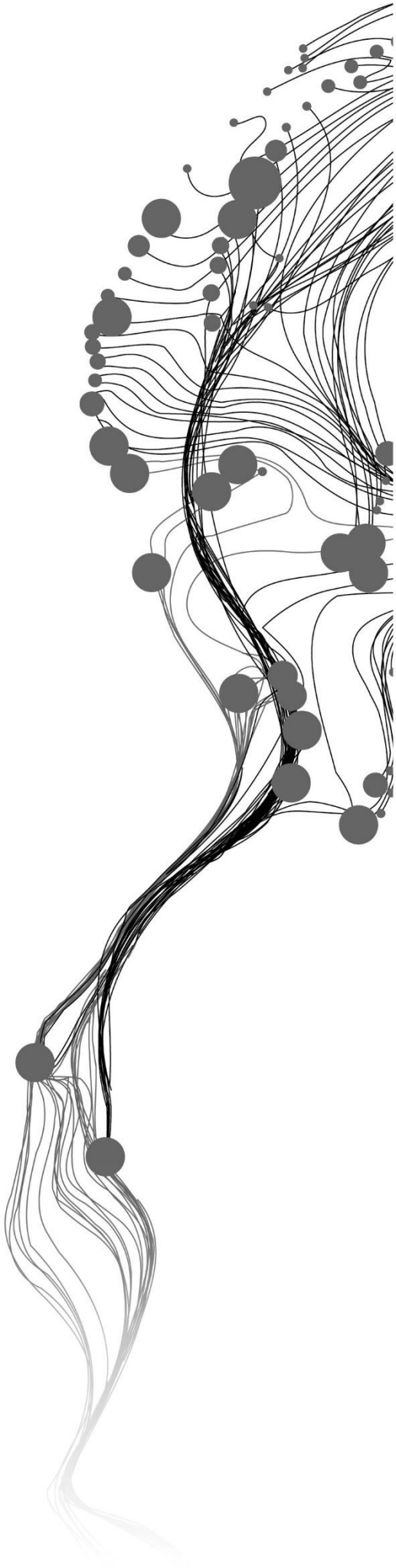
DENSE IMAGE MATCHING USING CONVOLUTIONAL NEURAL NETWORKS

KIMEU J. MAMBA
June 2020

SUPERVISORS:
dr. M. Yang
dr. F. Nex

ADVISOR:
Yaping Lin MSc.



DENSE IMAGE MATCHING USING CONVOLUTIONAL NEURAL NETWORKS

KIMEU, J. MAMBA
Enschede, The Netherlands, June, 2020

Thesis submitted to the Faculty of Geo-Information Science and Earth Observation of the University of Twente in partial fulfilment of the requirements for the degree of Master of Science in Geo-information Science and Earth Observation.
Specialization: Geoinformatics

SUPERVISORS:
dr. M. Yang
dr. F. Nex

ADVISOR:
Yaping Lin MSc.

THESIS ASSESSMENT BOARD:
Prof. dr. Ir. M.G. Vosselman (Chair)
dr. F. Remondino (External Examiner, Bruno Kessler Foundation, 3DOM Research Unit, Italy)

ABSTRACT

Demand for 3D geospatial products such as digital surface models (DSM) has increased rapidly over the past decades. They are finding applications in areas such as urban modelling, planning, construction and building, environmental mapping etc. Ground surveying, stereo-photogrammetry, and Airborne Laser Scanning (ALS) are some of the methods that have long been used to derive these products, but they are expensive and time-consuming. The emergence of Unmanned Aerial Vehicles (UAVs) platforms as a cost-effective mode of capturing aerial imagery has gained popularity in the field of geospatial engineering and remote sensing. Researchers and professionals are utilizing UAV systems to generate high-resolution 3D models for different applications. The automatic and fast generation of high-resolution 3D information presents an efficient and reliable alternative to the traditional (hand-crafted) methods. In this study, we developed a methodology to generate digital surface models using convolutional neural networks (CNNs). CNN's have been widely studied in computer vision for object recognition and segmentation tasks. They have also been applied in classification and stereo matching tasks and have shown to outperform hand-crafted methods due to their capability to learn high-level features. In this work, we designed a CNN architecture, optimized its parameters, and trained it end-to-end for disparity estimation using UAV imagery. We then recovered the 3D scenes from the disparity images and generated digital surface models. We compared our products with the state-of-the-art Pix4D software that relies on hand-crafted features. The results show that deep learning approaches are extremely fast than hand-crafted methods. On the quality of the products, our experimental results show our method generates a high-density point cloud that enables full recovery of scenes. Our DSM has smooth and sharp edges that resolve details in objects and structures. On comparison, we conclude that deep learning methods are quite fast and perform almost at par with conventional methods and have potential.

