; i1 <= ia; i1++) {
    Z->data[i0 * (i1 + CO->size[0])] = DI->data[i1 + DI->size[0] * l];
  }

  //  Pass inputs through sigmoid (logistic) function
  i0 = a->size[0] * a->size[1];
  a->size[0] = Wmask->size[0];
  a->size[1] = Wmask->size[1];
  matrix_cap((matrix__common *)a, i0, (int)sizeof(double));
```

```
ia = Wmask->size[0] * Wmask->size[1];
for (i0 = 0; i0 < ia; i0++) {
  a->data[i0] = -(Wmask->data[i0] * W->data[i0]);
}

if ((a->size[1] == 1) || (Z->size[0] == 1)) {
  i0 = COL->size[0] * COL->size[1];
  COL->size[0] = a->size[0];
  COL->size[1] = Z->size[1];
  matrix_cap((matrix__common *)COL, i0, (int)sizeof(double));
  ia = a->size[0];
  for (i0 = 0; i0 < ia; i0++) {
    cr = Z->size[1];
    for (i1 = 0; i1 < cr; i1++) {
      COL->data[i0 + COL->size[0] * i1] = 0.0;
      br = a->size[1];
      for (ar = 0; ar < br; ar++) {
        COL->data[i0 + COL->size[0] * i1] += a->data[i0 +
        a->size[0] * ar] * Z->data[ar + Z->size[0] * i1];
      }
    }
  }
} else {
  k = a->size[1];
  iv2[0] = a->size[0];
  iv2[1] = Z->size[1];
  m = a->size[0];
  i0 = COL->size[0] * COL->size[1];
  COL->size[0] = iv2[0];
  matrix_cap((matrix__common *)COL, i0, (int)sizeof(double));
  i0 = COL->size[0] * COL->size[1];
  COL->size[1] = iv2[1];
  matrix_cap((matrix__common *)COL, i0, (int)sizeof(double));
  ia = iv2[0] * iv2[1];
  for (i0 = 0; i0 < ia; i0++) {
    COL->data[i0] = 0.0;
  }

  if ((a->size[0] == 0) || (Z->size[1] == 0)) {
  } else {
    c = a->size[0] * (Z->size[1] - 1);
    cr = 0;
    while ((m > 0) && (cr <= c)) {
      i0 = cr + m;
      for (ic = cr; ic + 1 <= i0; ic++) {
        COL->data[ic] = 0.0;
      }

      cr += m;
    }

    br = 0;
    cr = 0;
```

```
        while ((m > 0) && (cr <= c)) {
          ar = 0;
          i0 = br + k;
          for (ib = br; ib + 1 <= i0; ib++) {
            if (Z->data[ib] != 0.0) {
              ia = ar;
              i1 = cr + m;
              for (ic = cr; ic + 1 <= i1; ic++) {
                ia++;
                COL->data[ic] += Z->data[ib] * a->data[ia - 1];
              }
            }

            ar += m;
          }

          br += k;
          cr += m;
        }
      }
    }

    i0 = COL->size[0] * COL->size[1];
    for (k = 0; k < i0; k++) {
      COL->data[k] = exp(COL->data[k]);
    }

    i0 = COL->size[0] * COL->size[1];
    matrix_cap((matrix__common *)COL, i0, (int)sizeof(double));
    cr = COL->size[0];
    br = COL->size[1];
    ia = cr * br;
    for (i0 = 0; i0 < ia; i0++) {
      COL->data[i0] = 1.0 / (1.0 + COL->data[i0]);
    }

    //  COL is output at t+1
    //  Update for partial derivatives at k-th output ij-th position
    for (k = 0; k < N; k++) {
      for (br = 0; br < N; br++) {
        d0 = (double)N + (double)I;
        for (ar = 0; ar < (int)d0; ar++) {
          for (cr = 0; cr < (int)P; cr++) {
            //  update of all derivatives
            if (Wmask->data[br + Wmask->size[0] * ((int)(((1.0 +
          (double)ar) - 1.0) * P + (1.0 + (double)cr)) - 1)]
          == 0.0) {
            } else {
              S = 0.0;
              for (ia = 0; ia < N; ia++) {
                //  formula for calculation
                b_i1 = ((((1.0 + (double)ia) - 1.0) * ((double)N * P *
                ((double) N + (double)I)) + ((1.0 + (double)br) - 1.0)*
                P*((double)N + (double)I)) + ((1.0 + (double)ar) - 1.0)*
        P) + (1.0 + (double)cr);
                for (ib = 0; ib < (int)P; ib++) {
                  if (Wmask->data[k + Wmask->size[0] * ((int)(((1.0 +
            (double)ia) - 1.0) * P + (1.0 + (double)ib)) -
            1)] == 0.0) {
                  } else {
                    S += W->data[k + W->size[0] * ((int)(((1.0 +
                    (double)ia) - 1.0) * P + (1.0 + (double)ib)) - 1)]
                    * Wd->data [Wd->size[0] * ((int)b_i1 - 1)];
                  }
                }
              }

              b = Z->data[(int)(((1.0 + (double)cr) - 1.0) * ((double)N +
                (double)I) + (1.0 + (double)ar)) - 1];
              Wd_m->data[Wd_m->size[0] * ((int)(((((1.0 + (double)k) -
        1.0) * ((double)N * P * ((double)N + (double)I)) + ((1.0 +
                (double)br) - 1.0) * P * ((double)N + (double)I)) + ((1.0 +
                (double)ar) - 1.0) * P) + (1.0 + (double)cr)) - 1)] =
                COL->data[k] * (1.0 - COL->data[k]) * (S + (double)(1.0 +
                (double)br == 1.0 + (double) k) * b);
            }
          }
        }
      }
    }

