Towards a 3D building reconstruction using spatial multisource data and computational intelligence techniques

Georgios Papadopoulos

Thèse de doctorat

UNIVERSITE DE LIMOGES

ECOLE DOCTORALE SCIENCES ET INGENIERIE POUR

L' INFORMATION, MATHEMATIQUES / S2IM

UNITE DE RECHERCHE XLIM – UMR CHRS No 7252

COMPOSANTE DMI/SIR

THÈSES

Pour obtenir le grade de

DOCTEUR DE L' UNIVERSITE DE LIMOGES

Discipline / Specialite : Informatique

presentee et soutenue par

Georgios Papadopoulos

Le 20 Septembre 2019

Towards a 3D building reconstruction using spatial multisource data and computational intelligence techniques

These dirigee par Professeur Djamchid GHAZANFARPOUR, Professer en Informatique, Faculté des Sciences et Technologies, Université de Limoges

Co-encadrement Professeur Nikolaos VASSILAS, Department of Informatics and Computer Engineering, University of West Attica

Members of Jury

President of jury: M. Jean-Pierre Jessel, Professor at Paul University of Toulouge, Institut de Recherche en Informatique de Toulouse.Reporters.

Reporters

M.Paris Mastorocostas, Professor of Computational Intelligence, Department of Informatics and Computer Engineering, University of West Attica.

M. Jean-Pierre Jessel, Professor at Paul University of Toulouge, Institut de Recherche en Informatique de Toulouse.

Examiners

M.Paris Mastorocostas, Professor of Computational Intelligence, Department of Informatics and Computer Engineering, University of West Attica.

M. Jean-Pierre Jessel, Professor at Paul University of Toulouge, Institut de Recherche en Informatique de Toulouse.

Guests

M. Georgios Miaoulis, Professor of Information Systems & Applications, Department of Informatics and Computer Engineering, University of West Attica.

M. Voulodimos Athanasios, Professor of Informatics, Department of Informatics and Computer Engineering, University of West Attica.

To my wife and parents who supported me throughout this endeavor and to my daughter to whom the future belongs.

Acknowledgements

I would like to express my deep gratitude to my advisors Dr. Djamchid Ghazafanpour: professor of the Department of Informatics at the University of Limoges; Dr. Nikolaos Vassilas: professor of the Department of Computer Engineering at the University of Western Attica; and Dr. Anastasios Kesidis: associate professor of the Department of Surveying and Geoinformatics Engineering at the University of Western Attica, who supported me throughout all stages of my doctoral thesis with their expertise, their generous counseling and by pointing me toward the right direction when results were not as expected.

Furthermore, I would like to thank professor Dr. Georgios Miaoulis professor of the Department of Computer Engineering at the University of Western Attica who trusted me with this doctoral dissertation and for formulating my scientific thought.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

Abstract (English)

Building reconstruction from aerial photographs and other multi-source urban spatial data is a task endeavored using a plethora of automated and semi-automated methods ranging from point processes, classic image processing and laser scanning. In this thesis, an iterative relaxation system is developed based on the examination of the local context of each edge according to multiple spatial input sources (optical, elevation, shadow & foliage masks as well as other pre-processed data as elaborated in Chapter 6). All these multisource and multiresolution data are fused so that probable line segments or edges are extracted that correspond to prominent building boundaries.

Two novel sub-systems have also been developed in this thesis. They were designed with the purpose to provide additional, more reliable, information regarding building contours in a future version of the proposed relaxation system. The first is a deep convolutional neural network (CNN) method for the detection of building borders. In particular, the network is based on the state of the art super-resolution model SRCNN (Dong, Loy, He, & Tang, 2015). It accepts aerial photographs depicting densely populated urban area data as well as their corresponding digital elevation maps (DEM). Training is performed using three variations of this urban data set and aims at detecting building contours through a novel super-resolved heteroassociative mapping. Another innovation of this approach is the design of a modified custom loss layer named Top-N. In this variation, the mean square error (MSE) between the reconstructed output image and the provided ground truth (GT) image of building contours is computed on the 2N image pixels with highest values[1]. Assuming that most of the N contour pixels of the GT image are also in the top 2N pixels of the re-construction, this modification balances the two pixel categories and improves the generalization behavior of the CNN model. It is shown in the experiments, that the Top-N cost function offers performance gains in comparison to standard MSE. Further improvement in generalization ability of the network is achieved by using dropout.

The second sub-system is a super-resolution deep convolutional network, which performs an enhanced-input associative mapping between input low-resolution and high-resolution images. This network has been trained with low-resolution elevation data and the corresponding high-resolution optical urban photographs. Such a resolution discrepancy between optical aerial/satellite images and elevation data is often the case in real world applications. More specifically, low-resolution elevation data augmented by high-resolution optical aerial photographs are used with the aim of augmenting the resolution of the elevation data. This is a unique super-resolution problem where it was found that many of -the proposed general-image SR propositions do not perform as well. The network aptly named building super resolution CNN (BSRCNN) is trained using patches extracted from the aforementioned

[1] N is the number of contour pixels in the GT.

data. Results show that in comparison with a classic bicubic upscale of the elevation data the proposed implementation offers important improvement as attested by a modified PSNR and SSIM metric. In comparison, other proposed general-image SR methods performed poorer than a standard bicubic up-scaler.

Finally, the relaxation system fuses together all these multisource data sources comprising of pre-processed optical data, elevation data, foliage masks, shadow masks and other pre-processed data in an attempt to assign confidence values to each pixel belonging to a building contour. Confidence is augmented or decremented iteratively until the MSE error fails below a specified threshold or a maximum number of iterations have been executed. The confidence matrix can then be used to extract the true building contours via thresholding.

Abstract (Français)

La reconstruction de bâtiments à partir de photographies aériennes et d'autres données spatiales urbaines multi-sources est une tâche qui utilise une multitude de méthodes automatisées et semi-automatisées allant des processus ponctuels au traitement classique des images et au balayage laser. Dans cette thèse, un système de relaxation itératif est développé sur la base de l'examen du contexte local de chaque bord en fonction de multiples sources d'entrée spatiales (masques optiques, d'élévation, d'ombre et de feuillage ainsi que d'autres données prétraitées, décrites au chapitre 6). Toutes ces données multisource et multirésolution sont fusionnées de manière à extraire les segments de ligne probables ou les arêtes correspondant aux limites des bâtiments.

Deux nouveaux sous-systèmes ont également été développés dans cette thèse. Ils ont été conçus dans le but de fournir des informations supplémentaires, plus fiables, sur les contours des bâtiments dans une future version du système de relaxation proposé. La première est une méthode de réseau de neurones à convolution profonde (CNN) pour la détection de frontières de construction. Le réseau est notamment basé sur le modèle SRCNN (Dong C. L., 2015) de super-résolution à la pointe de la technologie. Il accepte des photographies aériennes illustrant des données de zones urbaines densément peuplées ainsi que leurs cartes d'altitude numériques (DEM) correspondantes. La formation utilise trois variantes de cet ensemble de données urbaines et vise à détecter les contours des bâtiments grâce à une nouvelle cartographie hétéroassociative super-résolue. Une autre innovation de cette approche est la conception d'une couche de perte personnalisée modifiée appelée Top-N. Dans cette variante, l'erreur quadratique moyenne (MSE) entre l'image de sortie reconstruite et l'image de vérité de sol (GT) fournie des contours de bâtiment est calculée sur les 2N pixels de l'image avec les valeurs les plus élevées. En supposant que la plupart des N pixels de contour de l'image GT figurent également dans les 2N pixels supérieurs de la reconstruction, cette modification équilibre les deux catégories de pixels et améliore le comportement de généralisation du modèle CNN. Les expériences ont montré que la fonction de coût Top-N offre des gains de performance par rapport à une MSE standard. Une amélioration supplémentaire de la capacité de généralisation du réseau est obtenue en utilisant le décrochage.

Le deuxième sous-système est un réseau de convolution profonde à super-résolution, qui effectue un mappage associatif à entrée améliorée entre les images d'entrée à basse résolution et à haute résolution. Ce réseau a été formé aux données d'altitude à basse résolution et aux photographies urbaines optiques à haute résolution correspondantes. Une telle différence de résolution entre les images optiques / satellites optiques et les données d'élévation est souvent le cas dans les applications du monde réel. Plus spécifiquement, des données d'altitude à faible résolution, augmentées par des photographies aériennes optiques à haute résolution, sont utilisées dans le but d'augmenter la résolution des données d'altitude. Il s'agit d'un problème de super-résolution unique dans lequel il a été constaté que nombre des propositions de SR en image générale proposées ne fonctionnent pas aussi bien. Le réseau CNN (BSRCNN), qui porte bien son nom, est formé à l'aide de correctifs

extraits des données susmentionnées. Les résultats montrent que, par rapport à une élévation bicubique classique des données d'élévation, la mise en œuvre proposée offre une amélioration importante, comme l'atteste une métrique modifiée PSNR et SSIM. En comparaison, d'autres méthodes de SR à image générale proposées ont obtenu des résultats inférieurs à ceux d'un agrandisseur bicubique standard.

Enfin, le système de relaxation fusionne toutes ces sources de données multisource comprenant des données optiques pré-traitées, des données d'élévation, des masques de feuillage, des masques d'ombre et d'autres données pré-traitées dans le but d'attribuer des valeurs de confiance à chaque pixel appartenant à un contour de bâtiment. La confiance est augmentée ou décrémentée de manière itérative jusqu'à ce que l'erreur MSE échoue au-dessous d'un seuil spécifié ou qu'un nombre maximal d'itérations ait été exécuté. La matrice de confiance peut ensuite être utilisée pour extraire les véritables contours du bâtiment via le seuillage.

1. Introduction

There is a widespread demand across multiple business domains for the automatic detection and 3D reconstruction of building boundaries in urban settings from aerial or satellite photographs. Applications, among many, range from urban planning, virtual tourism, transportation navigation and creation of virtual 3D models, which can assist in further models like the propagation of radio waves in an urban environment. These images could be optical aerial photographs or digital surface models (DSM)/digital elevations models (DEM) constructed by point-clouds of LIDAR (Light, Imaging, Detection and Ranging) equipment and taken from fixed-wing aircraft; helicopters; balloons; UAVs and other vehicles. Further pre-processing can then be performed in order to filter out irrelevant patterns by creating foliage or synthetic object maps. The first step towards building a realistic 3D model is the extraction of edge chains that are part of building contours, which is in broad terms the main objective of the thesis.

1.1 Previous work

There are many proposals as to how to extract building contours from aerial /satellite imagery, based on three broad categories. The first category only makes use of monocular optical or elevation data; the second is a fusion of the first two using multiple sources while the third uses 3D image provider datasets. Only the first two cases are examined in this thesis. Early methods of the first category were severely constrained due to their reliance on a generic model, which assumed that buildings follow a certain pattern, and thus failed to provide consistent results when applied to varied urban environments (Mason & Balisavias, 1997). Unfortunately, such models were also hampered due to low-resolution ground sampling data, occlusions and shadows. Other researchers have used photogrammetric techniques which avail of stereoscopic images with several of these methods using optical images while others elevation data. Examples of the former category are Lang (Lang F. , 1996) as well as Fraser (Fraser, Baltsavias, & Gruen, 2002) who reconstructed 3D buildings from high-resolution IKONOS stereoscopic imagery. Hierarchical processing and correlation schemes of optical data were used in Paparoditis et al. (Paparoditis, Cord , & Corquerez, 1998)

A graph-based approach was presented by Kim et al. (Kim & Muller, 1999) who utilized four stages: line extraction; line-relation-graph generation; building hypothesis generation and building hypothesis. They were able to achieve robustness by considering only the mathematical and geometric relations between lines in the course of generating building hypotheses verification. Ok (Ok, Senaras, & Yuksel, 2012) utilized a fuzzy landscape generation approach to model the spatial relation between building and their shadows. They then applied a pruning process to eliminate generated landscapes inconsistent with an urban environment. In addition, a shadow model was used by Peng (Peng & Liu, 2004) in order to extract buildings in monocular urban aerial images. Raw segmentations of buildings were first extracted and verified by the shadow model.

Airborne laser scanning equipment became more reliable and refined during the late 1990s and early 2000s, thus becoming an important source of obtaining digital surface maps (DSM). Mass and Moleman developed two approaches to detecting building contours using DSMs (Mass & Vosselman, 1999), but they were limited to gable roof types. They used pseudo 3D point clouds and their calculated invariant moments to determine the parameters of standard gable roof types. Furthermore, observed point clouds from LIDAR data have been used by Rottensteiner et al. (Rottensteiner & Briese, 2002). More specifically the researchers applied a hierarchical robust interpolation using a skew error distribution function of the point cloud in order to discriminate between points belonging to building contours and others. Another application of point clouds LIDAR building contour detections was presented by Cho et al. (Cho, Jwa, Chang, & Lee, 2004) who introduced a pseudo grid that prevented the loss of information due to interpolation. The pseudo grid then passed through several stages of processing (noise removal; segmentation; grouping for building detection; linearization and simplification of building boundary) which resulted in a 3D model. A segmentation approach to point cloud data was demonstrated by Ramiya (Ramiya, Nidamanuri, & Krishnan, 2017) who applied a novel histogram based methodology to separate the building clusters from non-building ones with very good accuracy. Yet another method based on point clouds but this time specifically aiming to improve the detected building edges by removing jagged contours was presented by Mineo (Mineo, Pierce, & Summan, 2019). Their approach used dynamic thresholds to detect points belonging to sharp edges and creases by applying FFT based reconstruction. This eliminates the need for the predefinition of a specific polynomial function order for optimum polynomial curve fitting, according to the authors.

Probabilistic methods that avail of digital elevation models (DEM) have also been presented. For instance Ortner et al. (Ortner, Descombes, & Zerubia, 2007) used marked point processes and an energy function, which took into account the height of the building as well as prior knowledge about the general layout of buildings in urban settings. Simulated annealing was then employed in order to minimize the energy function. A variation of the previous method was presented by Lafarge et al. (Lafarge, Descombes, Zerubia, & Pierot-Deseilligny, 2008) which again used marked point processes to roughly approximate building contours via rectangular structures. These rectangular footprints were then regularized by taking into account the local context of each rectangle and detecting roof height discontinuities.

Fusing optical and elevation data was early on another promising avenue to examine, thus Haala et al. (Haala & Nrenner, 1999) combined altimetry data with multi-spectral images in order to extract buildings and trees in an urban environment. They combined this extraction, in a second step, with 2D ground plan information in order to obtain a 3D reconstruction. Similarly, Rottensteiner (Rottensteiner F. &., 2003) integrated LIDAR cloud point data with aerial images. They firstly detected building regions from the point cloud and applied a curvature-based segmentation technique to identify roof planes, which were then grouped to create polyhedral building models. Also, Sohn (Sohn, Sohn, & Dowman, 2007) used high-resolution IKONOS multispectral imagery with low-sampled airborne laser scanning. They initially detected building objects by investigating the height property of the point cloud and the normalized index vegetation indices (NDVI) from the IKONOS data.

More recently, deep convolutional networks have been added to the plethora of methods aiming to extract building contours from aerial/satellite images of various sources. For example, Yuan (Yuan, 2017) demonstrated a deep convolutional network that aimed to detect building contours. The author created a simple deep neural network model that integrated activation from multiple layers for pixel wise prediction. He used a signed distance function of building boundaries to represent output. Similarly, Alshehhi (Alshehhi, Marpu, Woon, & Dalla, 2017) presented an 8-layer deep convolutional network that extracted roads and building contours from high-resolution remote sensing data. They used a post-processing stage on the output of the convolutional network to integrate the low-features of the roads & buildings with those of the network. Finally, Vakalopoulou et al. (Vakalopoulou, Karantzalos, Komodaki, & Paragios, 2015) used single very high-resolution satellite images to build an automated building detection framework with a deep convolutional neural network. The network was trained using a supervised classification procedure

The proposed solution is an expansion of the third category and aims to reconstruct building contours using a fusion of multisource spatial data with the aid of computational intelligence and classic image processing techniques. All primary data used for this research were courtesy of the Archimedes III research program[2] and In particular, kind amenities have to be given to GeoIntelligence for providing the DSM and DTM elevation data as well as the National Cadastre & Mapping Agency of Greece for providing the high-resolution aerial photographs. In addition to the available data, two additional sub-systems were created for this research, a) a deep convolutional network, which extracts building, contours from multisource spatial data; b) a super resolution deep convolutional network augments the resolution of our data using optical & elevation data. These systems offer additional post-processed versions of the available data, which could prove important for a future version of the iterative relaxation system. Finally, the iterative relaxation system is constructed per se. This system accepts a plethora of multi-resolution spatial data with the aim to render a faithful building contour.

[2] Archimedes Research Program with title “Intelligent Pattern Recognition Techniques for the Development of Multimodal Representations of Urban Areas”, co-funded by the European Union (European Social Fund) and Greek national resources.

1.2 Objectives of the thesis

The objective of this work is to employ spatial multisource and multiresolution urban data towards an improved 3D building modeling. It explores image processing and pattern recognition techniques in order to determine salient features of the input data that could efficiently be used to obtain a 3D reconstruction. For this purpose, the multisource and multiresolution data were fused so that probable line segments or edges could be extracted that correspond to prominent building boundaries.

In order to achieve these objectives an iterative relaxation system was developed which accepts the multisource and multiresolution input. The aim of this system is to assign a confidence value to each pixel of the final image, which is a direct measure of how certain the system is that it corresponds to a true building boundary. In order to do this the system performs some one-time pre-processing and then enters into an iterative process which increments/decrements the confidence of each pixel according to the local context of the confidence matrix. The iterative process is terminated after a predetermined number of iterations and Otsu's method is used to binarize the image so that only pixels belonging to true edges are retained.

Two further systems were developed[3], that can operate autonomously or offer further post-processed data to a future version of the proposed iteration system. The first one is a deep neural network, which detects building boundaries by directly generating a real valued ([0..1] range) building contour image and that is named BCDCNN. It implements a super-resolved heteroassociative mapping, since low-resolution elevation data are mapped onto their associated high resolution building contours available during training from the ground truth data. The model is comprised of 3-layers, which perform in succession: a) building features patch extraction; b) non-linear map transformation; c) building contour reconstruction. The network then creates a building contour image.

Besides the building contour detector, a super-resolution deep convolutional network was specifically trained in an attempt to augment the resolution of digital elevation maps depicting urban areas. Similarly, to the previous implementation this model performs a mapping between the low-resolution image and high-resolution counterpart. However, this network now performs a hybrid auto-associative mapping between these two images, in the sense that the network performs the mapping assisted by corresponding high-resolution optical data. Elevation data are usually of lower resolution than the corresponding optical data and there is great need for such applications. Furthermore, elevation data have some intricacies of their own rendering generic super-resolution techniques inappropriate. The network was trained with various configurations and with the same urban data set that the previous network used. Results show that in comparison with a classic bicubic upscale of the

[3] These systems can also operate independently as a building contour detector and for optical images depicting urban areas super-resolution.

optical data this implementation offers important improvement as attested by a modified PSNR and SSIM metric.

In summary, the objective of the thesis is to create a relaxation system that when given all these multisource input data will produce a reconstruction of the real building contours more accurately than any of the previous methods described used in isolation. In order to assist the relaxation iterative method a great deal of work was done towards the development of the two deep neural networks BCDCNN AND BSRCNN.

1.3 Areas of contribution

The main contributions of this thesis are in the following areas:

1. Building reconstruction using optical aerial imagery & LIDAR elevation data:

   Obtaining an accurate building contour image given a) an optical aerial/satellite image; b) LIDAR elevation data; c) a combination of both is a very difficult problem to automate. This problem has been tackled with varying levels of success by many researchers as presented in the introduction. Nonetheless, there remains significant room for improvement.

   The approach followed in this paper is twofold. Firstly, a deep convolutional neural network has been applied which does not operate as a classifier but attempts to directly derive a building contour image from the available aerial and elevation data. Using two channels of data has the advantage of overcoming most difficulties that arise from the usage of a single image. For instance, the contrast of an image severely affects methods based solely on optical data. Despite this, the problem remains difficult because as explained in detail later in this dissertation not all) optical edges are also elevation edges and vice versa while there also exist cases of implied edges which are the most difficult case of all. Our solution differs from other proposals because it does not operate as a classifier but directly derives an output image, which depicts building boundaries. The deep building contour detector convolutional neural network (BCDCNN) performs remarkably well given the amount of available training data and proves that CNNs are not only suitable for classification.

   Secondly, an iterative relaxation system has been developed. This is based on the implementation by Prager (Prager J. , 1980) who developed a system that extracted line segments as connected sets of edges, labeled them, and computed features for them such as length and confidence. This initial implementation has been much expanded taking into account multiple sources of spatial data. It begins by applying a modified Canny operator to the optical image depicting a heavily populated urban area. The output of this operator is given as input to the relaxation system, which begins by performing one-time pre-processing on this building boundaries image according to the elevation data, the shadow masks and the foliage masks. The result is an initial confidence matrix with values in the range [0..1], where 0 denotes total confidence that a pixel does not belong to a building boundary and 1 that full confidence that it does. The system then enters into an iterative process, which takes into account the local context and more specifically their gradient directions in order to augment/decrement the confidence of an edge. A binary image is then derived using Otsu's method to calculate a cutoff threshold.

2. Urban area imagery super-resolution.

   Super resolution has been extensively researched during the previous years with methods ranging from sparse dictionary representations to deep convolutional neural networks using one or multiple images as source. This thesis examines how deep neural networks can be trained using high-resolution optical aerial/satellite images in conjunction with low-resolution DEM maps obtained from the processing of LIDAR data. This combination of high-resolution and low-resolution data combos is often the case in real world applications despite the falling cost of LIDAR equipment. More specifically, the work done aims to increase the resolution of low-resolution elevation data given corresponding high-resolution optical data of urban areas.

PART I - DEEP CONVOLUTIONAL NETWORKS

2. BACKGROUND ON NEURAL NETWORKS

Artificial Neural Networks constitute a parallel-distributed computing model, which has been inspired from the functioning of the mammalian cognitive system. They intend to mimic the highly distributed processing and representation of the mammalian brain. The following Table depicts the main differences between artificial neural networks and classical computing systems.

Table 1 - von Neuman vs Neural Networks

	von Neuman computing systems	Artificial Neural Networks
Parallel processing	Limited	Yes
Fault tolerance	Limited	Yes
Graceful degradation	No	Yes
Generalization	No	Yes

Classical von Neuman architecture has limited parallel processing capabilities, expensive fault tolerance solutions, generally does not gracefully degrade and proposed solutions cannot generalize. On the contrary, artificial neural networks are inherently parallel processing, have fault tolerance even if many neurons fail, degrade their performance gracefully when part of the network fails and can generalize much better than hand-crafted computing models.

2.1 Simplified Neural Network Model

A simplified model of a mammalian brain as proposed by McClelland (McClelland, Rumelhart, & PDP Reserach Group, 1987) is comprised of eight major elements:

- A set of processing units.
- A state of activation.
- An output function for each unit.
- A pattern of connectivity among units.
- A propagation rule for propagating patterns of activities through the network.
- An activation rule for combining the inputs impinging on a unit with the current state of the unit to produce a new level of activation for the unit.
- A learning rule whereby patterns of connectivity are modified by experience.
- An environment within which the system must operate.

An analysis of each element appears in Tables 2-9.

Table 2 - Neural Network Architecture (processing units)

Set of processing units
These units are simple abstract elements over which meaningful patterns can be assigned as a distributed representation. A simple processing unit carries no meaningful information; it is the pattern as a whole, which holds meaningful content. This is in contrast to one-unit-one-concept representational systems, in

which a single unit represents entire concepts. A unit acts on receiving weighted inputs from its neighbors and transmits an output signal according to its activation function. Classical neural networks employed three layers input, hidden and output due to the computational complexity of adding further layers. Modern deep network models have overcome this limitation due to the availability of powerful GPU architectures. During the past ten years power powerful GPUs have enabled the training of very deep neural networks These GPUs can be pooled or used for distributed training of deep neural networks enabling the development of models previously thought impossible. Deep networks employed up to 1000 layers in 2017 **(Fog, 2019)**. Recently focus has shifted from the number of layers and processing units that a deep neural network features to the examination of idea of using a block as a structural unit instead of a layer **(Khan, Sohail, Zahoora, & Qureshi, 2019)**.

Table 3 - Neural Network Architecture (state of activation)

State of activation
The representation of the state of the system at time t, which is specified by a vector N of real numbers $a(t)$. Each element of the vector stands for the activation of one of the units at time t. The pattern of activation over the set of units captures what the system is representing at any time. However, for practical analytical purposes only the status of the output layer neurons are utilized since all intermediate layers represent the computational details of this output. Activation values can be continuous or discrete, bounded or unbounded.

Table 4 - Neural Network Architecture (output of the unit)

Output of the unit
Units interact by transmitting signals to their neighbors. Associated with each unit $u_i(t)$, is an output function, $f_i(a_i(t))$, which maps the current state of activation $a_i(t)$ to an output signal $o(t)$. Usually f is a sort of threshold function so that a unit has no effect on another unit except if its activation exceeds the threshold.

Table 5 - Neural Network Architecture (patterns of activation)

Pattern of Connectivity
Units are connected to one another, which produces a pattern of activation that constitutes what the system knows and determines how it will respond to any arbitrary input. Each unit contributes to the input of the units to which it is connected. Thus, the total input to a unit is a weighted sum of each individual input. The weights between the neurons of layer i and layer j are usually represented as matrix w_{ij} whereby each entry represents the strength of the connection between unit i and j.

Table 6 - Neural Network Architecture (propagation rule)

Propagation rule
This rule accepts the outputs $o(t_j)$ for every neuron j of a specific layer and combines it with the weight matrices to produce the net input provided to the succeeding layer. It accepts the outputs $o(t_j)$ for every neuron j of a specific layer and combines it with the weight matrices to produce the net input provided to the succeeding layer.

Table 7 - Neural Network Architecture (activation rule)

Activation rule
The rule whereby the net inputs impinging on a particular unit are combined with one another and with the current state of the unit to produce a new state of activation. The rule whereby the net inputs impinging on a particular unit are combined with one another and with the current state of the unit to produce a new state of activation.

Table 8 - Neural Network Architecture (learning rule)

Learning rule
The weights connecting the outputs of neurons of a previous layer to the input of the next layer are modified according to the expected output through this algorithm. The most commonly used algorithm used today is the backpropagation algorithm, which commences execution at the final output layer and back propagates the error differences according to the chain rule.

Table 9 - Neural Network Architecture (operational environment)

Operational environment
The environment is represented as a time-varying stochastic function over the space of input functions. The environment is represented as a time-varying stochastic function over the space of input functions.

Neural networks can be trained using the backpropagation (BP) algorithm, which was developed in a non-neural network context the 1960s and elaborated in the 1970s. Paul Werbos was the first to discover how the back-propagation algorithm could be used to train multi-layer neural networks during his doctoral thesis in 1974 but he did not report on his findings due to the AI-winter context of the period. He did publish on it in 1982. The classic BP algorithm has evolved since then to more complex implementations like the ADAM and RMSProp, which converge to a solution much faster.

2.2 Historical Overview

Artificial neural networks first appeared in the 1940s. What follows is a brief history of major milestones and developments.

- McCullough and Pitts researched in 1943 the operation of a biological neuron and the result of their work was the first mathematical models of a neuron. As shown in Figure 1, a biological neuron is comprised of a) dendrites: which accept input from other neurons; b) soma: in which protein synthesis occurs; c) axon through which the output of a neuron is transmitted to dendrites; d) and the synapse which controls how much of the output of a neuron will reach the dendrite of another.

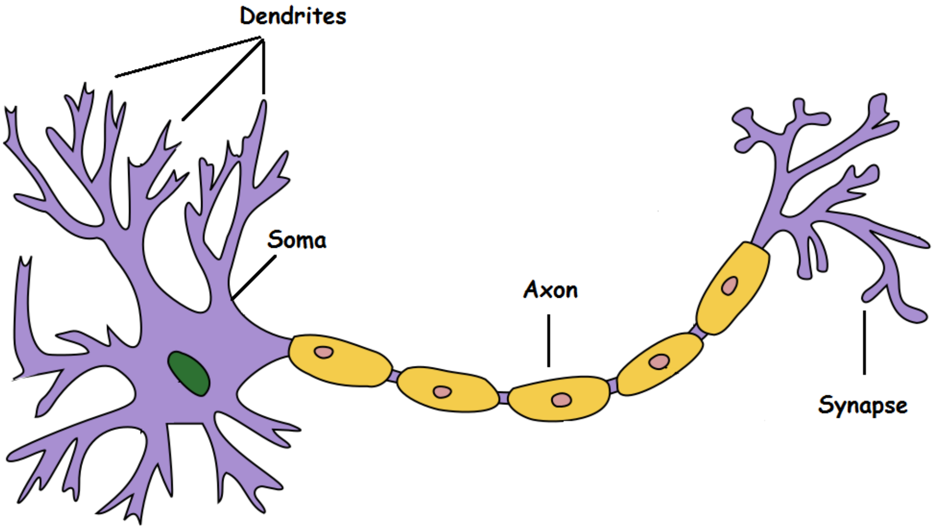


Figure 1 - A simple model of a biological dendrite (source -wikipedia)

The researchers presented a neuron model which summed binary inputs, outputting a logical 1 if these inputs exceeded a threshold or 0 otherwise. By using such a model of neuron, he and his colleagues designed mathematical models of the basic AND/OR/XOR logical functions. However, their model offered no mechanism for learning something, which limited its applications. A depiction of their model can be seen in Figure 2. As shown in the figure, it is composed of two parts. The left part named *g*, accepts a number of inputs $\mathrm{x_n} \in \{0,1\}$ and performs an aggregation (calculates the sum). The right part named f is a function$\mathrm{f}(\sum_1^n x_n)$, which outputs either a positive (1) or negative (0) decision based on the inputs.

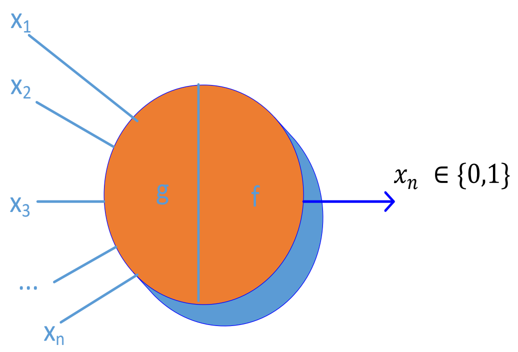


Figure 2 - McCullough & Pits neuron model

McCullough defined two categories of input: a) inhibitory, which force the neuron not to activate when they are active b) excitatory, which contribute to whether the neuron will activate. As an example, let us use a simple neuron to make a decision on whether I should go for a walk or stay inside. Three excitatory variables determine whether the neuron will activate as follows:

- ○ x_1 (excitatory): Is it raining? (0: not raining, 1: raining)
- ○ x_2 (excitatory): Is it warm outside? (0: no, 1: yes)
- ○ x_3 (excitatory): Am I feeling well? (0: no, 1: yes)

The neuron will accept these inputs and calculate their sum. Expressed mathematically:

$$\boldsymbol{g(x) = \sum_{i=1}^{N} x_i} \qquad (1)$$

, which is a simple sum of the input. The neuron activates according to whether the sum of the inputs exceeds a threshold θ. In mathematical terms:

$$\boldsymbol{y = f\big(g(x)\big) = 1, if\ g(x) \geq \theta} \qquad (2)$$

$$\boldsymbol{= 0, if\ g(x) < \theta}$$

- 1954: Donald Hebb and the IBM research group presented the first simulations of the McCullough and Pitts model. What has since been called the Hebb rule, stated '*When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased*'. The revolutionary idea that learning occurs in the brain through the formation and change of synapses in the brain has had a deep impact on the development of neural networks.
- 1958: Frank Rosenblatt's perceptron: Rosenblatt expanded on the work of Hebb by adding weights to the inputs of a neuron and introduced a learning

mechanism for his perceptron. Given a training set of input-output pairs the 'Perceptron' learns via increasing the weights if the output for a specific input is too low compared to the expected output and vice versa. The perceptron is a more generic computational model that was proven capable of solving linearly separable problems. It is depicted in Figure 3.

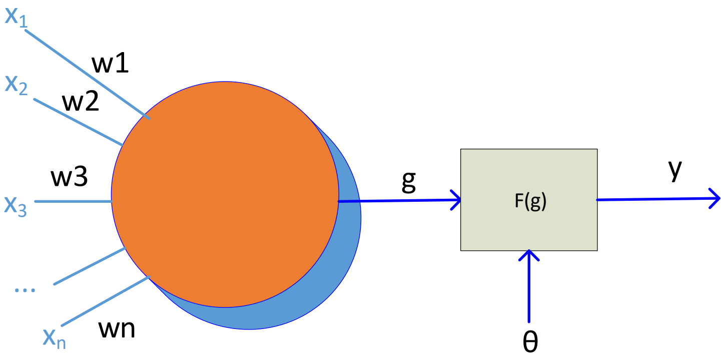


Figure 3 - Perceptron model

Mathematically the perceptron can be described as:

$$\boldsymbol{y = \sum_{i=1}^{N} f(w_i * x_i) + \theta} \tag{3}$$

Just like the neuron proposed by McCullough & Pits, the perceptron will activate if$y \geq \theta$, where θ is a pre-defined threshold. The most important difference compared to the McCulloughs-Pits neuron is that a perceptron uses synaptic weights with which each input is multiplied. By adjusting the weights the perceptron can learn to activate or not according to the presented input. A more practical model of the perceptron is depicted in Figure 4. In Figure 4, the threshold θ is passed as a synaptic weight to a unit input. This simplifies the model since now it will only activate if y > 0.

- 1969: Marvin Minsky proved that the Perceptron learning mechanism had severe limitations (Minsky & Papert, 2017). This was mostly a reaction by scientists engaged in algorithmic artificial intelligence systems who remained skeptical at all the fervor regarding artificial neural networks. His most notable criticism was the failure of the Perceptron to model the simple XOR logical function because it is not linearly separable. This publication ushered in an era of disenchantment with neural networks and artificial intelligence in general, aptly named the *AI Winter*.

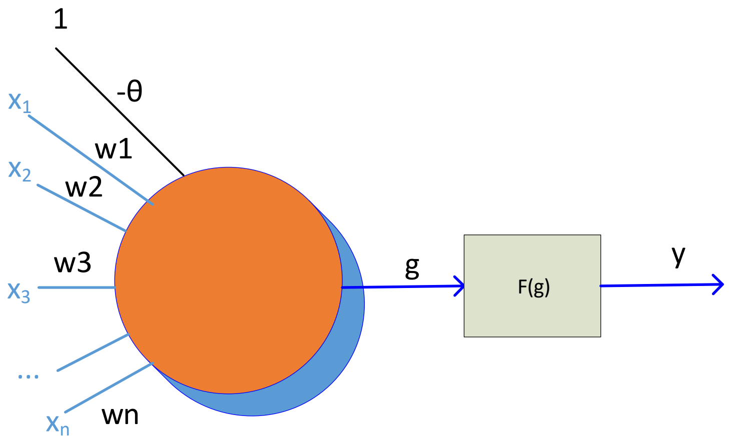


Figure 4 - Perceptron generic model

- 1985 - Reemergence of interest in neural networks: The handicaps that Minsky had presented back in 1969 were shown to be overcome through the usage of multi-layer networks and the back-propagation algorithm in order to train them. The hidden intermediate layers could find features within the training data allowing following layers to operate on this intermediate representation than on the initial raw data. For example, in a 4-layer fully connected layer designed to categorize human faces, the first hidden layer could find abstract features of the input images like circles, rectangles and other basic geometric elements while the second hidden layer could combine these and enable the output layer to categorize the faces according to their ethnicity. Nonetheless, progress was cumbersome and slow due to the inadequacy of the available hardware. Training requires many simple processing elements working in tandem, something that would not be massively available for another two decades.

2.3 Neural networks

The late 1950s and until Minsky's paper, which proved that a single-layer neural network could not solve non-linearly separable problems, was a period in which many developments occurred regarding artificial intelligence. There was great faith that AI would solve many engineering problems and that AI cognition would surpass human natural intelligence. Minsky's seminal work ushered in an era of stagnation for AI, which would last until the early 1980s. That is not to say that important work was not being done, since many multi-layered neural networks were being developed during the late 1970s and early 1980s. Fukushima for instance, developed the first convolutional neural network in 1980 (Fukushima, 1980). This area was characterized by a disenchantment and a loss of belief that AI would solve many of the important challenges of the era. In any case, AI started to reemerge as a credible discipline during the 1980s and by mid 1980s, multi-layered neural networks were being once again successfully used for many applications.

A typical multi-layered neural network is shown in Figure 5.

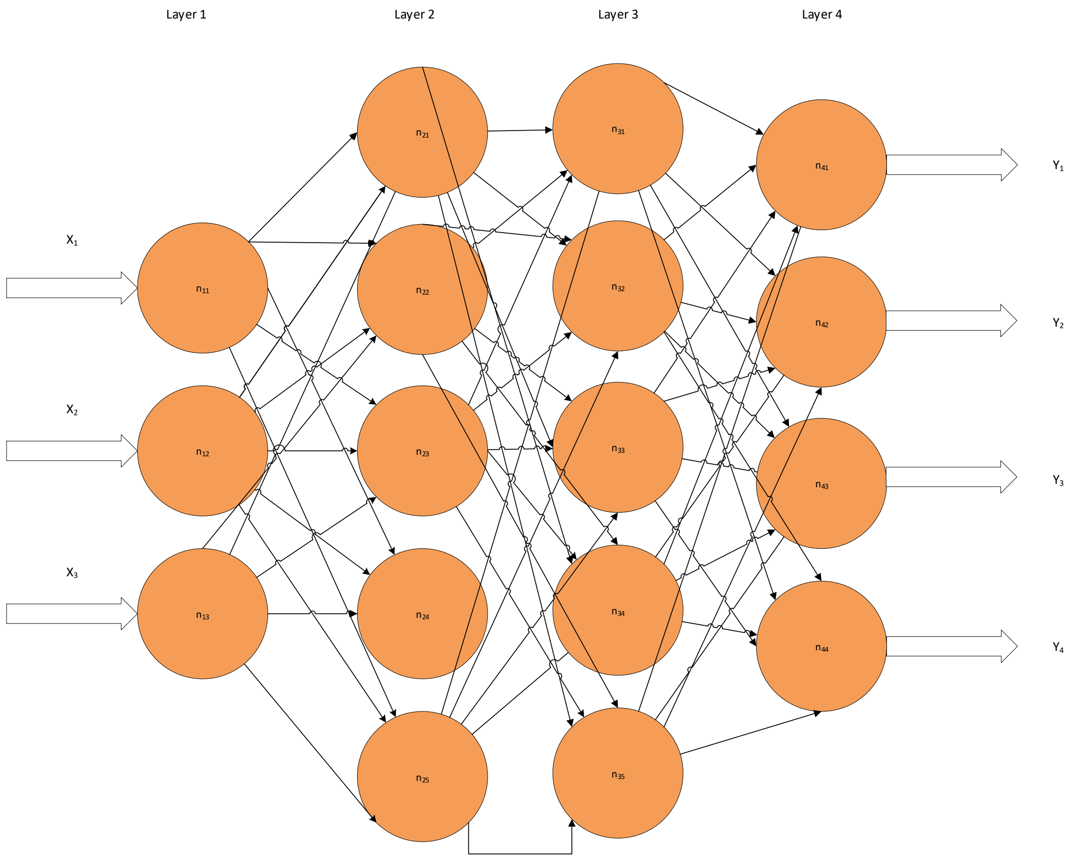


Figure 5- A simplified model of a fully connected neural network (synaptic weights not shown)

As seen in Figure 5, classical neural networks consist of many simple, connected processors, called neurons, each producing a sequence of real-valued activation. The number of layers comprising these networks were few due to processing power limitations of the era. They are now classified as *shallow networks* according to Fog et al. (Fog, 2019).

2.3.1 The neuron as a basic computing element

The basic computing unit of a neural network is called a neuron (depicted as a circle in Figure 5), which models the operation of a physiological neuron. A typical artificial neuron is depicted in Figure 6, where it can be seen that it simply sums the weighted inputs up. It then adds a bias (threshold) and changes its status according to a non-linear activation function ϕ (in order to simplify the model, the bias (threshold) is usually added as further input to the summation with a value of 1 as depicted in Figure 4). There are many variations of the activation functions, most of which are based on the functioning of biological neurons. Biological neurons can be either active or inactive. When active, they transmit an analog signal of varying intensity.

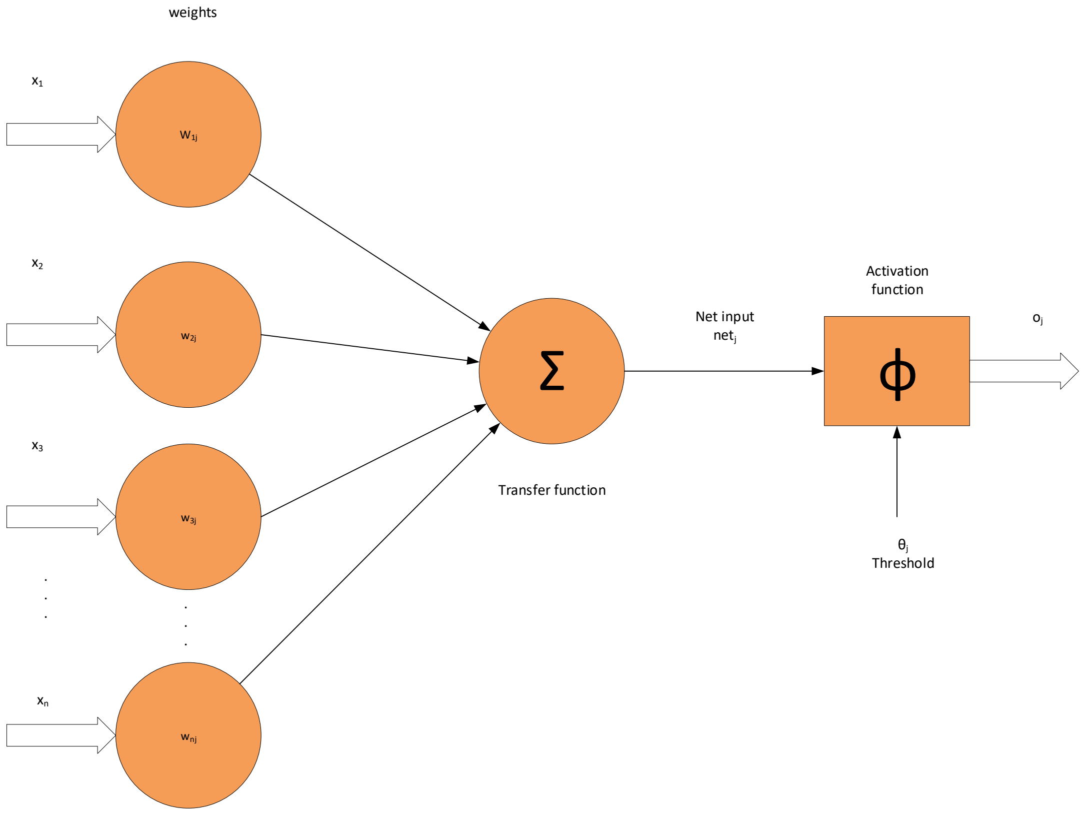


Figure 6 - Analytic model of a neuron

In the following section, the most typical activation functions are elaborated (Ramachandran, Zoph, & Le, 2017), (Sharma, 2019):

2.3.2 Activation functions

- Step function: The simplest of the activation functions, which activates simply if the weighted sum of all inputs is above a specified threshold. This function operates well for a binary classification problem but fails when a categorization problem has multiple categories. Non-linearly separable problems cannot be solved with a single neuron and this activation function. The plot of the step function can be seen in Figure 7.

- Linear activation function: Theoretically, this function solves the multiple categorization failure of the step function. Unfortunately, the linear function given by the mathematical expression $Y = cX$ has another problem. When moving on to the back-propagation stage, this produces a constant derivative regardless of the input X which would in turn cause the weights of the network to be updated by a constant factor in every iteration and no learning would be possible. Another important problem of this activation function is that if two or more layers are connected via linear activation functions then these layers can be collapsed to a single linear layer with the result that the network would lose its structure.