Keywords: convolutional neural networks, unmanned aerial vehicles, digital surface models, deep learning, dense image matching, computer vision, LiDAR

ACKNOWLEDGEMENTS

I wish to thank my supervisors, Dr. M. Yang and Dr. F. Nex, and my advisor Yaping Lin for their support, feedback, discussions, and insights throughout the entire research period. Sincere appreciation to all the staff at ITC Faculty for the knowledge, skills, and experiences I acquired from you all. I salute my GFM colleagues with whom we have walked the journey together. To all the new friends I made, I will always cherish the moments we shared.

Special thanks to my family and relatives for their support and concern through phone calls, Skype calls, messages, and prayers. I will always be indebted to you.

Lastly, I wish to thank NUFFIC for financial support through the OKP scholarship. It is through this opportunity I have fulfilled my desire to further my studies in Engineering. I hope you will continue to support many more.

" The starting point of all achievement is desire" ~ Napoleon Hill

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Motivation and Problem statement

3D modeling provides a potential benefit in many geospatial applications. Digital surface models (DSM) are increasingly used in many applications such as urban modeling and planning (Wang & Li (2007); Yan, Shaker, & El-Ashmawy (2015); Chai (2017)), disaster management (Bandrova, Zlatanova, & Konecny, 2012), building detection (Rottensteiner, Trinder, Clode, & Kubik 2005) and environmental mapping (Sadeghi, St-Onge, Leblon, & Simard, 2016). The automatic generation of accurate 3D models with complete building shapes and geometries is therefore crucial. Airborne Laser Scanning (ALS), Interferometric Synthetic Aperture Radar (InSAR), stereo-photogrammetry, and ground surveying are the most popular methods used for generating high-resolution 3D information (Jacobsen, 2003). LiDAR and ground surveying techniques have the advantage that they generate high quality and detailed 3D information but are costly and time-consuming (Sefercik, 2013). Compared to optical imagery, SAR imagery can be captured under different weather conditions and during day and night. However, due to its side-looking imaging system, it doesn't capture buildings well and therefore leads to poor reconstructions. Photogrammetric stereo matching methods generate high-resolution 3D information especially when images have no background variations. Background variations such as homogenous areas, occlusions, texture, and illumination changes cause matching errors that lead to poor reconstructions of objects in the scene.

The availability of Unmanned Aerial Vehicles (UAVs) platforms has led to fast and cost-effective image acquisition techniques. These systems capture high-resolution images of a scene from different viewpoints thereby minimizing occlusions which pose a challenge in the reconstruction of complex scenes from images (Gerke, Nex, & Jende, 2016). This presents a potential benefit in the generation of 3D models from 2D images. However, UAV images may suffer from complex background variations which may affect 3D mapping. Photogrammetric methods derive 3D products through stereo image matching techniques. Image matching is a challenging task, especially when using images with background variations such as illumination and viewpoint changes, occlusions, and texture-less regions. These variations make it difficult to find correspondences in regions affected. They lead to noise in disparity maps that impacts the quality of the 3D products.