    //  Update for delta W: dW for point t+1
    guard1 = false;
    if (EO->size[1] == 1) {
      guard1 = true;
    } else {
      i0 = DT->size[0];
      if (i0 == 1) {
        guard1 = true;
      } else {
        k = EO->size[1];
        iv2[0] = EO->size[0];
        m = EO->size[0];
        i0 = ET->size[0];
        ET->size[0] = iv2[0];
        matrix_cap((matrix__common *)ET, i0, (int)sizeof(double));
        ia = iv2[0];
        for (i0 = 0; i0 < ia; i0++) {
          ET->data[i0] = 0.0;
        }

        if (EO->size[0] == 0) {
        } else {
          cr = 0;
          while ((m > 0) && (cr <= 0)) {
            for (ic = 1; ic <= m; ic++) {
              ET->data[ic - 1] = 0.0;
            }

            cr = m;
          }

          br = 0;
          cr = 0;
          while ((m > 0) && (cr <= 0)) {
            ar = 0;
            i0 = br + k;
            for (ib = br; ib + 1 <= i0; ib++) {
              if (DT->data[ib + DT->size[0] * 1] != 0.0) {
                ia = ar;
                for (ic = 0; ic + 1 <= m; ic++) {
                  ia++;
                  ET->data[ic] += DT->data[ib + DT->size[0] * 1] *
                  EO->data[ia - 1];
                }
              }

              ar += m;
            }

            br += k;
            cr = m;
          }
        }
      }
    }

    if (guard1) {
      ia = DT->size[0];
      i0 = b_DT->size[0];
      b_DT->size[0] = ia;
      matrix_cap((matrix__common *)b_DT, i0, (int)sizeof(double));
      for (i0 = 0; i0 < ia; i0++) {
        b_DT->data[i0] = DT->data[i0 + DT->size[0] * 1];
      }

      i0 = ET->size[0];
      ET->size[0] = EO->size[0];
      matrix_cap((matrix__common *)ET, i0, (int)sizeof(double));
      ia = EO->size[0];
      for (i0 = 0; i0 < ia; i0++) {
        ET->data[i0] = 0.0;
        cr = EO->size[1];
        for (i1 = 0; i1 < cr; i1++) {
          ET->data[i0] += EO->data[i0 + EO->size[0] * i1] * b_DT->data[i1];
        }
      }
    }
```

```
//  this is mapping for target data
for (k = 0; k < N; k++) {
  if (OM->data[k] > 0.0) {
    E = ET->data[k] - CO->data[k];
    SE += E * E;
  }
}

for (br = 0; br < N; br++) {
  d0 = (double)N + (double)I;
  for (ar = 0; ar < (int)d0; ar++) {
    for (cr = 0; cr < (int)P; cr++) {
      S = 0.0;
      if (Wmask->data[br + Wmask->size[0] * ((int)(((1.0 + (double)ar)
    - 1.0) * P + (1.0 + (double)cr)) - 1)] == 0.0) {
        dW->data[br + dW->size[0] * ((int)(((1.0 + (double)ar) - 1.0)
    * P + (1.0 + (double)cr)) - 1)] = 0.0;
      } else {
        for (k = 0; k < N; k++) {
          if (OM->data[k] == 0.0) {
          } else {
            S += (ET->data[k] - CO->data[k]) * Wd->data[Wd->size[0] *
              ((int)(((((1.0 + (double)k) - 1.0) * ((double)N * P *
              ((double) N + (double)I)) + ((1.0 + (double)br) - 1.0) *
              P * ((double)N + (double)I)) + ((1.0 + (double)ar) - 1.0)
      * P) + (1.0 + (double)cr)) - 1)];
          }
        }
      }

      dW->data[br + dW->size[0] * ((int)(((1.0 + (double)ar) - 1.0) * P +
        (1.0 + (double)cr)) - 1)] = alpha * S;
    }
  }
}

//  Updating weights and outputs before going to next time sample
i0 = W->size[0] * W->size[1];
matrix_cap((matrix__common *)W, i0, (int)sizeof(double));
ia = W->size[1];
for (i0 = 0; i0 < ia; i0++) {
  cr = W->size[0];
  for (i1 = 0; i1 < cr; i1++) {
    W->data[i1 + W->size[0] * i0] += dW->data[i1 + dW->size[0] * i0];
  }
}

i0 = CO->size[0] * CO->size[1];
CO->size[0] = COL->size[0];
CO->size[1] = COL->size[1];
matrix_cap((matrix__common *)CO, i0, (int)sizeof(double));
ia = COL->size[0] * COL->size[1];
for (i0 = 0; i0 < ia; i0++) {
      CO->data[i0] = COL->data[i0];
    }

    ia = Wd_m->size[1] - 1;
    for (i0 = 0; i0 <= ia; i0++) {
      Wd->data[Wd->size[0] * i0] = Wd_m->data[Wd_m->size[0] * i0];
    }

    l++;
  }

  matrix_free(&b_DT);
  matrix_free(&a);
  matrix_free(&ET);
  matrix_free(&dW);
  matrix_free(&Wd_m);
  matrix_free(&OM);
  *rel_mean_sq = sqrt(SE / ((double)DT->size[0] * (double)DT->size[1]));
}
```