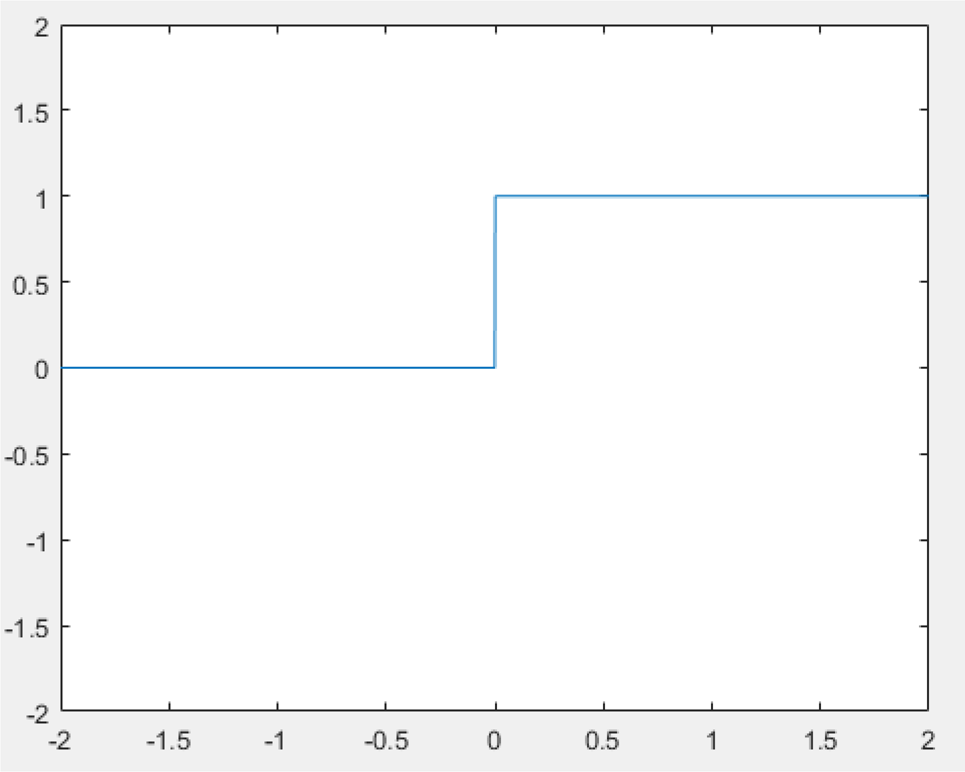


Figure 7 - step activation

- Sigmoid activation function: The sigmoid function was heavily applied in classical neural networks and with good reason. The mathematical notation for the sigmoid is $\varphi(z) = 1/(1 + e^{-z})$ and its graph is given in Figure 8. This function exhibits several very agreeable characteristics. Firstly, it performs well as a binary classifier as it has the tendency to push the Y values towards the edges of the graph (0 or 1). Secondly, it is nonlinear, which means that the layers of the neural network can be stacked one after another without bringing about the collapse of its structure. Lastly, the output is always in the range [0..1] as opposed to the linear function.

  Unfortunately, there is also a significant drawback. Notice that the function has a near linear operating region between the ordinate range [-2..2]. Outside that region large changes in the X value lead to very small changes in the Y values, something which means that the neural network will stop learning in that region.

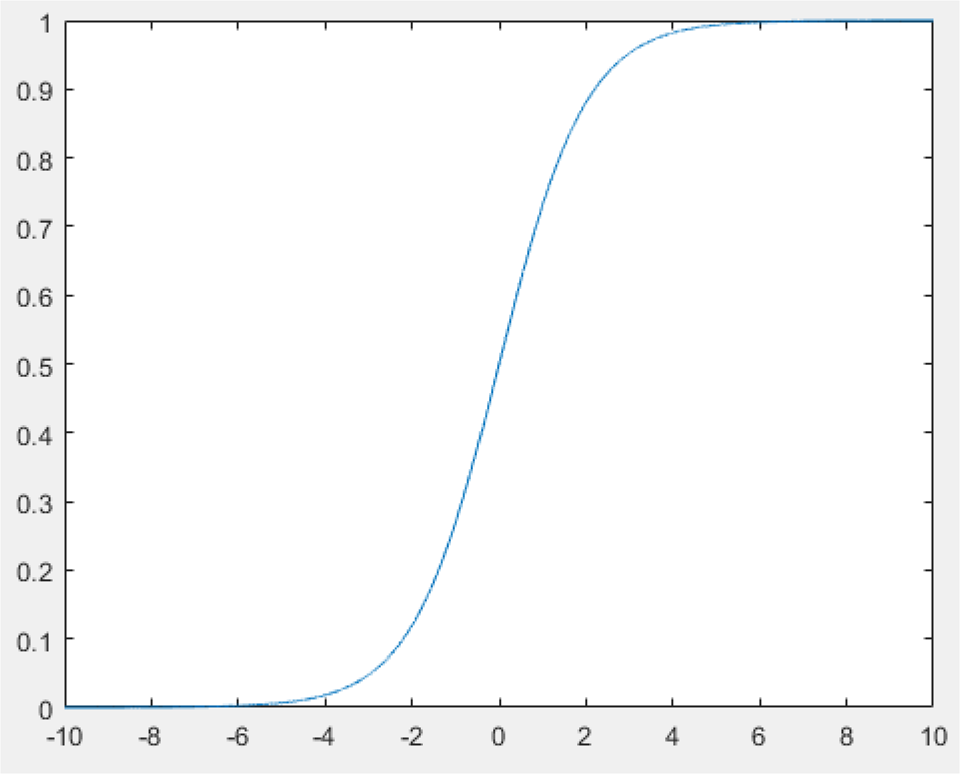


Figure 8 - sigmoid activation function

- Hyperbolic tangent function (Tanh): This activation function has many similarities to the sigmoid function. As can be seen from Figure 9, it is bounded to the range [-1..1] which guarantees a stable learning process in the linear operating region of the function. The gradient is also slightly steeper than the sigmoid, which can lead to faster convergence of the learning process. Nonetheless, it suffers from the same constraints as the sigmoid as the neural network stops learning outside the linear operating region.

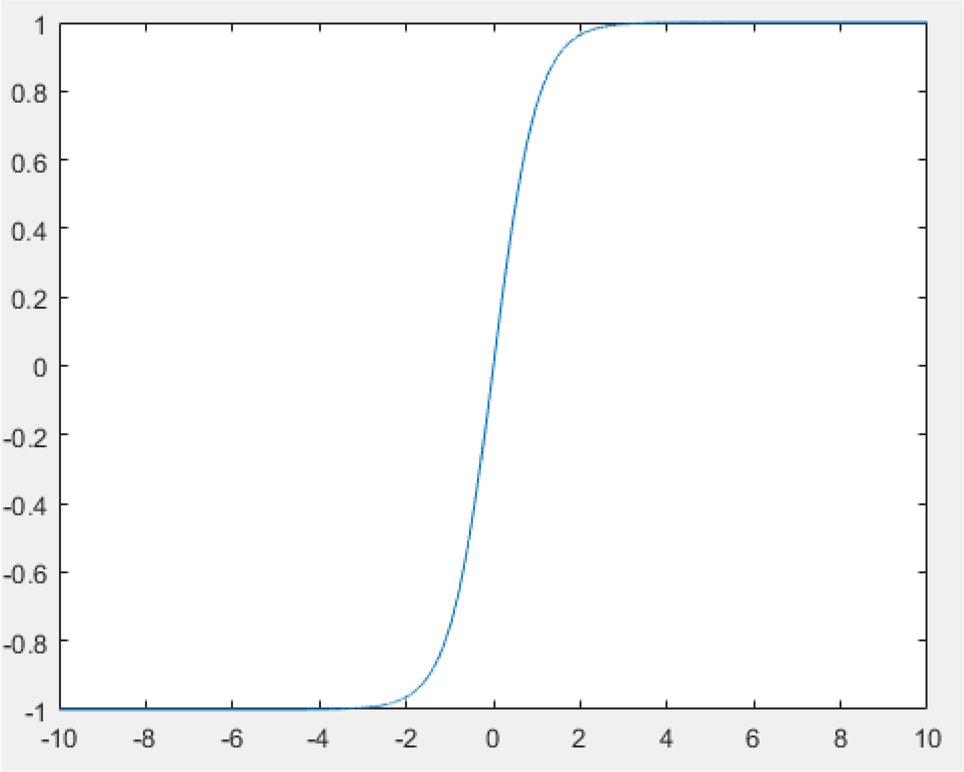


Figure 9 - Tanh activation function

- Rectified Linear Unit (ReLU): The ReLU is a rather modern activation function that offers the best of both worlds that the step function and the sigmoid function have to offer. It is linear in nature in the positive X axis but returns zero for negative values and is shown in figure 64. The benefits of using this function are its simplicity since it is linear for positive but non-linear for negative values, which means that layers connected via this function can be stacked up without the network's structure collapsing. A final significant benefit of the ReLU is that it leads to sparse activation in comparison to the sigmoid or Tanh. Consider a case where the initial weights of the neural network are randomly initialized. This will lead to 50% of the neurons to be initially inactive. After training, a specific representation will lead to a sparse activation of neurons. This is because of the nature of the ReLU activation function. As can be seen in Figure 10, all neurons whose weighted summed inputs are less than zero, will be rendered inactive. This leads to situation where half of the neurons in the hidden units will be inactive, as Bengio pointed out in his work (Bengio, Bordes A., & Glorot X., 2011). In comparison, a corresponding representation for Tanh or sigmoid will lead to a much denser activation of hidden neurons.

  The classic ReLU can however lead to the dying neurons problem, wherein when the gradient goes towards 0 the weights will stop adjusting. Such neurons will stop responding to the variations of the error function. Leaky ReLU bypasses this problem by allowing for a small incline in the output for negative values so that the gradient never is zeroed out and the learning process can continue.

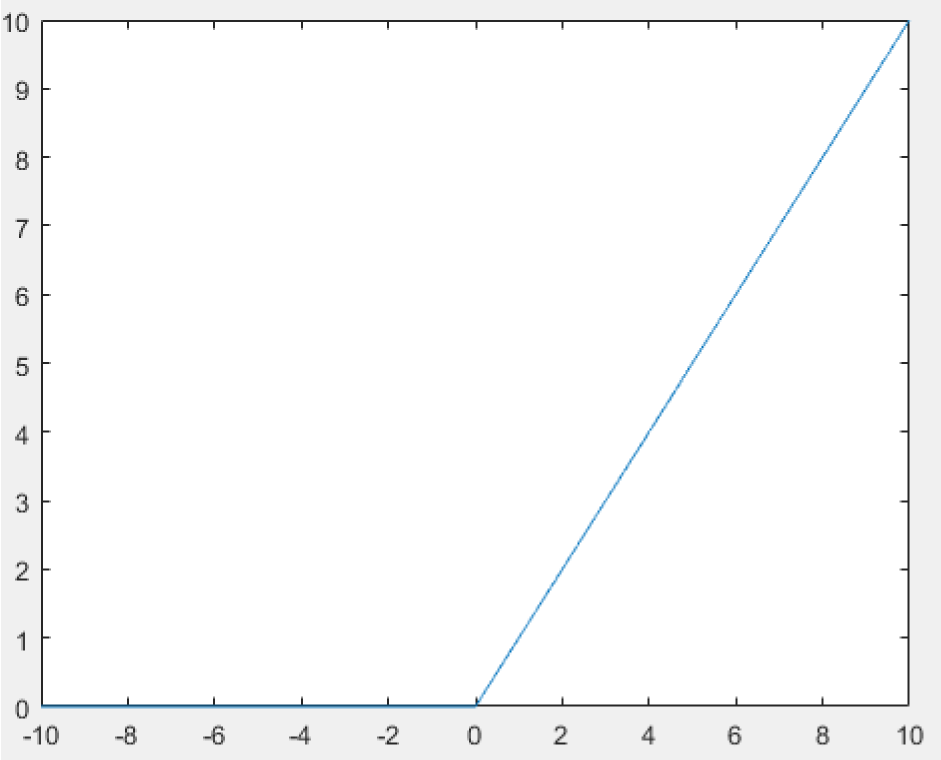


Figure 10 -ReLU activation function

2.3.3 Perceptron Learning Algorithm

The goal of the Perceptron learning algorithm is to adjust the values of the elements of the weight vector $\boldsymbol{w} = \{\boldsymbol{w_1}, \boldsymbol{w_2}, \dots, \boldsymbol{w_n}\}$ with which the input vector $\boldsymbol{x} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_n}\}$s multiplied (dot product), so that the neural element can learn to classify correctly positive and negative inputs. For this reason, a supervised learning algorithm is applied through which the weights are adjusted in a way so that the output of the neuron is as close as possible to a given real value L. A scaled multiple of the difference between the output of the neuron and L is used to adjust the weights. The full algorithm is given in Table 10.

Table 10 - Perceptron training algorithm

```
Initialize w randomly;
$\boldsymbol{c_s} = \{\boldsymbol{TE_1}, \boldsymbol{TE_2}, \dots, \boldsymbol{TE_n}\}$
While $size(C_s) > 0$ {
  i = random($size(C_s)$);
  $TE = c_s(i)$;
 y = w.x(TE); // this is the dot product
if positive(TE) & y > 0 {
  $C_s.remove(TE)$;
  break;
}
else If positive(TE) & y <= 0 {
    w = w + x(TE);
  }

 If negative(TE) & y < 0 {
  $C_S.remove(TE)$
 }
 else If negative(TE) & y $\geq 0$ {
    w = w – x(TE);
  }
 }
```

The algorithm commences by randomly initializing the weights. It then creates a set of all training exemplars before entering into a loop that will adjust the weights until all training exemplars have been correctly classified. Once in the loop it randomly extracts a training exemplar and examines if it has been correctly classified. There are two cases:

- The exemplar is positive: A positive exemplar is correctly classified if the neuron outputs a positive value. If this is the case, the training exemplar is removed from the set and another iteration of the loop follows. Otherwise, the weights of the neuron are incremented by the value of the input for that exemplar.

- The exemplar is negative: A negative exemplar is correctly classified if the output of the neuron is negative. In this case, the exemplar is removed from the exemplar training set. Otherwise, the weights of the neuron are decremented by the value of the input for that exemplar.

The algorithm continues until all exemplars are correctly classified.

3. Deep learning in neural networks

The emergence of powerful Graphics Processing Units (GPU) in the past decade as well as the development of distributed computing frameworks like CUDA and OpenCL have enabled the deployment of much more powerful neural networks consisting of a multitude of layers in contrast to the typical 3 fully connected layers (input-hidden-output) of classical neural networks. The GPUs can operate in parallel and/or a distributive manner. Such networks have been called Deep Neural Networks[4].

A deep neural network maps inputs to outputs by finding correlations between them. Essentially, for input vector x it seeks to approximate the unknown function $y = f(x)$ which fully describes the relationship between input and output. In addition, it has been described as a universal approximator, because assuming a causal or correlational relationship does exist, a deep neural network will find it (Nicholson, 2019).

3.1 Deep neural network applications

Some of the most important applications of deep neural networks (DNN) are:

- Classification: The process through which a DNN learns to find abstract similarities between images and assigning them a label. This is done through supervised learning, whereby the network is presented a training set of exemplars, which have been labeled by an expert or through crowd labeling efforts. A class of DNNs called convolutional neural networks (CNNs) are especially adept for this application due to their ability to learn the filters that classic image processing methods would have to apply in order to achieve a result of similar level. Classification applications range from face detection, object identification, voice recognition and hate speech recognition.
- Clustering: Clustering is the process of identifying similarities between objects of a data set. In algorithmic pattern recognition, it would be done via the k-means algorithm and its variants. This process does not require labelled data sets and is thus a case of unsupervised learning. Given large data sets, deep neural networks trained using unsupervised learning can produce very accurate models. Typical applications are sound & documents retrieval based on similarity metrics.

  Besides detecting similarities, deep neural networks can perform the opposite. They can find outlying members of a data set. There have been examples in the literature of deep neural networks trained to detect out of baseline performance on a computer network. This can be an indication of

[4] Although the term deep network was initially proposed in the 1970s, it did not receive any widespread usage until 2006.

a denial of service attack or other illicit behavior. For instance, (Tang T.A, Mhamdi, McLernon, Zaidi, & Ghgho, 2016) trained a deep neural network for flow-based anomaly detection in an SDN environment. They thus demonstrated an effective NN intrusion detection system.

- Regression: Neural networks can also be used in a more generic context to predict future events. For instance, (Nicholson, 2019) states that '*deep learning is able to establish correlations between present events and future events. It can run regression between the past and the present*'. Regression applications of deep neural networks are especially prominent in the health sciences. Stroke predictors, kidney failure predictors and heart attack predictors are among the few of many similar applications. For instance, in (Lee, Kim, Kim, & Kang, 2017), a review was conducted on the then current state of play of neural network imaging. The authors found that in most stroke cases examined, neural networks had comparable or better performance than health professionals and/or more established prognostic solutions.

3.2 Deep neural networks

3.2.1 Deep convolutional neural networks

Convolutional Neural Networks are a form of deep network that was initially inspired from the organization of the visual cortex of the human brain. It is especially tuned to the processing of data that comes in the form of multiple arrays with a typical example being a color image that comes in the RGB format. Such images contain a matrix for the pixel intensities of each channel. Convolutional Neural Networks are especially suitable for categorization problems with a typical architecture depicted in Figure 11.

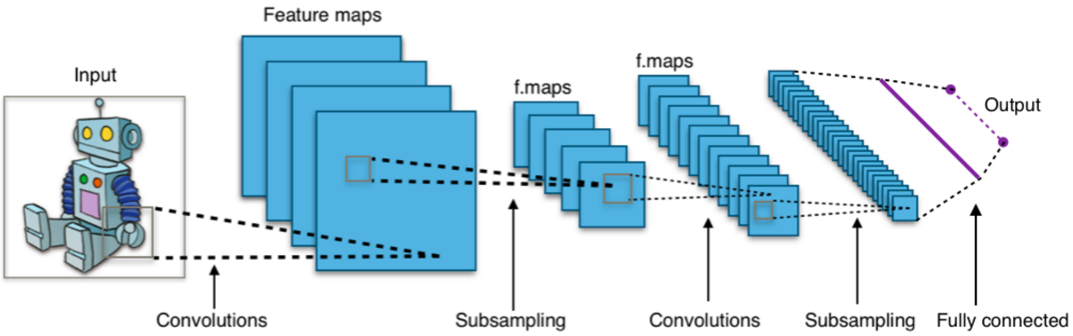


Figure 11 - convolutional neural network (wikipedia)

As depicted in Figure 11, a CNN is composed of many pairs of convolutional and subsampling (pooling) layers followed by a fully connected layer and a Softmax cost function layer at the output[5]. CNNs exploit four properties of natural signals: local connections; shared weights; pooling; and the use of many layers. Each convolutional layer accepts a set of feature maps and usually applies 3D convolution on these maps outputting another set of feature maps. A single neuron conducts each convolution by accepting local patches of the feature maps of the previous layer multiplied by the corresponding weights while a set of convolution kernels that renders the feature maps is called a filter bank. The result is then passed through a non-linear activation function, such as the ReLU. Such an architecture can easily detect local groups of values that are often highly correlated, forming patterns that can easily be distinguished. Furthermore, the local statistics of images and other signals are invariant to location. A pattern can be repeated many times across an image. Thus, a filter bank trained to detect a specific pattern in one location of an image can easily generalize to recognize a similar pattern in another part of the image.

The neural network depicted in Figure 11 is a typical implementation of a categorization convolution neural network. These networks are comprised of a succession of convolution and max pooling layers followed by several fully connected layers and a finally, a softmax layer that asserts one or more categorization neurons according to their probabilities scores[6]. The foremost application of Deep CNNs is image and video classification. Such a classifier has the task of taking an input image and categorizing it as belonging to one or more classes with their respective probabilities. This is a basic trait of human beings, which is learned even from an infant age and is constantly being perfected in our adult lives. We develop early on the ability to be able to discern different images in our environment and assign a category to each object in it. Prior knowledge plays a crucial role in this process and can easily be adapted to different environments.

Despite advances in machine learning, computers had a very hard time generalizing in image classifications tasks before the advent of effective deep convolutional networks. Hand engineered methods were successful in recognizing images of a specific domain only in an ideal or artificially generated context. For a computer, an image is simply of collection of pixel values in an array data structure. A color image of resolution 640x480 has three-color channels (RGB) and thus would have a representative array of size 640x480x3. The task of the computer given these numbers is to calculate the probabilities that the input belongs to a specific class and then return the class or classes with the highest probabilities.

An analysis of the basic layers of which a CNN is comprised in the context of image classification follows:

- Convolutional Layer: This is the main layer of a CNN. It aims to process a local patch around a pixel and to create translation invariant feature

[5] Not shown in the Figure.

[6] The softmax is an activation function that converts the inputs it receives into a probability vector whose values sum to 1. It was not presented in the activation functions section because it is primarily used in classifier CNNs, which is not the case for any of the developed CNNs for this thesis.

detectors, which can detect various patterns in an image. The main advantage against typical fully connected networks is firstly computational, since there exists a vastly reduced number of connections (weights) that need to be stored and trained and secondly, performance wise it can easily focus on the local context and fine tune to it (Le, 2015). The complexity of a network can be further reduced via weight sharing, in which certain connections that aim to discover the underlying pattern on a local context share the same weights.

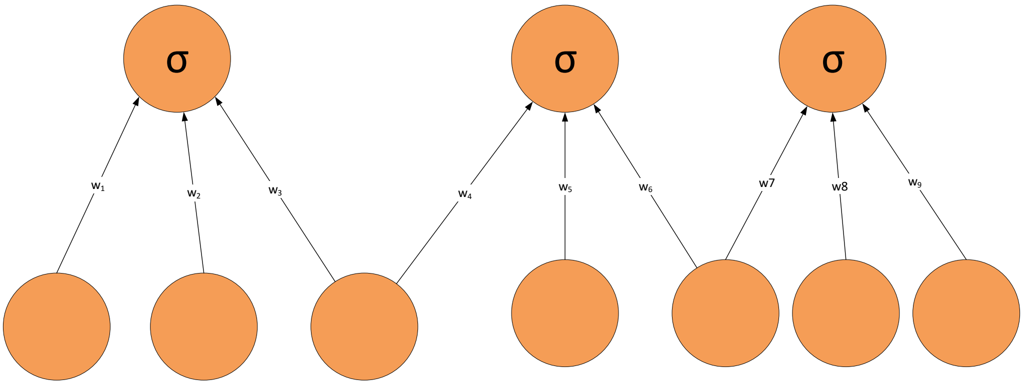


Figure 12- Weight sharing in a convolutional network

For instance in Figure 12, weights $w_1 = w_4 = w_7, w_2 = w_5 = w_8, w_3 = w_5 = w_9$. This way, the network can be compact in terms of storage requirements, as only three different weights need to be stored instead of nine. This can also be viewed as a necessity due to the nature of convolutional networks, which require the same filters to be applied to different parts of an image or video. An effect of the nature of the convolution operator is the reduction of the resolution of the feature maps of the previous layer. This is because the convolution kernels do not exactly fit at the corners of the image. A compensation measure, in case this resolution reduction has an adverse effect on an application, is to zero pad all dimensions of the feature maps so that the convolution kernel precisely fits the original data.

- Max Pool Layer: This layer takes the outputs of several neurons of a convolution layer and only let us through the one with the maximum value (Krizhevsky, Sutskever, & Hinton, 2012). It enables the network to better generalize since two input vectors x that enter this layer and have the same values but different permutation will output the same value. Take for example, the input vectors$v1 = \{0,1,0,0\}\ and\ v2 = \{1,0,0,0\}$, which will both return a value of one after the max pool layer. This layer can also be found in the literature as the sub-sampling layer, since it has the effect of reducing the resolution of the output. The immediate effect of this layer is the reduction of the resolution of the produced feature maps. The max pool layer can be seen in Figure 13.

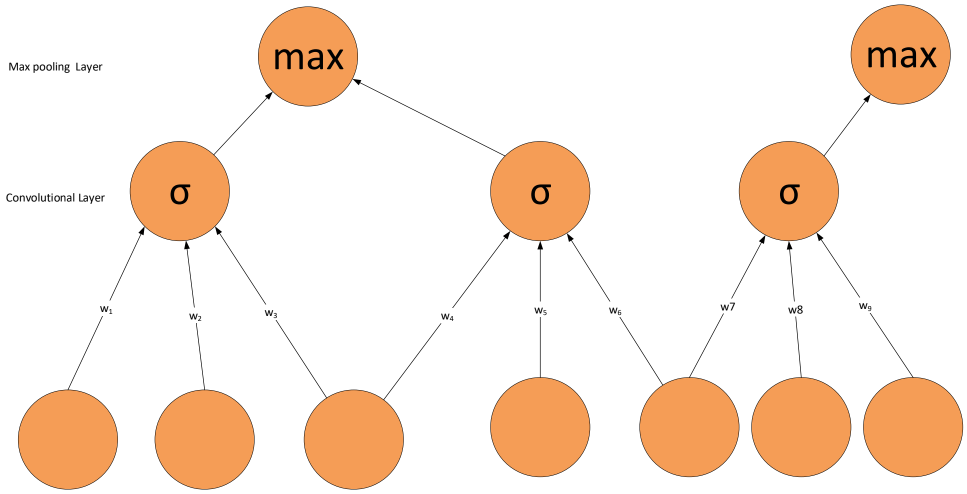


Figure 13- Max Pooling Layer

In a deep convolutional network, the previous two layers are repeated in succession according to the application and depth of a network. This has the effect of creating a hierarchy of feature maps that learn to recognize more intricate features of the input data as a function of the network depth. In Figure 14, the architecture of the deep convolutional network used by Hinton, Krizhevsky and Sutskever (Krizhevsky, Sutskever, & Hinton, 2012) in order to categorize 1.2 million images of the ImageNet database into 1200 categories is depicted.

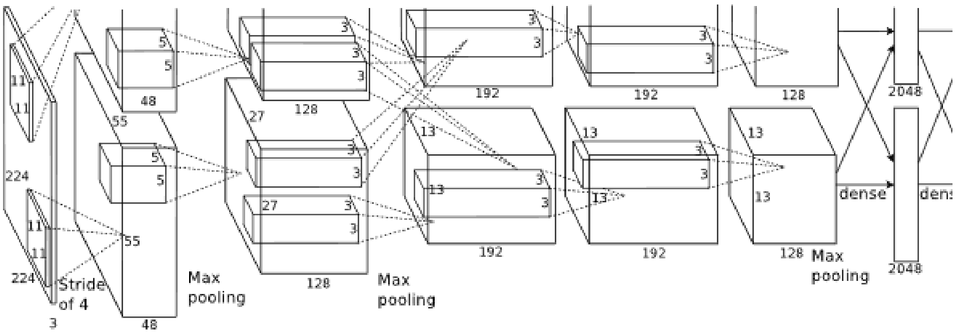


Figure 14 - Deep Neural Network used by Hinton, Kizhevsky and Sutskever to categorize ImageNet images into 1200 categories

- Fully Connected Layer: One or more fully connected layers typically follow the last Max-pooling layer. The preceding convolutional layers have created a series of features maps that describe the details of input data. It is up to the fully connected layers to take these feature maps and reach a classification decision. The first fully connected layer accepts a single vector that is a summary of the details in the feature map. During training, the weights of this fully connected layer are adjusted in order to learn how to classify each input vector.

The output of convolution/pooling is flattened into a single vector of values, each representing a probability that a certain feature belongs to a label. For example, if the image is of a cat, features representing things like whiskers or fur should have high probabilities for the label “cat”. The process is shown in Figure 15 (AI, Misssing Link, 2019).

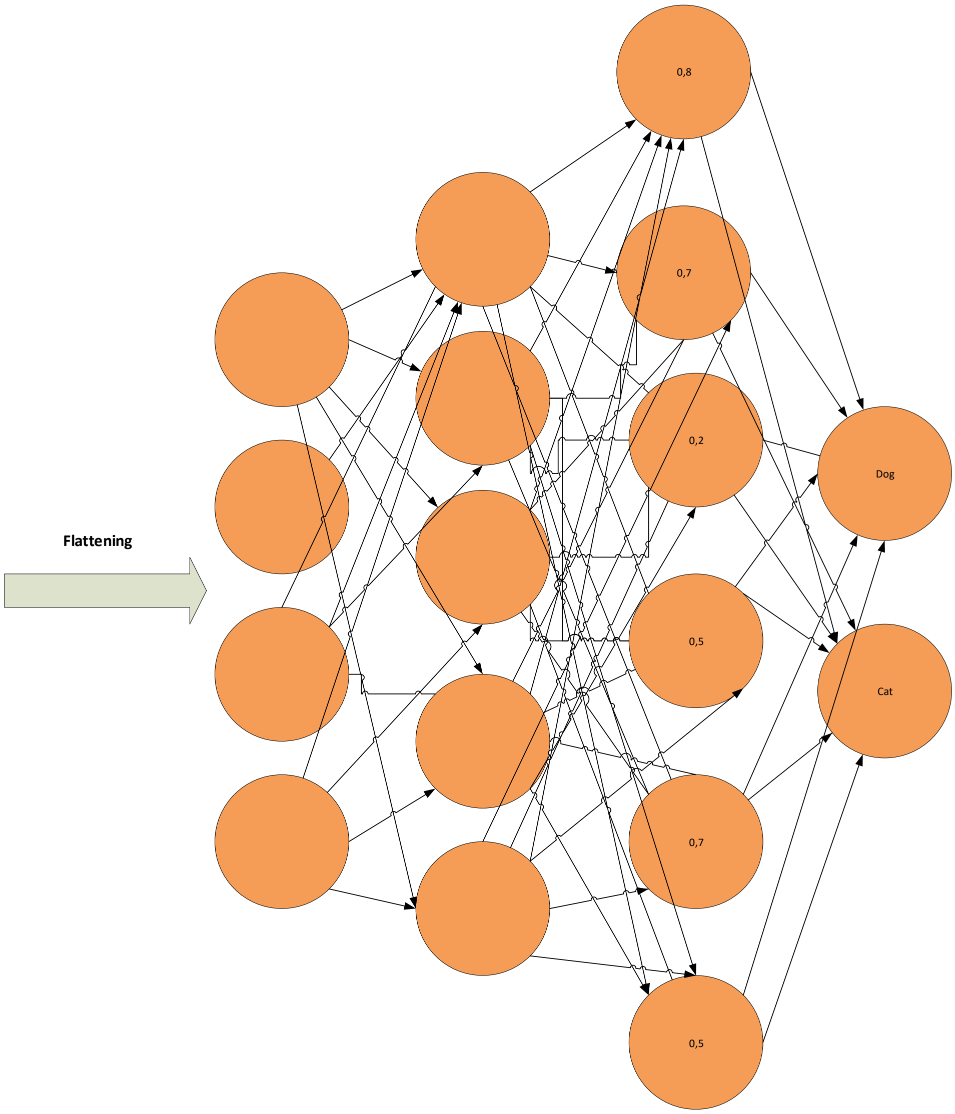


Figure 15 - Role of fully connected layer in a CNN

- Loss layer: The final layer of a deep neural network is comprised of loss function that is necessary so that the stochastic gradient descend commences back propagation of the error and perform the weight adjustment, in order to facilitate learning. The loss function[7] assess a candidate solution and produces an error that denotes how far away the neural network is from the optical solution[8]. Its function is extremely important, as the single value produced by it will be used to assess how well the complete neural network is performing and will guide the backpropagation of errors (Brownlee, 2019).

The category of loss functions characterized as maximum-likelihood estimators (MLE) are used for neural networks. MLE is a framework for inference for finding the best statistical estimates of parameters from historical training data. Under maximum likelihood, a loss function estimates how closely the distribution of predictions made by a model matches the distribution of target variables in the training data. When modeling a classification problem where we are interested in mapping input variables to a class label, we can model the problem as predicting the probability of an example belonging to each class. In a binary classification problem, there would be two classes, so we may predict the probability of the example belonging to the first class. In the case of multiple-class classification, we can predict a probability for the example belonging to each of the classes. (Brownlee, 2019).

3.3 Learning procedure

Learning in neural networks can be either supervised or unsupervised. Supervised learning requires labeled training exemplars and it is mostly used in the context of neural networks for classification purposes. On the contrary, unsupervised learning only requires large amounts of training data of which it tries to determine similarities. Since this thesis deals with convolutional neural networks that use labeled data, the focus is on supervised learning.

3.3.1 Supervised learning and the back propagation algorithm

The modern back-propagation algorithm and its variations has been evolving since the early 1960s but were only successfully applied to the training of neural networks in the early 1980’s. For instance, Kelley et al. (Kelley, 1960) derived analytic formulas for flight performance optimization according to the method of steepest descent (gradient). A year later Bryson (Bryson A. E., 1961) presented an early version of the back-propagation algorithm, which he developed further in (Bryson & Denham, 1962)

[7] Also called an objective function.

[8] This is typical for a neural network but in other cases, we may seek to maximize the objective function.

in the context of developing optimum programs for nonlinear systems with terminal constraints. A common denominator of the many variations of the back-propagation algorithm that have been presented since then, is the requirement of finding the steepest descent in the weight space by iterating the chain rule.

Early implementations were inefficient because they relied on back propagating Jacobian information through the standard Jacobian matrix calculations from one layer to the previous, something, that requires a lot of memory. More efficient error back-propagation was first introduced in the master‘s thesis of Linnainmaa (Linnainmaa, 1970) albeit with no reference to neural networks (NN). Speelpenning, also wrote a program for automatically deriving and implementing BP for given differentiable systems (Speelpenning, 1980). These initial implementations did not pertain to neural networks but they paved the way for variations of the BP, which could modify the control parameters (weights) of a NN driven by a cost function. According to (Schmidhuber, Deep learning in neural networks. An overview, 2015) the first application of the BP algorithm to NN appeared in 1981 by Werbos (Werbos, 1982). Much important work was published in the early 1980s. For example, Lecun (Lecun, Une procedure d'apprentissage ponr reseau a seuil asymetrique, 1985) analyzed neural networks with low Kolmogorov complexity and high generalization capability. soon thereafter, while Rumelhart et al. (Rumelhart, Hinton, & Williams, 1985) made an impact through a significant contribution, whose major theoretical & practical contribution was the procedure now known as error propagation, whereby the gradient can be determined by individual units of the network based only on locally available information.

According to the back-propagation algorithm and after an episode of activation spreading through the differentiable activation functions of individual neurons culminating with the calculation of the total error cost E, a single iteration of gradient descend computes changes of all w_i . Weights are usually initialized to zero or according to a Gaussian distribution function. The algorithm is described in Table 11 (Nielsen, 2019).

Table 11 - Back propagation algorithm

1. Input ***x*****:** Set the corresponding activation a^1 for the input layer.

2. Feedforward: For each l=2,3,…,L compute $z_l = w^l\, a^{l-1} + b^l 1$ and $a^l = \sigma(\mathrm{z}^{\mathrm{l}})$, where w$_l$: the weights of layer l, a$_{l-1}$: the input to layer l, b$_l$: the bias of layer l and σ: the chosen activation function.

3. Compute the vector$\delta^L = \nabla_a C \odot \sigma^l(z^L)$, where C: the training example and the symbol ⨀ is the Hadamard product (Nielsen, 2019).

4. Back propagate the error: For each l=L−1,L−2,…,2 compute $\delta^{l} = ((w^{l+1})^{T} d^{l+1}) \odot \sigma'(z^{l})$

5. The gradient of the cost function is given by $\frac{\partial C}{\partial w_{jk}^{l}} = a_{k}^{l-1} and \frac{\partial C}{\partial b_{j}^{l}} = \delta_{j}^{l}$.

It is called back propagation precisely because of the way it operates. Execution commences only after the total error has been calculated, computing the backward errors δ^{λ} starting from the final layer.

No matter how efficient it proved when first developed, various shortcomings of back propagation were quickly noticed. By the late 1980's most neural network designers considered the basic BP algorithm applicable only to shallow networks. This was nonetheless not considered to be much of a problem since the Kolomogorov theorem (Kůrková, 1992) stated that a single layer network with enough hidden units can approximate any multivariate continuous function with arbitrary accuracy. Thus, the basic BP algorithm notwithstanding its drawbacks[9] remained the most popular neural network-training algorithm for shallow FNN.

BP is essentially a linear least-squares problem where second order gradient information is passed up to preceding layers as noted in Biegler et al. (Biegler-König & Bärmann, 1993) and since the inception of the basic BP algorithm in the 1960's various other improvements have been proposed. Least-square methods like those proposed by Stoer (Stoer, Bauer, & Bulirsch, 1989) or quasi Newton methods (Shanno, 1970) were proven computationally expensive. Momentum was introduced by Rumelhart et al., (Rumelhart, Hinton, & Williams, 1985), in order to speed up calculation and avoid small pitfalls in the solution space. It quickly became mainstream, proving more efficient than other contemporary and future proposals like ad-hoc constants added to the slope of linearized activation functions (Fahlman, 1991) or exaggerating the non-linearity of the slope of activation functions (West & Saad, 1996). Another popular variation is the R-Prop, which only takes into account the sign of the error derivatives as well as it's more (Riedmiller & Rprop, 1994) robust variant named irProp+.

A variant of the classic BP algorithm widely employed in modern neural networks is stochastic gradient descent (SGD), in which a batch of inputs is presented to the network, which then computes the a) outputs; b) the errors; c) the average gradient for those examples and only then proceeds to adjust the weights accordingly. This process continues for many batches until the average of the objective or cost function stops decreasing (Lecun, Bengio, & Hinton, Deep learning., 2015). The stochastic adjective is used because each batch gives a noisy estimate of the average over the

[9] For instance, the rather slow convergence to an acceptable solution in comparison with newer adaptations considered later in this chapter.

total gradient for all examples. Although simple, it usually offers a quick convergence to weight values that produce a good result. A test set is then usually applied in order to assess the performance of the trained network to unseen inputs, which is used as a metric of the generalization ability of the network.

3.3.2 Learning in a feed forward deep network

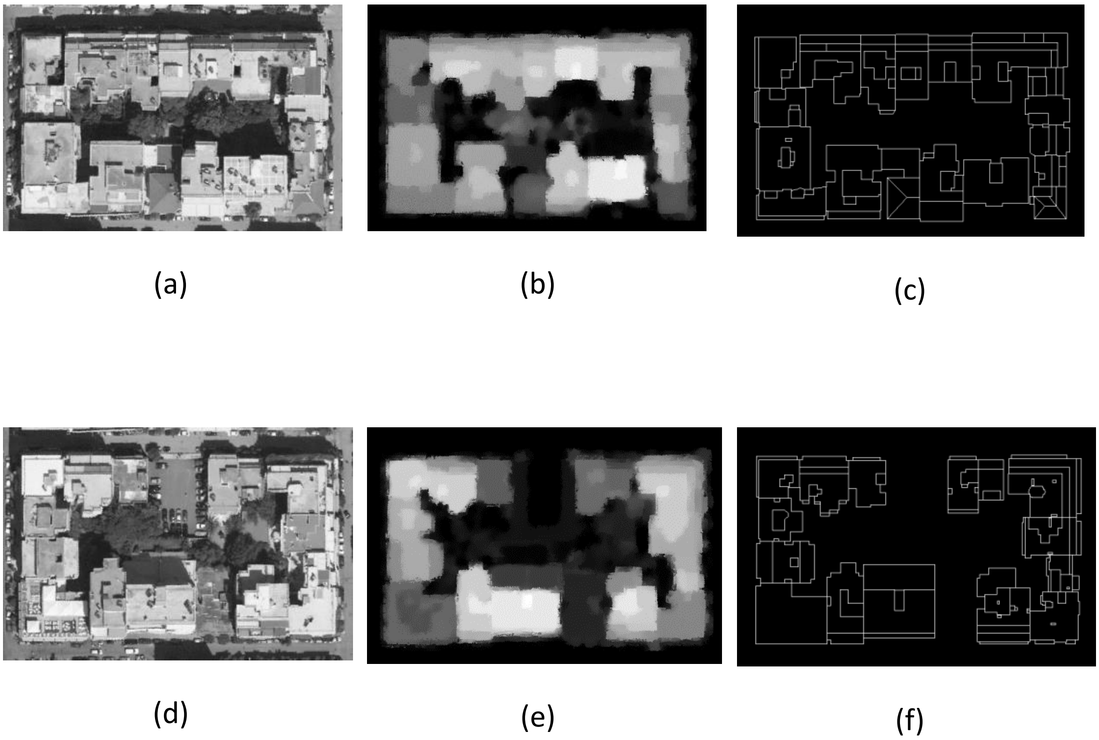
A deep leaning network learns to map a fixed sized input to a fixed sized output. The weighted sum of the inputs to a layer are computed and then passed on to the next layer after being moderated by an activation function. As stated above the most widely used activation function is the ReLU, which offers a straightforward and fast implementation while also having non-linear features. It can be seen as the half-wave rectifier. The ReLU typically has a faster learning rate than other activation functions like the tanh or sigmoid (Lecun, Bengio, & Hinton, Deep learning., 2015). A typical deep neural network has many hidden layers whose function is to distort the input in a non-linear way so that the output layer can linearly separate the input into categories.

Neural Networks in general were mostly forsaken until the late 1990s because it was believed that simple gradient descent or stochastic gradient descent would be trapped in local minimum, which would not lead to optimal solutions. Since the early 2000's evidence has slowly accrued which points to the other direction. Poor local minima are rarely a problem with deep networks (Lecun, Bengio, & Hinton, Deep learning., 2015). It has been shown theoretically and empirically that the solution space is full of saddle points where the gradient is zero. Almost all of these saddle points have a few downward curving directions, which have similar values for the objective function. This means that the network will find a more or less similar solution irrespective of the saddle point it could temporarily stuck at. Neural Networks in their deep variation were reintroduced in the scientific literature owing to Hinton et al. working under the auspices of the Canadian Institute for Advanced Research (CIFAR), who introduced an unsupervised learning procedure that could create layers of feature detectors without requiring labeled data (Hinton , Osindero, & Teh, 2006). The objective of the network was to model the activities of features detectors in previous layers, which was accomplished via the pre-training of several layers of progressively more complex feature detectors using this reconstruction objective. The whole network could then be trained after adding an output layer using standard backpropagation techniques. Such networks were applied for recognizing hand-written digits and worked very well.

The advent of easily programmable powerful GPUs allowed the training of gradually more complex networks much faster than could be done in the past. Speech recognition was one of the first signal analysis domains to avail of deep neural networks. In 2009, the previous described technique of Hinton was employed to map short temporal windows of coefficients extracted from a sound wave to a set of probabilities for the various fragments of speech that might be represented by the frame in the center of window. Variations of this technique were being used in Android phones by 2012 with remarkable success.

4. DEEP CONVOLUTIONAL NEURAL NETWORKS FOR BUILDING CONTOUR DETECTION

For this thesis, research was conducted on using convolutional neural networks for building contour detection. The result of this research is a deep convolutional neural network that can directly detect building contours. Due to the nature of the work in Dong et al. (Dong, Loy, He, & Tang, 2015) which exhibits several features that were considered akin to this application, a modified version of this network was applied as the basis of the solution. In this case the modified SRCNN, which was named BCDCNN (Building Contour Detector Convolutional Neural Network) accepts a tuple of available data in the form <[optical, DEM], GT> which is comprised of an optical and a DEM input pair along with the corresponding ground truth output. The network is expected to approximate the GT data given the [optical, DEM] pair. The data used for this research originated from a densely populated area of Kallithea – Attica –Greece. They comprise of sets of optical – digital elevation maps and handcrafted ground truth images. Of the whole dataset two blocks were selected as shown in Figures 16(a-f).



(a) (b) (c)

(d) (e) (f)

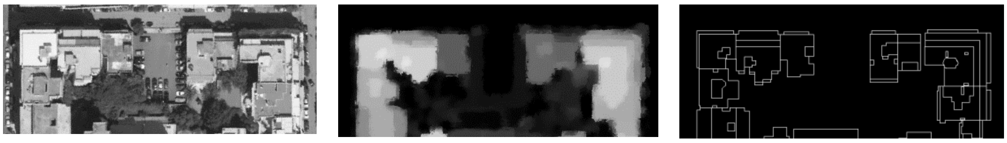
Figure 16 – Original data used for the building contour detector. (a) Optical image of block 1. (b) Elevation data of block 1. (c) Ground truth data of block 1. (d) Optical image of block 2. (e) Elevation data of block 2. (f) Ground truth data of block 2.

The data of Figure 16 were segmented into three categories:

- Train data: Used for training the neural network model. The whole of block 1 was used (Figures 16(a-c)) and the top 2/3 portion of Figures 16(d-f). The top 2/3 of Figures 16(d-f) used for training are shown in Figures 17(a-c).

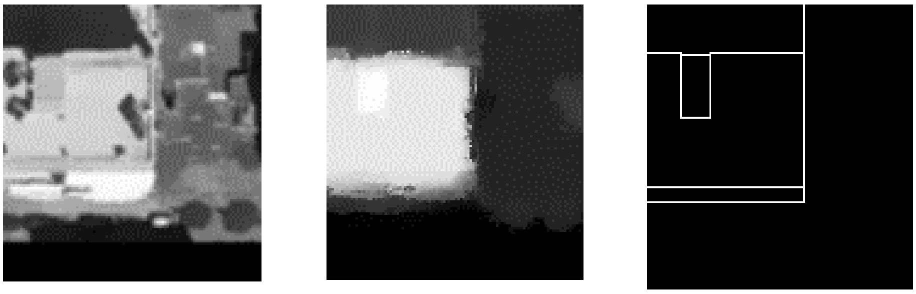
- Validation data: This set of data was used for validation purposes during training. No weight adjustment was performed with this data set. The data are displayed in Figures 18(a-c).