Traditional dense stereo-matching methods consist of matching cost computation, cost aggregation, disparity computation, and disparity refinement (Scharstein & Szeliski, 2002). They are based on hand-crafted features that rely on set parameters and cannot be easily adapted and hence lack flexibility (Verdie, Yi, Fua, & Lepetit, 2015). To estimate disparity, matching constraints such as window-based correlation, smoothing and occlusion are applied (Kanade & Okutomi (1994); (Jang & Ho (2011)). However, the challenge is that these constraints cannot be handcrafted. Window-based algorithms such as sum-of-squared-differences compute disparity based on the intensity values within a window and usually make implicit smoothness assumptions. Global algorithms, on the other hand, make an explicit smoothness assumption and solve an optimization problem. They don't perform the aggregation step and instead compute a disparity that minimizes global cost function (Szeliski, 2010). For real-time applications, global optimization is however not possible. Different cost aggregation strategies are applied. For instance, Semi-Global Matching (SGM) applies 1D cost aggregations to simulate a 2D optimization problem (Hirschmüller, 2008) while Graph Cut (Boykov & Jolly, 2001) employs a graph model to minimize energy in a 2D neighborhood.

Recently, deep learning approaches have successfully been applied in stereo-matching and have shown to outperform traditional methods. These methods consist of network layers that learn multi-level representations from input data hierarchically and enable predictions to be made (Lecun, Bengio, & Hinton, 2015). These multi-level features learned by CNNs have proved to be effective than hand-crafted ones. The powerful learning ability of deep convolutional neural networks has led to their popularity in stereo-matching and classification. A review of deep learning methods for 3D reconstruction was done by (Han, Laga, & Bennamoun, 2019). The study highlights the different network architectures and areas they have been applied.

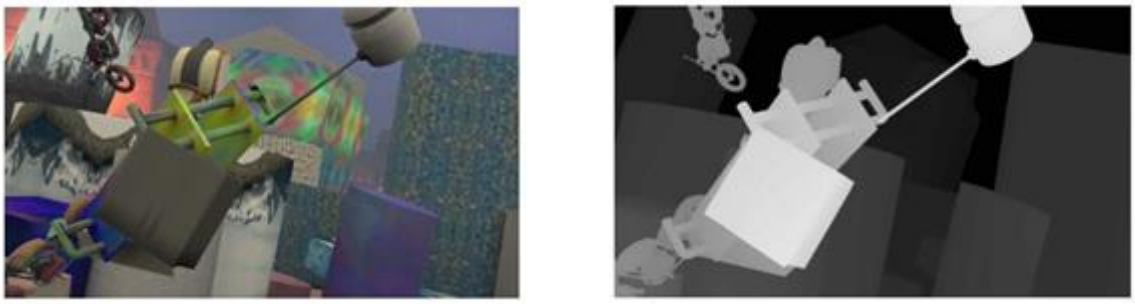
Various studies have proposed architectures that estimate disparity for single images such as (Kuznietsov, Stückler, & Leibe (2017); Eigen, Puhrsch, & Fergus (2014)) while others such as (Yang, Manela, Happold, & Ramanan (2019); Žbontar & Lecun (2016)) implement CNNs using stereo image pairs. The above architectures have been implemented in a supervised way where a vast amount of ground truth labels is used during training. However, the need to acquire labeled data for supervised learning is quite challenging especially for realistic image datasets such as aerial imagery. Some studies have implemented unsupervised learning methods to train networks and use ground truth to evaluate the accuracy of the results. Unsupervised learning methods such as (Garg, Vijay Kumar, Carneiro, & Reid (2016); Godard, Mac Aodha, & Brostow (2017); Zhong, Dai, & Li (2017) & Mahjourian, Wicke, & Jun (2017)) leverage constraints of stereo geometry with image warping loss function to estimate disparity. These studies have shown the capability

of deep learning approaches in stereo matching tasks. However, the methods have been applied in computer vision using close-range images and therefore their suitability in geospatial applications using aerial imagery needs to be investigated.

1.2. Research Identification

The availability of high-resolution UAV images presents benefits as well as challenges in generating 3D map products. Using these images results in accurate and complete models but poses a challenge to algorithms due to the complex variations they suffer from. Approaches that are not affected by these variations are therefore needed. End-to-end CNN architectures can overcome these variations and have recently shown to achieve improved results in dense disparity and flow estimation. They consist of encoder that extracts features from the input and a decoder that up samples feature maps from the encoder to the same resolution as input images and make predictions.