- Test data: This is the part of block 2 that was used to assess the performance of the proposed model. It measures the generalization capability of the model and can be seen in Figures 19(a-c).



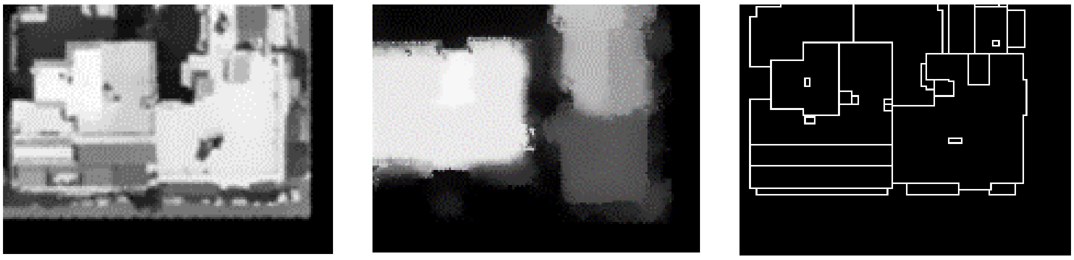
(a) (b) (c)

Figure 17 – Remaining training data (top 2/3 of block 2)



(a) (b) (c)

Figure 18- Original validation data



(a) (b) (c)

Figure 19 - Original test data

4.1 Methodology

The proposed BCDCNN model is based on the Super-Resolution Convolutional Neural Network (SRCNN) presented by Dong et al. (Dong, Loy, He, & Tang, 2015). However, if one can say that SRCNN implements a super-resolved auto associative mapping in the sense that a low resolution image is mapped onto the high resolution version of itself, BCDCNN implements a super-resolved heteroassociative mapping since low resolution elevation data are mapped onto their associated high resolution building contours available during training from the ground truth data. In particular, similar to the first convolutional layer of SRCNN, BCDCNN accepts a low-resolution elevation map at the input which is up sampled to the desired higher resolution (the up sampling scale is determined by the corresponding high resolution optical image) using the joint (optical + DEM) mean-shift based up sampling algorithm described in (Vassilas, Tsenoglou, & Ghazanfarpour, 2015). Following Dong et al. (Dong, Loy, He, & Tang, 2015), it is assumed that the high-resolution optical image combined with the preprocessed low-resolution elevation map constitute the mixed resolution input X to the network. The goal of the convolutional network is then to reconstruct an image F(X) that is similar to the corresponding ground truth high-resolution building contours image Y. In order to accomplish this, BCDCNN uses a mapping F from input to reconstruction (output) which consists of the following three operations:

- Patch extraction and representation. Patches from the mixed resolution image X are extracted, then processed by the filter bank of the first convolutional layer and, finally, represented as a set of feature maps. This can be mathematically expressed as the operation:

$$\boldsymbol{F_1(X) = max(0, W_1 * X + B_1)} \qquad (4)$$

where $\boldsymbol{W_1} = \{\, \boldsymbol{W_1 k} \mid 1 \leq k \leq \boldsymbol{N_1} \,\}$ and $\boldsymbol{B_1} = \{\, \boldsymbol{B_1 k} \mid 1 \leq k \leq \boldsymbol{N_1} \,\}$ with $\boldsymbol{w_1^k}$ being the k-th 3-D filter of the first layer's filter bank $\boldsymbol{w_1}$, $\boldsymbol{B_1^k}$ the corresponding bias term and $\boldsymbol{F_1(X)}$ the set of $\boldsymbol{N_1}$ feature maps. As induced by eq. (1), this layer includes a ReLU non-linearity. Each of the $\boldsymbol{N_1}$ filters is of size $\boldsymbol{s_1 x s_1 x\ N_0}$, with $\boldsymbol{N_0}$ denoting the number of channels in the input image ($\boldsymbol{N_0}$=2 for the first layer). Finally, operator '*' signifies convolution.

- Non-linear feature map transformation. In the second convolutional layer, the$\boldsymbol{N_1}$ feature maps generated by the previous operation are non-linearly transformed into another set of $\boldsymbol{N_2}$ feature maps by applying $\boldsymbol{N_2}$ filters of size $\boldsymbol{s_2 x s_2 x N_1}$ 1 and then, as before, passing the results from a ReLU. This operation can be described mathematically as

$$\boldsymbol{F_2(X) = \max(0, W_2 * F_1(X) + B_2} \tag{5}$$

where $\boldsymbol{W_2}$ contains $\boldsymbol{N_2}$filters of size $\boldsymbol{s_2 x s_2 x N_1}$ and $\boldsymbol{B_2}$is $\boldsymbol{N_2}$ dimensional.

- Building contour reconstruction: Finally, the feature maps of the previous stage are aggregated to generate the high-resolution building contour image. The reconstruction operation is implemented as a linear convolution layer,

$$\boldsymbol{F_3(X) = W_3 * F_2(X) + B_3} \tag{6}$$

where $\boldsymbol{w_3}$ corresponds to a single filter of size $\boldsymbol{s_3 x s_3 x N_2}$_2 and $\boldsymbol{B_3}$ is the final layer's bias term.

The network architecture is illustrated in Figure 20 in which the input to the network, the optimal output and the size of the convolution kernels applied at each layer are shown.

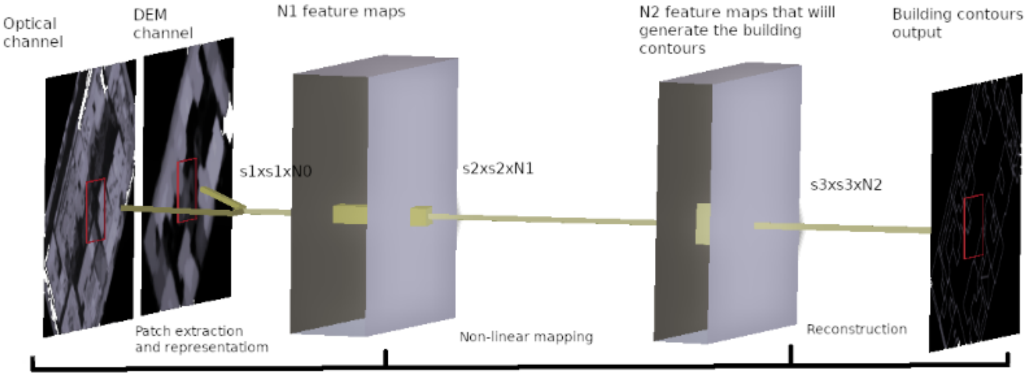


Figure 20 - proposed 3-layer convolutional system architecture

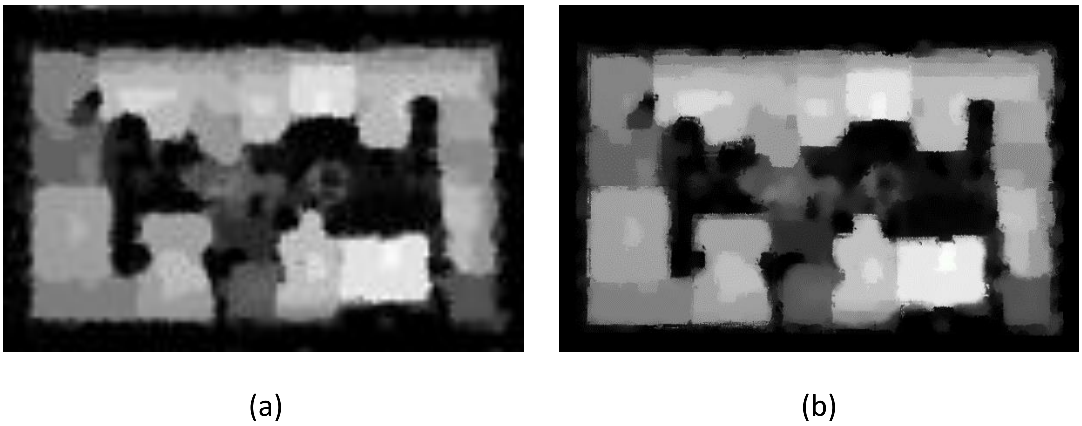
The goal is to get a building contour map F(Y) that is as close as possible to the ground truth. However, unlike classification type of applications in which the training procedure associates input images to, usually, a few class labels, the proposed system is presented with a far more difficult and challenging problem. That is because it is learning a heteroassociative mapping from a quite limited training set of <input, output> pairs and then expecting to generalize on new pairs of building top-view images. Further adding to the complexity of our data sources there are four different types of edges that the network must learn to differentiate.

- Elevation edges that are simultaneously optical edges, which is mostly the case.
- Optical edges that are not elevation edges: For instance, rooftops of neighboring buildings of different colors but same heights.
- Elevation edges that are not optical: For example, a rooftop of the same color as an adjacent street and at different heights.
- Implied edges: For instance, rooftops with the same color and same height. This is the most difficult case.

Just to make the problem even more difficult, the available elevation data – carrying most of the building contours information are at a five times lower spatial resolution than the optical images and the associated building contours. Hence, the proposed CNN architecture is actually performing a combination of elevation data super-resolution assisted by available high-resolution optical images and a heteroassociative mapping to building contours.

4.2 Data pre-processing

The proposed network is trained using high-resolution aerial orthophotographs of Kallithea, a densely populated area in Attica, Greece, as well as the corresponding low-resolution digital elevation model and the high-resolution ground truth building contours. Figures 16(a) and 16(b) depict the optical and elevation data of a building block (named BLOCK1), respectively. In particular, to arrange the two data sources as two channels of a multimodal image, the depicted DEM has been up sampled with a scale of 5 using the joint mean shift algorithm (Vassilas, Tsenoglou, & Ghazanfarpour, 2015). A second block of buildings from the same area has also being selected and sliced to produce a complementary dataset for training (Figure 17) and testing (Figure 19). As before, the corresponding DEM channels have been 5x up-sampled using joint mean shift. The original and mean-shift processed elevation data for block 1 and 2 are shown in Figures 21-22(a-b), respectively.



(a)

(b)

Figure 21 - Original and MS processed elevation data (block 1)

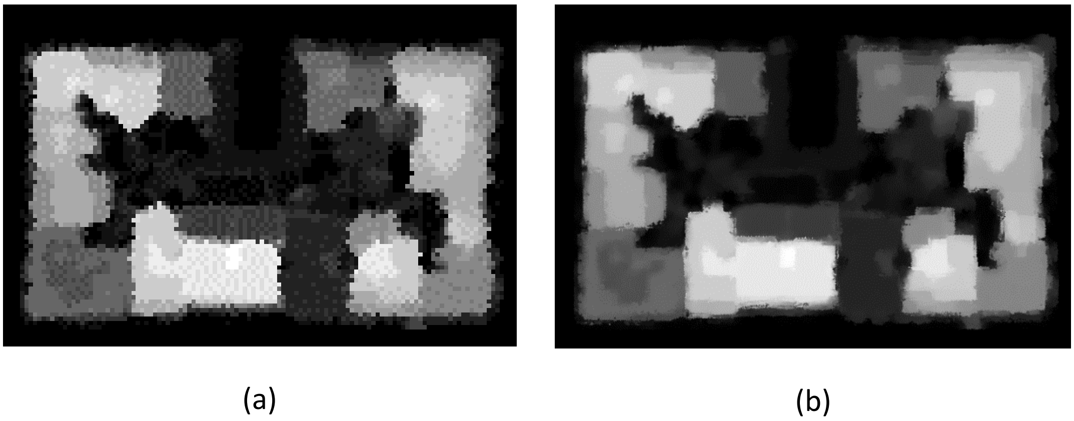


Figure 22 - Original and MS processed elevation data (block 2)

Furthermore, three variations of the training data were used. More specifically, the first variation is comprised of the original optical data and the mean shift up sampled DEM data [Figures 23-24(a-c) – Train, Figures 25(a-c) – Validation, Figures 26(a-c) - Test]. Moving on to the second variation, the optical channel has also been filtered with the mean shift edge preserving smoothing algorithm [Comanciu Meer, 2002]. The full data set can be seen in [Figures 27-28(a-c) – Train, Figures 29(a-c) – Validation, Figures 30(a-c) – Test]. Finally, in the third variation the mean shift optical & DEM data have been filtered by a Laplacian of Gaussian (LoG) operator [see Figures 30-31(a-c) - Train, Figures 32(a-c) - Validation, Figures 33(a-c) - Test]. The last variations has been considered as an attempt to reduce the effective dimensionality of the input data and improve the generalization ability of the proposed system.

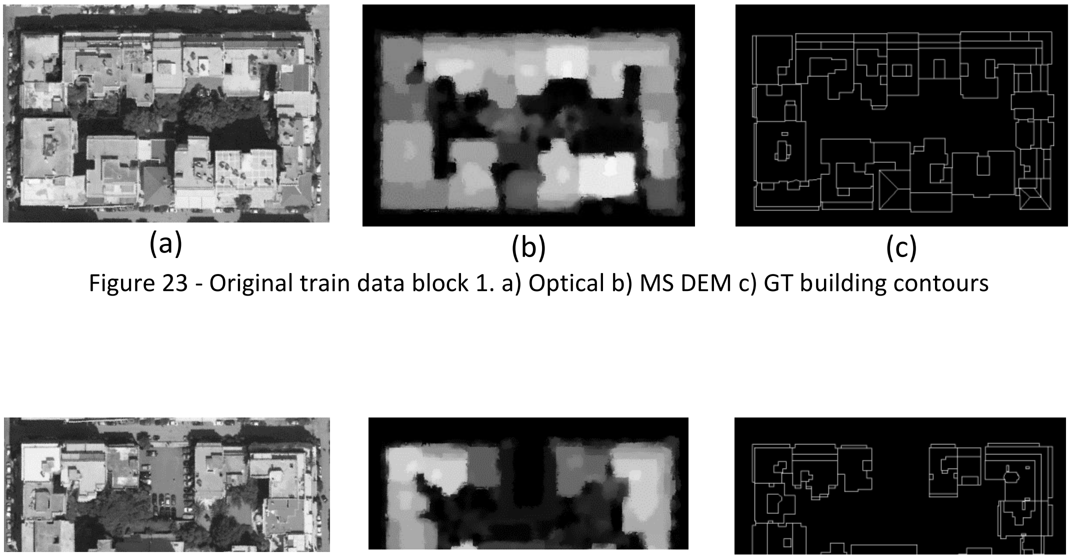


Figure 23 - Original train data block 1. a) Optical b) MS DEM c) GT building contours

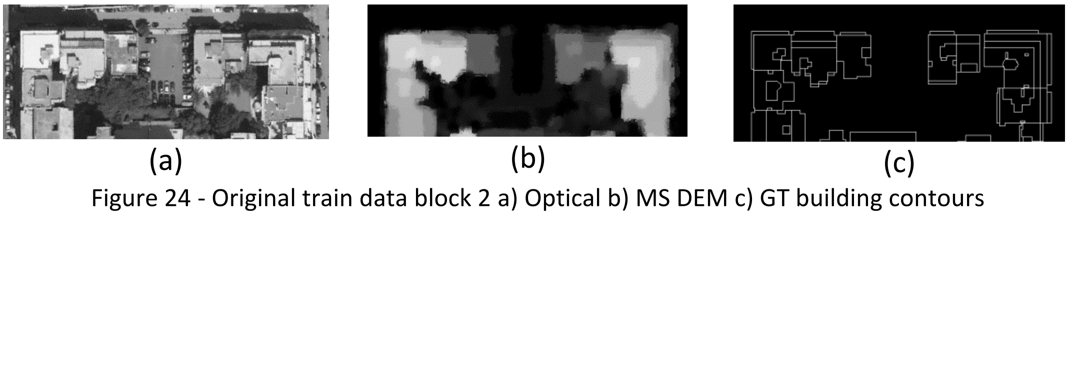


Figure 24 - Original train data block 2 a) Optical b) MS DEM c) GT building contours

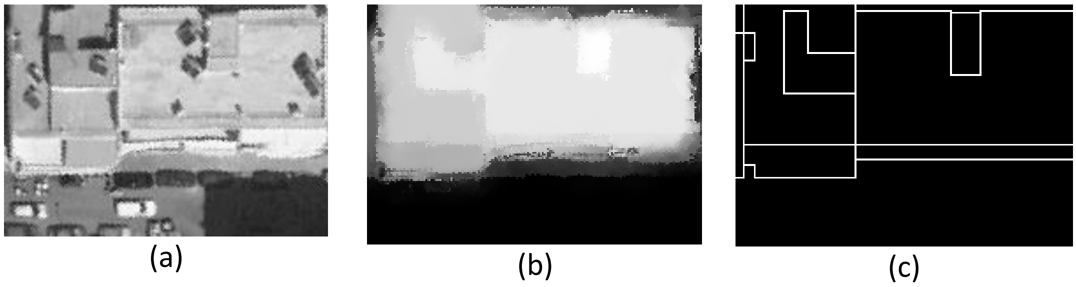


Figure 25 - Original validation data a) Optical b) MS DEM c) GT building contours

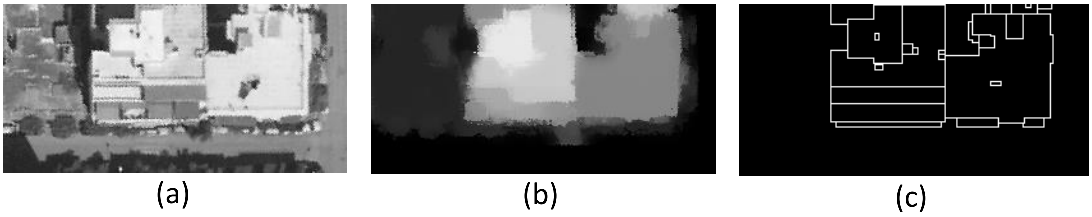


Figure 26 - Original test data a) Optical b) MS DEM c) GT building contours

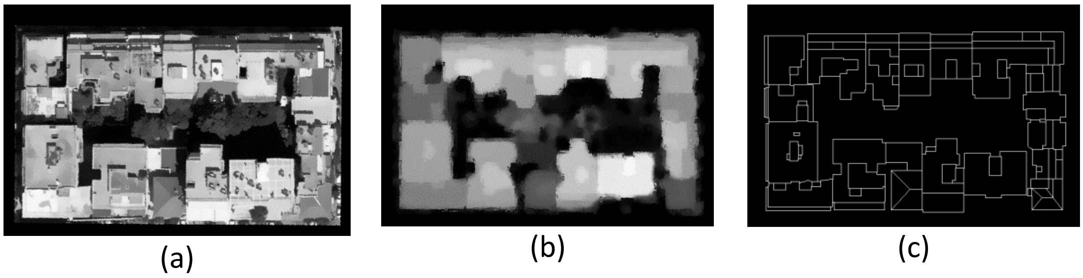


Figure 27 - MS train data block 1 a) MS Optical b) MS DEM c) GT building contours

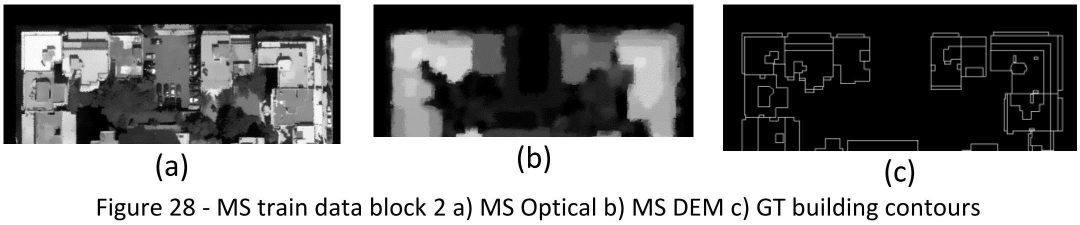


Figure 28 - MS train data block 2 a) MS Optical b) MS DEM c) GT building contours

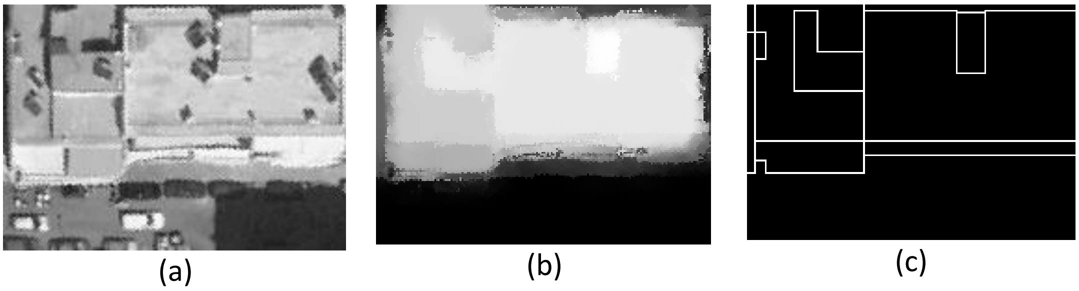


Figure 29 - MS validation data block a) MS Optical b) MS DEM c) GT building contours

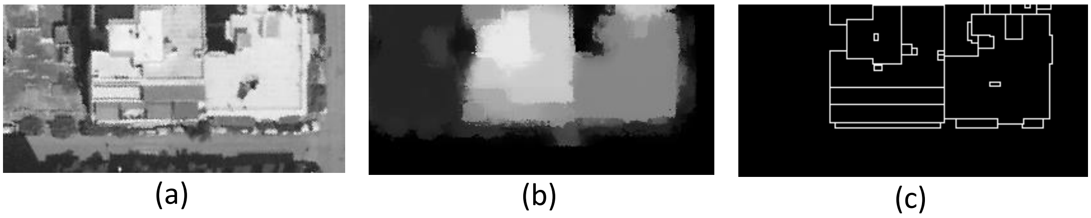


Figure 30 - MS test data a) MS Optical b) MS DEM c) GT building contours

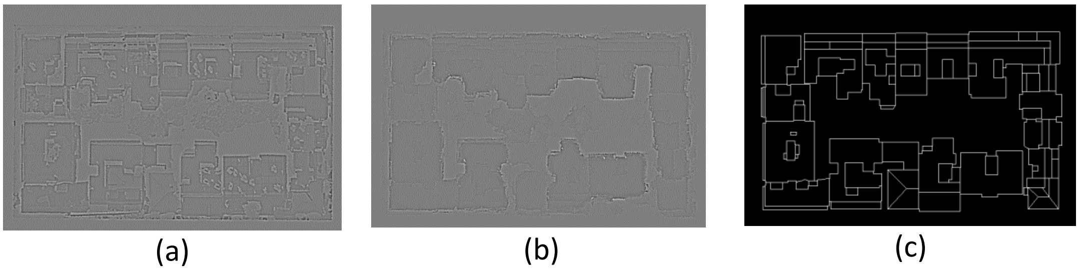


Figure 31 - Log train data block 1 a) LoG Optical b) Log DEM c) GT building contours

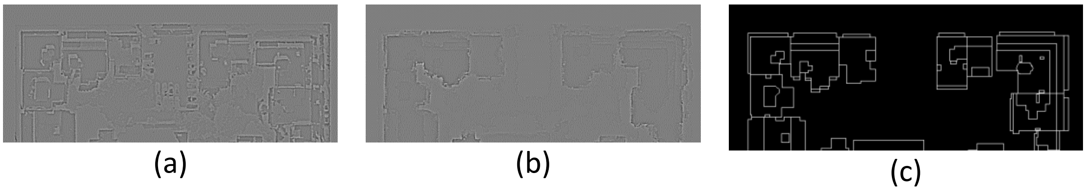


Figure 32- Log train data block 2 a) LoG Optical b) Log DEM c) GT building contours

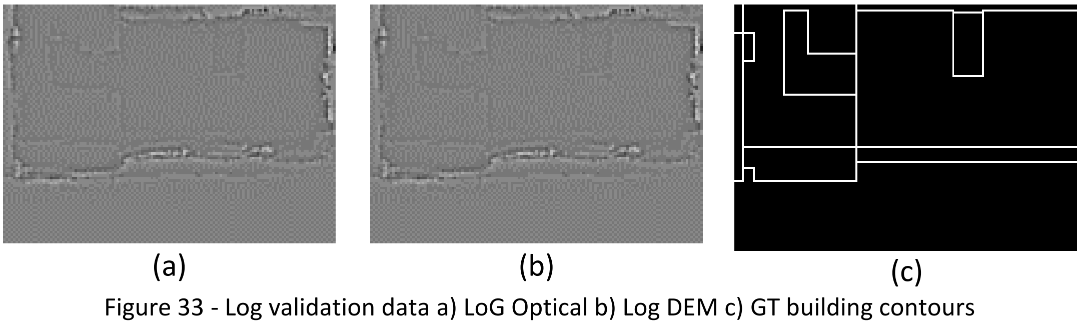
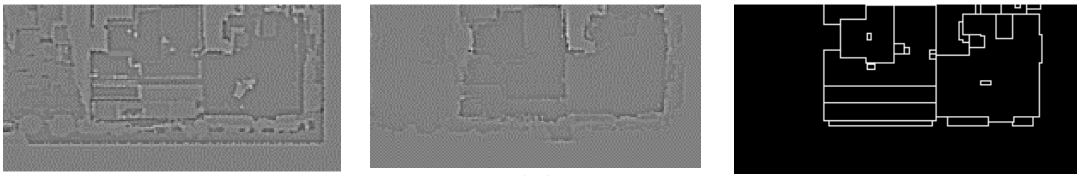


Figure 33 - Log validation data a) LoG Optical b) Log DEM c) GT building contours



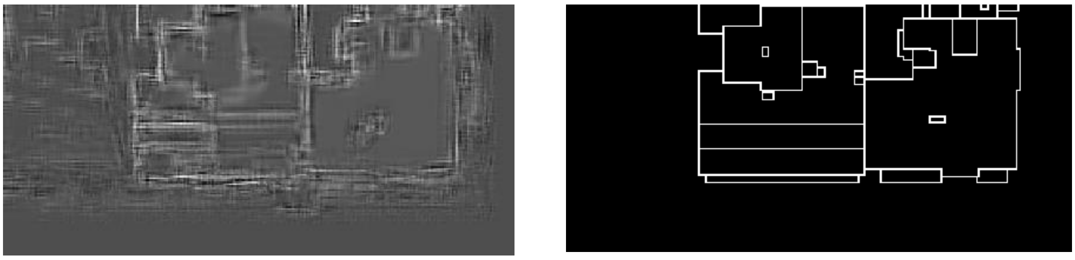
(a) (b) (c)

Figure 34 - Log test data a) LoG Optical b) Log DEM c) GT building contours

4.3 Cost function modification

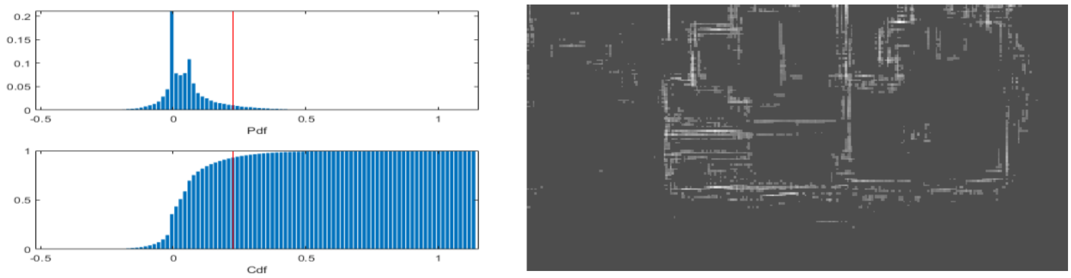
A typical cost function to be minimized during network training is the root mean square error (RMSE) between the actual reconstruction and the ground truth, which in this particular application is a binary image with ones for pixels belonging to the building contour and zeroes for all other pixels. However, only a very small proportion of pixels in the GT image (as well as its sub images and patches thereafter) will have a value of one and will pull the corresponding neuron outputs of the reconstruction layer. All the remaining neuron outputs (or pixel values), no matter how close to zero they are will be pushed towards zero. Although no post processing stage has been included to clean up the reconstructed binary contours, it is intuitively evident that the neuron outputs of the reconstructed image that correspond to background and have close to zero or negative values, could be set aside from the derivative computations of the back-propagation phase, e.g. by setting them to zero. On the other hand, neuron outputs wrongly close to 1 should play a role in the back-propagation phase in order to be pushed down to lower values. A second point we can make regarding weight adaptation in this application is that all output neurons share the same weights and that these weights should be given a chance to adapt in such a way as to satisfy confronting demands: to push some output neurons to one and other neurons to zero. Since the proportion of 1-pixels is much smaller than that of 0-pixels, it is expected that the shared weights will prioritize minimizing the error of the “many” background pixels instead of the “few” contour pixels. This comment highlights network-training difficulties in heteroassociative mappings that arise due to unequal pixel-class probabilities and resembles the necessity for class-balanced datasets in classification problems. In order to balance the weight adaptation process to serve equally well the contour and non-contour pixels, it is proposed to substitute the typical RMSE cost criterion that involves all neuron outputs of the reconstruction layer by a novel custom cost layer, which was named Top-N. Under this scheme, the RMSE between the reconstructed image and the corresponding GT is calculated only for those pixels that belong to the 2N pixels with highest values[10]. Assuming that most of the N contour pixels of the ground truth image are also in the top 2N pixels of the reconstruction, this scheme satisfies the imposed balancing criterion.

[10] N is the number of contour pixels in GT



(a) (b)

Figure 35 - Proposed Top-N custom cost layer. (a) low-quality reconstruction. (b) Corresponding ground truth data



(a) (b)

Figure 36 - Proposed Top-N custom cost layer. c) pdf and cdf of intensity levels and Top-N threshold, and d) Top-N version of the reconstruction.

In practical terms, the threshold used to specify the top 2N pixel values is calculated as follows: The probability distribution function and cumulative distribution function of the intensity levels for each image used during training are calculated and only the pixels that have an intensity above the 2N threshold[11] are retained. This is depicted in Figures 35(a-b) and 36(a-b), which show a low quality reconstruction of the test data, the corresponding ground truth, the Top-N threshold calculated as the percentage of pixels above the Top-N intensity and the Top-N version of the reconstruction, respectively.

4.4 Experiments

4.4.1 Training Set Preparation

To satisfy the requirement of large numbers of training data to properly train deep neural networks I performed data augmentation (Simonyan & Zisserman, 2014). Firstly, 33x33 data patches were extracted from the input data (optical + DEM) along with the corresponding 21x21 patches of the GT data (GT patches are smaller due to "valid" convolutions with 9x9, 1x1 and 5x5 kernels). The data were then augmented with rotations at multiples of 90° and with their vertical flips. In this manner tuples of

[11] Actually, the average value of the intensity level for a whole batch is used in order to accelerate the computation procedure.

input data and GT were constructed in the form <[optical_section, DEM_section], GT_section>. The procedure described in the following sections was followed for each of the three variations of the training data set (original, Mean-Shift processed, LoG processed).

4.5 Experimental results

All presented results pertain to the 9-1-5 or 9-3-5 convolution kernel choices and to the 64-32-1 feature maps configuration, i.e. number of feature maps at the output of each convolutional layer. Actually, several tests were conducted to assess how the number of feature maps affect the performance of the network. Specifically, networks of with 128-64-1 and 256-128-1 feature map configurations were tested. However, even though performance is increased (the RMSE for the Original data sets at epoch 60 decreases from 3,4048 to 3,3384 and then to 3,2077 for the larger configurations), the heavy computational costs prohibited their further use).

Three deep learning framework were used before making a final decision on which to use. These were:

- Tensorflow: Tensorflow is the brainchild of Google brain. It supports general machine learning and deep learning. It has grown considerably since its inception in 2015 because of its capabilities and the reputation of Google. Although initial experiments were done under this framework, it was not selected because it could not integrate directly with other code written in Matlab.

- Caffe: This framework was created by Yangqing Jia during his PhD at Berkley. It has impressive expressive capability and it is very fast. Regarding the programming interface, it can be programmed using Python and Matlab. The support for Matlab is crucial for researchers using Matlab and this is why it was used. Unfortunately, it can only efficiently be extended using c++ and CUDA programming which requires a recompilation of the framework.

- MatConvNet: This a specialized framework for Matlab. It is suitable for applications using convolutional neural networks only. Since it is written in Matlab it is can seamlessly change between CPU and GPU processing according to whether it is given GPU arrays or standard arrays. It is also easily extensible via standard Matlab code since the source code is readily available. Finally, it is almost as fast as Caffe and Tensorflow.

4.5.1 Comparison between MSE and Top-N Custom Loss Layers

The proposed Top-N custom cost layer leads to lower RMSE and higher PSNR values as shown in Tables 12-14. The tables depict the RMSE and PSNR of the test data for the case of training on the Original, the Mean Shift and LoG data sets, respectively. In all cases, the custom Top-N layer exhibits lower RMSE and higher PSNR values than the typical MSE cost layer. For instance, in Table 12 regarding the Dropout 50% 9-1-5 case the proposed Top-N cost layer produced an RMSE 3.37% lower than the corresponding

MSE cost layer. Comparing corresponding entries for the PSNR for the Original data (Table 12), 5 out of 6 entries have a higher value for the 9-3-5 network. Likewise, for the Mean Shift and LoG processed data (Tables 13 and 14) most PSNR entries are higher for the 9-3-5 network. Nonetheless, since the 9-3-5 configuration was by 16.5% slower during training than the 9-1-5 configuration and since as shown in Tables 12 through 14, there was only a slight improvement either in RMSE or in PSNR compared to 9-1-5, we decided to consider the more practical 9-1-5 configuration as was also argued in Dong et al. (Dong, Loy, He, & Tang, 2015).

Table 12 - RMSE and PSNR metrics for Original dataset

Loss Layer	Dropout 50%		Dropout 50%-50%		NoDropout	
	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR
	Min	Max	Min	Max	Min	Max
Top-N (9-1-5)	0,10442	15,205	0,10537	15,087	0,10565	15,268
Top-N (9-3-5)	0,10620	15,217	0,10591	15,269	0,11408	15,257
MSE (9-1-5)	0,10806	14,293	0,10961	14,780	0,10802	14,919
MSE (9-3-5)	0,10886	14,866	0,10816	14,888	0,10793	14,930

Table 13 - RMSE and PSNR metrics for MS dataset

Loss Layer	Dropout 50%		Dropout 50%-50%		NoDropout	
	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR
	Min	Max	Min	Max	Min	Max
Top-N (9-1-5)	0,10423	15,257	0,10549	15,079	0,10486	15,067
Top-N (9-3-5)	0,10263	15,263	0,10343	15,263	0,10865	15,283
MSE (9-1-5)	0,10833	14,903	0,10977	14,807	0,10926	14,832
MSE (9-3-5)	0,10817	14,912	0,10912	14,841	0,10923	14,842

Table 14 - RMSE and PSNR metrics for LoG dataset

Loss Layer	Dropout 50%		Dropout 50%-50%		NoDropout	
	RMSE	PSNR	RMSE	PSNR	RMSE	PSNR
	Min	Max	Min	Max	Min	Max
Top-N (9-1-5)	0,10948	14,831	0,10811	14,929	0,10779	15,127
Top-N (9-3-5)	0,10608	15,031	0,10637	15,106	0,10763	15,148
MSE (9-1-5)	0,11005	14,647	0,11314	14,489	0,11093	14,803
MSE (9-3-5)	0,10955	14,005	0,10988	14,746	0,10839	14,824

4.5.2 BCDCNN Configurations & Applied Metrics

Experiments were run with two convolution kernel sizes for the second layer using 1x1 and 3x3 mapping kernels. Across all three layers of our model, the sizes of the convolution kernels we tested for were 9-1-5 and 9-3-5. In addition, the network used 64 feature maps for the first layer, 32 for the second and 1 for the last, which is henceforth denoted as 64-32-1. Furthermore, for each training data set and each convolution kernel size performance was assessed for three cases: a) No dropout; b) Dropout 50%, i.e. dropout rate of 50% after the RELU activation function of the first layer; c) Dropout 50%-50%, i.e. dropout rate of 50% after the RELU activations of the first and second layers. A learning rate of 10^{-4} was used for layers 1 and 2 while the learning rate was 10^{-5} for layer 3. In addition, the weight decay was $5 * 10^{-3}$ for all layers and the batch size was set to 128. In total, 18 experiments were run for the 64-32-1 configuration. Finally, the RMSE and PSNR metrics were utilized to assess performance of our network.

4.5.3 Detection of building contours

In Figures 37 & 38, two typical reconstructions for the training data Block1 are shown for the TopN & MSE configurations of the network. BCDCNN was able to learn the association of building contours to the input data sources.

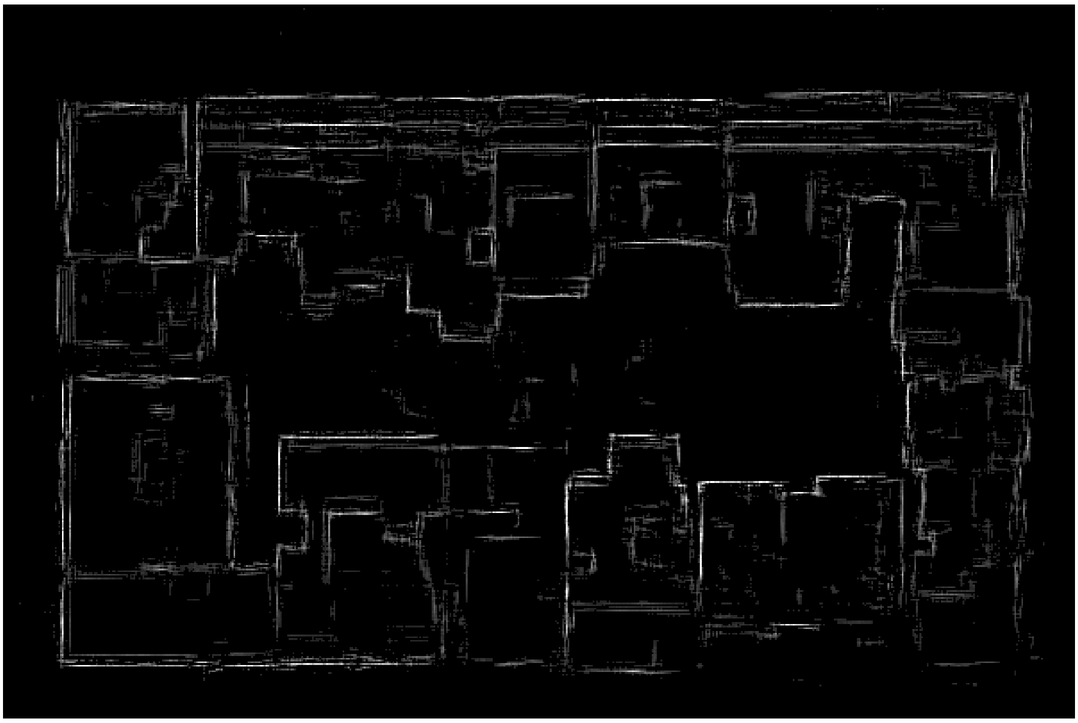


Figure 37 - Reconstruction of Train data for Original data set and Top-N Cost Layer

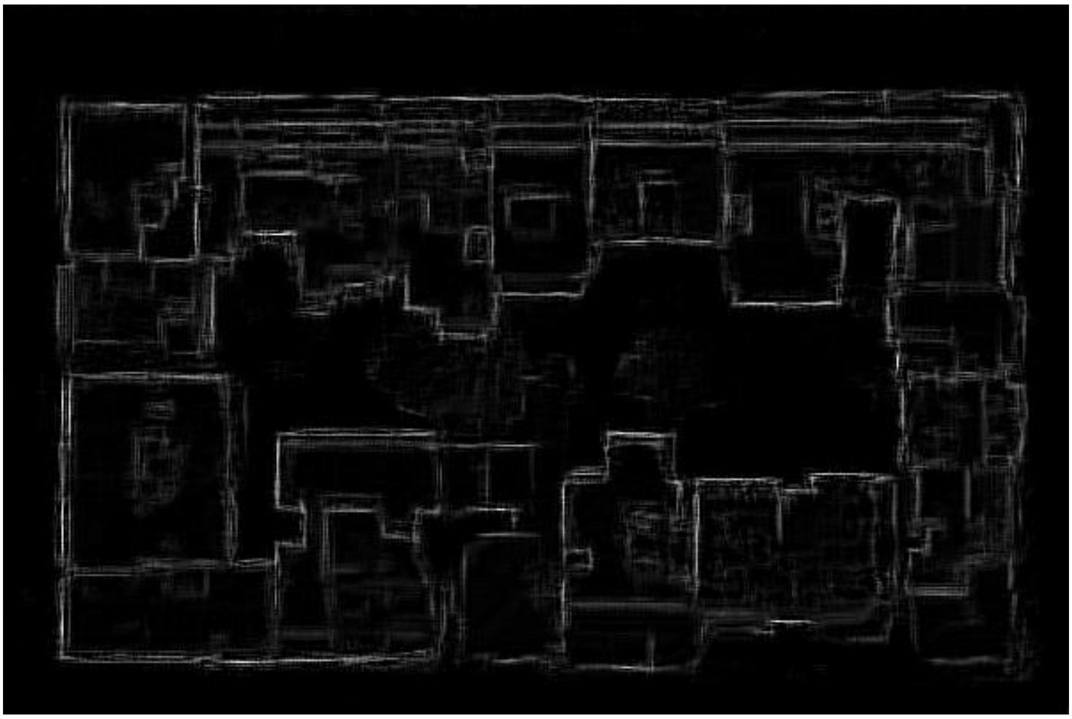


Figure 38 - Reconstruction of Train data for Original data set and MSE Cost Layer

The above reconstructions are not the best that this research produced. By increasing the number of feature maps to 256-128-1 for the three layers respectively, the network learned to detect building contours with extreme precision. This however was at the cost of training & reconstruction speed and quite detrimental to the model's generalization ability. For instance in the following Figures 39 & 40, a reconstruction obtained in early 2017 using the Caffe framework is presented. Figure 39 displays the reconstruction for the Train data and Figure 40 for the Test. From Figure 39, it is shown that given enough data and training epochs a deep neural network can learn to detect building contours with extreme precision. Unfortunately, when the data are not enough this is to the expense of the generalization capability as shown in Figure 40.

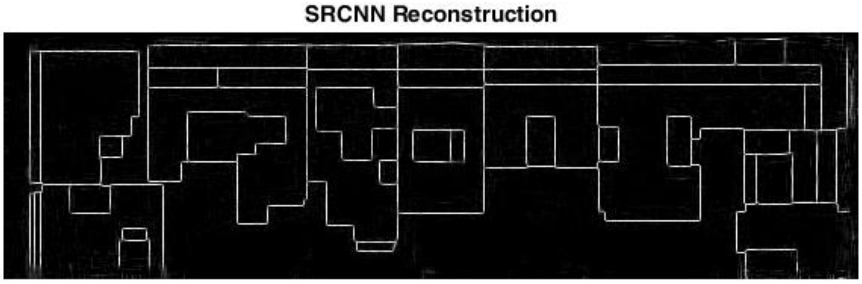


Figure 39 - Dual channel reconstruction for 256-128-1 feature maps (Train data)

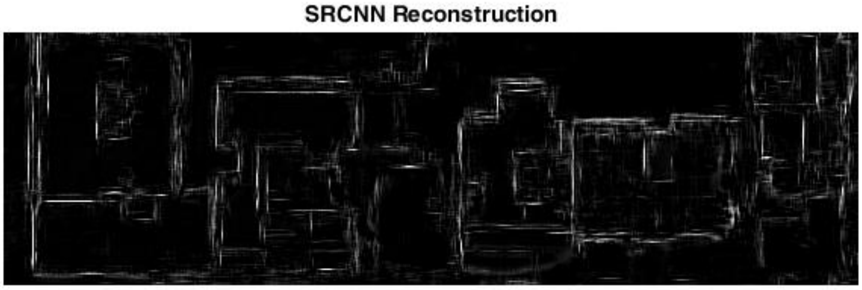


Figure 40 - Dual channel reconstruction for 256-128-1 feature maps (Test data)

However, the 64-32-1 model trained under MatConvNet can also generalize. The reconstructions of the test data for networks trained on the three variations for the proposed Top-N and MSE loss layer are shown in Figures 41 – 43.

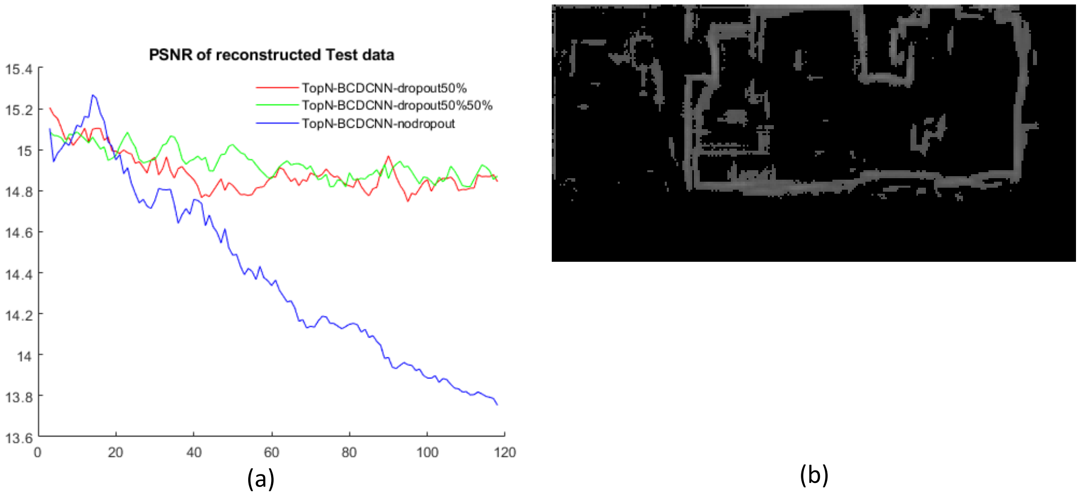
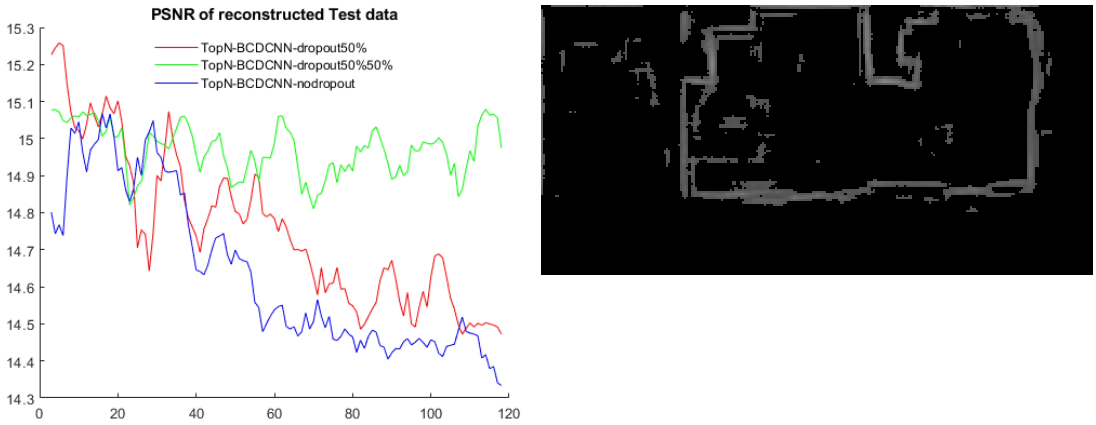


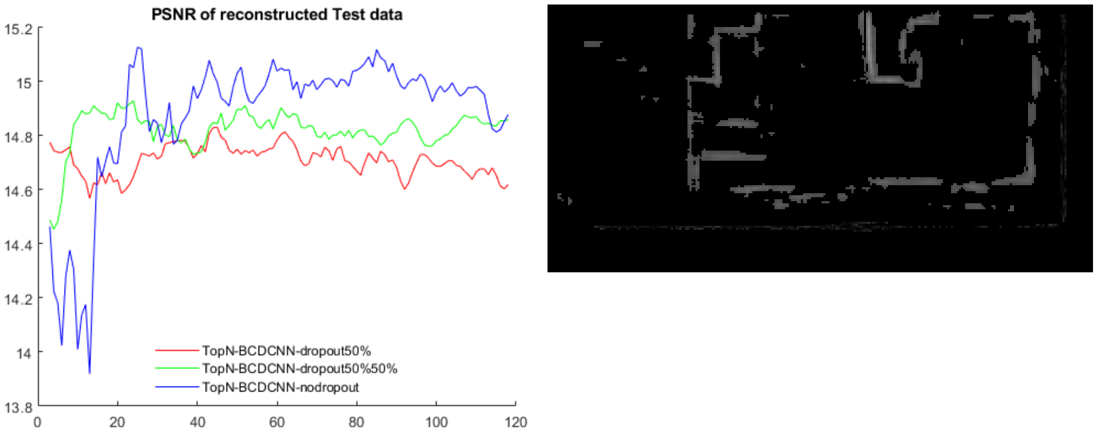
Figure 41 - Top-N reconstruction of test data. a) PSNR curve for test data on Original dataset training b) Corresponding reconstruction of test data



(a)

(b)

Figure 42 - Top-N reconstruction of test data. PSNR curve for test data on MS dataset training b) Corresponding reconstruction of test data

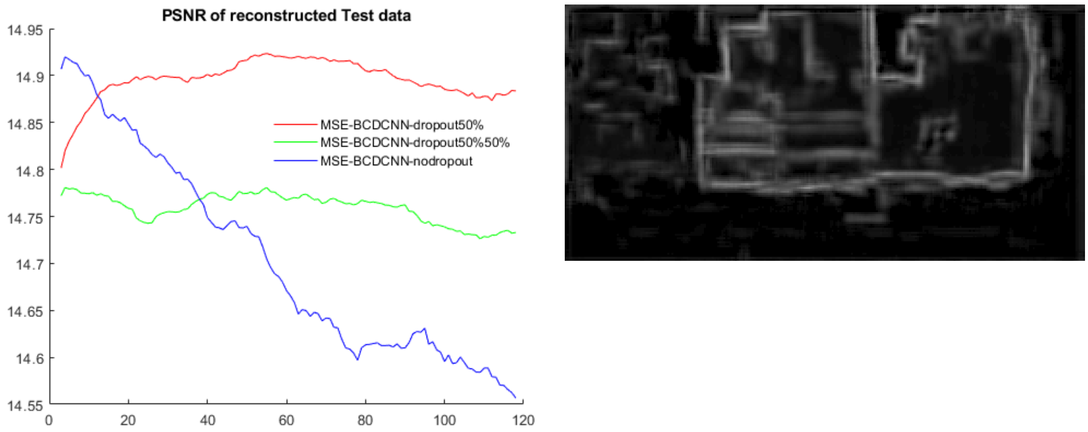


(a)

(b)

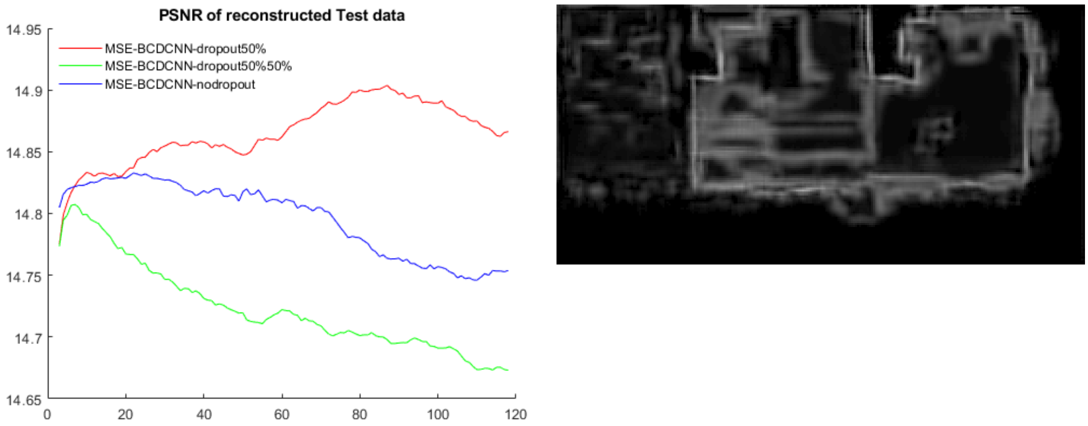
Figure 43 – Top-N reconstruction of test data. a) PSNR curve for test data on LoG dataset training b) Corresponding reconstruction of test data

According to Figure 41(a), the highest PSNR for the test data set was at epoch 55 for the dropout 50% case and the reconstruction at that specific instant is presented (Figure 41b). This process was repeated for all our training data variations and the resulting reconstructions are shown in Figures. 42(b) and 43(b). The corresponding experiments for the MSE cost layer are shown in Figures 44 to 46. It has to be noted that no post-processing was used at this stage to improve the obtained building contours, as this is the case of the relaxation system presented in chapter 6. From Figures 41-46, it can be readily observed that deciding about how to improve the generalization ability of the network is not straightforward. Perhaps, one can say that when the effective input dimensionality is high (i.e. when the variance of the input pixel values is large) as is the case for the Original data sets, the network exhibits poor generalization behavior (see the blue curves of Figures 41(a) and 42(a). As the effective dimensionality is progressively reduced through the imposed smoothing from the Mean Shift and LoG data preprocessing the generalization ability of the network is improved and, in the case of LoG, even surpasses the cases that use dropout in one or two layers.



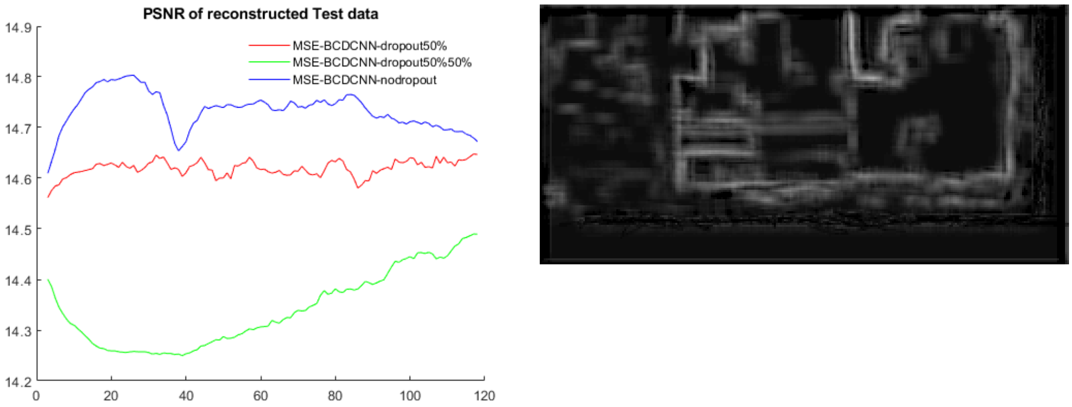
(a) (b)

Figure 44 - MSE reconstruction of test data. a) PSNR curve for test data on Original dataset training b) Corresponding reconstruction of test data



(a) (b)

Figure 45 - MSE reconstruction of test data. a) PSNR curve for test data on MS dataset training b) Corresponding reconstruction of test data



(a)

(b)

Figure 46 - MSE reconstruction of test data. a) PSNR curve for test data on LoG dataset training b) Corresponding reconstruction of test data

A second remark that we can make is that by using 50% dropout on one or two layers the network resists better to overfitting. Specifically, in the case of training data sets with relatively high effective dimensionality as is the case with the Original and Mean Shift processed data sets, dropout (either in one or in two layers) proves to be the necessary choice for network generalization. Finally, in accordance to the comparative results of Tables 12–14, a comparison of Figures 41-46 shows that there is a slight improvement in PSNR under any training data set variation when using the Top-N cost layer.

5. Super-Resolution of low-resolution digital elevation data

5.1 Context

High-resolution optical cameras for urban aerial photographs are readily available these days but LIDAR technology, despite recent advances, still produces images of comparatively lower resolution while it also remains more expensive. It is for this reason that usually aerial camera optical photographs are of much higher resolution than matching digital elevation maps (DEM) produced from the processing of LIDAR data taken from the same plane or drone.

Image super resolution reconstruction has diverse applications ranging from medical image augmentation (Baranov, Olea, & van den Bogaart, 2019) to geological applications (Wang, Armstrong, & Mostaghimmi, 2019), to the augmentation of general purpose images (Dong, Loy, He, & Tang, 2015). Regardless of the application, the goal is to upscale a low-resolution image {I_L} and produce a super resolution reconstruction {I_H'} as close as possible to the original high-resolution image {I_H}. It is a well-studied problem of computer vision with a variety of solutions but no unanimity yet on how to optimally assess the similarity of the reconstructed image to the original (Benecki , Kawulok , Kostrzewa, & Skonieczny, 2018). Super resolution can be attained through either a single image or multiple images. In the second case, the two channels can either be displaced versions of a single image or images depicting corresponding but of different nature information.

This chapter of the thesis aims to apply convolutional neural networks to augment the low resolution of elevation data augmented by corresponding optical high-resolution data. The elevation data available were of very low resolution (120x80); five times lower than the corresponding optical resolution data (600x400). Two blocks of elevation data of the fore-mentioned resolution were available. This chapter will briefly examine some previous work on the topic, the methodology followed in this thesis to accomplish super-resolution of elevation data; the experiments conducted and will reach several conclusions on the matter.

5.2 Previous work

In recent years, there has been a great interest in super-resolution applications as can be concluded by numerous published research results. Only a very brief presentation in the field of image super-resolution will thus be performed in this thesis. Facial recognition from low-resolution cameras for instance, is an important super-resolution application and Huang and He (Huang, 2010) used super resolution for facial recognition from low-resolution security cameras by applying nonlinear mappings to infer coherent features that favored higher recognition of nearest neighbor (NN) classifiers for recognition of single low-resolution face images. Furthermore, old manuscripts were given the super-resolution treatment, when Datsenko & Elad (Datsenko & Elad, 2007) applied super-resolution in order to enhance manuscripts containing text & equations. For this, they assigned to low resolution patches several high-quality candidate patches, using the nearest neighbor metric in an image database that contained low and high resolution corresponding patches.

They then used a penalty function to reject some of the irrelevant examples, keeping the rest for image reconstruction. Medical and microscopy imaging are also a very active research topic regarding super resolution and on this Huang et al. (Huang B., Wang, Bates, & Zhuang, 2008) applied super-resolution to fluorescence microscope imaging by using optical astigmatism in order to determine both axial and lateral positions of individual fluorophores with nanometer accuracy.

The majority of applications based on a single image are based on the exemplar paradigm in which learning attempts to match <low-resolution, high-resolution> image pairs that are then used to reconstruct general high-resolution images from low-resolution ones. Two of the most prominent methods in modern scientific literature are sparse representation and deep neural networks. An example of the first case is Yang et al. (Yang, Wright, Huang, & Mia, 2010) who created sparse representations for low-resolution patches of the input image and then used the coefficients of these representations to generate high-resolution reconstructions. On the contrary, Dong et al. created a deep neural network model with 3 layers that performed a single-channel super-resolved non-linear mapping between low-resolution patches and high-resolution ones during training which was used to adjust the weights of the convolution filter-bank (Dong, Loy, He, & Tang, 2015).

Super-resolution from multiple images has also offered excellent results. It is usually based on the definition of a parametrized image model (PIM) where the multiple images required for training are derived from applying image processing operators like warping, blurring, down sampling or contamination with noise. An optimization technique is then applied to find the optimal parameter values. For instance, Vilenna et al. (Villena, Abad, Molina, & Katsaggelos, 2004) implemented an iterative technique based on a Bayesian classifier, which obtained a set of under sampled and degraded frames by shifting displacements of high-resolution images. The researchers then applied an iterative Bayesian method to estimate the unknown shifts and the high-resolution image from the corresponding low-resolution one. Multi-channel deep learning techniques have been especially prominent in the past several years with many advancements and new proposals. In their research, Lee et al. (Lee, Chen, Tseng, & Lai, 2016) utilize complementary RGB-D images (color and depth) to achieve recognition that is more accurate. They first trained their network with a color RGB dataset and then fine-tuned with the depth dataset using transfer learning with the results showing a higher accuracy than a single image RGB solution.

5.3 Contribution to multi-channel super-resolution

A dual-channel input convolutional neural network learning approach for super resolution (SR) is proposed that performs a mixed-input associative mapping between a low-resolution elevation image depicting the height of an urban area in Kalliithea – Attica - Greece and a corresponding high-resolution optical image of the area. The network, named building super resolution auto-associative convolutional neural network (BSRCNN), performs a mixed-input associative mapping in the sense that it is assisted by high-resolution optical data to augment low-resolution elevation data, in order to associate them to a high-resolution rendition of themselves. Furthermore, the network was trained with a rather limited available dataset[12]. The nature of elevation data has several subtleties that differentiate this application from general-purpose SR approaches. Firstly, elevation data are a by-product of LIDAR data that usually comes in the form of a point-cloud. The result is a digital-surface-model (DSM) or a digital elevation model (DEM) that is a real-valued matrix denoting the true height of the ground/buildings, respectively. This matrix can be further processed to attain a graylevel image that corresponds to the real-valued data and this is precisely the case of the DEM data used in this chapter. The resulting image shifts from subtle graylevel differences to steep changes in the graylevel of adjacent pixels making it difficult for a nearest neighbor or a bicubic up-sampler to make an optimum choice. Furthermore, deep convolutional neural networks (CNNs) trained to perform super-resolution on generic real-world images are also not expected to perform very well because of this difference in high & low frequency content even within a convolution kernel. It was expected that a CNN trained with elevation data & augmented with high-resolution optical data of the urban would perform at least on par or better than the corresponding neural network trained only with the low-resolution elevation data.

In order to test this assumption the performance of the dual-channel approach was compared to a single-channel rendition using the MSE, PSNR and SSIM metrics. Furthermore, comparisons were made against other state of the art generic super-resolution models like that of [Dong, Loy, He, Tang, 2015]. In addition, it was tested how well various forms of super-low resolution data are augmented from high-resolution optical data. For this reason, the LR DEM data were transformed into a very low-resolution version by either the bicubic or the nearest neighbor interpolation method.

5.4 Data preparation

The optical images of the dataset were considered as the high-resolution (HR) input (600x400) while the corresponding DEM had a resolution five times lower than the optical (120 x 80), aka the characterization of low-resolution data. The high-resolution optical data and corresponding low-resolution DEM are shown in Figure 47 (a-d).

[12] Two blocks of elevation data each of resolution 120x80 and the corresponding five-time higher optical data (upscaled & downscaled to 120x80) were used for the research, as elaborated later in the chapter. These data were taken from the research of [Vassilas N., Charou, Petsa, Grammatikopoulos, 2013].

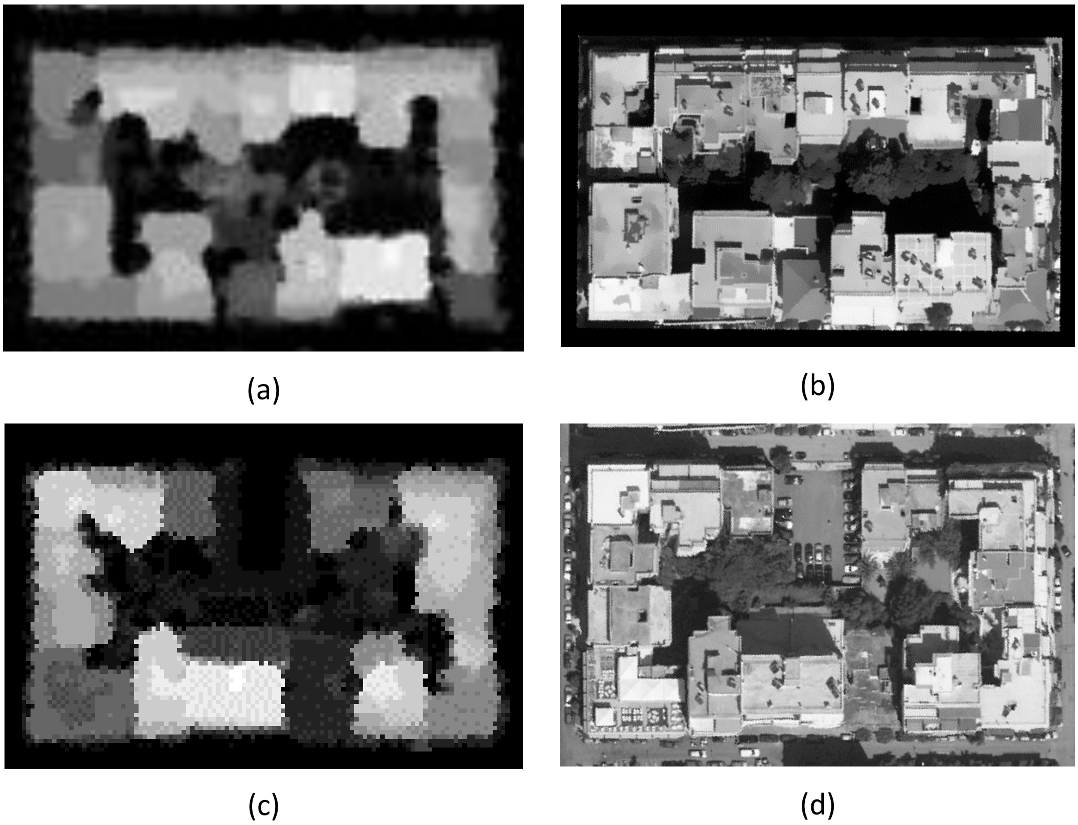


Figure 47 -DEM & Optical data. a) DEM block1 LR b) Optical block1 HR c) DEM block2 LR d) Optical block2 HR

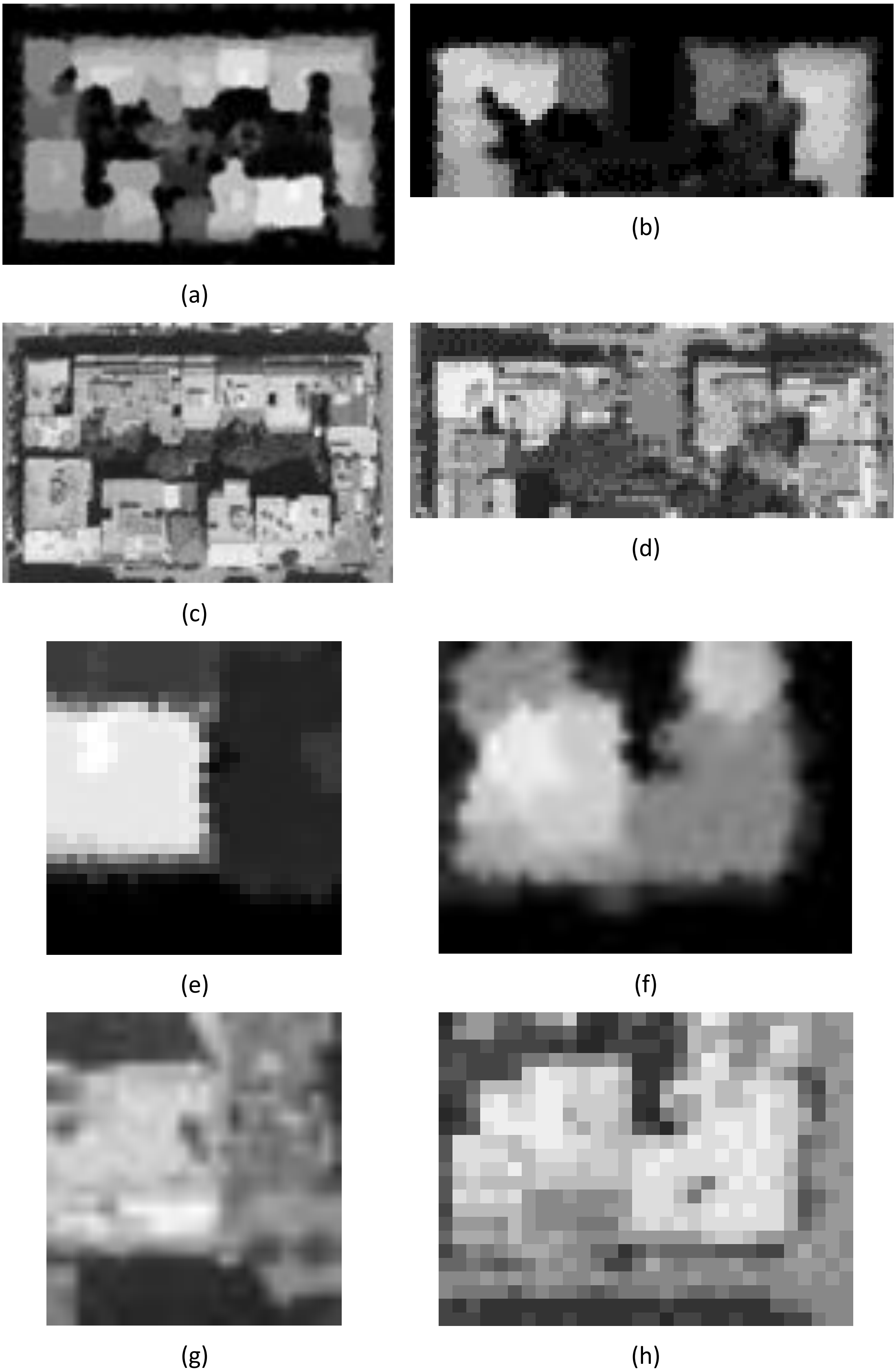
The data shown in Figure 47 (a-d) were used for training of the CNN, verification of its performance and testing its generalization capability. Two sets of training data were constructed from the aforementioned data with the second being noisier and thus a harder problem to solve. In order to render the data in a form suitable for this application, the following pre-processing was performed:

- Elevation data set 1: 120x80 (original resolution) → downscale BC by two → upscale BC x two → DEM_SLR (120x80). The original elevation data that the CNN had to augment.

- Elevation data set 2: 120x80 (original resolution) → downscale NN by two → upscale NN x two → DEM_SLR (120x80). A more difficult elevation data set with noise and spurious gray levels.

- Optical data (600x400): downscale BC x five → optical LR (120x80).

The pre-processed elevation data sets one and two as well as the pre-processed optical data are shown in Figures 48 and 49.



(a)

(b)

(c)

(d)

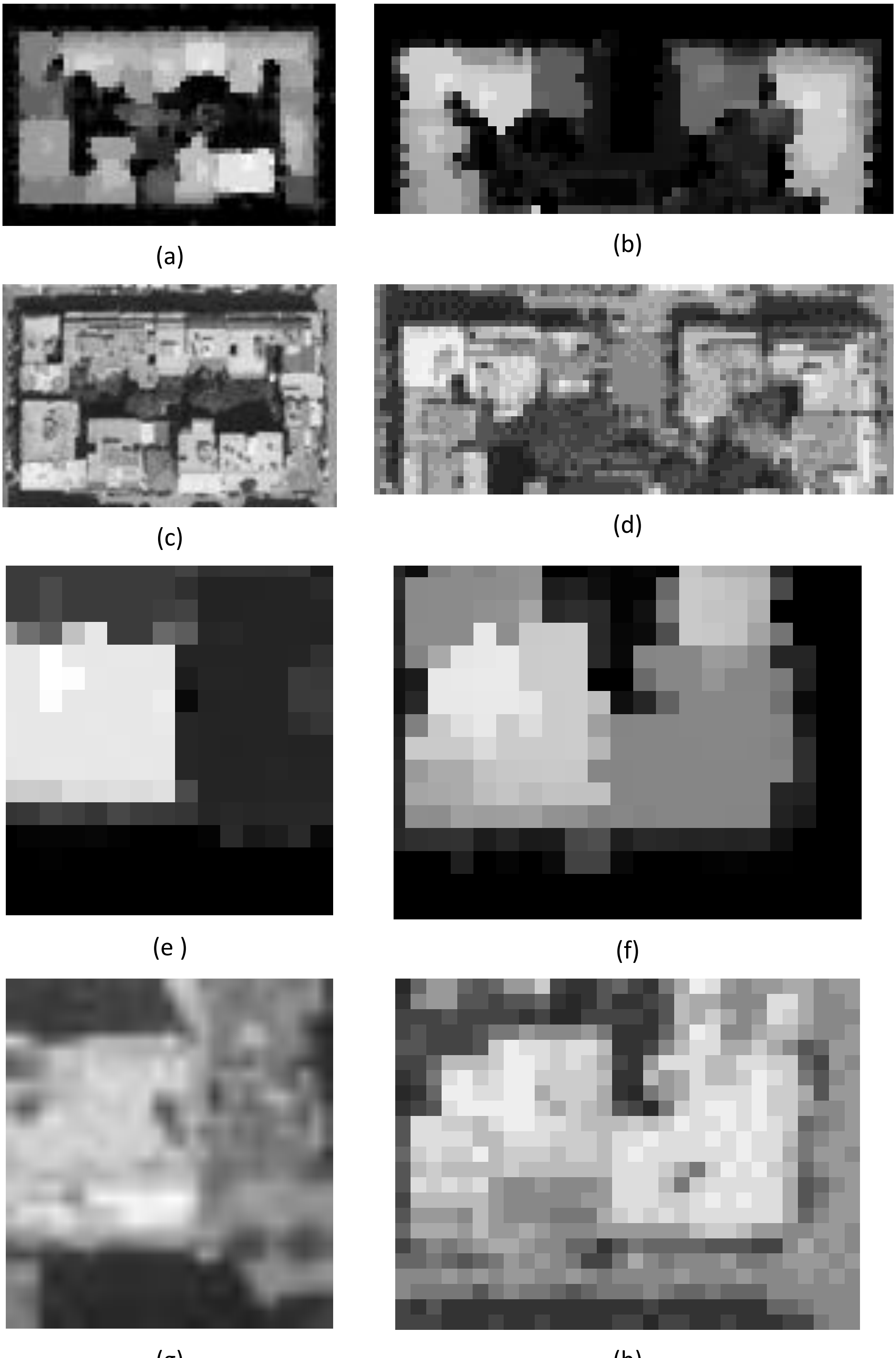
(e)

(f)

(g)

(h)

Figure 48 - Train & validation data with elevation set one. a) Block 1 LR bicubic DEM (120x80) b) part of Block 2 LR bicubic DEM (120x48) c) Block 1 optical HR (120x80) d) part of BLock 2 optical HR (120x48) e) part of Block 2 LR validation 1 DEM (29x31) f) part of Block 2 LR validation 2 (41x31) g) part of Block 2 HR validation 1 optical (29x31) h) part of Block 2 HR validation 2 optical (41x31)



(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

Figure 49 - Train and validation data with elevation set two. a) Block 1 SLR NN DEM (120x80) b) part of BLock 2 SLR NN DEM (120x48) c) Block 1 optical LR (120x80) d) part of Block 2 optical LR HR (120x48) e) part of Block 2 LR validation 1 DEM (29x31) f) part of Block 2 LR validation 2 (41x31) g) part of Block 2 HR validation 1 optical (29x31) h) part of Block 2 HR validation 2 optical (41x31)

Two sets of data were used for the training and validation of the neural network model as shown below:

- Elevation set 1: This was comprised of tuples <<Elevation SLR set one, Optical HR>, Elevation LR>[13], where Elevation set one are the data shown in [Figures 48(a-h)]. The elevation data of the data series have been down & up scaled two times from the original resolution (120x80) using bicubic resizing to create the very low-resolution Elevation (slr_dem) set one. Furthermore, this data set was further divided for training and validation purposes as follows:
  - Train: The whole of block 1 down & up scaled using bicubic resizing [Figures 48(a, c)] and two-thirds of block 2 down & up scaled using bicubic resizing[14] [Figures 48(b, d)]. The train set is comprised of very low-resolution elevation (slr_dem) and optical low-resolution (opt_lr) data for input as well as the low-resolution elevation GT data (lr_dem)[15].
  - Validation 1: A part of block 2 down & up scaled using bicubic resizing for the elevation data [Figure 48(e, g)]. Validation set 1 was comprised of very low-resolution elevation (slr_dem) and optical low-resolution (opt_lr) data for input as well as the low-resolution elevation GT data (lr_dem).
  - Validation 2: A part of block 2 down & up scaled using bicubic resizing for the elevation data [Figure 48(f, h)].)]. Validation set 2 was comprised of very low-resolution elevation (slr_dem) and optical low-resolution (opt_lr) data for input as well as the low-resolution elevation GT data (lr_dem).

Finally, sliding windows of various resolutions were used to slice the very low-resolution elevation slr_dem) and LR optical data (opt_lr) into segments for training and validation purposes. The resolution depended on the convolution kernels used[16]. Similarly, a smaller window was used to slice

[13] The first tuple of the training scheme <Elevation LR set one, Optical HR> stands for a dual channel configuration where the first channel is comprised of elevation data and the second of optical. Similarly, the last tuple <, Elevation_LR> stands for the GT LR elevation data. The corresponding single-channel configuration scheme would be <Elevation_SLR, Elevation_LR> where the network would only accept a single elevation channel Elevation_SLR as input and Elevation_LR would be the GT.

[14] Only the elevation data have been resized during this step. The same is true for the validation and test data.

[15] The input elevation data are very low resolution because they were down and up scaled from the initial low-resolution elevation data (slr_dem) while the optical data are low-resolution since they have been downscale 5 times to reach the elevation data resolution (opt_lr). Finally, the GT data are the original low-resolution elevation data (lr_dem).

[16] As explained in Section 5.5 the proposed model is comprised of three layers. A typical convolution kernel used was (9-1-5), which means 9x9, 1x1, 5x5 kernels for the first, second and third layer, respectively. For this case, the sliding window could be 33x33 for the SLR elevation and LR optical data and 21x21 for the ground truth LR elevation data. The resolution discrepancy for the GT data is because the convolution kernels do not precisely fit on the corners of the image, leading to a reduction of resolution (8 pixels 1st layer, none for the second and 4 pixels for the third) of the feature maps as they

the ground truth LR elevation data into segments for training & validation purposes. The data segments were artificially augmented in much the same way as was done for BCDCNN but this time using a stride of one[17] due to the much lower resolution of the available data. Finally, tuples were created comprising of the data segments in the form <<slr_dem, lr_opt>, lr_dem>, where slr_dem is the very-low resolution elevation data; lr_opt: the low-resolution optical data; and lr_dem: the low-resolution elevation data.

- Set 2: This was comprised of tuples <<Elevation SLR set two, Optical HR>, Elevation LR> where Elevation set two are the data shown in [Figures 49(a-h)]. The elevation data of the data series have been down & up scaled two times from the original resolution (120x80) using nearest neighbor resizing to create the Elevation SLR set two. Furthermore, this data set was further divided for training and validation purposes as follows:
    - Train: The whole of block 1 down & up scaled using nearest neighbor [Figures 49(a, c)] and two-thirds of block 2 down & up scaled using nearest neighbor [Figures 49(b, d)]. The train set is comprised of very low-resolution elevation (slr_dem) and optical low-resolution (opt_lr) data for input as well as the low-resolution elevation GT data (lr_dem)[18].
    - Validation 1: A part of block 2 down & up scaled using nearest neighbor resizing for the elevation data [Figure 49(e, g)]. Validation set 1 was comprised of very low-resolution elevation (slr_dem) and optical low-resolution (opt_lr) data for input as well as the low-resolution elevation GT data (lr_dem).
    - Validation 2: A part of block 2 down & up scaled using nearest neighbor resizing for the elevation data [Figure 49(f, h)]. Validation set 2 was comprised of very low-resolution elevation (slr_dem) and optical low-resolution (opt_lr) data for input as well as the low-resolution elevation GT data (lr_dem).

The data were then sliced into segments as described for the bicubic-resized data and were artificially augmented in much the same way as was done for BCDCNN but this time using a slice of one due to the much lower resolution of the available data. Finally, tuples were created comprising of the data segments in the form <<slr_dem, lr_opt>, lr_dem>, where slr:dem the super-low resolution elevation data; lr_opt: the low-resolution optical data; and lr_dem: the low-resolution elevation data.

are propagated through the network. The same is true of other sliding windows size used (23x23->11x11, 21x21->9x9, 19x19->7x7, 17x17->5x5) for a 9-1-5 model.

[17] A stride of two was used for the training data of BCDCNN due to the larger training set. This means that every second row and column was skipped.

[18] The input elevation data are very low resolution because they were down and up scaled from the initial low-resolution elevation data (slr_dem) while the optical data are low-resolution since they have been downscale 5 times to reach the elevation data resolution (opt_lr). Finally, the GT data are the original low-resolution elevation data (lr_dem).

The total number of used for both set one or set two training examples were:

- 120x80 (block 1) = 9600 examples * 4 (rotations for 90°, 270° and 360°) * 4 (vertical flipping & rotations for 90°, 270° and 360°) = 153.600.
- 120x80 (block 2) * 2/3 = 6336 examples * 4 (rotations for 90°, 270° and 360°) * 4 (vertical flipping & rotations for 90°, 270° and 360°) = 101.376.

The number of training examples were thus substantially less than the data available for the training of BCDCNN due to the lower resolution of the available data.

5.5 Proposed model

The proposed model is based on the single channel super-resolution architecture by Dong et al. (Dong, Loy, He, & Tang, 2015), which was modified by adding another input channel. The modified version has been named BSRCNN (Building super resolution convolutional neural network). In contrast to SRCNN, which performs a super-resolved auto associative mapping between the low-resolution image and the high-resolution version of itself, this network implements a super-resolved enhanced input auto associative mapping. This means that the network can map a low-resolution image, depicting in this case elevation data, onto a high-resolution version of itself with the assistance of a high-resolution optical image. In essence, BSRCNN accepts as input a super low-resolution elevation image combined with a corresponding low-resolution optical image and this constitutes the mixed resolution input X. The goal of the convolutional network is then to reconstruct an image F(X) that is similar to the corresponding ground truth high-resolution elevation image Y. In order to accomplish this and in a similar manner to BCDCNN, BSRCNN uses a mapping F from input to reconstruction (output) which consists of the following three operations:

- Patch extraction and representation. Patches from the mixed resolution image X are extracted, then processed by the filter bank of the first convolutional layer and, finally, represented as a set of feature maps. This can be mathematically expressed as the operation:

$$\boldsymbol{F_1(X) = max(0, W_1 * X + B_1)} \qquad (7)$$

where $\boldsymbol{W_1} = \{\boldsymbol{W_1{}^k} \mid 1 \le k \le \boldsymbol{N_1}\}$ and $\boldsymbol{B_1} = \{\boldsymbol{B_1{}^k} \mid 1 \le k \le \boldsymbol{N_1}\}$ with $\boldsymbol{w_1^k}$ being the k-th 3-D filter of the first layer’s filter bank $\boldsymbol{w_1}$, $\boldsymbol{B_1^k}$ the corresponding bias term and $\boldsymbol{F_1(X)}$ the set of $\boldsymbol{N_1}$ feature maps. As induced by eq. (1), this layer includes a ReLU non-linearity. Each of the $\boldsymbol{N_1}$ filters is of size $\boldsymbol{s_1 x s_1 x\, N_0}$, with $\boldsymbol{N_0}$ denoting the number of channels in the input image ($\boldsymbol{N_0}$=2 for the first layer). Finally, operator ‘*’

signifies convolution.

- Non-linear feature map transformation. In the second convolutional layer, the $\boldsymbol{N_1}$ feature maps generated by the previous operation are non-linearly transformed into another set of $\boldsymbol{N_2}$ feature maps by applying $\boldsymbol{N_2}$ filters of size $\boldsymbol{s_2 x s_2 x N_1}$ 1 and then, as before, passing the results from a ReLU. This operation can be described mathematically as

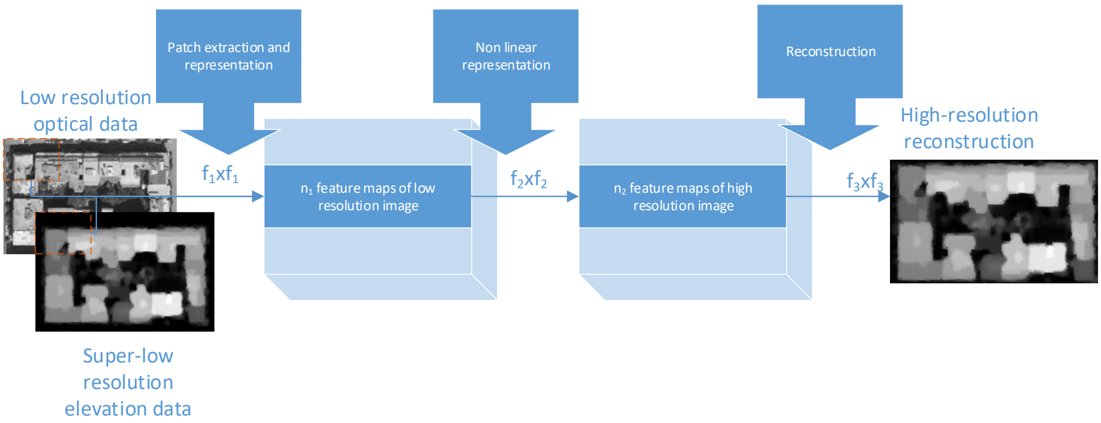
$$\boldsymbol{F_2(X) = \max(0, W_2 * F_1(X) + B_2} \qquad (8)$$

where $\boldsymbol{W_2}$ contains $\boldsymbol{N_2}$filters of size $\boldsymbol{s_2 x s_2 x N_1}$ and $\boldsymbol{B_2}$is $\boldsymbol{N_2}$ dimensional.

- High-resolution elevation data reconstruction: Finally, the feature maps of the previous stage are aggregated to generate the high-resolution elevation data image. The reconstruction operation is implemented as a linear convolution layer,

$$\boldsymbol{F_3(X) = W_3 * F_2(X) + B_3} \qquad (9)$$

One thing that has to be said about the network is its extreme sensitivity to training parameters since the final layer has no bounding activation function. The network architecture can be seen in Figure 50.



Patch extraction and representation
Non linear representation
Reconstruction
Low resolution optical data
f_1xf_1
n_1 feature maps of low resolution image
f_2xf_2
n_2 feature maps of high resolution image
f_3xf_3
High-resolution reconstruction
Super-low resolution elevation data

Figure 50 - BSRCNN Proposed three-layer architecture

Similarly to BCDCNN, the proposed model has a difficult task to solve. This time the complexity increases even further due to the scarcity of available data (1 and 2/3 120x80 mixed-resolution image was available as opposed to 1 and 2/3 600x400 mixed-resolution image for BCDCNN). Furthermore, the original high-resolution optical data of 600x400 were down sampled five times to reach the low-resolution of the elevation data.

5.6 Experiments

5.6.1 Neural network configuration & metrics

The neural network was trained under the MatConvNet (MatConvNet, n.d.) Matlab framework, which is a specialized framework for deep convolutional networks. It can operate both in CPU and GPU mode. The experiments were conducted in GPU mode.

A series of experiments were conducted using various combinations of the configurations in Table 15. For example, a 7-1-5 convolution kernel x 17x17 window size x 32-16-1 feature maps, would be a full configuration of an experiment[19].

Table 15 - Experiments configurations

Convolution kernel size	**7-1-5**	**9-1-5**	**9-3-5**		
Window size	17x17	19x19	21x21	23x23	33x33
Feature maps	32-16-1	64-32-1			

The full list of experiments (dual channels) is listed in Table 16. No tests were conducted with more than 64-32-1 feature maps due to the relative low amount of available training data. This will be made clear when the experimental results are presented in the next section.

Table 16 - Experiments configuration (dual-channel)

Experiment configuration	
Elevation set one (bicubic)	Elevation set 2 (NN)
7-1-5 x 17-17 x 32-16-1	7-1-5 x 17-17 x 32-16-1
7-1-5 x 19-19 x 32-16-1	7-1-5 x 19-19 x 32-16-1
7-1-5 x 21-21 x 32-16-1	7-1-5 x 21-21 x 32-16-1
7-1-5 x 23-23 x 32-16-1	7-1-5 x 23-23 x 32-16-1
7-1-5 x 33-33 x 32-16-1	7-1-5 x 33-33 x 32-16-1
7-1-5 x 17-17 x 64-32-1	7-1-5 x 17-17 x 64-32-1
7-1-5 x 19-19 x 64-32-1	7-1-5 x 19-19 x 64-32-1
7-1-5 x 21-21 x 64-32-1	7-1-5 x 21-21 x 64-32-1

[19] This experiment would be denoted as 7-1-5 x 17-17 x 64-32-1 and this denotation is followed henceforth.

7-1-5 x 23-23 x 64-32-1	7-1-5 x 23-23 x 64-32-1
7-1-5 x 33-33 x 64-32-1	7-1-5 x 33-33 x 64-32-1
9-1-5 x 17-17 x 32-16-1	9-1-5 x 17-17 x 32-16-1
9-1-5 x 19-19 x 32-16-1	9-1-5 x 19-19 x 32-16-1
9-1-5 x 21-21 x 32-16-1	9-1-5 x 21-21 x 32-16-1
9-1-5 x 23-23 x 32-16-1	9-1-5 x 23-23 x 32-16-1
9-1-5 x 33-33 x 32-16-1	9-1-5 x 33-33 x 32-16-1
9-1-5 x 17-17 x 64-32-1	9-1-5 x 17-17 x 64-32-1
9-1-5 x 19-19 x 64-32-1	9-1-5 x 19-19 x 64-32-1
9-1-5 x 21-21 x 64-32-1	9-1-5 x 21-21 x 64-32-1
9-1-5 x 23-23 x 64-32-1	9-1-5 x 23-23 x 64-32-1
9-1-5 x 33-33 x 64-32-1	9-1-5 x 33-33 x 64-32-1
9-3-5 x 17-17 x 64-32-1	
9-3-5 x 19-19 x 64-32-1	
9-3-5 x 21-21 x 64-32-1	
9-3-5 x 23-23 x 64-32-1	
9-3-5 x 33-33 x 64-32-1	

In order to assess how better the dual-channel proposed network performed in comparison to the equivalent single channel network[20], the following single-channel experiments were conducted.