Both supervised and unsupervised learning methods have shown good results in stereo matching tasks. A major drawback in supervised learning is the need to acquire a large amount of labeled data for training (Garg et al., 2016). In areas where there is no labeled data for training or the available ground truth is sparse, unsupervised learning methods can be used. A common approach mostly applied in deep learning is initially training on large synthetic datasets such as Scene flow (Mayer et al., 2016) and fine-tuning on real-world target dataset. Nevertheless, there is a challenge of overfitting, where a network trained on synthetic datasets performs poorly when tested on realistic data. This means that the network has low generalization ability. In this study, we investigate the suitability of deep learning stereo matching methods on aerial images. We develop a methodology for 3D reconstruction using UAV images based on CNN by optimizing a pre-trained network. A comparison with hand-crafted methods will be done to evaluate the performance of deep learning approaches. **Figure 1-1** below shows part of SceneFlow dataset.



(a) (b)

Figure 1-1: Scene Flow dataset (a) FlyingThings3D image and (b) Corresponding dense ground truth by (Mayer et al., 2016)

1.2.1. Research objectives

The main objective of this study is to investigate deep learning approaches for 3D reconstruction using UAV imagery. The following sub-objectives are derived.

1. Review convolutional neural networks as applied in stereo matching.
2. Develop a methodology for disparity estimation from UAV images.
3. Reconstruct the scene using the estimated disparity images.
4. Compare the performance of CNN on aerial imagery with handcrafted methods.

1.2.2. Research questions

The following research questions are addressed.

Specific objective 1:

i. What are the existing networks that have been applied for stereo matching? And what network architecture is suitable?

Specific objective 2:

i. Do pre-trained CNN models have generalization ability that enables transfer learning to target dataset?
ii. What parameters are to be considered when adapting a CNN model for a specific task?

Specific objective 3:

i. What is the quality of the point cloud and DSM?

Specific objective 4:

i. What is the performance of deep learning approaches compared to handcrafted methods?

1.3. Innovation

The study applies state-of-the-art deep learning approach in generating a digital surface model of a scene captured using high-resolution images. Considering the challenges high-resolution imagery poses to matching algorithms, this indeed is an innovation geared towards accurate 3D geoinformation using cost-effective imagery.

1.4. Thesis structure

The thesis consists of six chapters. Chapter 1 presents the motivation of the work and problem statement, research identification, research objectives, and questions. Chapter 2 presents an overview of stereo matching, convolutional neural networks, and related work. Chapter 3 describes the methodology followed in the study. In chapter 4, we describe the data used, experiments conducted, results obtained and their analysis. Chapter 5 presents the discussions of the results. Lastly, chapter 6 presents conclusions drawn from the study and recommendations for further work.

2. LITERATURE REVIEW

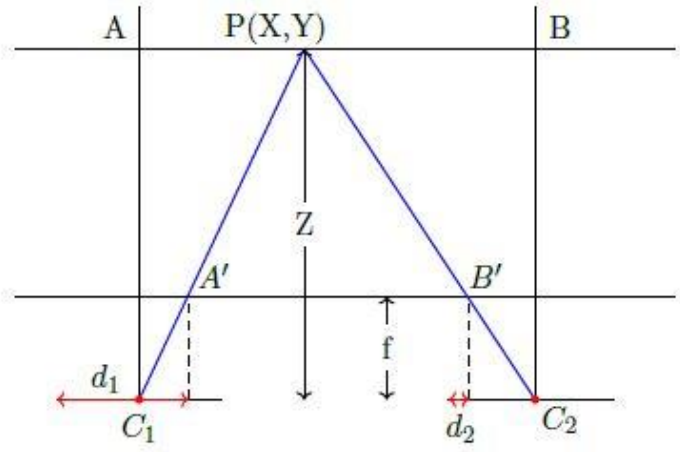
This chapter briefly reviews stereo matching algorithms and CNN as applied in stereo matching. An overview of stereo-vision is presented in section 2.1. A brief introduction to CNNs and related work is presented in 2.2 and 2.3 respectively. Finally, transfer learning is discussed in section 2.4.