Table 17 - Experiments configuration (single-channel)

Experiment configuration
Elevation set one (bicubic)
7-1-5 x 17-17 x 64-32-1
9-1-5 x 17-17 x 32-16-1
9-1-5 x 19-19 x 32-16-1
9-1-5 x 21-21 x 32-16-1
9-1-5 x 23-23 x 32-16-1

[20] The single channel equivalent was trained with the same data but only for elevation set one. The tuples were of the form <slr_dem, lr_dem>, where slr_dem: the super-low resolution elevation data and lr_dem: the GT low-resolution elevation data.

9-1-5 x 33-33 x 32-16-1
9-1-5 x 17-17 x 64-32-1
9-1-5 x 19-19 x 64-32-1
9-1-5 x 21-21 x 64-32-1
9-1-5 x 23-23 x 64-32-1
9-1-5 x 33-33 x 64-32-1

It was previously noted that the model is very sensitive to variations of the learning parameters. For this reason and after extensive testing, a fixed learning rate of 10^{-4} for layers 1 and 2 and 10^{-5} for layer 3, were selected. In addition, the weight decay was set to $5 * 10^{-3}$ for all layers and the batch size to 128. In total, 25 experiments for the elevation set one and 20 experiments for the elevation set two were conducted (dual-channel) while a further 11 experiments were conducted for a single channel (DEM only). Finally, the MSE, PSNR and SSIM metrics were used to assess performance of the network. A brief explanation of each used metric follows:

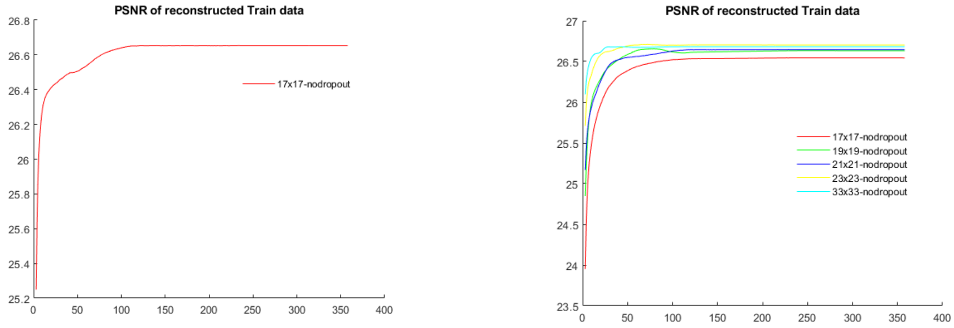
- MSE: 'The mean squared error (MSE) or mean squared deviation (MSD) of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value. MSE is a risk function, corresponding to the expected value of the squared error loss. The fact that MSE is almost always strictly positive (and not zero) is because of randomness or because the estimator does not account for information that could produce a more accurate estimate' (Wikipedia)
- PSNR: 'Peak signal-to-noise ratio, is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. Because many signals have a very wide dynamic range, PSNR is usually expressed in terms of the logarithmic decibel scale'. (Wikipedia)
- SSIM: 'The structural similarity (SSIM) index is a method for predicting the perceived quality of digital television and cinematic pictures, as well as other kinds of digital images and videos'. (Wikipedia). It ranges in values from 0 to 1, where 0 signifies total irrelevance between two images and 1 total similarity.

5.6.2 Experimental results

The first test the proposed network had to pass was the performance improvement it could offer versus a single-channel (optical) solution. Having been then proven to offer superior performance in this regard, it was tested against well-accepted image processing resizing techniques like the bicubic up-sampler. Finally, it was also tested against state of the art techniques like the single-channel Dong et al. (Dong, Loy, He, & Tang, 2015) model of which BSRCNN is a modification.

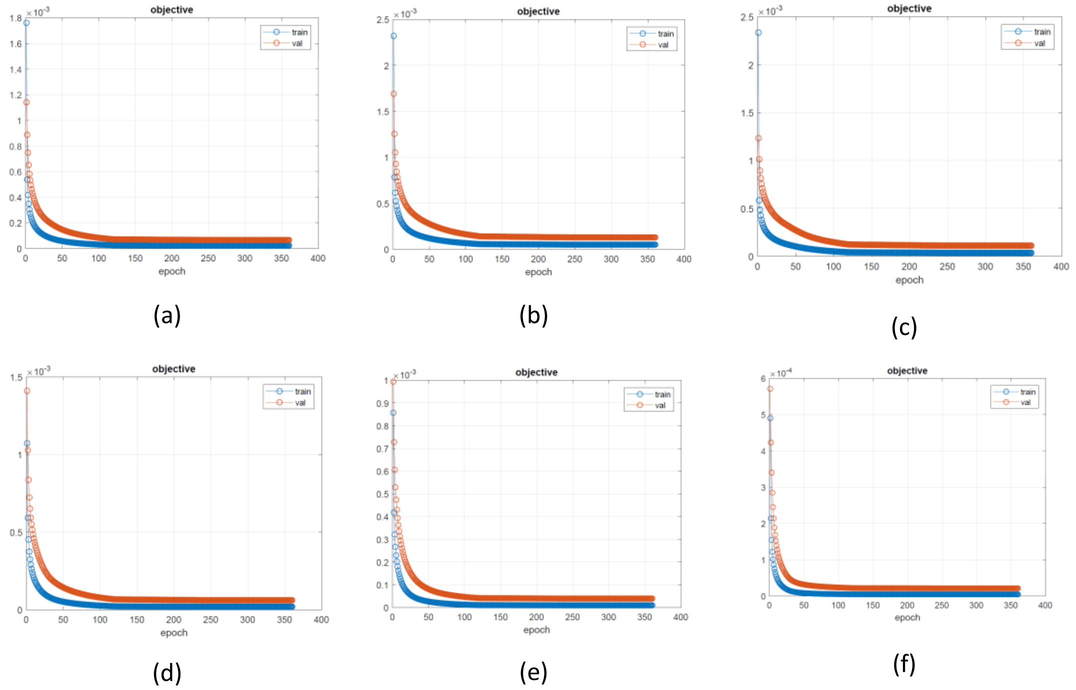
5.6.2.1 Single channel results (elevation set 1)

The single channel performance of the network can be seen for the Train data set (Figure 51). The network had been trained on these data so the good performance was expected. Figures 51 depicts the PSNR for the whole image while Figures 52(a-f) display the performance during training. More analytically, the curves of Figure 51 depict the PSNR between the reconstructed LR DEM and the original LR DEM, when given the SLR DEM as input. Similarly, Figures 52(a-f) depict the running average of RMSE error for the patches used during training and thus have a finer granularity. It can be clearly seen that the RMSE is constantly dropping for both the training and validation patches. Furthermore, the larger convolution kernel of 9-1-5 offers a slight performance increase in comparison to the 7-1-5 case both for the PSNR and SSIM metrics as can been from Table 18.



(a) (b)

Figure 51 - PSNR single channel. a) 7-1-5 64-32-1 b) 9-1-5 64-32-1

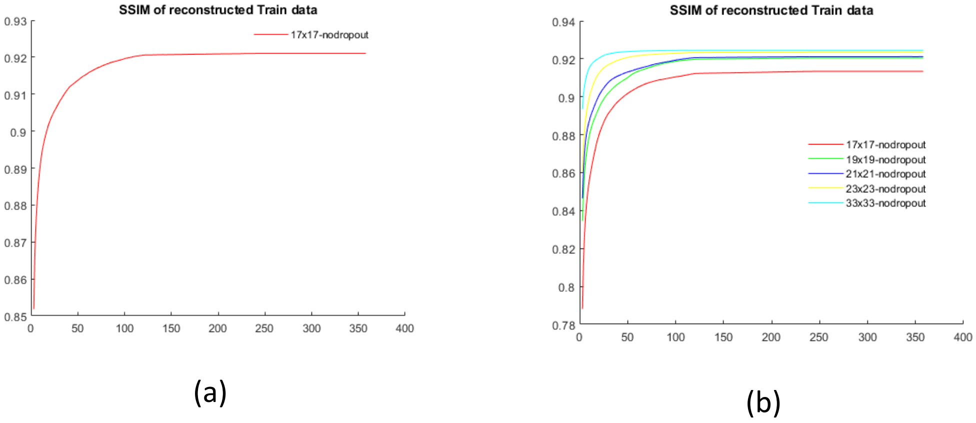


(a) (b) (c)

(d) (e) (f)

Figure 52 - Training curves single channel 64-32-1. a) 7-1-5 17x17 b) 9-1-5 17x17 c) 9-1-5 19x19 d) 9-1-5 21x21 e) 9-1-5 23x23 f) 9-1-5 33x33

The SSIM metric curves are similarly shown in Figures 53(a-b). Regarding, the reconstruction it is good and clearly better than the corresponding bicubic reconstruction as can be seen in Figures 54(a-c).

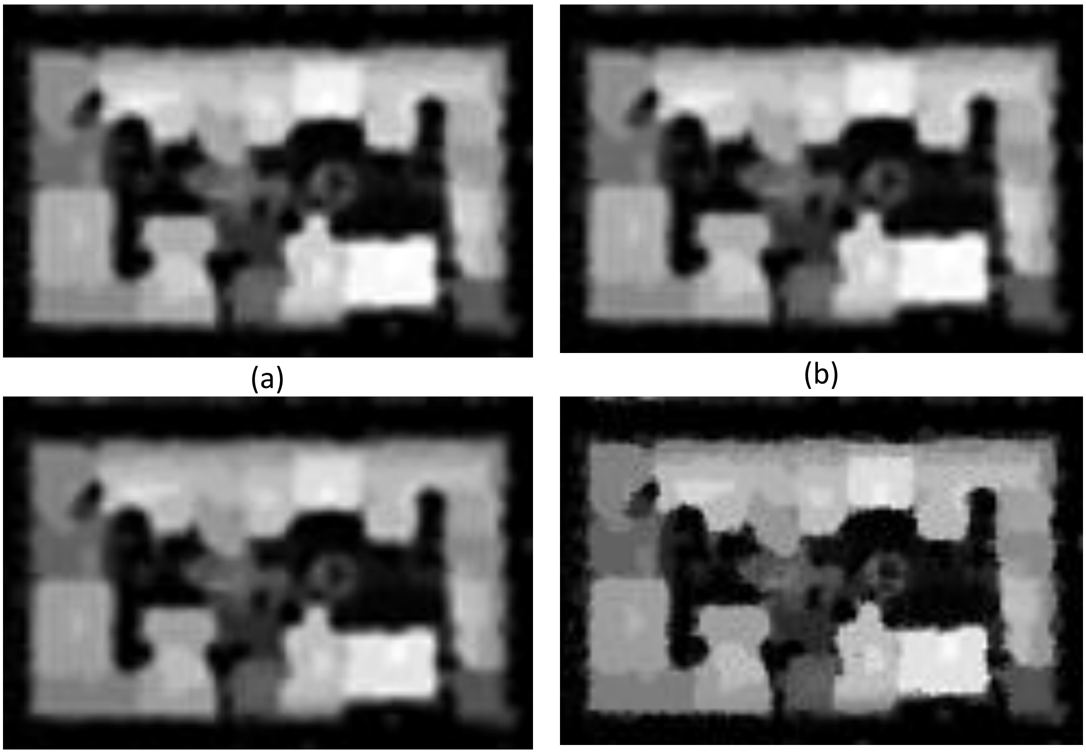


(a) (b)

Figure 53 - SSIM Single channel Train data. a) 7-1-5 64-32-1 b) 9-1-5 64-32-1

Table 18 - Single channel PSNR and SSIM for Train data set.

Metric	Convolution kernel size		Bicubic
	7-1-5	9-1-5	
PSNR	26,65361	26,70987	22,742064
SSIM	0,9210329	0,9245114	0,901115

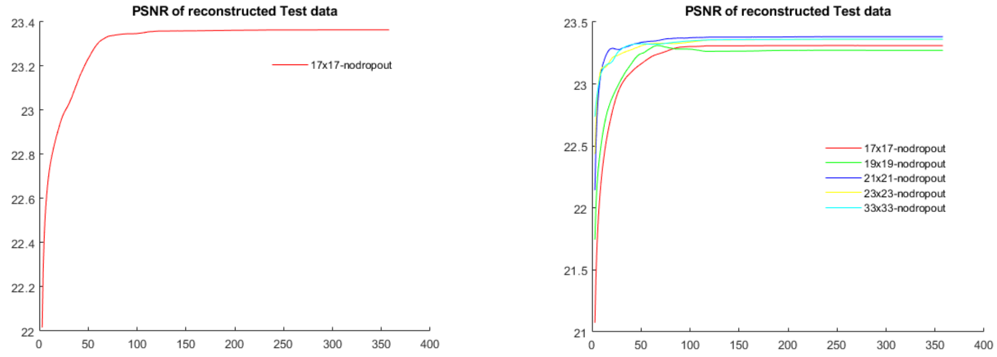


(a) (b)

(c) (d)

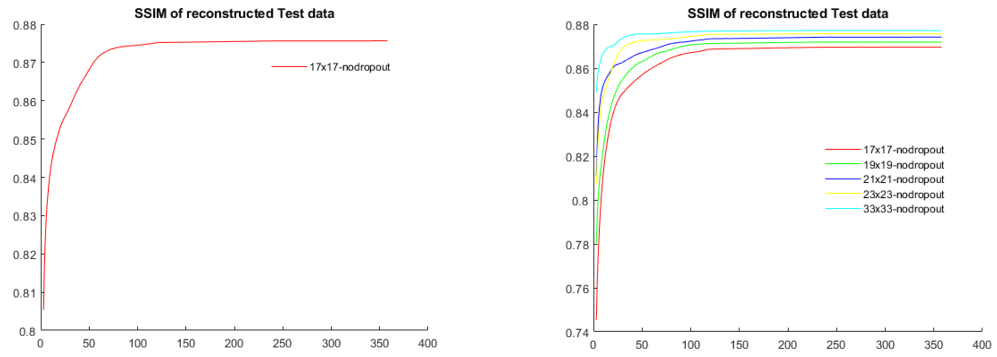
Figure 54 - 64-32-1 Single channel Train reconstruction. a) 7-1-5 b) 9-1-5 c) bicubic d) GT

Moving on to the validation set 2 data[21], the corresponding PSNR and SSIM curves can be seen in Figures 55(a-d). Furthermore, as can be seen from Table 19 the proposed method performs better than the bicubic method even with the relative limited training set that was used. On top of that, there is slight performance increase when moving to a bigger convolution kernel (7-1-5 -> 9-1-5). Both the PSNR and SSIM metric are better for the 9-1-5 kernel.



(a) (b)

[21] Validation set 2 is named Test data on the curves.



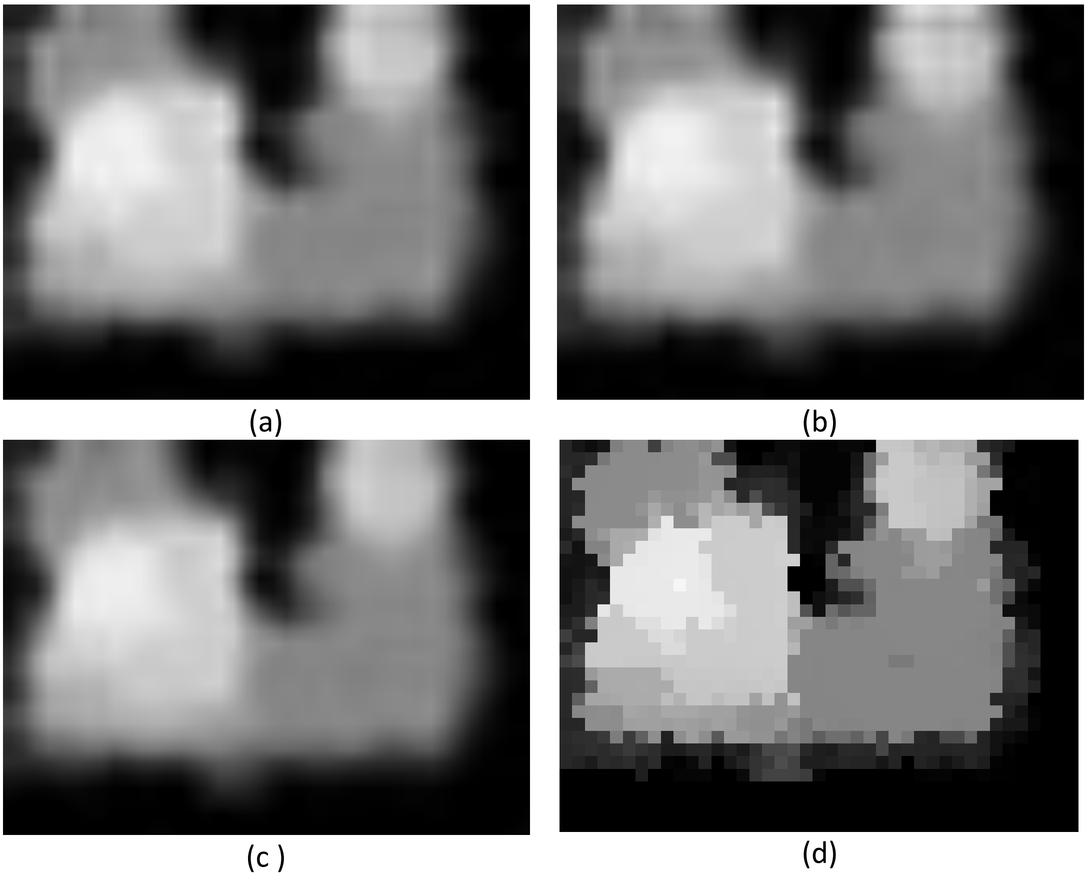
(c) (d)

Figure 55 - PSNR & SSIM single channel validation set 2 64-32-1. a) PSNR 7-1-5 b) PSNR 9-1-5 c) SSIM 7-1-5 d) SSIM 9-1-5

Table 19 - Single channel PSNR and SSIM for validation set 2 data set

Metric	Convolution kernel size		Bicubic
	7-1-5	9-1-5	
PSNR	23,36433	23,38185	21,012714
SSIM	0,8756447	0,8771589	0,858634

As can be seen from Table 19, the proposed single-channel model trained with the bicubic down and up scaled data performs much better in comparison to the bicubic case. The reconstructions can be seen in Figure 56.



(a)

(b)

(c)

(d)

Figure 56 - 64-32-1 Single channel validation set 2 reconstruction. a) 7-1-5 b) 9-1-5 c) bicubic d) GT

5.6.6.2 Single channel results (elevation set 2)

This data set is a different case since the data have been downscaled and up scaled using the nearest neighbor algorithm. This algorithm produces jaggy edges and is noisier. The trained model was expected to perform worse than the previous case. The PSNR and SSIM curves for this case can be seen in Figures 57(a-b).

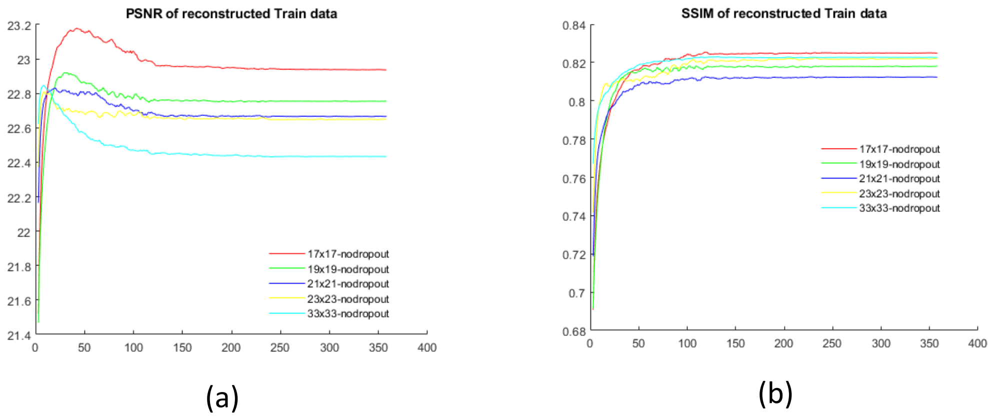


Figure 57 - Single channel PSNR & SSIM for Train data. Elevation set 2. 9-1-5 64-32- a) PSNR b) SSIM

Table 20 lists the maximum PSNR and SSIM values for the models trained with elevation set 2, 9-1-5 32-64-1 among the 17x17, 19x19, 21x21, 23x23 and 33x33 cases. Comparing Tables 18 (elevation set 1 Train results) & 20 (elevation set 2 Train results) it can be clearly seen that the second network does not learn to augment the resolution as well. The noisier data of the nearest neighbor interpolation have an adverse effect on performance. The reconstruction can be seen in Figure 58.

Table 20 - Single channel PSNR and SSIM for Train data set.

Metric	Convolution kernel size	Bicubic
	9-1-5	
PSNR	23,17907	22,742064
SSIM	0,8254921	0,901115

The corresponding validation set 2 curves are shown in Figures 59(a-b) and the analytic values in Table 21. Comparing Tables 19 and 21, it is clear that the model can reconstruct far better when trained with elevation set 1. The added complexity that the nearest neighbor noisy upscaling brings about to the data is the main cause. Lastly, the reconstruction can be seen in Figure 60.

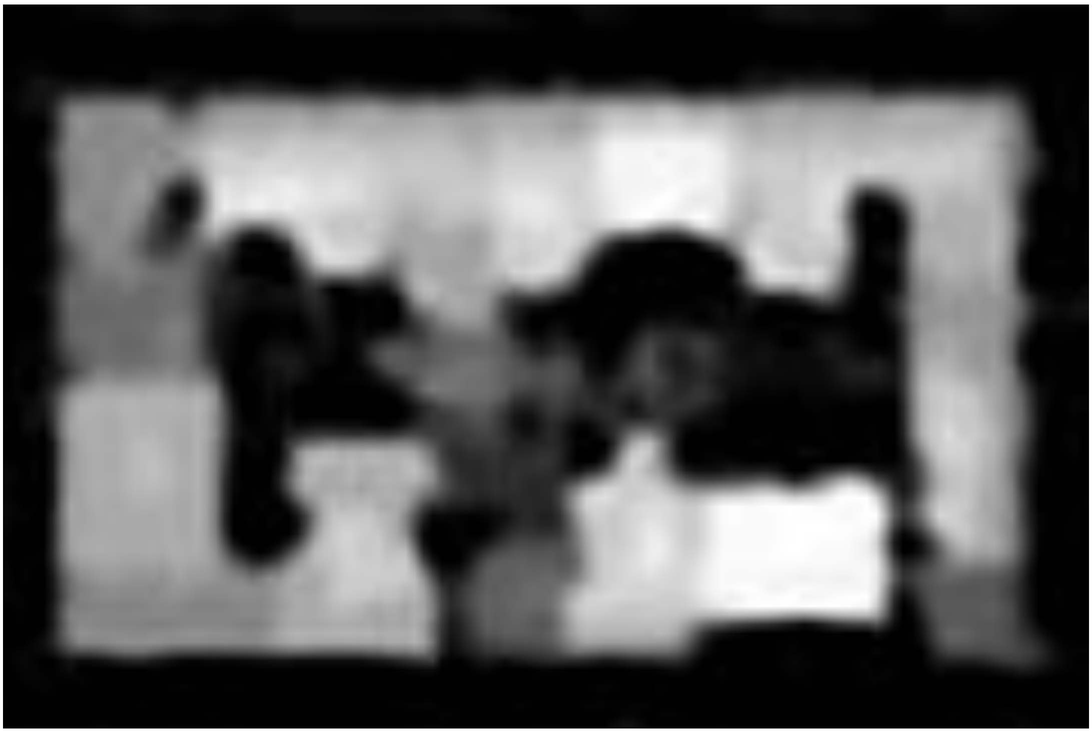
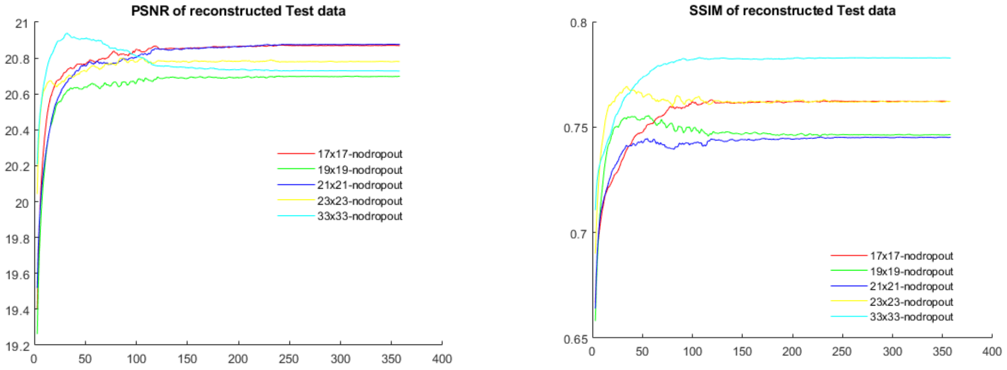


Figure 58 - 64-32-1 Single channel Train reconstruction (elevation set 2)



(b)

(a)

Figure 59 - Single channel PSNR & SSIM For validation set 2 data (Elevation set 2) 9-1-5 64-32- a) PSNR b) SSIM

Table 21 - Single channel PSNR and SSIM for validation set 2 data set

Metric	Convolution kernel size	Bicubic
	9-1-5	
PSNR	20,93913	21,012714
SSIM	0,7829993	0,858634

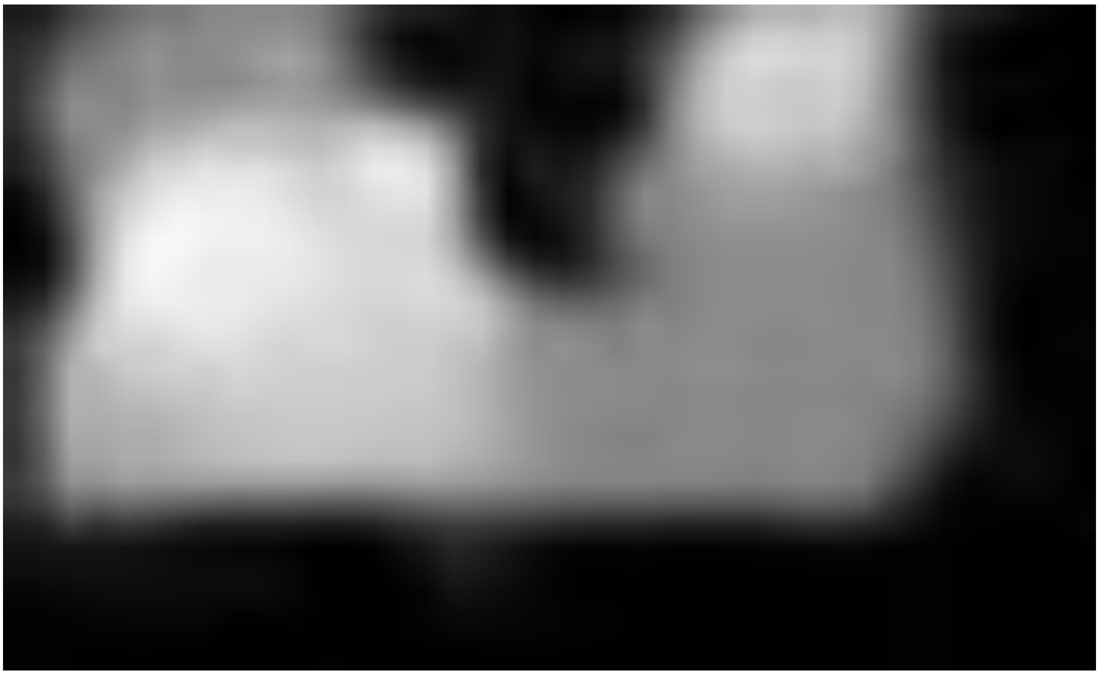


Figure 60 - 64-32-1 Single channel validation set 2 reconstruction (elevation set 2)

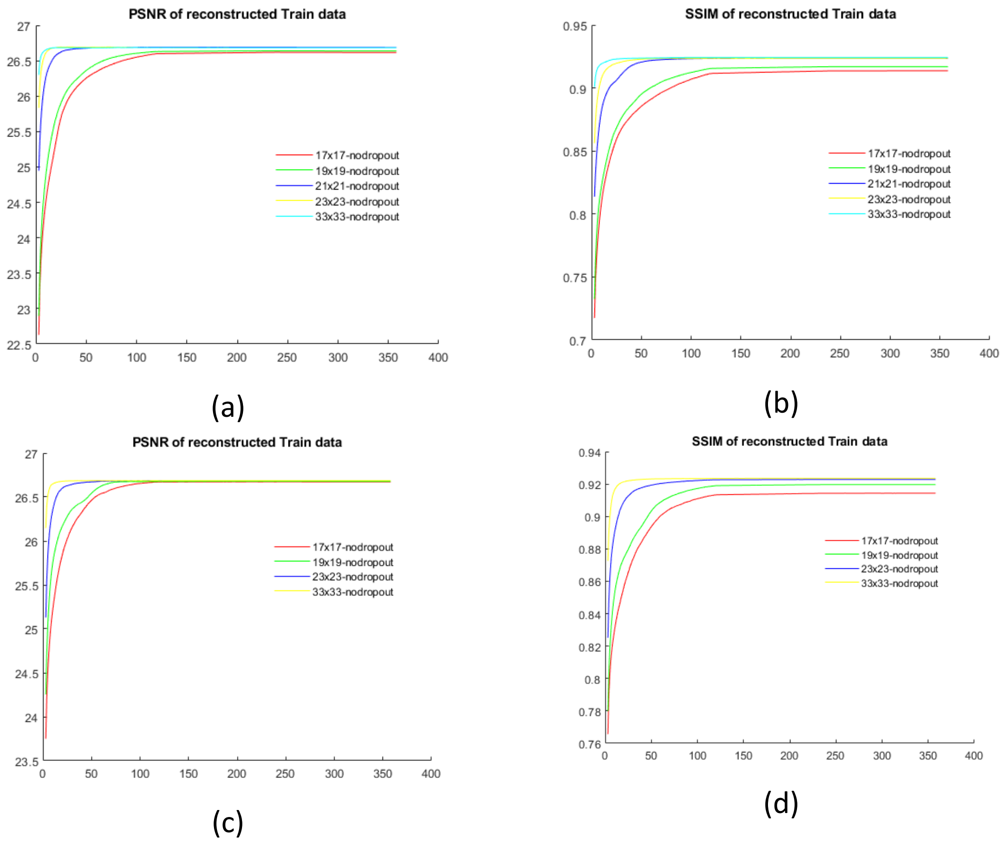
We can notice from Table 21 that the single channel version fails to perform better than the bicubic interpolation method for elevation set 2. This is in contrast to the dual channel method as will be seen in section 5.6.2.3.

5.6.2.3 Dual channel results

A second channel with the low-resolution optical data was added in order to assess how this would affect the reconstruction. Experiments using 32-16-1 & 64-32-1 feature maps were conducted with both the elevation set 1 and elevation set 2. In addition, kernel sizes of 7-1-5, 9-1-5 & 9-3-5 were examined. Finally, the results of the two different elevation sets (Bicubic & NN) used will be presented separately.

5.6.2.3.1 Elevation set one results

As a rule, the more the available data the more a neural network model avails of additional feature maps. Under this assumption, the results for the 9-1-5 case will be presented for the 32-16-1 and 64-32-1 cases, which will help decide whether the 64-32-1 should be examined further. From comparison of Figures 61(a, c) and 61(b, d) it is clear that the available training data were not enough to make use of the extra features maps. As a result, performance has deteriorated for the 64-32-1 case. Both renditions surpass the performance of the bicubic up-scaler for the PSNR metric but the 32-16-1 feature maps version is marginally better as can be from Table 22.



(a)

(b)

(c)

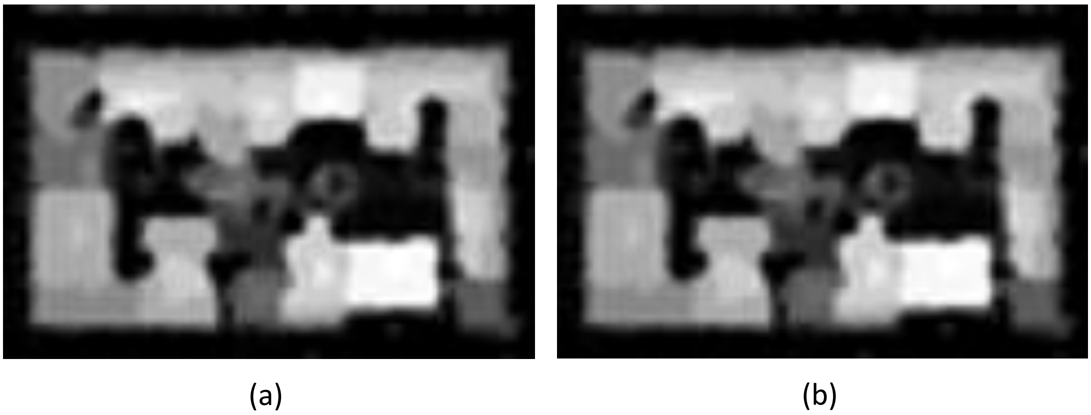
(d)

Figure 61 -Dual channel elevation set one PSNR & SSIM metrics for Train data. a) PSNR 32-16-1 b) SSIM 32-16-1 c) PSNR 64-32-1 d) SSIM 64-32-1

Table 22 – Dual channel Train 9-1-5 PSNR & SSIM. Comparison between 32-16-1 and 64-32-1 feature maps

Metric	Feature maps		Bicubic
	32-16-1	64-32-1	
PSNR	26,69095	26,68689	22,742064
SSIM	0,9242242	0,9235429	0,901115

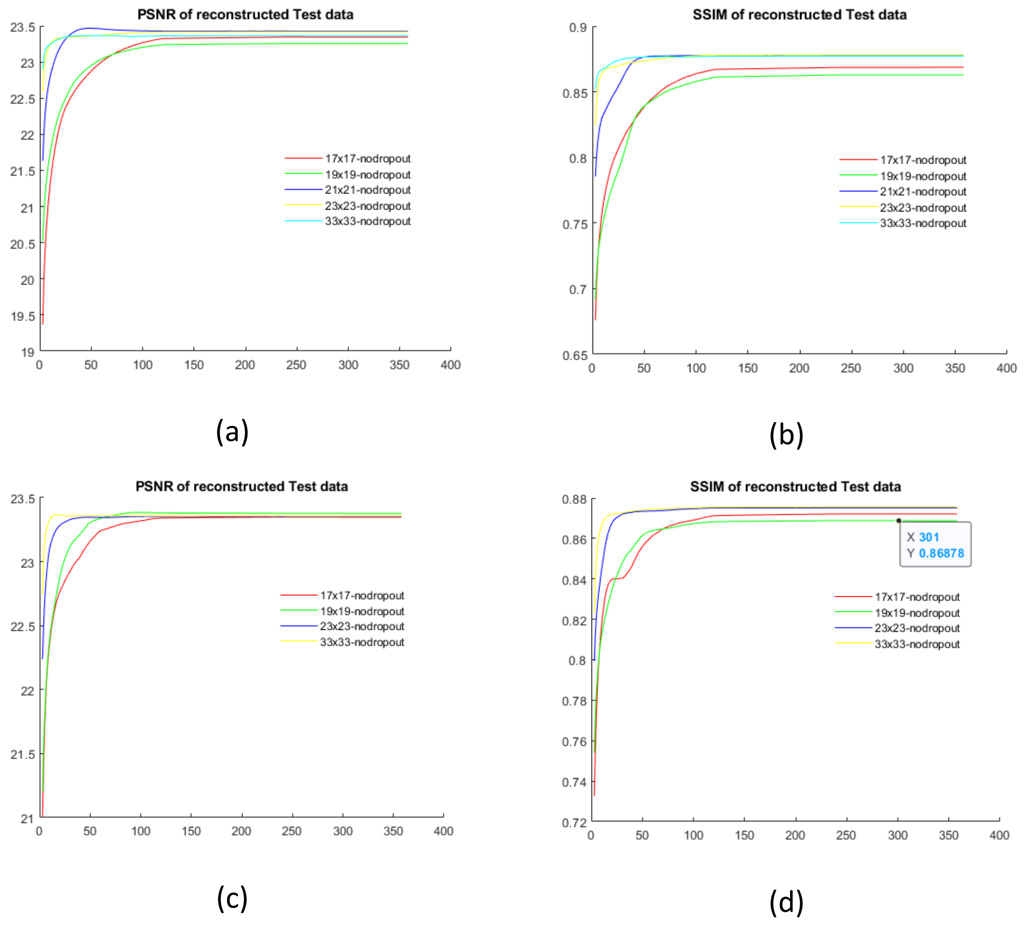
The reconstructions can be seen in Figure 62(a, b) for the 32-16-1 and 64-32-1 case, respectively.



(a) (b)

Figure 62 - Dual channel Train reconstruction. 32-16-1 & 64-32-1 comparison

Similarly, the performance of the two versions are compared for the validation set 2 in Figures 63(a-d). From Figures 63(a-d), Table 23 and the similar performance drop for the Train data, it can be safely assumed that performance steadily declines with larger feature maps for this data set. This is not to say that performance would not have been higher when using more feature maps had more training data been available. However, this is something that has to be verified with more training data.



(a)

(b)

(c)

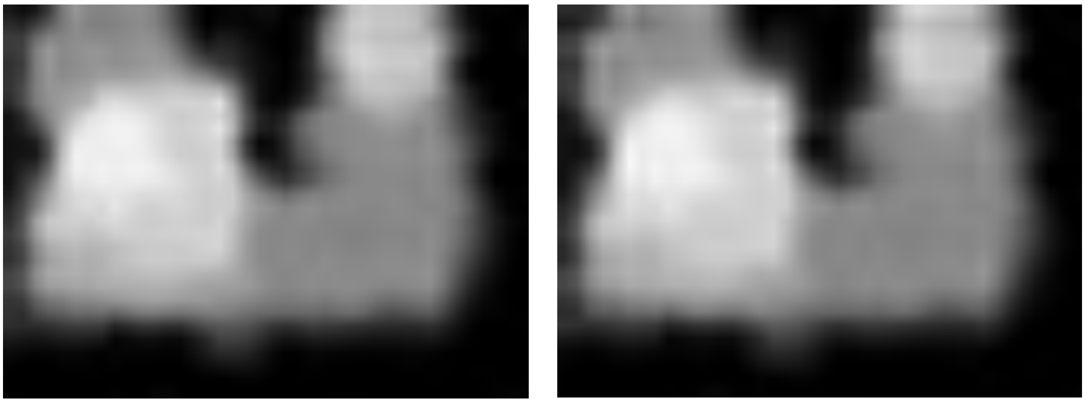
(d)

Figure 63 - Dual channel elevation set one 9-1-5 PSNR & SSIM metrics for validation set 2 data. a) PSNR 32-16-1 b) SSIM 32-16-1 c) PSNR 64-32-1 d) SSIM 64-32-1

Table 23 - Dual channel validation set 2 9-1-5 PSNR & SSIM. Comparison between 32-16-1 and 64-32-1 feature maps

Metric	Feature maps		Bicubic
	32-16-1	64-32-1	
PSNR	23,46689	23,38473	21,012714
SSIM	0,8781328	0,8756348	0,858634

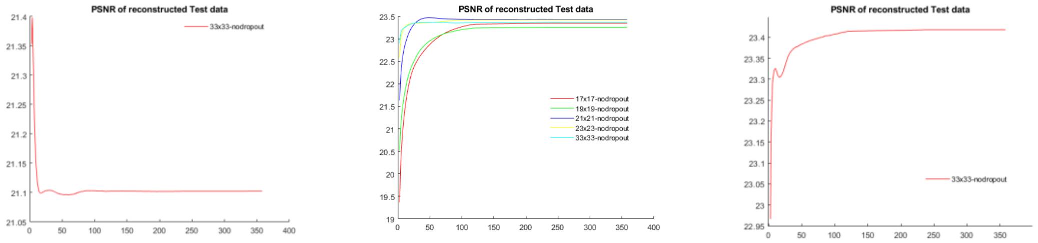
The reconstructions can be seen in Figure 64(a, b) for the 32-16-1 and 64-32-1 case, respectively. If we compare the PSNR value of the 64-32-1 network with the corresponding value of the single channel model, we can discern a slight performance increase for the dual channel version.



(a) (b)

Figure 64 - Dual channel validation set 2 reconstruction. 32-16-1 & 64-32-1 comparison

The convolution kernel size also had an impact on performance. As can be seen in Figure 65 and Table 24 the larger convolutional kernel 9-1-5 performed a lot better for the validation set 2 data in comparison to the 7-1-5 case. This pattern did not hold for larger kernels something that is attributed to the low-resolution of the available data and that elevation edges that correspond to optical edges may not fall within the convolution window.



(a) (b) (c)

Figure 65 - PSNR comparison of different kernel sizes 32-16-1 for validation set 2 data a) 7-1-5 b) 9-1-5 c) 9-3-5

Table 24 - PSNR & SSIM of validation set 2 data for different kernel sizes and 32-16-1 feature maps

Metric	Convolution kernel			Bicubic
	7-1-5	9-1-5	9-3-5	
PSNR	21,39834	23,46689	23,41802	21,012714
SSIM	0,8063351	0,8781328	0,8777893	0,858634

In order to assess real world performance, a comparison of the proposed model to that of Dong et al. (Dong, Loy, He, & Tang, 2015) was conducted. The model of Dong had been trained with two sets of data[22] and this comparison was done for the smaller dataset of the two datasets that used 91 images. In order to conduct a fair comparison, tests were run using the code provided by Dong et al. (Dong C. , 2019), so that all numbers including the PSNR of the bicubic method are given as was returned by the code of the original research[23]. More specifically, the Train, Validation and Test elevation images for elevation data set one were used. They were up scaled two times just as done for the proposed model. The results are presented in Table 25 while the reconstructions can be seen in Figures 66(a-c) for the Dong model; on Figures 66(d-f) for the proposed model; and Figures 66(g-i) display for the GT[24]. It can visually be seen from Figures 66(a-c) and 66(d-f) that the model of Dong returns more blurry reconstructions than the proposed model. Furthermore, the PSNR returned for all Dong reconstructions is clearly less than the bicubic method. This finding strengthens the argument that general-purpose super-reconstruction models trained with generic data are insufficient to process images of elevation data.

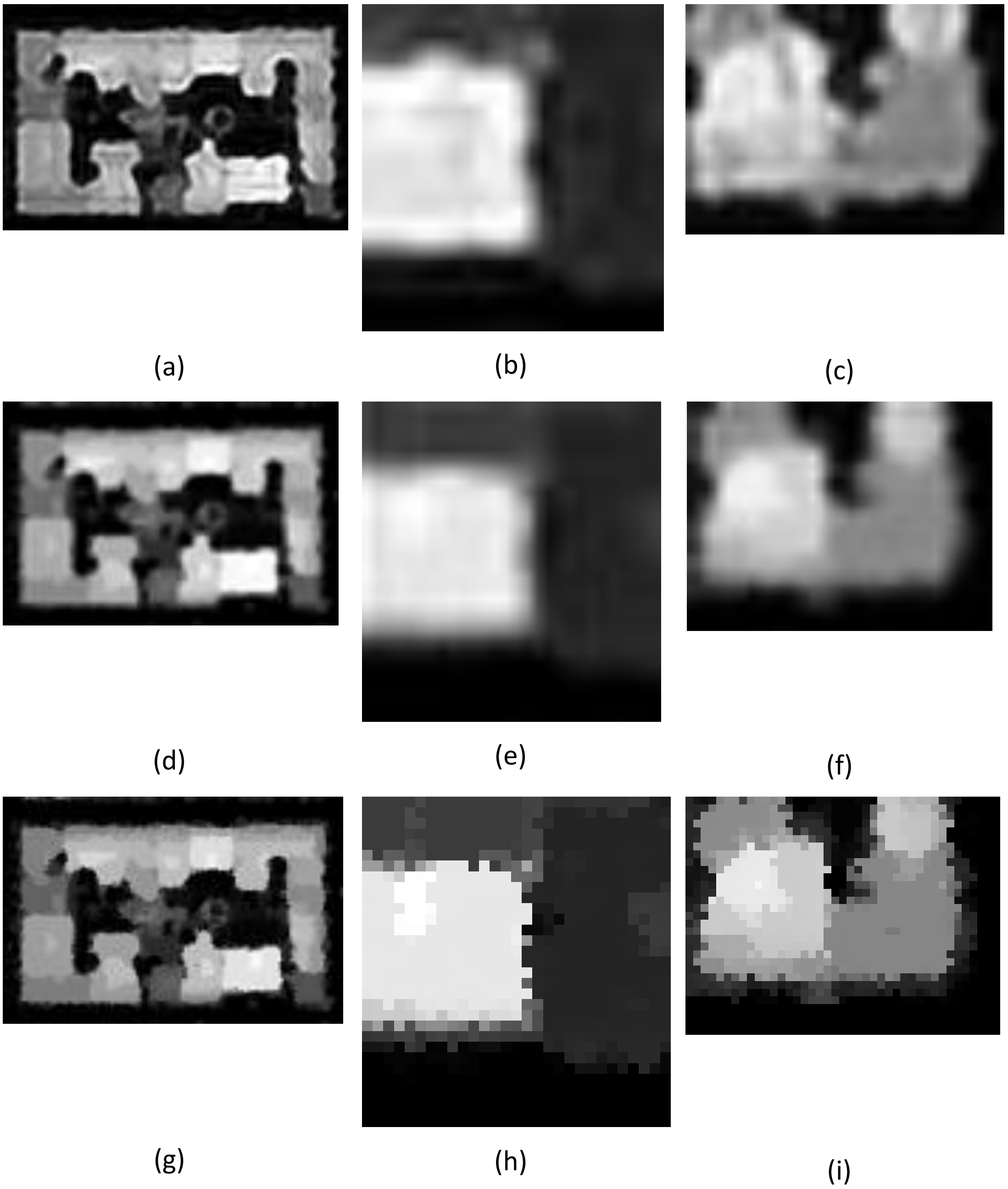
Table 25 - PSNR for Train, Validation sets 1 and 2 data returned by Dong's model

	PSNR of Dong Model	PSNR of bicubic method
Train	23,775215	26.926449
Validation set 1	20,381643	23.009273
Validation set 2	22,116246	22.796978

[22] The first was trained with a rather small set of 91 general nature & people images while the second with a much larger ImageNet data set.

[23] A discrepancy between the PSNR given for the proposed method and that given by the code of Dong is due to the normalization to the 0..1 range that was conducted in the case of the proposed method. Dong et al. worked directly with the 0..255 range gray levels.

[24] The proposed method is based on 9-1-5 32-16-1 21x21 dual-channel configuration, which was found to offer the best dual-channel performance.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 66 - SR Reconstructions of Train, Validation sets 1 and 2 data. a) Dong Train b) Dong Validation set 1 c) Dong validation set 2 d) proposed method Train e) proposed method validation set 1 f) proposed method validation set 2 g) GT Train h) GT Validation set 1 i) GT validation set 2

5.6.2.3.2 Elevation set two results

This variation of the training was considered as an attempt to assess the performance of the model under more adverse conditions. The elevation data have in this case been downscaled and up-scaled two times with the nearest neighbor method, which introduces much more noise and produces jaggy images (Figure 49). In this regard, the effect of the convolution kernel size on performance for the Train data can be seen in Figure 67, where the 9-1-5 kernel seems to offer slightly better performance than 7-1-5 rendition, as was the case for elevation set one. Furthermore, it can be seen in Table 26 that both the maximum PSNR and SSIM are higher for the 9-1-5 kernel. Lastly, the reconstructions can be seen in Figure 68 for the 7-1-5 and 9-1-5 case, respectively.

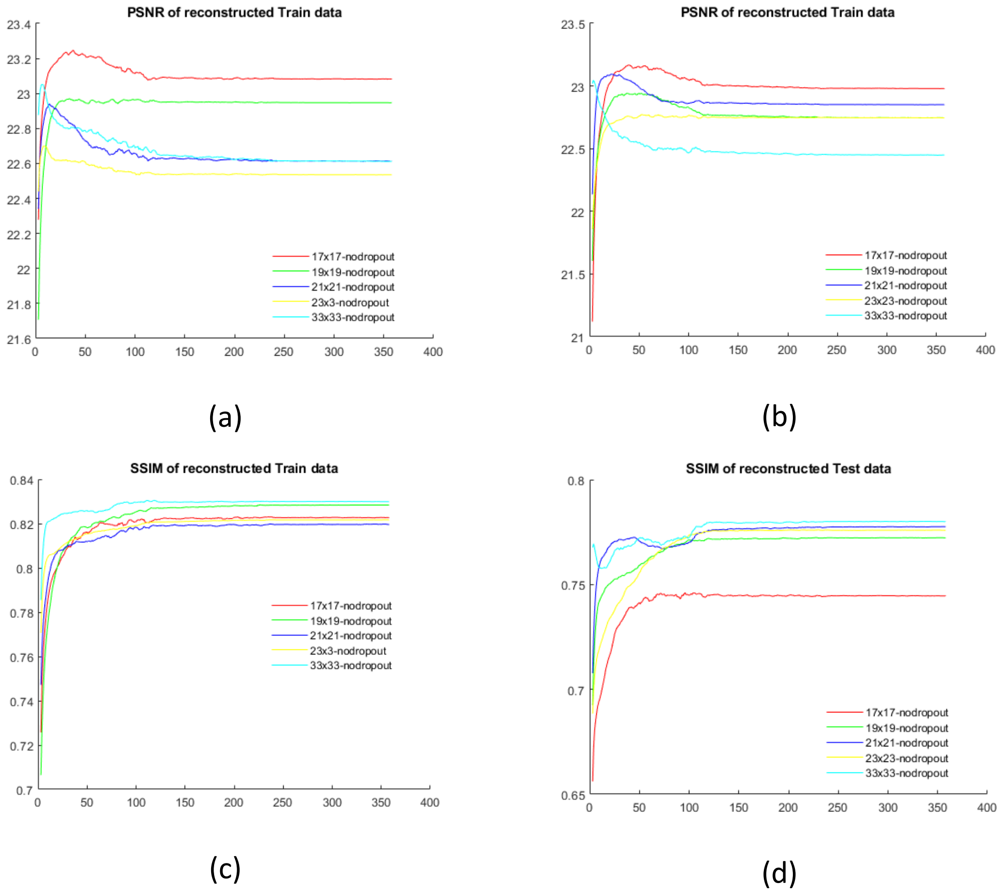
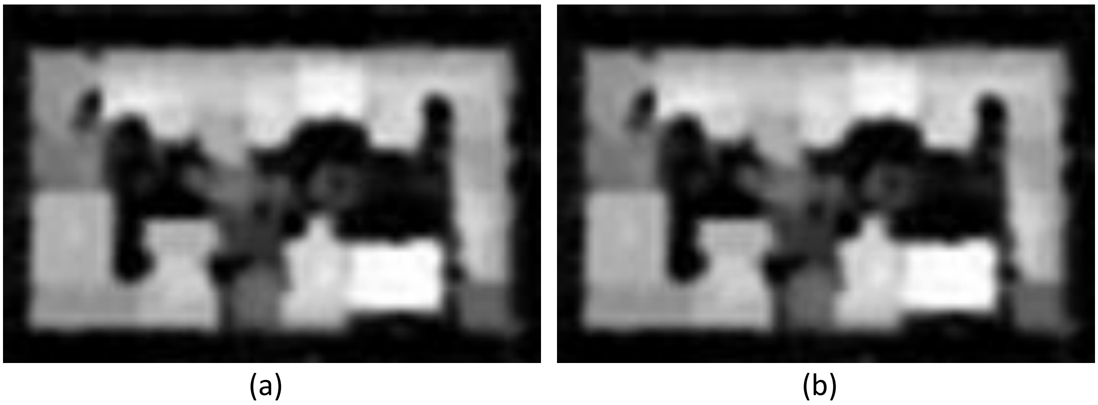


Figure 67 - Dual channel elevation set two PSNR & SSIM metrics for Train data. a) PSNR 7-1-5 b) PSNR 9-1-5 c) SSIM 7-1-5 d) SSIM 9-1-5

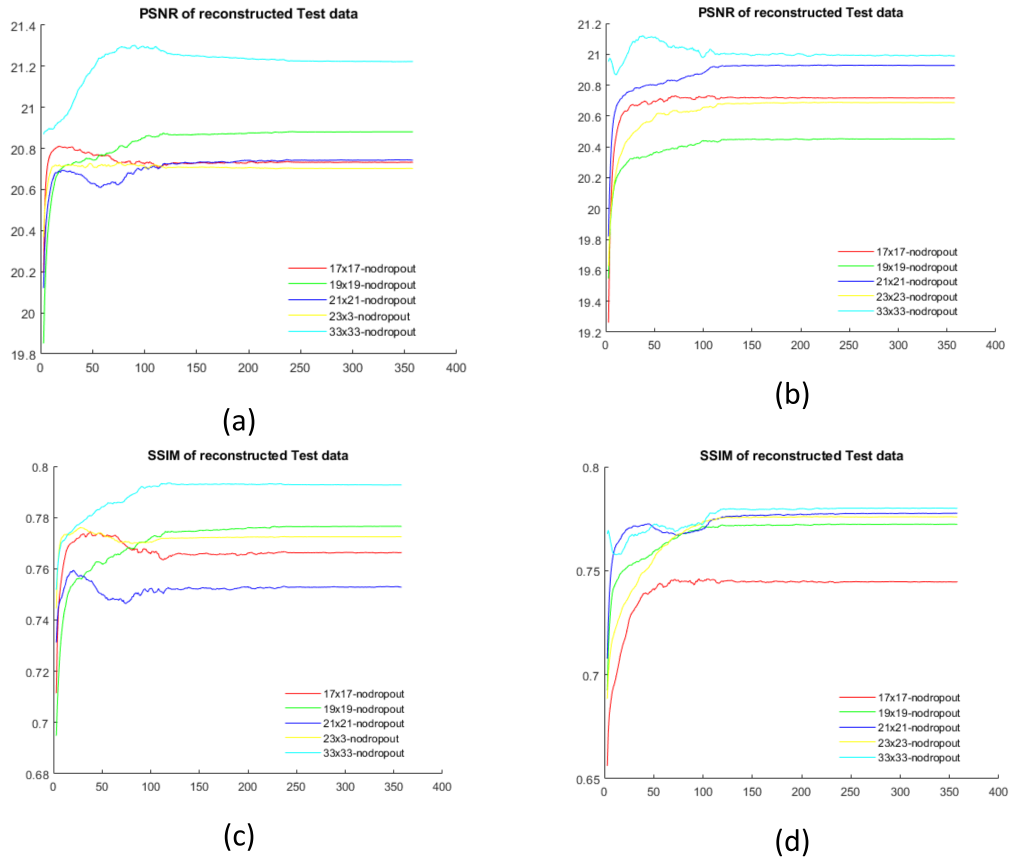
Table 26 - Dual channel 32-16-1 PSNR & SSIM Train Reconstruction. Comparison between 7-1-5 and 9-1-5 convolution kernels.

Metric	Convolution kernel		Bicubic
	7-1-5	9-1-5	
PSNR	23,05184	23,16803	22,742064
SSIM	0,8304677	0,8323113	0,901115



(a) (b)

Figure 68 - Dual channel Train reconstruction for elevation set two. 7-1-5 & 9-1-5 comparison



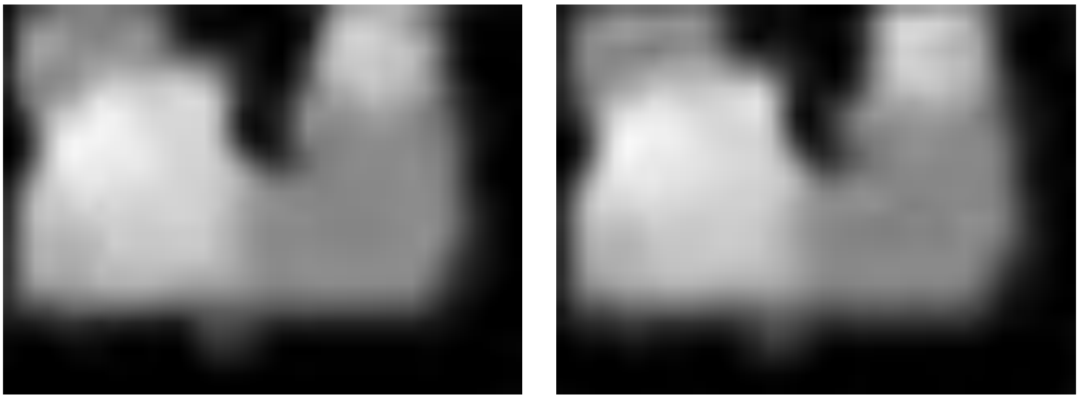
(a) (b) (c) (d)

Figure 69 - Dual channel elevation set two PSNR & SSIM metrics for validation set 2 data. a) PSNR 7-1-5 b) PSNR 9-1-5 c) SSIM 7-1-5 d) SSIM 9-1-5

Table 27 Dual channel 32-16-1 PSNR & SSIM validation set 2 reconstruction. Comparison between 7-1-5 and 9-1-5 convolution kernels.

Metric	Convolution kernel		Bicubic
	7-1-5	9-1-5	
PSNR	21,30144	21,11909	21,012714

SSIM	0,7934909	0,7801557	0,858634



(a) (b)

Figure 70- Dual channel validation set 2 reconstruction for elevation set two. 7-1-5 & 9-1-5 comparison

According to Table 27, the 9-1-5 version seems to, unexpectedly, offer slightly worse performance for both metrics when reconstructing on validation set 2, which is attributed to jaggy high-frequency content of the nearest neighbor upscaling. More importantly though, the dual-channel version of the network outperforms the corresponding best case for the single-channel version by approximately 0.2db (9-1-5 case). The reconstructions can be seen in Figure 70 (a-b) and the performance curves in Figure 69.

5.6.2.3 Single vs dual channel comparison

The research presented in this chapter was based on the assumption that the dual channel version would offer benefits to the performance of the network. This was not always the case, as can be seen from Table 28. More specifically, the single channel 7-1-5 convolution kernel size scored marginally lower as compared to the single-channel equivalent when reconstructing on the validation set 2 data and trained on the elevation set 1 data set. This can be seen in Figures 71(a-d) and Table 28.

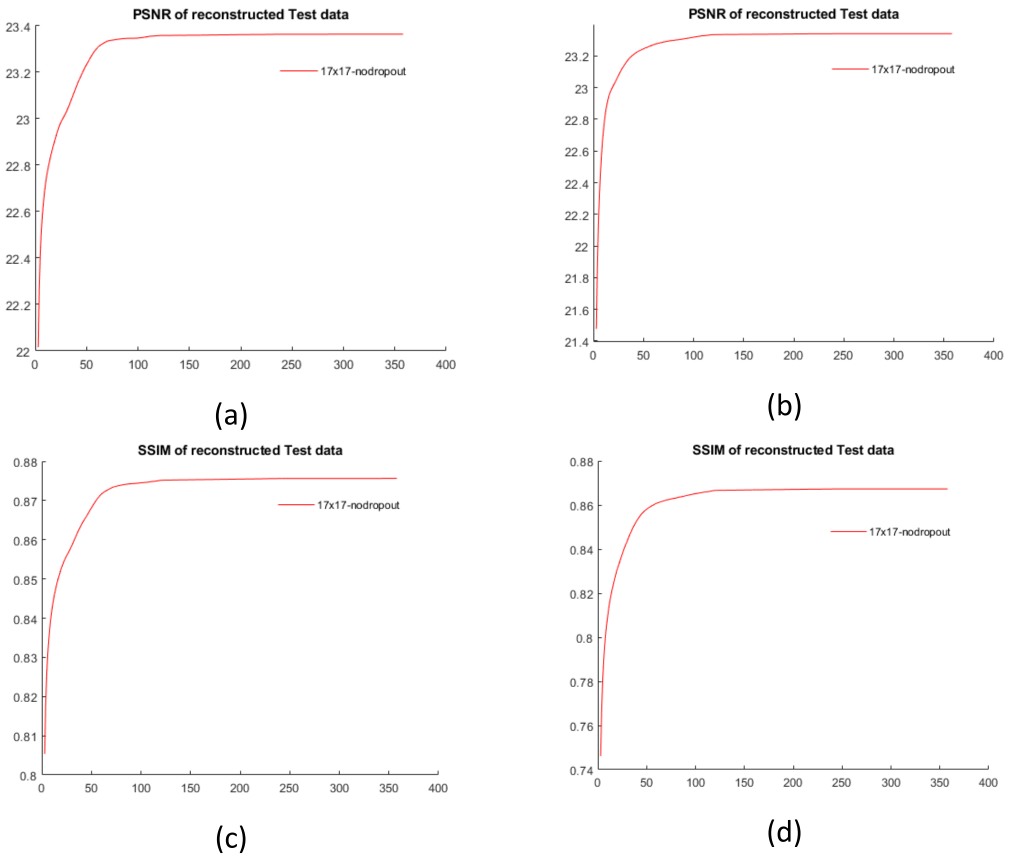


Figure 71 - PSNR and SSIM 64-32-1 for single and dual channel validation set 2 7-1-5 model. a) PSNR single channel b) PSNR dual channel c) SSIM single-channel d) SSIM dual-channel

Table 28 - PSNR and SSIM 7-1-5 single and dual channel validation 2 comparison

Metric	**7-1-5 Convolution kernel**		**Bicubic**
	Single channel	Dual channel[25]	
PSNR	23,36433	23,340	21,012714
SSIM	0,8756447	0,86729	0,858634

Further comparisons are shown for the 9-1-5 case. This time, as shown in Table 29, the dual channel scores better than the single channel version. The problem tackled in this chapter of my research is very difficult since the resolution of the elevation data were five times lower than the optical version and because of the rather limited training set. It surmised that more available or by reducing the resolution discrepancy between the elevation and optical data would help improve performance. Finally, a similar improvement of performance for the 9-1-5 dual-channel case trained on elevation set 2 over the equivalent 9-1-5 single channel performance was also observed, as can be attested by comparing Table 21 (single-channel) and Table 27 (dual-channel).

[25] Elevation set one.

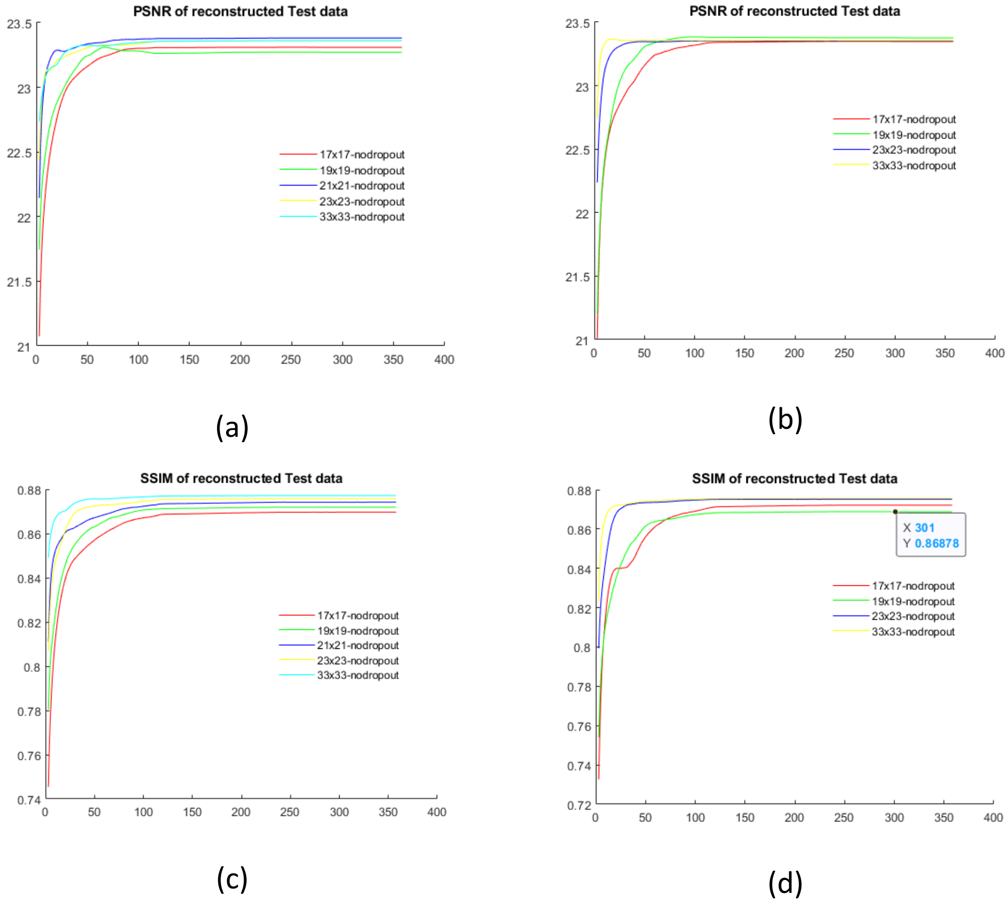


Figure 72 - PSNR and SSIM 64-32-1 for single and dual channel 9-1-5 validation set 2 model. a) PSNR single channel b) PSNR dual channel c) SSIM single-channel d) SSIM dual-channel

Table 29 - PSNR and SSIM 9-1-5 single and dual channel comparison (validation set 2)

Metric	9-1-5 Convolution kernel		Bicubic
	Single channel	Dual channel[26]	
PSNR	23,38185	23,46689	21,012714
SSIM	0,8771589	0,8781328	0,858634

[26] Elevation set one.

PART II – ITERATIVE RELAXATION SYSTEM

6. Relaxation System

The idea behind the relaxation system stems from image processing applications for edge detection of the early 1980's but applied in a more general context. Owing to the increased computing power available, a multitude of spatial information can come into play increasing the accuracy of the algorithm. The logic behind a relaxation system is to allow the smooth formation of an edge chain by examining the local context of an edge *e*[27]. If the local context contains evidence that the under investigation edge fits in with neighboring edges it has a confidence value augmented. Otherwise, its confidence value is decremented. The ultimate goal of such a system is to reach a point where all edges are positively labeled as belonging to an edge or not.

6.1 Historical origins

Many propositions for edge detection were made since Marr & Hildreth formulated a concrete theory of edge detection in 1980 (Marr D. & Hildreth, 1980). The most successful and influential proved to be the proposition of Canny (Canny, 1986), a method so efficient that variations of it are widely used even today. The significant contribution of Canny was the inception of non-maximal suppression of directional edge data, which discarded spurious low-intensity edges along the direction of the magnitude of an edge[28]. This was followed by a hysteresis stage that would classify the remaining edges as strong and weak, respectively. Weak edges that were not in the neighborhood of strong edges were discarded while weak edges near a strong edge were converted into strong edges themselves. The result was a crisp edge image free of most spurious edges.

Edge detectors are a vital component of many edge segmentation algorithms. The first step in such algorithms is to find the edges by applying edge detecting operators that detect discontinuities in the graylevel, color or texture of an image (Sonka M., Hlavac, & Boyle, 2014). Supplementary processing is then performed to combine the found edges into continuous chains with the continuity usually defined through a 4-way or 8-way connectivity operator. A problem regarding edge detectors is that borders still have noise with important edges missing. It is for this reason that edge properties in the context of the under inspection edge can yield important information for making the case whether it should be included in an edge chain. This procedure can be performed iteratively, with each successive iteration augmenting or decrementing the confidence of whether an edge belongs to an edge chain or not. For example, a weak edge placed between two strong edges can provide an indication that it should be included in an edge chain (Sonka M., Hlavac, & Boyle, 2014). This procedure has come to be known as *relaxation*.

The method proposed in this thesis, is a variation of Prager's method (Prager J. , 1980), which is based on the so-called crack-edges (virtual edges between the pixels of an images). Prager's initial proposal was to consider all crack edges emanating from

[27] An edge in this context is the pixel of a given image that is being examined if it belongs to a transition chain segmenting two regions.
[28] Also known as the orientation of an edge.

an edge as show in Figure 73 [taken from (Sonka M., Hlavac, & Boyle, 2014)]. Prager considered horizontal and vertical edge chains so he developed rules for finding continuity between the central edge *e* and all parallel horizontal edges[29] or all possible parallel vertical edges[30]. He then created categories according to the number of crack edges that had strong neighboring edges.

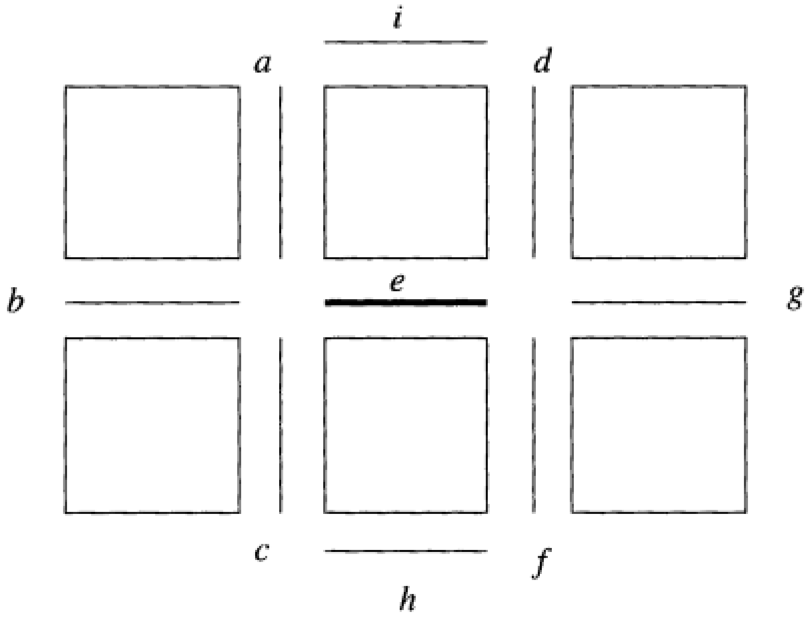


Figure 73 - Crack edges of central edge e

Edges pairings$(\mathbf{x}, \mathbf{y})$, were then created which were called the vertex-type. In this schema, x is the number of left strong neighboring crack edges and y is the number of right strong neighboring edges. Prager also defined various vertex-type categories and according to this categorization, the edge *e* confidence could either incremented or decremented. The categories Prager defined were:

- 0-0: isolated edge – negative influence on edge confidence.
- 0-1: uncertain – weak positive or no influence on edge confidence.
- 0-2. 0-3: dead end – negative influence on edge confidence.
- 1-1: continuation – strong positive influence on edge confidence.
- 1-2, 1-3: continuation to border intersection – medium positive influence on edge confidence.

[29] Horizontal case.

[30] Vertical case.

- 2-2, 2-3, 3-3: bridge between borders – not necessary for border creation, no influence on edge confidence.

The patterns of connectivity for the horizontal case can be seen in Figure 74 (from [Sonka M., Hlavac, Boyle, 2014]). A pairing (x,y) of crack edges is called the vertex-type.

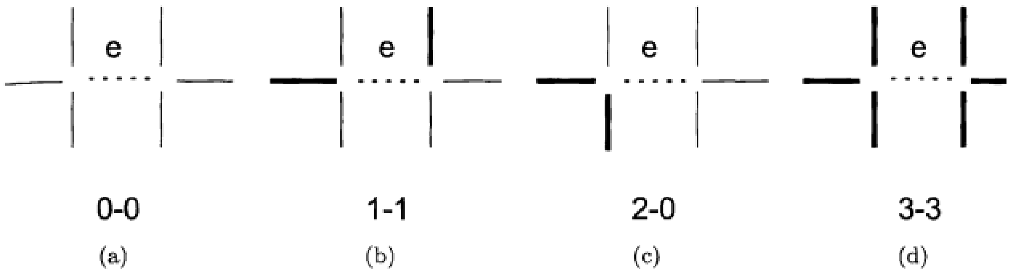


Figure 74 - Some typical crack edges connectivity patterns

To compute the vertex type choose the maximum confidence vertex conf(j) , i.e., the vertex is type j where j maximizes conf(j), as shown below:

Table 30 - Confidence of vertex type calculation

	$\boldsymbol{conf}(\mathbf{0}) = (\boldsymbol{m} - \boldsymbol{a})(\boldsymbol{m} - \boldsymbol{b})(\boldsymbol{m} - \boldsymbol{c})$	(7)
	$\boldsymbol{conf}(\mathbf{1}) = \boldsymbol{a}(\boldsymbol{m} - \boldsymbol{b})(\boldsymbol{m} - \boldsymbol{c})$	(8)
	$\boldsymbol{conf}(\mathbf{2}) = \boldsymbol{ab}$(m-c)	(9)
	$\boldsymbol{conf}(\mathbf{3}) = \boldsymbol{abc}$	(10)
	$\boldsymbol{m} = \mathbf{max}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{q})$, where a,b,c: normalized gradient for the three edges and q a constant (0.1 or something close).	(11)

Parameter m adjusts the vertex classification so that it is relative to the local maximum. For example, (a,b,c) = (0.25, 0.01, 0.01) is a type 1 vertex[31]. The parameter q forces weak vertices to type zero. After the determination of the left and right vertex-type, the edge-type is simply the concatenation of the left and right vertex-type. Finally, the edge confidence in each iteration is modified according to the following equations:

[31] If you do the math according to equations 7-11, conf(1) is the maximum value so the edge is classified as type 1.

Table 31 - Modification of confidence

Increment	$c^{k+1}(e) = min(1, c^k(e) + \delta)$	**(12)**
Decrement	$c^{k+1}(e) = max(0, c^k(e) - \delta)$	(13)
Leave as is	$c^{k+1}(e) = c^k(e)$	(14)

This iterative process smoothly relaxes the confidences of the edges so that they fit in with strongly aligned edges or atrophy.

6.2 Data sources and pre-processing

Multimodal data sources have been collected from a dense urban neighborhood of Athens, Greece. Since the development of many of the algorithms in this thesis was conducted under the Mathworks Matlab platform, part of the necessary pre-processing pertains to the transformation of these primary data sources to the appropriate Matlab formats. Specifically, the following tools were used: a) LAS Tools, b) Quantum GIS, c) Monteverdi, d) Global Mapper, e) TNT Mips. This thesis used the product of the pre-processing as conducted in the research project of the Technological Institute of Athens, Archimedes III in 2013 by Vassilas et al. (Vassilas N., Charou, Petsa, & Grammatikopoulos, 2013).

6.2.1 Data sources

1. LiDAR data from a neighborhood of Kallithea regarding a region 2km x 4km which were taken and delivered in 2003 by GoeIntelligence1. The initial data have been re-sampled through interpolation so that they have a spatial resolution (sampling step) of 1m. This re-sampling resulted in a LiDAR picture of 2000x4000 pixels from which a specific region of 1827x1793 was cropped and used for further processing. The height resolution of the initial LiDAR data was 20cm whilst that of the delivered data (after the re-sampling) was 1cm. These data form the digital terrain of the urban region (Digital Surface Map – DSM)[32].

2. A digital elevation map for the same area was also handed in by Geointelligence. The DEM was calculated through interpolation from the DSM and has a spatial resolution of 2m. An area of resolution 1827x1793 was cropped from the DEM and used in this thesis.

3. Normalized DSM (nDSM): This has been calculated by subtracting the DEM from the DSM data, that is nDSM = DSM – DEM. The nDSM contains the real height of the buildings from the ground up and can be used in the 3D reconstruction of the area.

[32] All cited data sources of this section come from the research of Vassilas et al. (Vassilas N., Charou, Petsa, & Grammatikopoulos, 2013) and for the sake of brevity are not shown.

Normalization was performed in Matlab. Possible negative values were replaced by a height of zero. Such negative values are errors, which are the product of the interpolation technique used to calculate the DEM. The delivered nDSM data came in two formats. First, is a grayscale image with a range of values 0..255 and secondly a Matlab m file which contains the absolute real value of the height.

4. Colored high-resolution RGB aerial-photograph from the National Cadastre & Mapping Agency S.A., which depicts the specific area of interest in Kalithea, Greece.

5. Multispectral Google Earth satellite image dated from 2003. This image was used in order to extract ground truth maps of the building boundaries in the region. It was also used to interpret possible differences between the optical (aerial) photograph given by the National Cadastre & Mapping Agency S.A and the Digital Elevation Map given by GeoIntelligence in 2007.

6. Three optical channels of the ICONOS orthogonal projected satellite image with zero cloud overlay and a spatial resolution of 1m (pan-sharpened). This image was granted for the research needs of the Archimedes III program by the Computational Intelligence Laboratory.

7. 12 bird's eye views of the region of interest. These data will assist in the qualitative assessment of the results and in the extraction of ground truth maps, which will facilitate the quantitative results of the experiments.

6.2.2 Implicit data sources

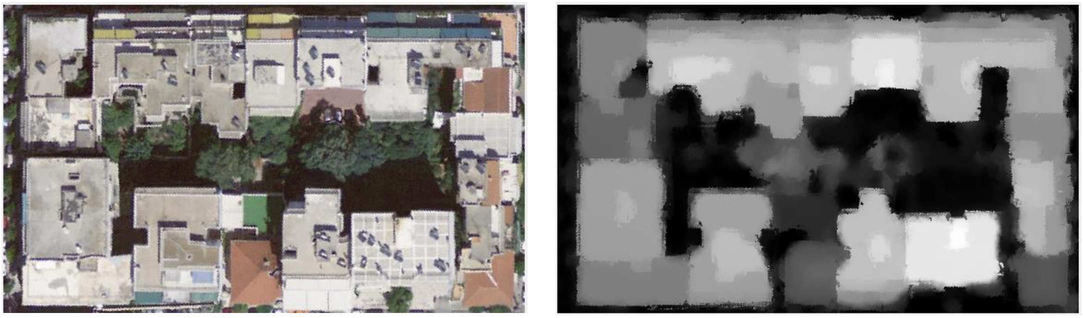
As part of the Archimedes III research program, masks were created that classify the pixels of the optical data according to various attributes of interest. These masks are binary images where a logical 0 denotes the existence of the attribute of interest and a logical 1 the absence hereof. Tree, grass and shadow masks were created by training neural networks to identify these structures.

6.2.3 Selection and pre-processing of building block

The building block selected has the facades of the buildings oriented across two dominant directions, which were found to be at -41º and 49º. In order to simplify calculations without losing the general features of the region the following pre-processing steps were applied:

- Rotation according to the dominant direction of -41º.

- Extraction of a rectangular region that corresponds to a building block from the initial data. Data were extracted for the optical and DEM channel during this process and are shown in Figure 75(a-b). The rotated and cropped green and tree masks were similarly extracted. These are shown in Figure 76 (a-b). Finally, the shadow mask was extracted in a similar manner and is shown in Figure 77.



(a) (b)

Figure 75 - rotated & cropped a) Optical data b) DEM data

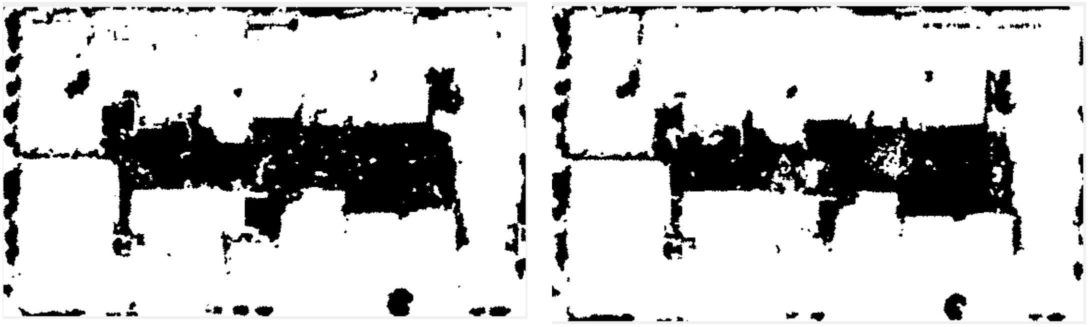


Figure 76 - rotated & cropped a) Grass mask b) Tree mask

Figure 77 - rotated & cropped shadow mask

6.2.4 Further pre-processing

Various necessary pre-processing is conducted in the sub-section.

- Creation of foliage mask: This mask is the logical conjunction of the grass and tree masks after a closure has been applied to it and thereafter some further dilation. It is show in Figure 78.

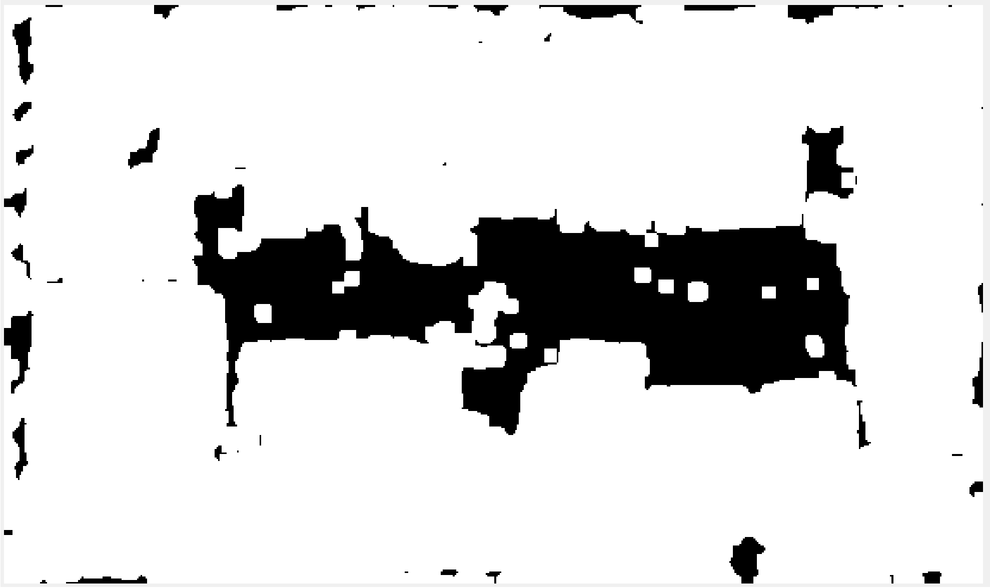


Figure 78 - Foliage mask

- Gray-scale version of optical data.

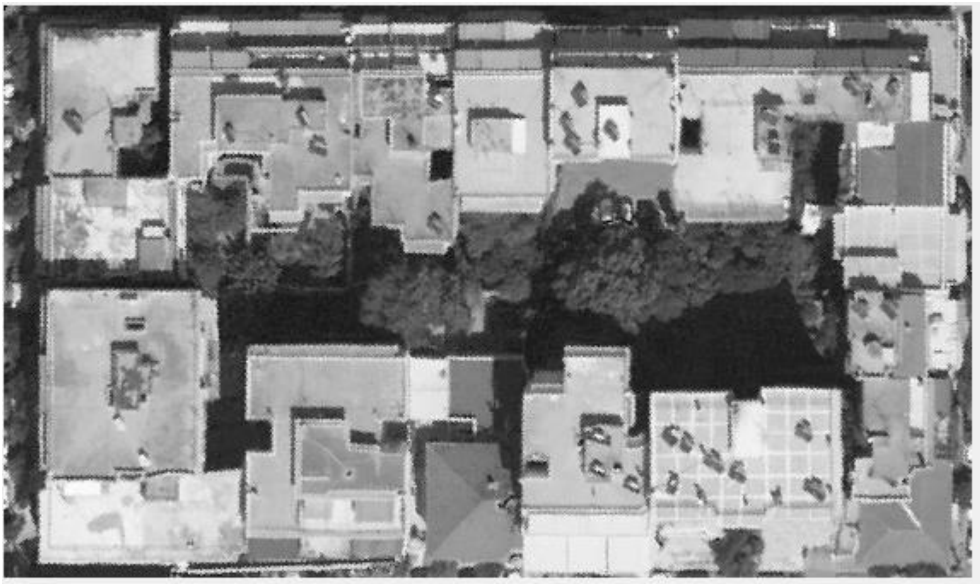


Figure 79 - Gray scale version of optical data

- Optical data with foliage masked out: The foliage is masked out according to the composite foliage mask.

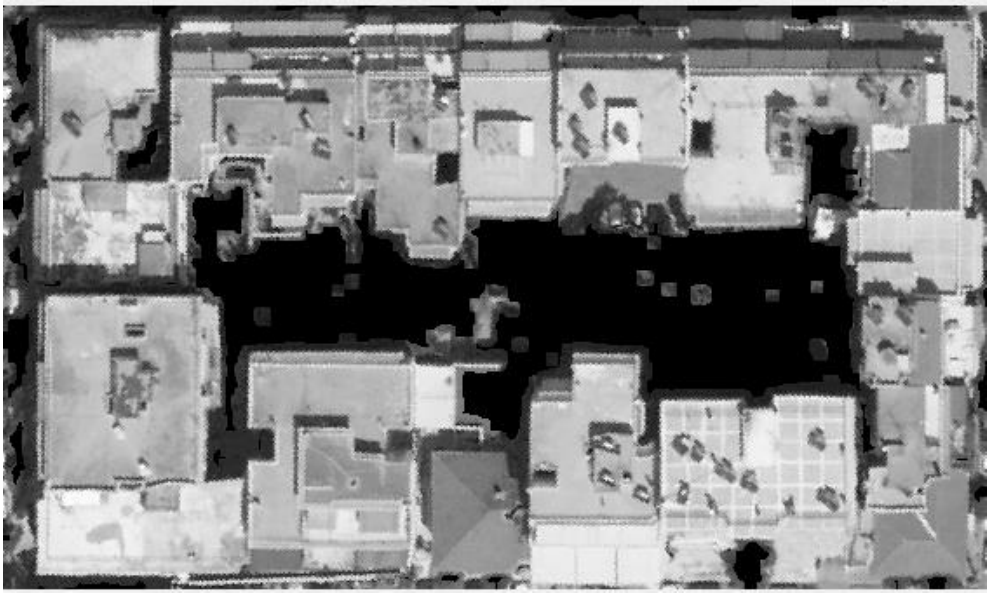


Figure 80 - Optical data with foliage masked out

- DEM data with foliage masked out: The foliage is masked out according to the composite foliage mask.

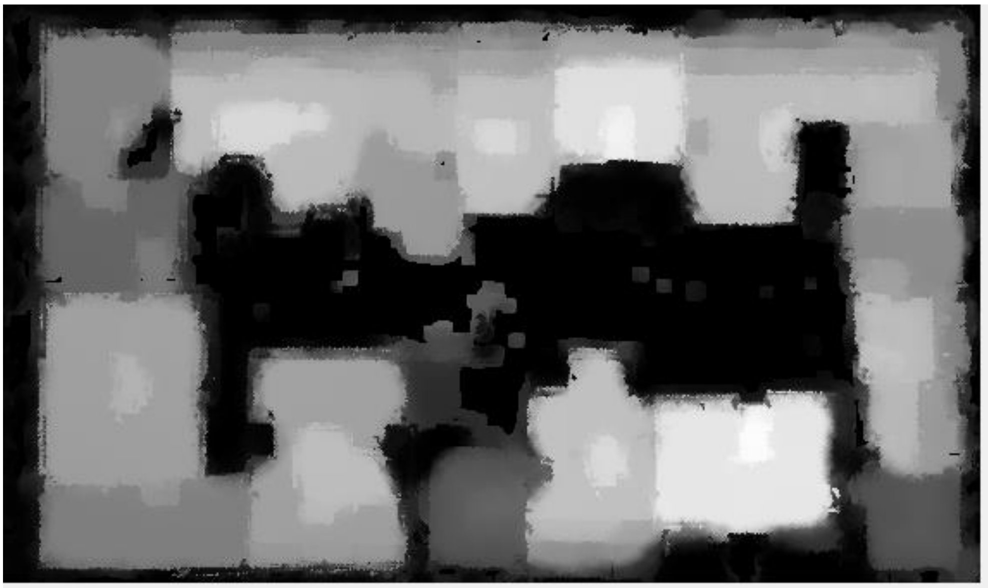


Figure 81 - DEM data with foliage masked out

- Height cohesive regions map: A height cohesive region map developed by (Vassilas N., Charou, Petsa, & Grammatikopoulos, 2013) was also used as input to the relaxation system. The algorithm utilizes 8-way connectivity to scan the image from the top-left to the bottom-right searching for neighbors along the three pixels of the previous scanline as well as along the pixel to the left[33]. The map can be seen in Figure 82.

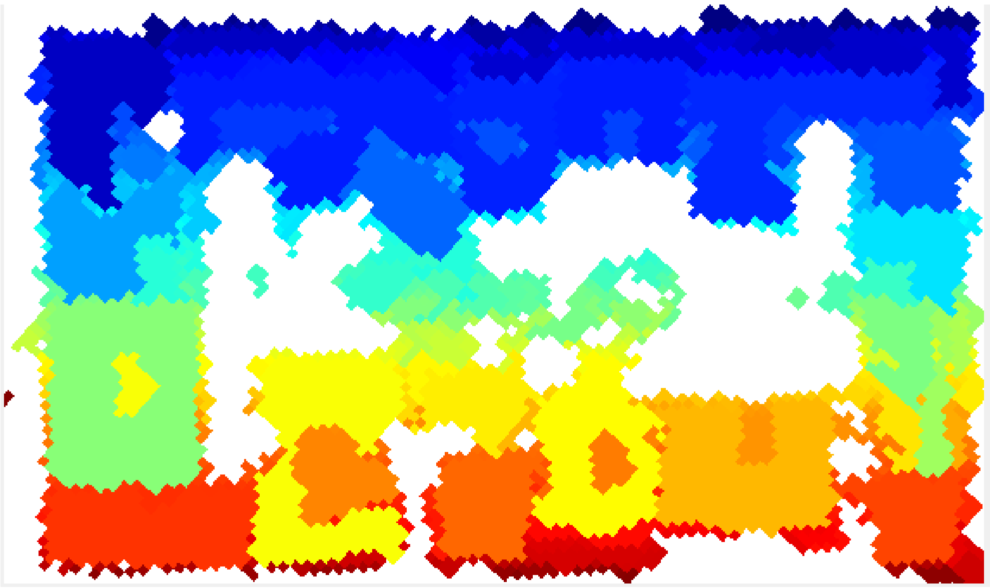


Figure 82 - Colored height cohesive region map

- A grayscale version of the cohesive regions image: This image was used in order to detect the edges of the cohesive regions. It can be seen in Figure 83.

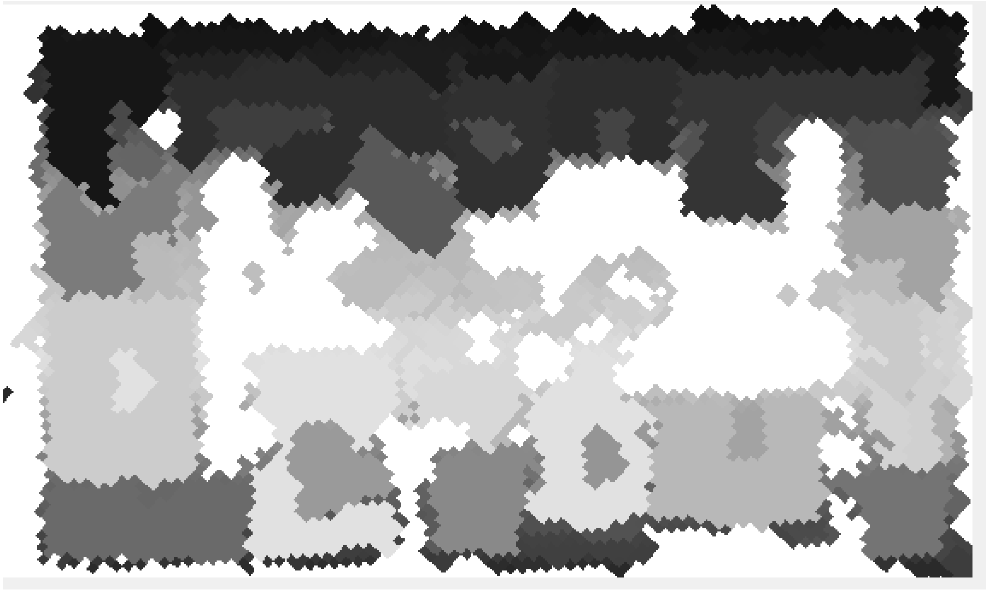


Figure 83 - grayscale height cohesive region map

[33] The algorithm is a generalization of the binary cohesive region generation algorithm. See the full paper of (Vassilas N., Charou, Petsa, & Grammatikopoulos, 2013) for more details.

- Smoothing of the grayscale version of the cohesive region image, which is standard procedure before applying any edge detection technique[34]. A Gaussian kernel with a unit standard deviation was used.

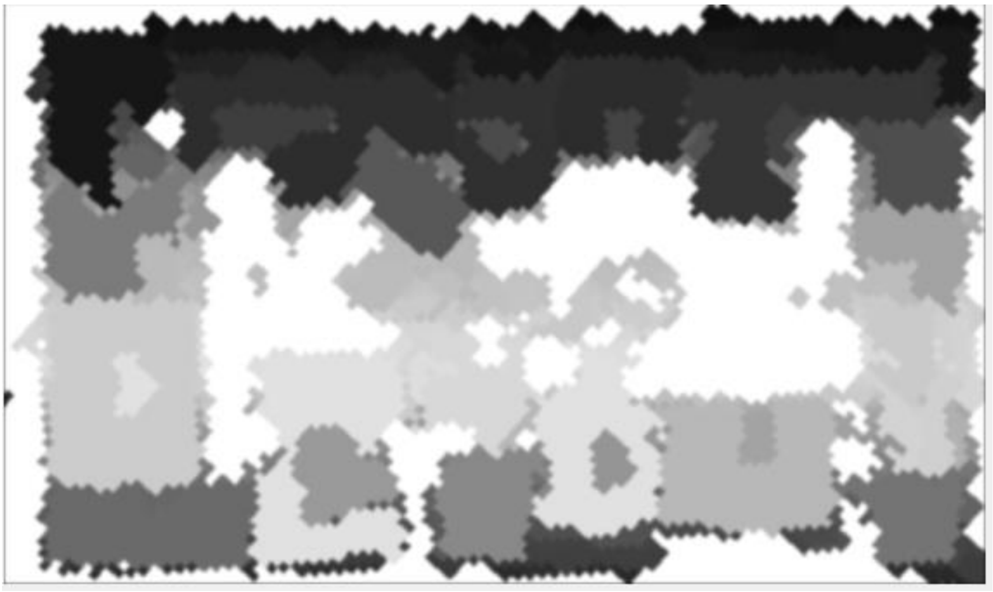
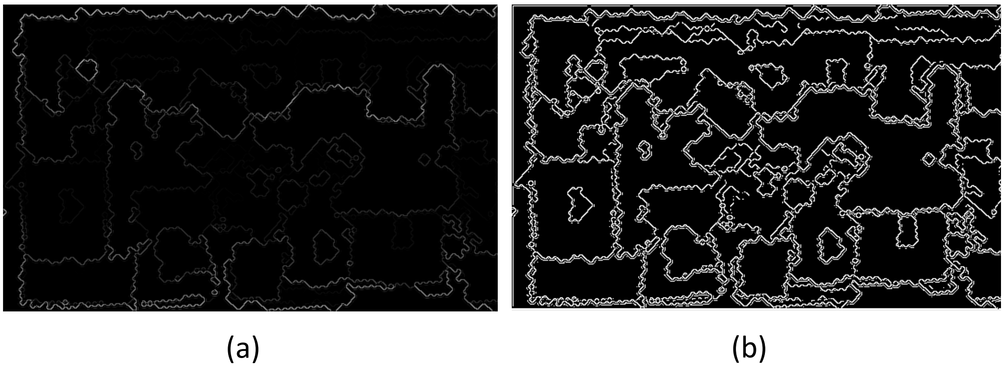


Figure 84 - smoothed height cohesive regions

- Edge detection of cohesive regions: The edges of the cohesive regions are found using the Canny edge detection algorithm. A proprietary version of the Canny method was written for this purpose. This variation of the Canny algorithm performs maximum suppression and hysteresis along the diagonals as well as along the horizontal & vertical directions. The magnitude and edges are shown in Figure 85(a-b).



(a) (b)

Figure 85 – a) Magnitude of cohesive regions b) Edges of cohesive regions

- Smoothing of optical & DEM data with a Gaussian filter (μ=0 & std=1).

[34] The edges of the height cohesive regions map are used in the Relaxation process.

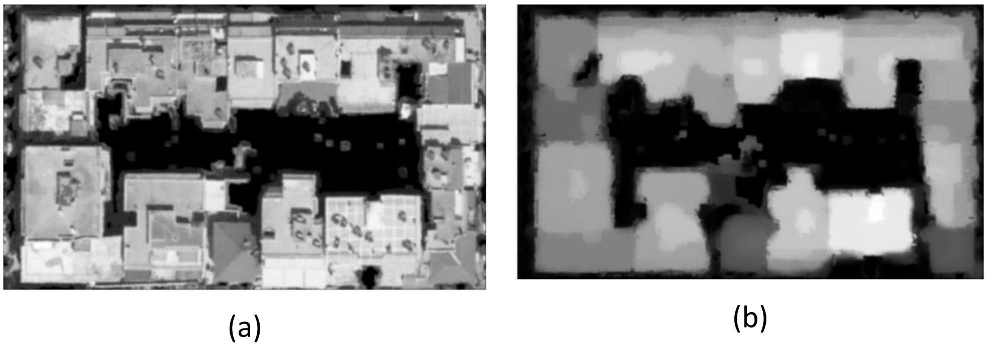


Figure 86 - Gaussian smoothed (μ=0, std=1) a) Optical data b) DEM data

- Edge detection of optical and DEM data: A proprietary implementation of the Canny edge detection is utilized in order to find the edges. The three stages of the Canny detection method as modified for the purpose of the thesis are described below.
    - Magnitude of optical & DEM data: The standard Sobel masks

$$\boldsymbol{G_x} = \begin{matrix} \mathbf{-1} & \mathbf{0} & \mathbf{1} \\ \mathbf{2} & \mathbf{0} & \mathbf{2} \\ \mathbf{-1} & \mathbf{0} & \mathbf{1} \end{matrix} \tag{15}$$

$$\boldsymbol{G_y} = \begin{matrix} \mathbf{1} & \mathbf{2} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{-1} & \mathbf{-2} & \mathbf{-1} \end{matrix} \tag{16}$$

are utilized in order to calculate the magnitude and orientation of the optical and DEM data and the gradient magnitudes are then calculated as the Euclidean distance measure according to the equation

$$|\boldsymbol{G}| = \sqrt{\boldsymbol{G_x^2} + \boldsymbol{G_y^2}} \tag{17}$$

while the orientation of the gradient according to the equation.

$$\boldsymbol{\theta} = \mathbf{arctan}(\boldsymbol{G_y}/\boldsymbol{G_x}) \tag{18}$$

The results are shown in Figures 87 and 88.

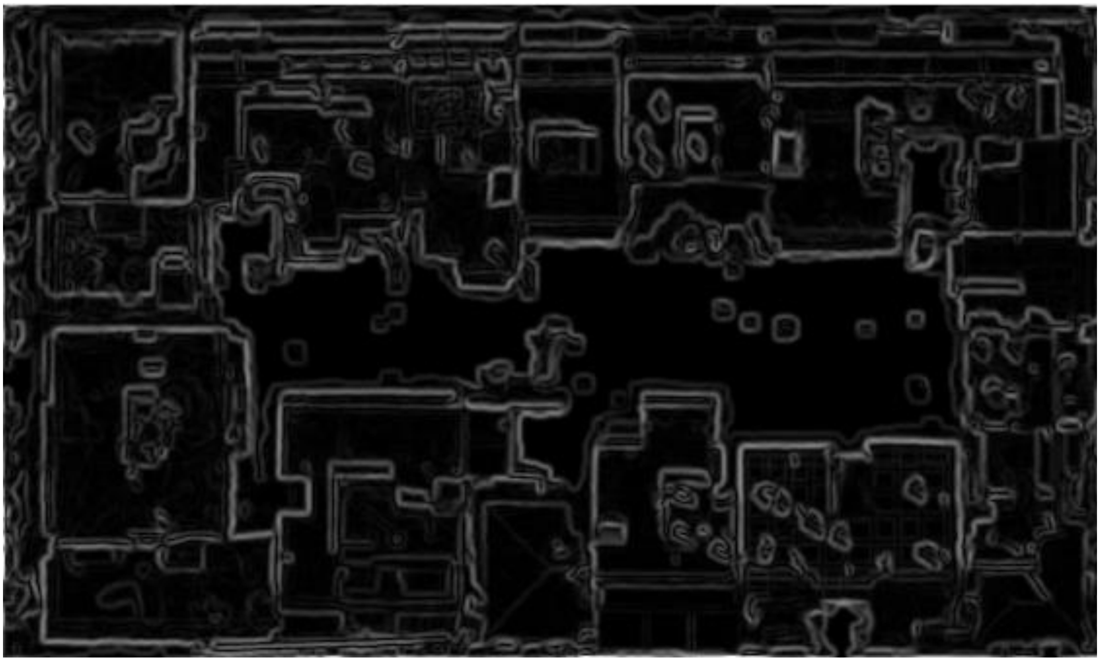


Figure 87 - Magnitude of optical image

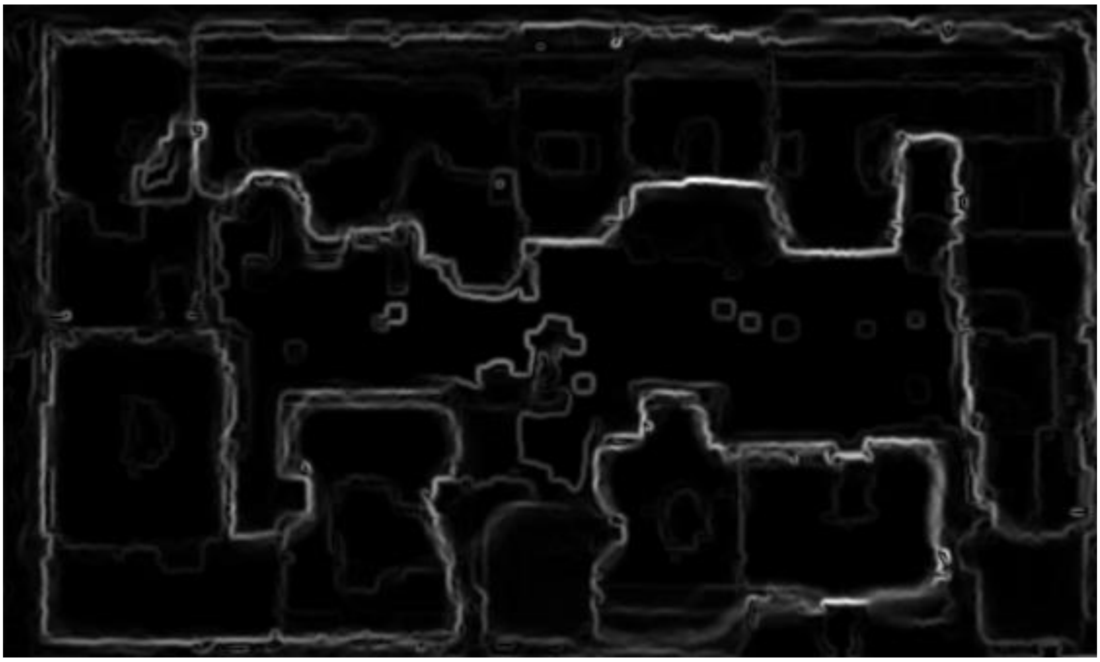


Figure 88 - Magnitude of DEM image

- Non-maximum suppression stage: The purpose of this stage of the Canny edge detection algorithm is to sharpen the edges. All local maxima of the gradient image are preserved while all other gradients are discarded. Since all calculations of this stage use 8-way connectivity, all orientations are rounded to the nearest 45º. Then the magnitude of each edge is compared to that of the magnitudes along the positive and negative gradient directions quantized to the nearest 8-neighbors. If the strength of the edge is the largest, then the edge is retained otherwise it is suppressed. The results of this stage are shown in Figures 89 and 90.

- Double thresholding stage: This final Canny stage has the goal of removing spurious edges due to noise. Two thresholds are used by Canny, a strong threshold above which edges are accepted and a weak threshold below which edges are discarded. Edges that have a magnitude between these two thresholds are marked as weak. A final stage called hysteresis is then applied. In this stage, weak edges are tested to ascertain whether they are flanked by strong edges. Such weak edges are turned into strong edges. The logic behind this is that noise in unlikely to result in strong edges, which are supposed to be due only to variations of the image. The results are shown in Figures 91 and 92.

- Create logical neighbor cell matrix: Four Matlab cell matrices[35] are used in order to extract all logical neighbors of a cell according to 8-way connectivity. These are the a) horizontal cell matrix; b) vertical cell matrix; c) 45° with the dimension being the same as that of the optical & DEM data. Each cell in turn holds a 3x3 matrix that represents the central edge and its eight neighbors. It has a logical one for each possible neighbor aligned along the orientation of the under consideration edge while all other entries have a logical zero. The four cell matrices are shown in Figures 93(a-d).

[35] Cell matrices are a Matlab complex array type in which each entry can hold another structure. For instance, each entry could contain a full matrix. These are essentially used as masks to extract the relevant neighbors according to the orientation.

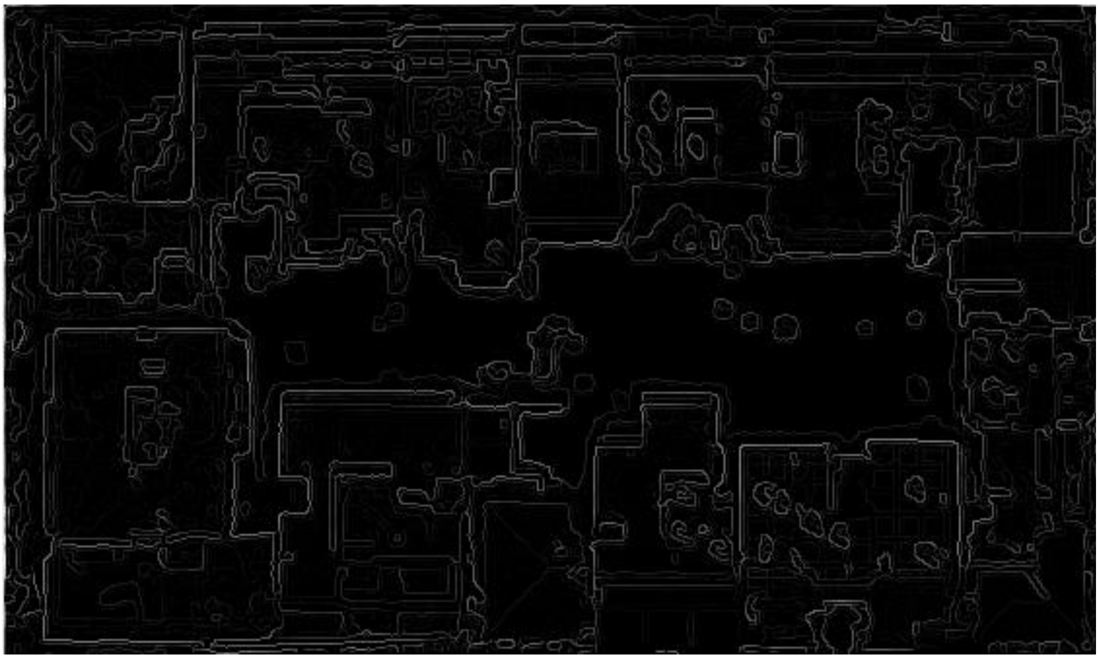


Figure 89 - Magnitude of gradient (optical) after non-maximum suppression

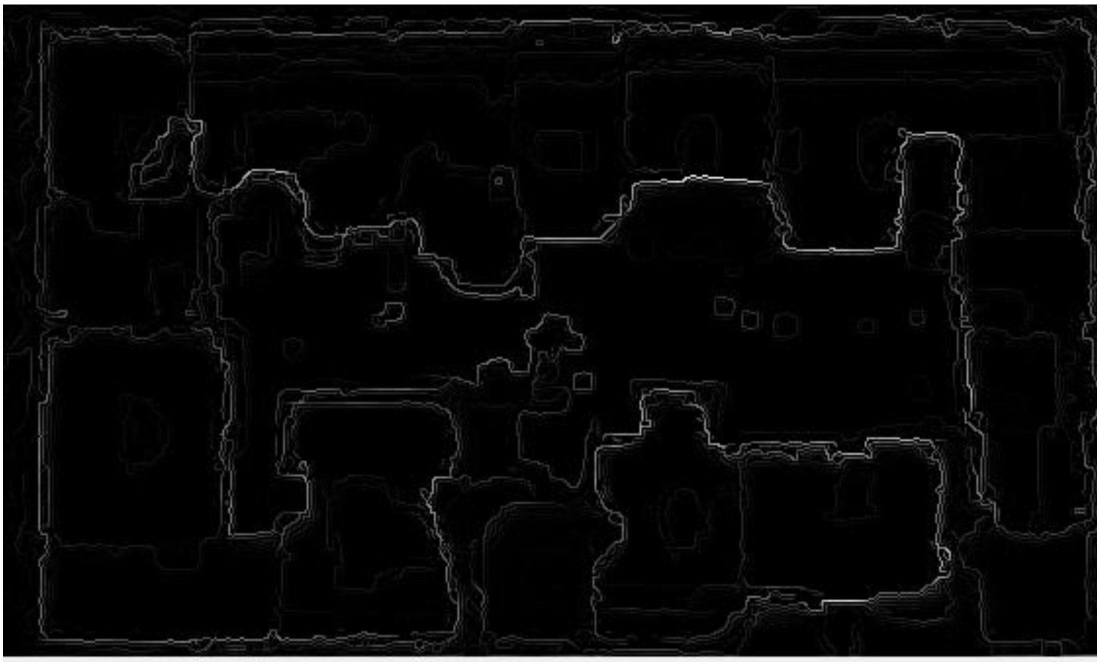


Figure 90 - Magnitude of gradient (DEM) after non-maximum suppression

- Create neighbor extract cell matrix A Matlab cell matrix with same dimensions as that of the optical & DEM data. Each cell holds a 3x3 matrix with the values of the magnitude along the asserted entries of the previous logical neighbor's cell matrix[36]. The previous logical neighbor's cell matrices are used as logical masks to extract the corresponding magnitude.

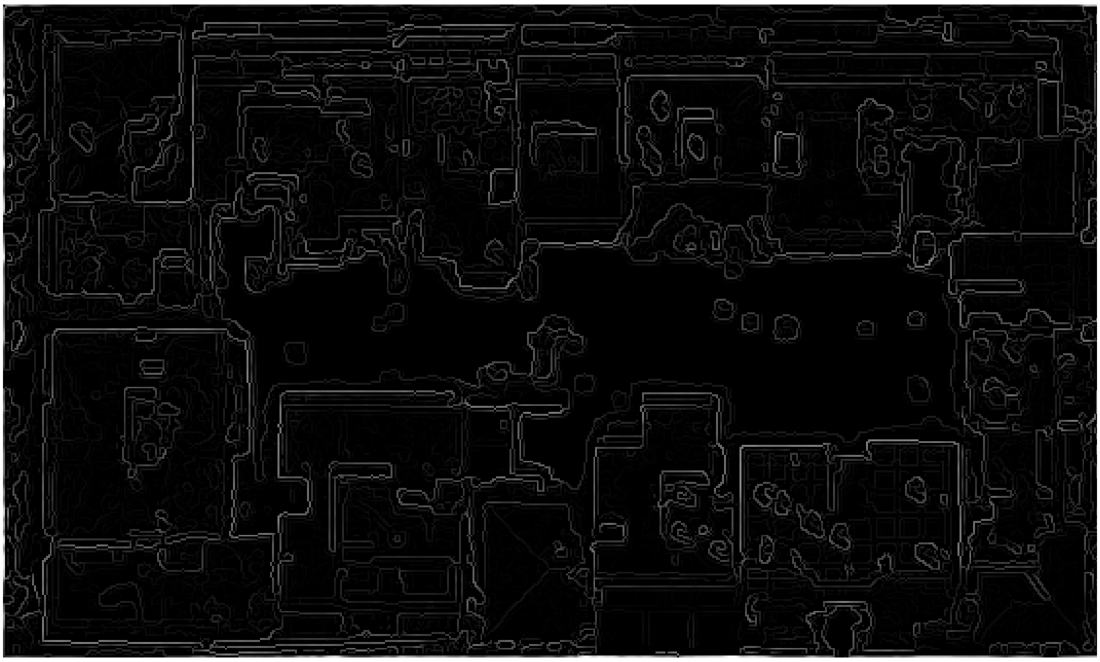


Figure 91 - Magnitude of optical image after double thresholding

[36] For instance an edge with horizontal orientation will have the neighbors extracted according to Figure 108(a); with vertical according to Figure 108(b); with 45° according to 108(c) and with -45° according to Figure 108(d).

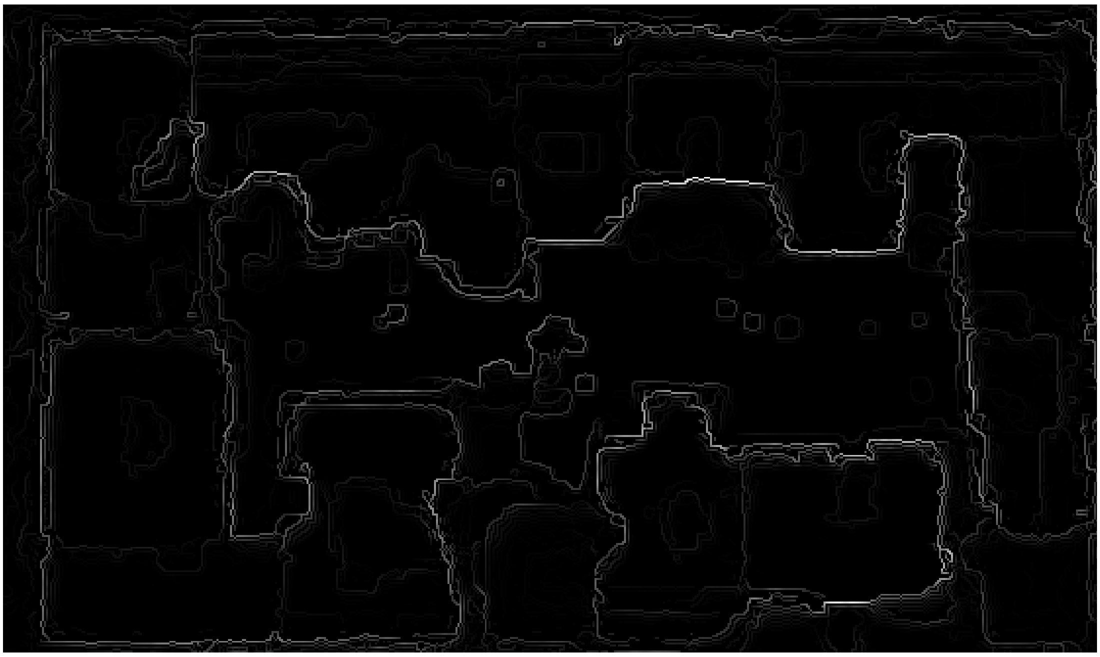
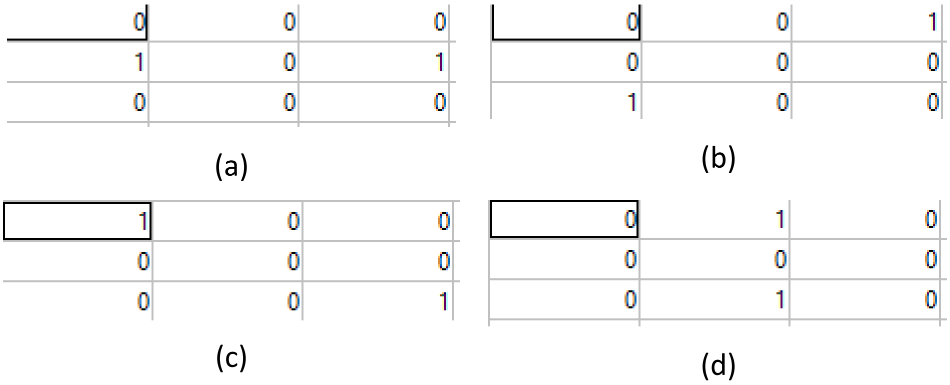


Figure 92 - Magnitude of DEM image after double thresholding



0	0	0
1	0	1
0	0	0

(a)

0	0	1
0	0	0
1	0	0

(b)

1	0	0
0	0	0
0	0	1

(c)

0	1	0
0	0	0
0	1	0

(d)

Figure 93 - Logical neighbors cell matrix

- Crack neighbors & logical Crack neighbors: These are derived cell matrices just like the previous two cell matrices. The difference is that they take into account the orientation of the possible neighbors and include them only if they are aligned with the central edge.

- Erosion of DEM data: An erosion morphological operator is applied to the DEM in order to facilitate the comparison with the optical data while minimizing the possibility of accidentally wiping off optical data that are not supported by the DEM.

6.3 Confidence matrix initialization

The goal of the proposed algorithm is to create orthogonal borders that correspond to the building facades of a densely populated urban area. In order to achieve this the edge borders of the optical image as attested by the magnitude of Figure 96 are utilized in an iterative technique called Relaxation. The confidence assigned to each edge is merely a measure of the certainty that it is a true edge belonging to a building contour. The algorithm processes edges differently according to their type. As previously stated we define four type types of edges (Vassilas N., Charou, Petsa, & Grammatikopoulos, 2013):

- '*Optical and elevation edges. Most edges belong to this category. Optical edges separate regions of different graylevel in the optical image while elevation edges separate regions of different height. This the typical case'*.

- '*Optical edges that are not elevation edges. These edges separate regions of different graylevel but same height. For instance, adjacent roofs with different graylevel*'.

- '*Elevation edges that are not optical edges: These edges separate regions of different height but same color. For instance, roofs that have the same graylevel as the pavement*'.

- '*Edges that are neither optical nor elevation edges: This is the most difficult case. For instance, adjacent roofs with the same height and same graylevel*'.

The proposed relaxation system is divided into two sub-systems. The first sub-system performs one time confidence adjustment. This is because of the nature of some of the data they cannot be used in an iterative procedure since this would always yield the same result. The confidence matrix would thus be distorted and not correspond to reality. It is for this reason that these data affect the confidence matrix once before the relaxation process commences.

The initial confidence matrix is set to the magnitude of the optical data after the pre-processing that occurred in the previous section. It can be seen in Figure 94.

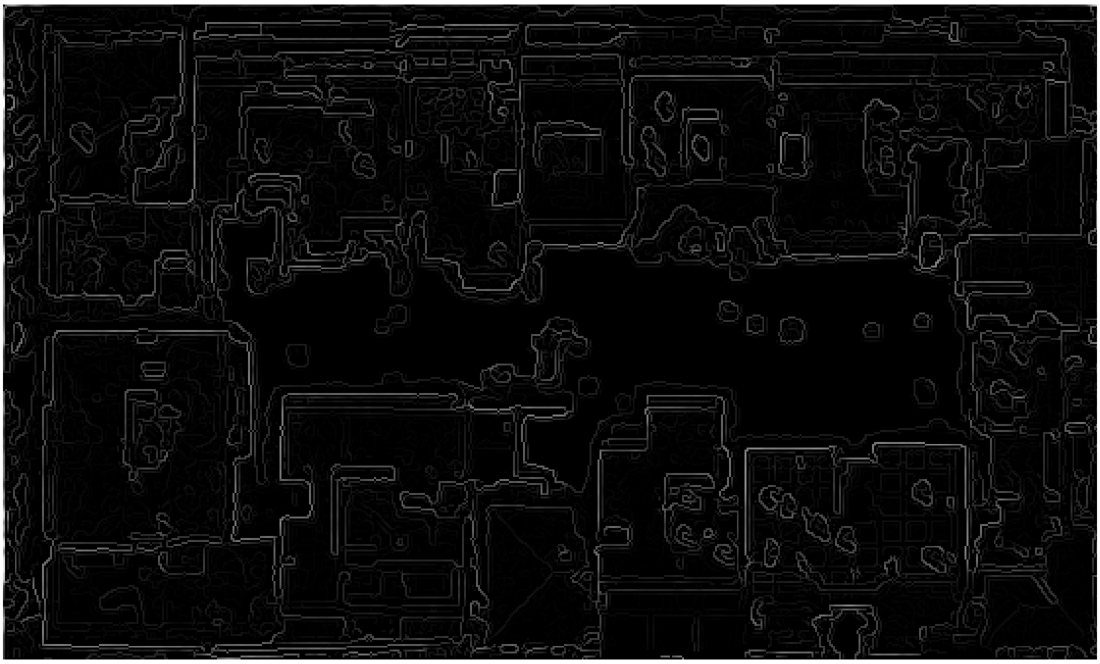


Figure 94 - Initial confidence matrix (set to magnitude of optical data)

One time confidence adjustment[37]

The confidence adjustment presented in this section are performed only once before the iterative procedure commences.

1. Pruning of all edges below 3.7m: Any structure below the threshold of 3.7m can be safely removed since it is assumed not to belong to a building facade. This is done by examining the DEM data as it has been pre-processed in the previous section. The modified confidence matrix is graphically depicted in Figure 95. Table 32 lists the number of edges that were pruned according to this criterion.

Table 32 -Low height edges pruned after low-height object removal

Edges pruned	**51730**

[37] As explained in introduction to this section, due to the static nature of some of the data they cannot be used in the iterative relaxation process. They are however used to adjust the confidence once before the iterative relaxation procedure commences.

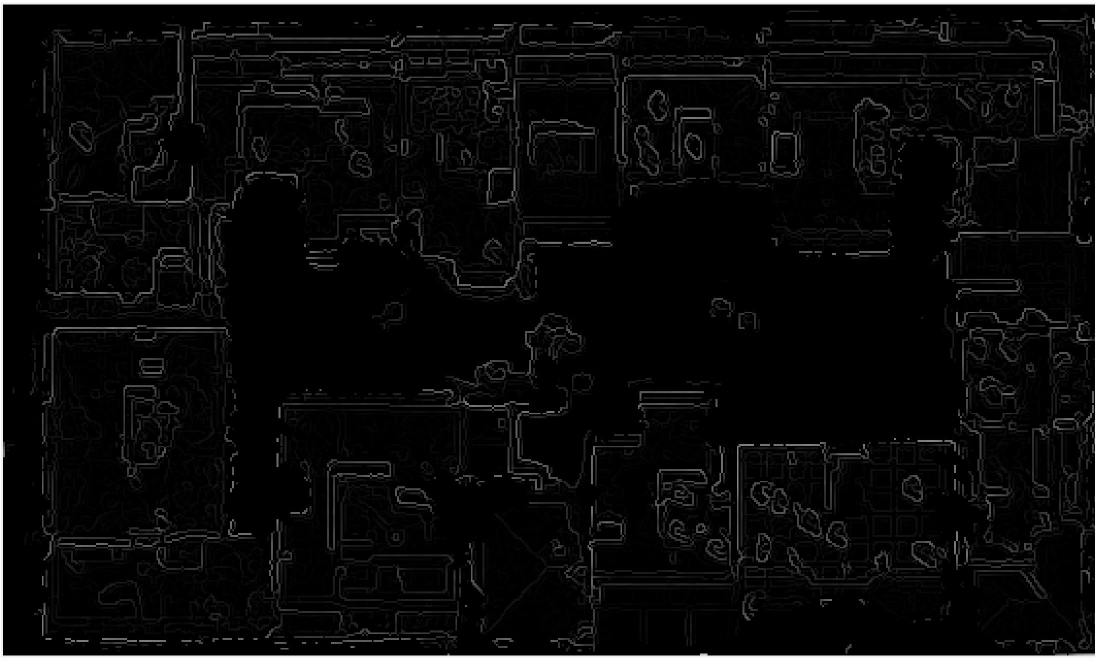


Figure 95 - Confidence matrix after low-height object removal

2. Enhancement of edges that coincide with cohesive regions edges: Edges that coincide with those of figure 90(b) have their confidence set to 100% since the optical edges fall upon the cohesive regions boundaries. This almost certainly only occurs for a true edge. Nine thousand two hundred fifty two edges were found to coincide. The modified confidence matrix is show in Figure 96.

Table 33 - Number of optical edges coinciding with cohesive regions edges

Positive edges	9257

3. Enhancement of optical edges that are also DEM edges: If an optical edge coincides with a DEM edge then we can be almost certain that this a true edge. The confidence is boosted to 100% and is displayed in Figure 97.

Table 34 - Number of optical edges coinciding with DEM edges

Positive edges	15349

4. Suppression of edges delimiting low height spanning objects: Edges of objects with a relative height span less than 3.5m are labeled as low height-spanning objects and are suppressed. This is done with the aid of the elevation edge image (Figure 92).

Table 35 -Number of optical edges suppressed due to low-height spanning objects

Positive edges	26790

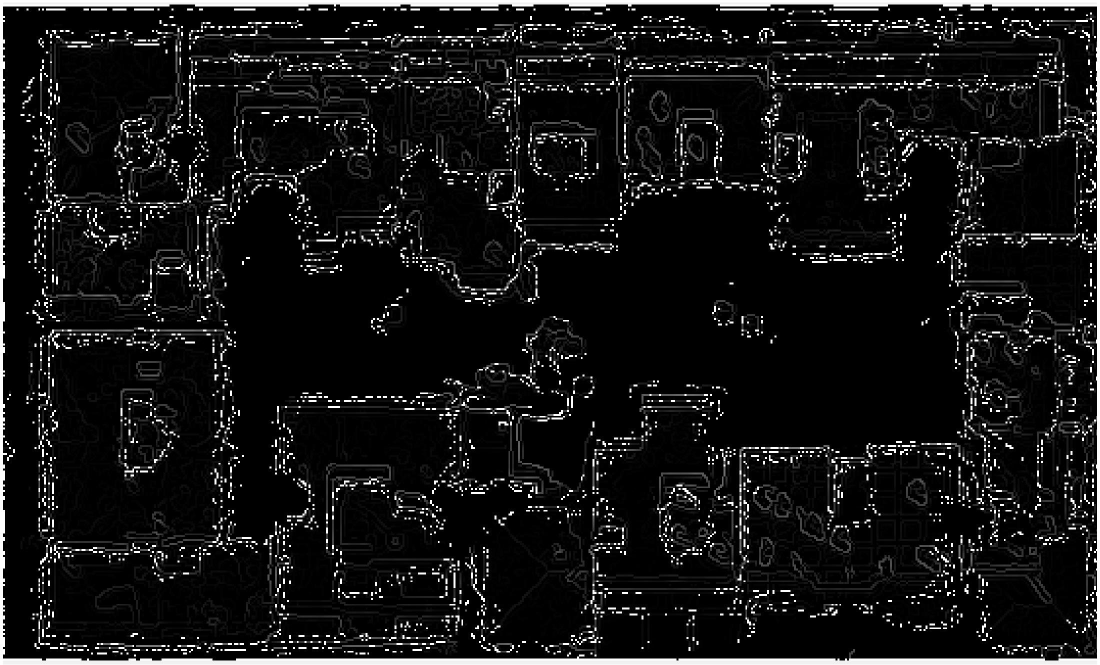


Figure 96 - Confidence after taking into account the coincidence of optical and cohesive region edges

5. Enhancement of edges that are optical and are above minimum height-span: All optical edges that are above the 3.7m height threshold have their credibility augmented. Likewise all optical edges that are above 3.7m and are at a cohesive region boundary are further augmented. Finally, all optical edges that are above 3.7m, are Canny optical edges and are at a cohesive region boundary are even further augmented. The new confidence matrix is shown in Figure 98.

Table 36 - Edges above 3.7m that have their confidence augmented

Edges above 3.7m	5791
Edges above 3.7m and at cohesive region boundary	3543
Edges above 3.7m, at cohesive region boundary and Canny output edges.	2007

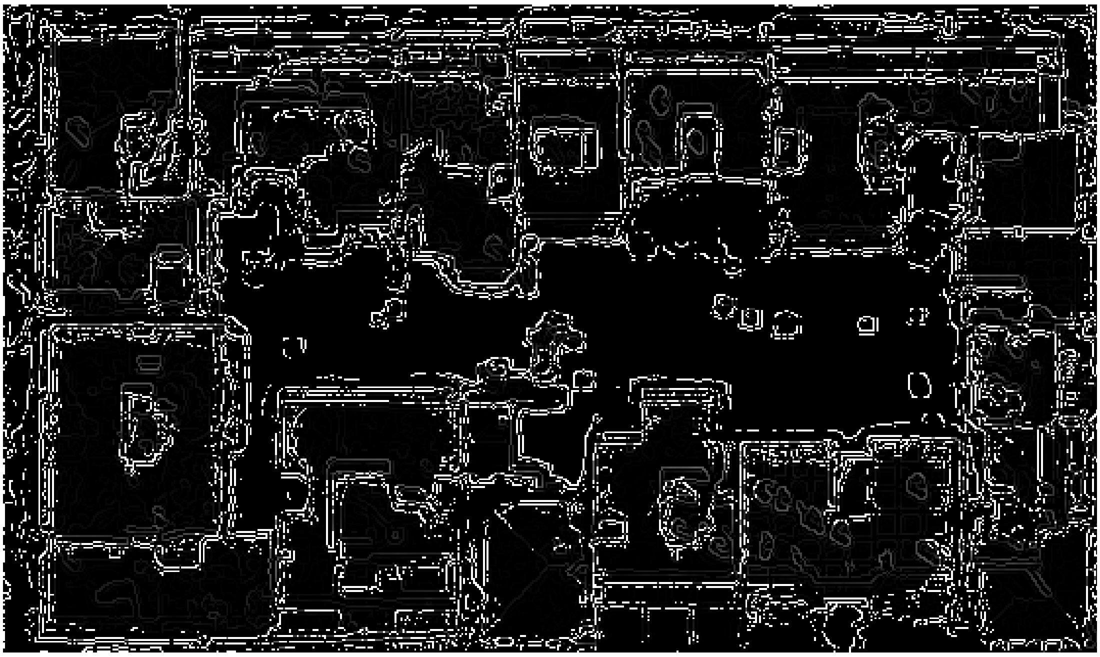


Figure 97 - Confidence matrix after coinciding optical & elevation edges have their confidence boosted to 100%

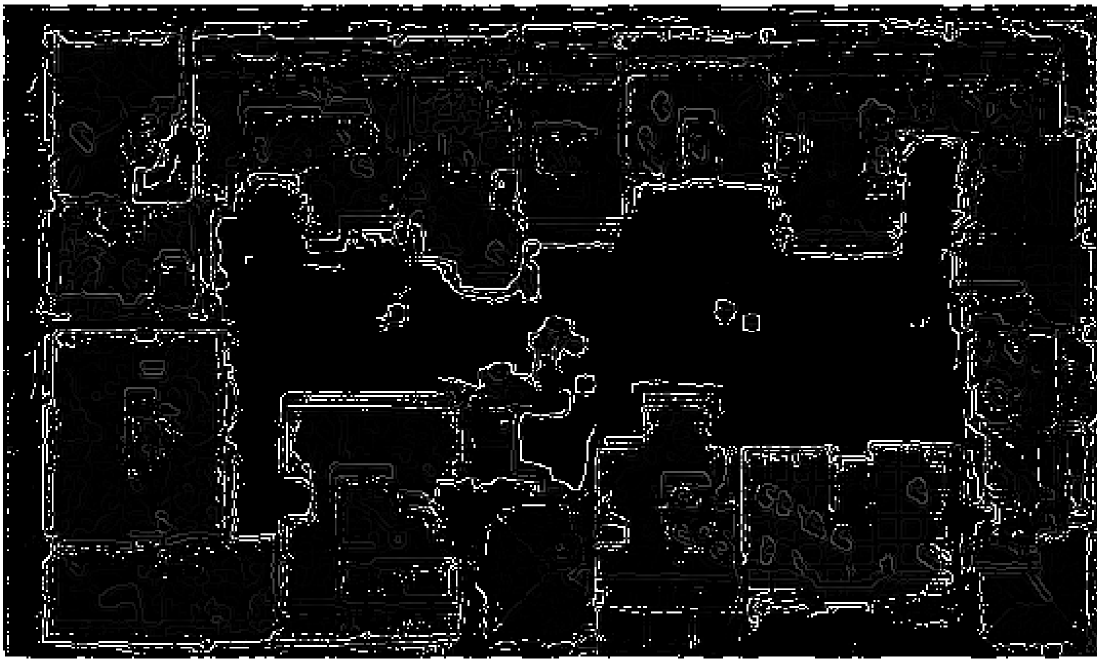


Figure 98 - Confidence matrix after augmenting confidence according to Table 36

6. Edges along dominant directions: All edges that have an orientation aligned with the dominant directions is given a small augmentation to their credibility. On the contrary, all edges that are not aligned with the dominant directions have their credibility decremented. Since, the initial data images have been compensated for the dominant directions, the default dominant directions are 0º and 90º. The new confidence matrix can be seen in Figure 99.

Table 37 - Edges aligned with dominant directions

Edges aligned with dominant directions	**14868**
Edges not aligned with dominant directions	49168

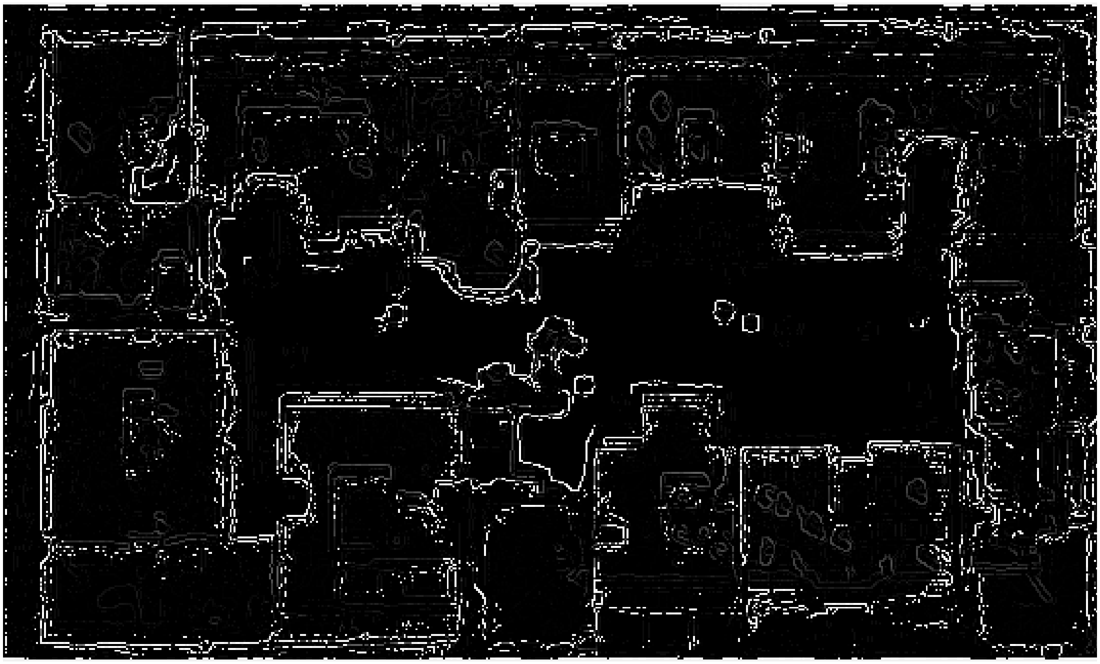


Figure 99 - Confidence matrix after augmenting confidence of edges along dominant directions

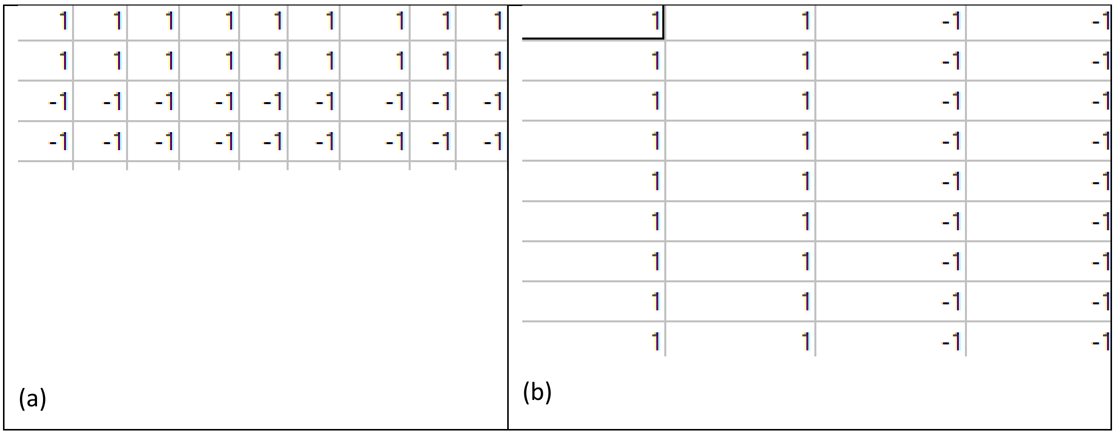
7. Shadow mask edges: Careful examination of the original shadow mask in (Vassilas N., Charou, Petsa, & Grammatikopoulos, 2013) reveals that the illumination source (sun) must have been placed at the bottom-center of the image. It was determined in the research by Vassilas et al. that the horizontal shadows must be the result of higher buildings casting their shadows north on lower surfaces while the vertical shadows are due to higher buildings casting their shadows to the east[38]. These borders (edges) give

[38] According to Vassilas et al. (2013) who originally created the shadow mask. 'The illumination source must have been to the southeast'. Since our data have been rotated to compensate for the dominant directions, the illumination source must have been to the bottom-center.

very important information about the boundaries of the building facades and thus have priority over other edges or height data.

The shadow mask used for this processing was created by turning the initial binary image of the shadow mask (Figure 82) to a bipolar map (0/1 to -1/+1). The bipolar map was then filtered with the templates of Table 38(a-b), for horizontal and vertical edges, respectively and then normalized with the Otsu method. This resulted in a usable shadow mask, which is depicted in figure 100. The shadow mask of Figure 100 is then placed on top of the optical edges and coinciding edges are found by performing a logical AND between them. The remaining edges have a high probability of being a true edge belonging to a building contour. Hence, their confidence is significantly boosted. The resulting confidence mask can be seen in Figure 101.

Table 38 a) Horizontal template b) Vertical template



1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

(a)

1	1	-1	-1
1	1	-1	-1
1	1	-1	-1
1	1	-1	-1
1	1	-1	-1
1	1	-1	-1
1	1	-1	-1
1	1	-1	-1
1	1	-1	-1

(b)

Table 39 - Number of optical edges that coincide with shadow edges

Number of optical edges that coincide with shadow edges	2858

8. Enhancement of credibility of edges that have strong neighboring edges along their orientation: Each edge that has a strong neighboring edge along its orientation is enhanced. Eight-way connectivity is used in order to ascertain whether a strong neighboring edge exists, thus all orientations are partitioned in 45° areas. Each edge that has at least one neighbor along its orientation has its credibility augmented. The resulting confidence matrix can be seen in Figure 102.

Table 40 - Number of edges with strong edges along their orientation

Number of edges that have strong neighbors along their orientation	11688

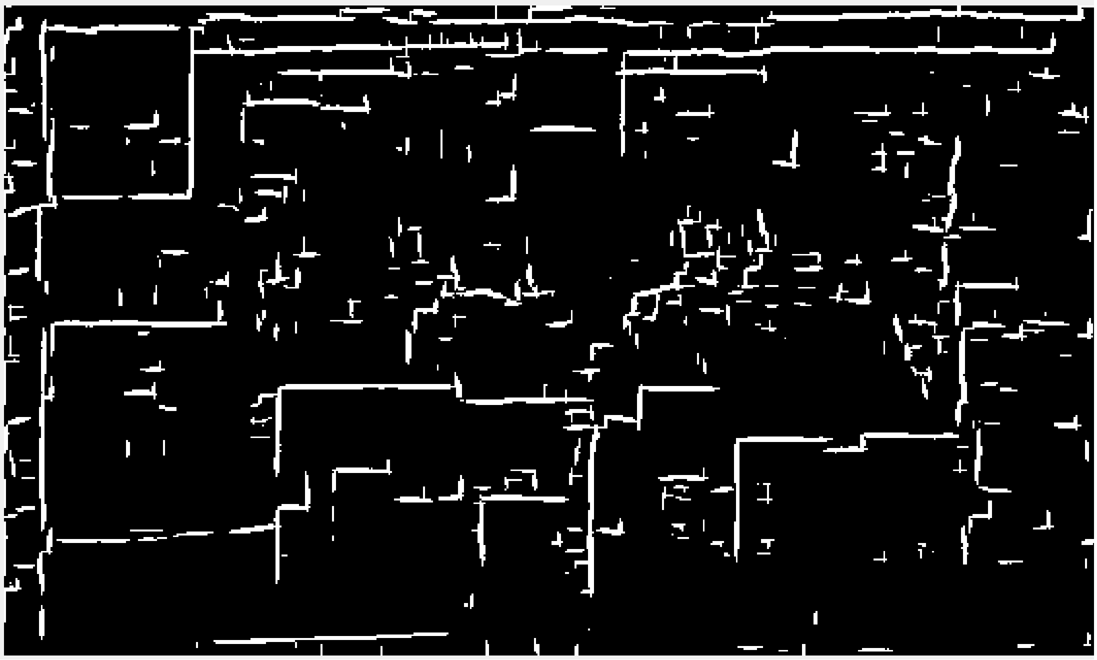


Figure 100 - Final shadow mask

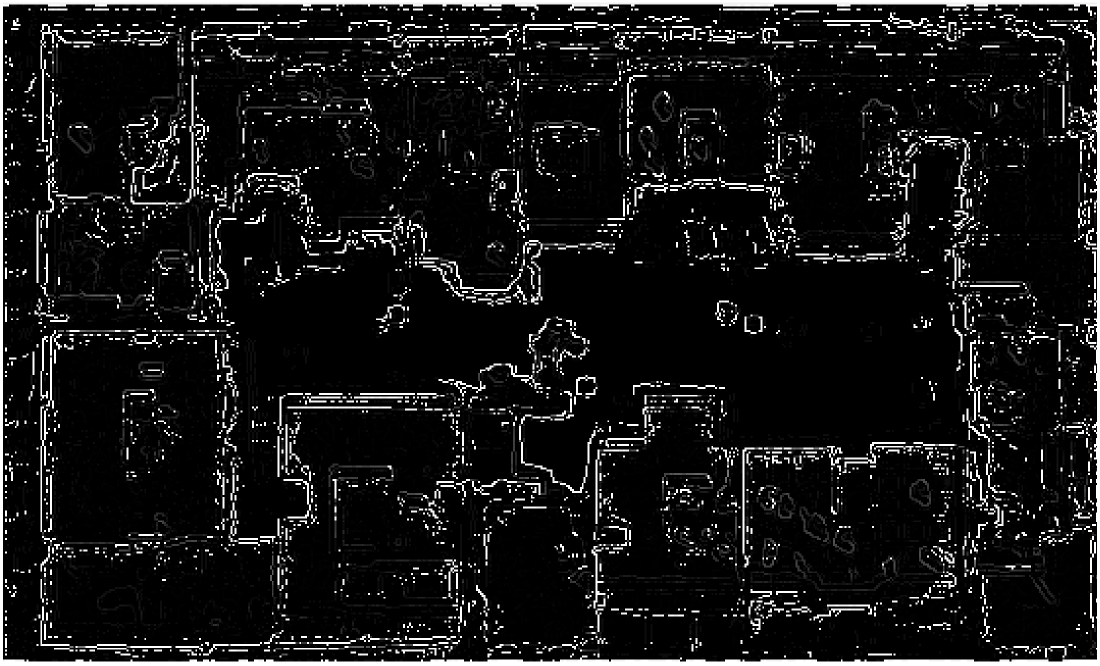


Figure 101 - Confidence after augmenting edges that coincide with shadow edges

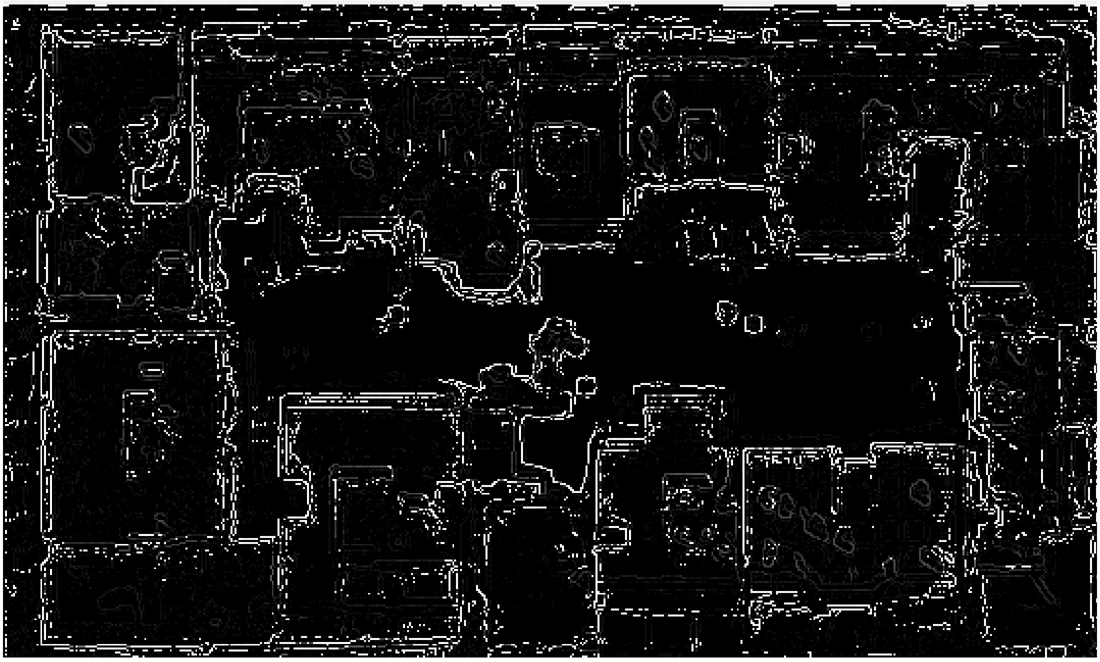


Figure 102 - Confidence after taking into account neighbors along edge orientation

This was the final step of the pre-processing before the iterative relaxation process.

6.4 Iterative relaxation

As stated at the beginning of this chapter, borders of regions or other edges are strongly affected by image noise. Therefore, considering the context of an edge can result in a crisper image. For example, 'a weak horizontal edge positioned between two strong horizontal edges is highly probable to be a true edge and should gain credibility. On the contrary, an edge that is positioned by itself with no supporting context should have its credibility decreased'. This is the basic idea behind Prager's work. The contribution of this thesis to his work is that connectivity is considered not only for horizontal and vertical edges but also for other diagonal edges.

Prager (Prager J. , 1980) proposed an iterative technique, which can easily be parallelized, that gradually increases/decreases the credibility or confidence of the edges until they asymptotically approach 0 or 1. The proposed algorithm is presented below:

Table 41 - Proposed relaxation algorithm (Prager, 1980)

1. Set the initial confidence of each edge as the gradient of the optical image, normalized to unity.

```
2. Enter loop.
    1. Compute edge-type and vertex-type based on the confidence of edge neighbors.
    2. Modify confidence of each edge based on its vertex type and previous confidence.
    3. End iterative loop when all confidences have asymptotically approached zero or one.
```

The two most important notions of this algorithm are:

- Edge-types: The number of left and right neighbors. There are two edge types ranging from 0 to 3 for each edge, as explained in Section 6.1.

- Vertex-type: Each vertex (edge) has a left and right edge-type, which are computed from the strength of edges emanating from a vertex. Their concatenation is the vertex-type.

The proposed variation of Prager's algorithm uses an 8-way connectivity scheme and is applicable to horizontal edges, vertical edges 45º and -45º edges[39]. It can easily be extended to other edge types as well[40]. Starting from the central edge *e* and considering a horizontal orientation, the left-vertex is the end-point for three other possible edges to the left. Likewise, the right-vertex is the end-point for three other possible edges to the right.

The variation of the Prager's algorithm implemented for the purposes of this thesis, assumes that each edge can continue along three edges to the left and/or three edges to the right for the horizontal case. Three more cases are consider which are the vertical, the diagonal type 1 and the diagonal type 2. The idea is illustrated in Figure 103. The central edge in Figure 103(a) has three possible neighbors to the left and three possible neighbors to the right. How many neighbors really exists is determined according to the equations, which were presented in section 6.1. Regarding the relaxation technique, each edge is evaluated according to the number of edges emanating from the vertex (left or right for the horizontal case, top or bottom for the vertical case, alternating diagonal connectivity for the two diagonal cases). The edge-type is then simply a concatenation of the left and right vertex-types, using an x-y pairing scheme, where x is the number of left neighbors and y the number of right neighbors. The confidence of each edge is then modified in an iterative scheme according to the edge type[41] where equations (7-11) are used to calculate the vertex-type. The vertex type calculations depend on the orientation of the under examination edge and the crack edges in its context, as shown in Figure 103. After the determination of the left and right vertex-type, the edge-type is simply the concatenation of the left and right vertex-type. Finally, the edge confidence in each

[39] Prager's algorithm only labeled edge-types according to horizontal and vertical crack edges.

[40] For instance, the granularity of the orientations could me made finer.

[41] Edges types were defined in section 6.1.

iteration is modified according to equations 12-14 and the vertex-type.

The algorithm is iterative and it aims to categorize all elements of the confidence matrix as certain edges (aka belonging to a building contour or not) or non-certain edge (aka those that the algorithm failed to verify as belonging to building contours or not). The categories are defined as shown below:

- Certain Edges: Edges that have a confidence less than 0.2 or greater than 0.8.
- Non-certain edges: Edges that have a confidence between 0.2 and 0.8.

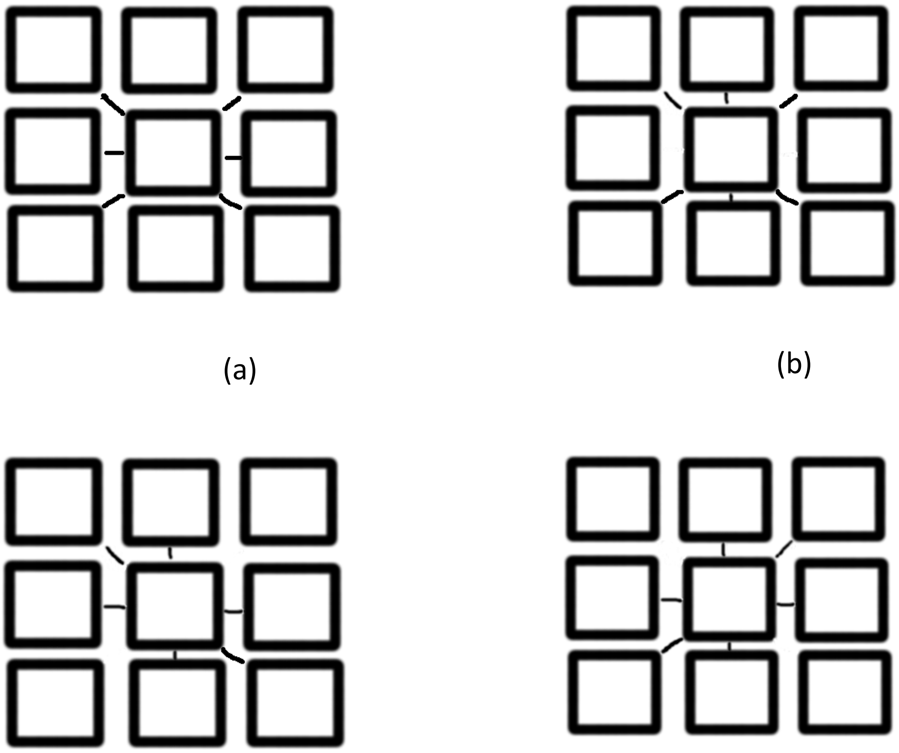


Figure 103 - Crack edges. a) Horizontal b) Vertical c) Diagonal case 1 (45°) d) Diagonal case 2 (-45°)

The iterative effect of the relaxation process on the confidence of the edges is shown in Table 42. As can be seen from the table, the algorithm has managed to classify the vast majority of edges with certainty. Only five (5) edges have remained unclassified. The final confidence matrix is illustrated in Figure 104.

After the final confidence has been calculated, a small region elimination procedure is executed on the image. The result of the small area elimination process is shown in Figure 105.

Table 42 - Iterative effect of relaxation process on confidence

Iteration	Edges classified with certainty	Non-certain edges
0	12081	2658
1	12081	4204
2	12081	8070
3	12082	8069
4	12088	8063
5	12103	8048
6	12143	8008
7	12236	7915
8	12382	7769
9	12584	7567
10	12983	7168
11	13750	6401
12	14855	5296
13	16280	3871
14	20146	5
15	20146	5

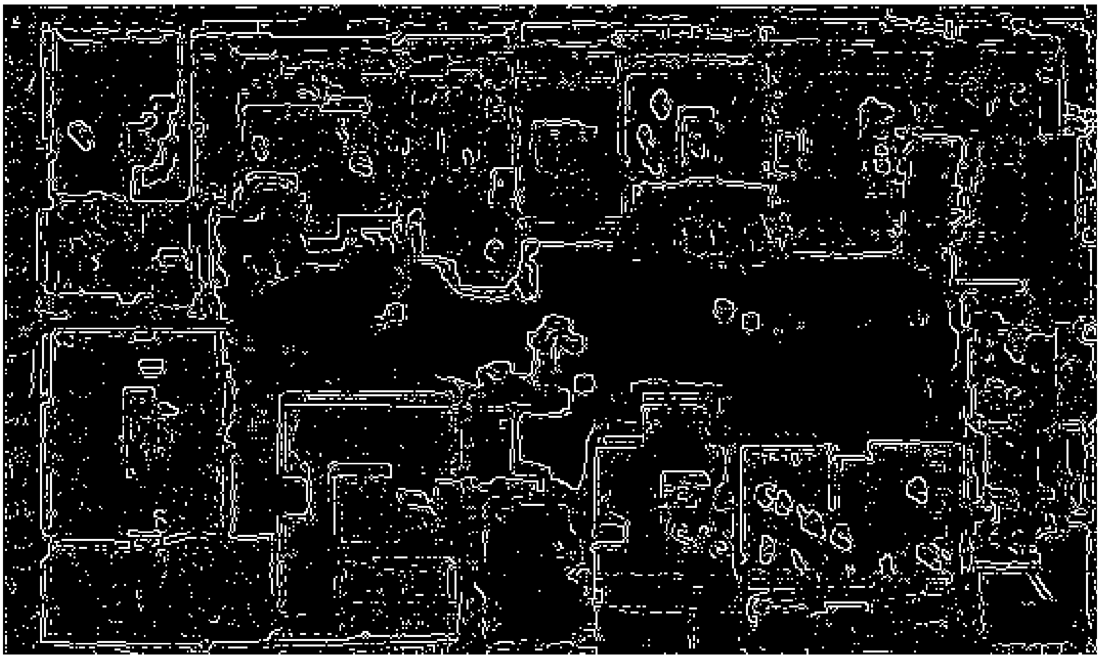


Figure 104 - The confidence matrix after the 15th iteration of the relaxation algorithm

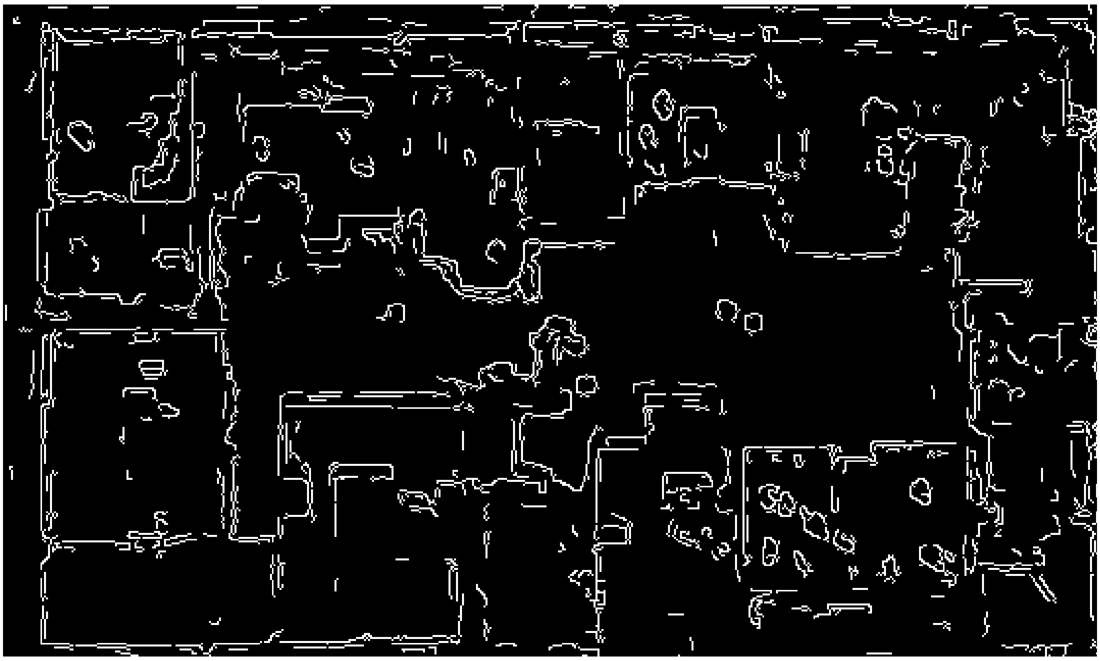


Figure 105- Final confidence after small area elimination (binary image)

6.5 Evaluation

As can be seen from visually inspecting Figure 105, many of the important edges have been discovered. However, there do exist gaps in the edge chain, which a deeper context examination might have found. This would be an interesting topic to pursue in future research. For the quantitative assessment of the performance of the algorithm, two metrics were utilized. The first was the mean square error (MSE) and the second was the peak signal to noise ratio (PSNR). The metrics regarding the Relaxation method were taken after performing small area elimination.

The ground truth image for the building block whose edges were discovered by the Relaxation system is shown in Figure 106. This same block was used to train the BCDCNN neural network proposed in Chapter 4 and a direct comparison is thus possible.

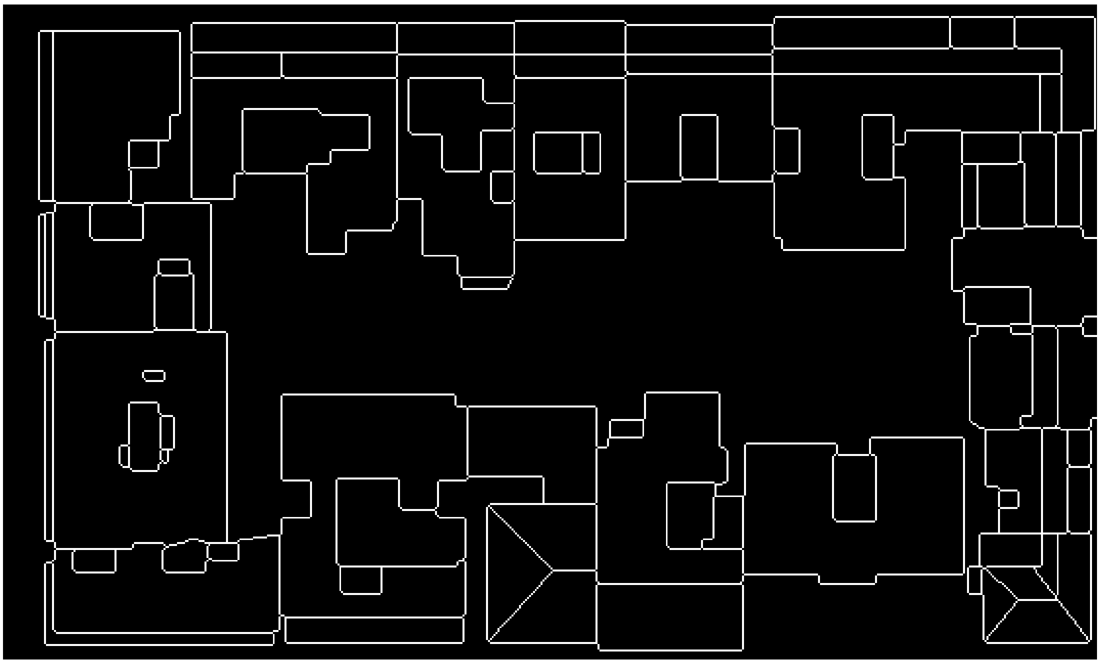


Figure 106 - Ground truth image

Table 43 - Quantitative comparison between Relaxation system and BCDCNN

	MSE	PSNR
Relaxation	0,1280	8,9282
BCDCNN	0,10423	15,269

From Table 43 it can be seen that the relaxation method performs worse than the proposed in Chapter 4 BCDCNN for both the MSE and PSNR metrics. The neural network has been trained with the GT image of Figure 106 so it would be very difficult for the relaxation method to perform better.

7. CONCLUSIONS

This thesis has presented innovative research methodologies towards the automatic detection of building contours. Building contours can be considered as a first step for a 3D model of urban areas.

The building contour detector presented in Chapter 4, which is based on convolutional neural networks, proved that CNNs are potent tools to obtain a full image reconstruction of the building contours. This is in contrast to most to date typical applications of convolutional neural networks, which operate as classifiers. The network that was named BCDCNN, accepts low-resolution elevation data of an urban area and corresponding high-resolution optical data of the same area. It then performs a hetero associative mapping to a new image, which contains the building contours. Another innovation of the proposed model is the Top-N custom layer, which offers performance benefits, wherein the RMSE and PSNR exhibit better performance for the Top-N layer as opposed to the typical MSE cost layer. The effect of adding more feature maps was also examined and it was shown that dropout is mostly necessary in order for the model to generalize. It is very interesting to notice that training with the LoG data set was the only case in which the network managed to generalize without using dropout, presumably a result of the reduced dimensionality of the LoG dataset. The tackled problem is extremely complex to solve using deep neural networks due to the varying context around true building contours in an urban environment. It is conjectured that given more training data the performance of the network will increase but handcrafting such ground truth data is a very tedious and time costly procedure. It would be interesting to see how the network would perform given more training data[42]. Further research proposal on CNNs and building contour detection would be to build a pixel classifier whose performance could be compared with this implementation.

The stimulus for building the super-resolution system presented in Chapter 5 was the BCDCNN network, since it had been initially designed to perform super-resolution with a single channel. This thesis examined how well this network would perform with elevation data when assisted by a second channel of optical data. It was designed to enhance the resolution of low-resolution elevation data augmented by corresponding high-resolution optical data. The research demonstrated the efficacy of deep neural networks for super-resolution applications regarding elevation data. It also exhibited some intricacies of elevation data. Foremost of them is the requirement that these models are trained with elevation data per se. Elevation data seem to have an increased ratio of low to high[43] frequency content as compared to generic images that makes it difficult for SR CNNs trained on generic images to perform well on them. Generic SR CNNs although offering top-notch performance on general-purpose images, failed to hold that performance when presented with elevation data. It was also demonstrated, as proof of concept that high-resolution optical data can help augment low-resolution optical data. This can be seen for elevation set 1 and 2 when reconstructing on Validation set 2. In this setup, the dual channel version performed

[42] Obtaining pairs of optical & DEM datasets is quite expensive. Therefore, a promising avenue to explore would be the synthesis of artificial optical-DEM data pairs.

[43] Or vice-versa.

better than the corresponding single channel version. Furthermore, using many feature maps does not scale well when using a small dataset. The number of feature maps must be commensurate with the available volume of training data while a similar trend seems to hold regarding the dimensions of the convolution kernel. This application was also hampered by the resolution discrepancy between the low-resolution elevation data and the high-resolution optical data. It is surmised that lowering the ratio of the discrepancy would lead to better results. In addition, it would be interesting to explore deeper architectures trained with more data. Since procuring pairs of optical - elevation data is costly, the previous proposition regarding BCDCNN to use synthetic pairs of optical - elevation data also holds for BSRCNN.

Finally, the developed relaxation system accepts multi-resolution spatial data from various sources and fuses them all together. It attempts to ease in pixels that belong to building contours to an edge chain while suppressing pixels that do not fit in. For this reason, it is an iterative process that examines the local context of every pixel of the image depicting an urban area taking into the orientation of the under examination edge and that of the local context. It then proceeds to either augment or decrement the confidence of the pixel belonging to a building contour. The proposed system was shown to provide clear-cut edge chains that mostly belonged to building contours. A possible enhancement to the system would examine a deeper local context.

This thesis presented two innovative deep CNN systems that can be used to detect building contours or perform super resolution on elevation data. These systems can operate independently. However, it would be worth exploring how the output of these two systems can be used by the proposed relaxation system. The combinatory power of the output produced by BCDCNN and the relaxation system could lead to even better building contour detection. On top of that, BSRCNN can be used to augment the resolution of the elevation data provided to the relaxation system. This would be a final proposal for future research based on this thesis.

Bibliography

AI, Misssing Link. (2019, 9 8). *Fully Connected Layers in Convolutional Neural Networks: The Complete Guide*. Retrieved from https://missinglink.ai/guides/convolutional-neural-networks/fully-connected-layers-convolutional-neural-networks-complete-guide/

Alshehhi, R., Marpu, P., Woon, W., & Dalla, M. M. (2017). Simultaneous extraction of roads and buildings in remote sensing imagery with convolutional neural networks. *ISPRS Journal of Photogrammetry and Remote Sensing, 130*, pp. 139-149.

Baranov, M., Olea, R., & van den Bogaart, G. (2019). Chasing Uptake: Super-Resolution Microscopy in Endocytosis and Phagocytosis. *Trends in cell biology*.

Benecki , P., Kawulok , M., Kostrzewa, D., & Skonieczny. (2018). Evaluating super-resolution reconstruction of satellite images. *Acta Astronautica, 153*, pp. 15-25.

Bengio, Y., Bordes A., & Glorot X. (2011). Deep sparse rectifier neural networks. *Proceedings of the fourteenth international conference on artificial intelligence and statistics (*, pp. 315-323.

Biegler-König, F., & Bärmann, F. (1993). A learning algorithm for multilayered neural networks based on linear least squares problems. *Neural Networks, 6(1), 127-131.*

Borman, S. &. (1998, August). Super-resolution from image sequences-a review., (pp. 374-378)).

Brownlee, J. (2019, 9 8). *Machine learning mastery*. Retrieved from https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/

Bryson, A. E. (1961). In Proc. Harvard Univ. Symposium on digital computers and their applications (Vol. 72). *A gradient method for optimizing multi-stage allocation processes.*

Bryson, A., & Denham, W. (1962). A steepest-ascent method for solving optimum programming problems. *Journal of Applied Mechanics, 29(2),*, pp. 247-257.

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence, (6),*, pp. 679-698.

Cho, W., Jwa, Y., Chang, H., & Lee, S. (2004). Pseudo-grid based building extraction using airborne LIDAR data. *Int. Arch. Photogramm. Remote Sens, 35*, pp. 378-381.

Comanciu, D., & Meer, P. (2002). Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24(5),*, pp. 603–619.

Datsenko, D., & Elad, M. (2007). Example-based single document image super-resolution: a global MAP approach with outlier rejection. *Multidimensional Systems and Signal Processing, 18(2-3)*, pp. 103-121.

Dayan, P., Hinton, G., Neal, R., & Zemel, R. (1995). The helmholtz machine. *Neural computation, 7(5),*, pp. 889-904.

Dong, C. (2019, 09 15). *Image Super-Resolution Using Deep Convolutional Networks*. Retrieved from Image Super-Resolution Using Deep Convolutional Networks: http://mmlab.ie.cuhk.edu.hk/projects/SRCNN.html

Dong, C., Loy, C., He, K., & Tang, X. (2015). Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence, 38(2)*, pp. 295-307.

Fahlman, S. E. (1991). An empirical study of learning speed in back-propagation networks.

Fog, A. (2019, 09 03). *A History of Deep Learning*. Retrieved from import.io: https://www.import.io/post/history-of-deep-learning/

Fraser, C., Baltsavias, E., & Gruen, A. (2002). Processing of Ikonos imagery for submetre 3D positioning and building extraction. *Journal of Photogrammetry and Remote Sensing, 56(3)*, pp. 177-194.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. In *Biological cybernetics, 36(4)* (pp. 193-202).

Haala, N., & Nrenner, C. (1999). . Extraction of buildings and trees in urban environments. In *. Isprs journal of photogrammetry and remote sensing, 54(2-3),* (pp. 130-137).

Hinton , G., Osindero, S., & Teh, Y. (2006). A fast learning algorithm for deep belief nets. *Neural computation, 18(7)*, pp. 1527-1554.

Huang B., Wang, W., Bates, M., & Zhuang, X. (2008). Three-dimensional super-resolution imaging by stochastic optical reconstruction microscopy. *Science, 319(5864)*, pp. 810-813.

Huang, H. &. (2010). Super-resolution method for face recognition using nonlinear mappings on coherent features. *IEEE Transactions on Neural Networks, 22(1),*, pp. 121-130.

Judd, J. S. (1990). *Neural network design and the complexity of learning.* MIT press.

Kelley, H. J. (1960). Gradient theory of optimal flight paths. *Ars Journal, 30(10)*, pp. 947-954.

Khan, A., Sohail, A., Zahoora, U., & Qureshi, A. (2019). A survey of the recent architectures of deep convolutional neural networks. In *arXiv preprint arXiv:.*

Kim, T., & Muller, J. (1999). Development of a graph-based approach for building detection. *Image and Vision Computing 17(1)*, pp. 3-14.

Krizhevsky, A. S. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, pp. 1097-1105.

Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, pp. 1097-1105.

Kůrková, V. (1992). Kolmogorov's theorem and multilayer neural networks. *Neural networks, 5(3)*, pp. 501-506.

Lafarge, F., Descombes, X., Zerubia, J., & Pierot-Deseilligny, M. (2008). Automatic building extraction from DEMs using an object approach and application to the 3D-city modeling. In *ISPRS Journal of photogrammetry and remote sensing, 63(3),* (pp. 365-381).

Lang, F. (1996). 3D-city modeling with a digital one-eye stereo system. *Proceedings of the XVIII ISPRS-Congress.*

Lang, K., Waibel, A., & Hinton, G. (1990). A time-delay neural network architecture for isolated word recognition. *Neural networks, 3(1)*, pp. 23-43.

Le, Q. V. (2015). *A tutorial on deep learning part 2: Autoencoders, convolutional neural networks and recurrent neural networks.* Google Brain, 1-20.

Lecun, Y. (1985). Une procedure d'apprentissage ponr reseau a seuil asymetrique. *Proceedings of Cognitiva 85*, (pp. 599-604).

Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521(7553)*, p. 436.

Lee, E., Kim, Y., Kim, N., & Kang, D. (2017). Deep into the brain: artificial intelligence in stroke imaging. *Journal of stroke, 19(3)*, p. 277.

Lee, Y., Chen, J., Tseng, C., & Lai, S. (2016, September). Accurate and robust face recognition from RGB-D images with a deep learning approach., (p. 123).

Linnainmaa, S. (1970). *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors. Master's Thesis (in Finnish).* Helsinski: Univ. Helsinki, 6-7.

Marr D., & Hildreth, E. (1980). Theory of edge detection, Proc. Roy.Soc. London B-207., (p. 187}217).

Mason, S., & Balisavias, E. (1997). Image-based reconstruction of informal settlements. *Automatic extraction of man-made objects from aerial and space images* (pp. 97-108). Birkhäuser, Basel.

Mass, H., & Vosselman, G. (1999). Two algorithms for extracting building models from raw laser altimetry data. *ISPRS Journal of photogrammetry and remote sensing, 54(2-3)*, pp. 153-163.

MatConvNet. (n.d.). *MatConvNet: CNNs for MATLAB*. Retrieved from http://www.vlfeat.org/matconvnet/

McClelland, J., Rumelhart, D., & PDP Reserach Group. (1987). *Parallel distributed processing (Vol. 1). Cambridge, MA::.* MIT press.

Mineo, C., Pierce, S., & Summan, R. (2019). Novel algorithms for 3D surface point cloud boundary detection and edge reconstruction. *Journal of Computational Design and Engineering, 6(1)*, pp. 81-91.

Minsky, M., & Papert, S. (2017). *Perceptrons: An introduction to computational geometry.* MIT press.

Mozer, M. C. (1995). *A focused backpropagation algorithm for temporal. Backpropagation: Theory, architectures, and applications, 137.*

Nicholson, C. (2019, 09 05). *A Beginner's Guide to Neural Networks and Deep Learning*. Retrieved from https://skymind.ai/wiki/neural-network

Nielsen, M. (2019, June). *How the backpropagation algorithm works*. Retrieved from neuralnetworksanddeeplearning: http://neuralnetworksanddeeplearning.com/chap2.html

Ok, A., Senaras, C., & Yuksel, B. (2012). Automated detection of arbitrarily shaped buildings in complex environments from monocular VHR optical satellite imagery. *Neural computation, 12(10),*, pp. 2385-2404.

Ortner, M., Descombes, X., & Zerubia, J. (2007). Building outline extraction from digital elevation models using marked point processes. In *International Journal of Computer Vision, 72(2),* (pp. 107-132).

Panchai, G., Ganatra, A., Shah, P., & Panchal, D. (2011). Determination of over-learning and over-fitting problem in back propagation neural network. *International Journal on Soft Computing, 2(2)*, pp. 40-51.

Paparoditis, M., Cord , M., & Corquerez, J. (1998, November). Building Detection and Reconstruction fromMid- and High-Resolution Aerial Imagery. *COMPUTER VISION AND IMAGE UNDERSTANDING*, pp. 122-142.

Peng, J., & Liu, Y. (2004). The role of context and model in urban aerial image interpretation focusing on buildings. *IEEE International Conference on Networking, Sensing and Control, IEEE*, (pp. (Vol. 1, pp. 1-12).

Prager, J. (1980). Extracting and labeling boundary segments in natural scenes. In *IEEE Transactions on Pattern Analysis and Machine Intelligence, (1)* (pp. 16-27).

Prager, J. M. (1980). Extracting and labeling boundary segments in natural scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence, (1),*, pp. 16-27.

Ramachandran, P., Zoph, B., & Le, Q. (2017). Searching for activation functions. *preprint arXiv:1710.05941*.

Ramiya, A., Nidamanuri, R., & Krishnan, R. (2017). Segmentation based building detection approach from LiDAR point cloud. *The Egyptian Journal of Remote Sensing and Space Science, 20(1)*, pp. 71-77.

Riedmiller, M., & Rprop, I. (1994). Rprop-description and implementation details.

Rottensteiner, F. &. (2003). In A. g. images.

Rottensteiner, F., & Briese, C. (2002). A new method for building extraction in urban areas from high-resolution LIDAR data. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences (Vol. 34, No. 3/A)*, p. 295.

Rumelhart, D. E. (1985). *Learning internal representations by error propagation (No. ICS-8506). California Univ San Diego La Jolla Inst for Cognitive Science.*

Rumelhart, D., Hinton, G., & Williams, R. (1985). *Learning internal representations by error propagation (No. ICS-8506).* California Univ San Diego La Jolla Inst for Cognitive Science.

Schmidhuber, J. (1996). *The neural heat exchanger.*

Schmidhuber, J. (2015). *Deep learning in neural networks. An overview.*

Shanno, D. F. (1970). Conditioning of quasi-Newton methods for function minimization. *Mathematics of computation, 24(111),*, pp. 647-656.

Sharma, S. (2019, 09 4). *Activation Functions in Neural Networks*. Retrieved from https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Shavlik, J., & Towell, G. (1989). *COMBINING EXPLANATION-BASED AND NEURAL LEARNING: AN ALGORITHM AND EMPmiCAL RESULTS.*

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:*, p. 1409.1556.

Sohn, G. &., Sohn, G., & Dowman, I. (2007). Data fusion of high-resolution satellite imagery and LiDAR data for automatic building extraction. In *ISPRS Journal of Photogrammetry and Remote Sensing, 62(1),* (pp. 43-63).

Sonka M., Hlavac, V., & Boyle, R. (2014). *Image processing, analysis, and machine vision.* Cengage Learning.

Speelpenning, B. (1980). *Compiling fast partial derivatives of functions given by algorithms (No. COO-2383-0063; UILU-ENG-80-1702; UIUCDCS-R-80-1002).* Urbana: Illinois Univ., Urbana (USA). Dept. of Computer Science.

Stoer, J., Bauer, F., & Bulirsch, R. (1989). *Numerische Mathematik (Vol. 5). Berlin: Springer-Verlag.*

Tang T.A, Mhamdi, L., McLernon, D., Zaidi, S., & Ghgho, M. (2016). Deep learning approach for network intrusion detection in software defined networking. *International Conference on Wireless Networks and Mobile Communications.*

Vakalopoulou, M., Karantzalos, K., Komodaki, N., & Paragios, N. (2015). Building detection in very high resolution multispectral data with deep learning features. *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, (p. 1873).

Vassilas N., Charou, Petsa, & Grammatikopoulos. (2013). Intelligent pattern recognition techniques for the development of multimodal representation of urban areas.

Vassilas, N., Tsenoglou, T., & Ghazanfarpour, D. (2015). Mean shift-based preprocessing methodology for improved 3D buildings reconstruction. *WASET Int. J. Civ. Environ. Struct. Constr. Architectural Eng, 9(5)*, pp. 575-580.

Vassillas N., Charou, Petsa, & Grammatikopoulos. (2013). Intelligent pattern recognition techniques for the development of multimodal representation of urban areas.

Villena, S., Abad, J., Molina, R., & Katsaggelos, A. (2004, October). Estimation of high resolution images and registration parameters from low resolution observations. *Iberoamerican Congress on Pattern Recognition . Springer, Berli*.

Wang, Y., Armstrong, R., & Mostaghimmi, P. (2019). Enhancing resolution of digital rock images with Super Resolution Convolutional Neural Networks. *Journal of Petroleum Science and Engineering, 106261.*

Werbos, P. J. (1982). Applications of advances in nonlinear sensitivity analysis. *System modeling and optimization (pp.). Springer, Berlin, Heidelberg.*, pp. 762-770.

West, A., & Saad, D. (1996). Adaptive back-propagation in on-line learning of multilayer networks. *Advances in Neural Information Processing Systems*, pp. 323-329.

Yang, J., Wright, J., Huang, T., & Mia, Y. (2010). Image super-resolution via sparse representation. *IEEE transactions on image processing, 19(11)*, pp. 2861-2873.

Yuan, J. (2017). Learning building extraction in aerial scenes with convolutional networks. *IEEE transactions on pattern analysis and machine intelligence, 40(11)*, pp. 2793-2798.