2.1. Overview of stereo matching

Stereo vision is the process of recovering the structure of a scene from two or more images by matching pixels and estimating a 3D point cloud from their 2D positions (Szeliski, 2010). Given stereo image pairs the relative distance from a point P to the focal plane can be recovered using equation (2.1) below.

$$Z = f\frac{B}{d} \tag{2.1}$$

where B is the baseline $i.e$ the distance between the two camera centers, d is the disparity, f is the focal length and Z is the corresponding depth. **Figure 2-1** below shows how to recover the 3D position of a point P from two camera positions C_1 and C_2.



AB = Baseline
P = Point on ground
C_1 = Camera 1 center
C_2 = Camera 2 center
f = focal length
A' = Projection of P on image 1
B' = Projection of P on image 2
Z = Distance to the camera plane

Figure 2-1: Geometry of stereo vision

To obtain depth information per pixel, objects captured in the scene should be visible in images from both cameras. This results in a dense depth map. However, this process is computationally demanding. Additionally, occlusions and texture-less surfaces result in poor point matches in both images, therefore, making it difficult to compute disparities. In such cases the resultant depth maps are sparse and the quality of the 3D point cloud is affected. To avoid errors in the estimation of depth, the correspondences between pixels in both images should be

unique, free of noise, illumination changes, and geometric differences. By exploiting epipolar geometry, the correspondence search between the stereo pair is reduced to one dimensional. This speeds up the disparity computation process.

Dense correspondence consists of four major steps – matching cost computation, cost aggregation, disparity computation, and disparity refinement. The computation of matching cost involves the determination of a similarity measure between pixels. Sum of squared differences (SSD) (Hannah, 1974) and the sum of absolute intensity differences (SAD) (Takeo Kanade, 1994) are some of the most common pixel-based matching costs. Although these algorithms are less complex, their drawback is the assumption of constant disparity within the same window. Other traditional matching costs are normalized cross-correlation (Hannah, 1974) and binary matching costs (Marr & Poggio, 1976) based on binary features such as edges. Binary matching costs have low discriminability and hence are no longer used in dense stereo matching.

Cost aggregation employs local or global information to reduce matching uncertainties. Local and window-based cost aggregation methods aggregate cost by summing or averaging over a support region in the disparity space. Two-dimensional aggregation methods such as shifting windows (Fusiello, Roberto, & Trucco, 1997) where for each pixel, the correlation with nine different windows and disparity with the smallest sum of squared differences is retained and sliding a low-pass filter overall target regions have been used. Three-dimensional aggregation methods such as limited disparity difference (Grimson, 1985) and limited disparity gradient (Pollard, Mayhew, & Frisby, 1985) have been proposed. Shah (1993) proposes another method of aggregation – iterative diffusion, implemented by repeatedly adding the weighted values of neighboring pixels costs to the cost of a pixel.

Local disparity computation methods emphasize on matching cost computation and cost aggregation. They perform a local winner take all (WTA) optimization at each pixel. Their drawback is that the uniqueness of matches is only enforced for the reference image while points in the other image might match multiple points (Scharstein & Szeliski, 2002). Global methods, on the other hand, perform optimization after disparity computation and often skip the aggregation step. They are formulated in an energy minimization framework and try to minimize a global energy function given by the equation below.

$$E(d) = E_d(d) + \lambda E_s(d) \tag{2.2}$$

where $E_d(d)$ is the initial matching cost and $E_s(d)$ is a smoothness term weighted by λ. The value λ encourages neighboring pixels to have similar disparities based on the assumption that the disparity map is locally smooth. After defining this global energy, various algorithms can be used

to compute the local minimum. Algorithms such as graph cut and belief propagation (Liang, Cheng, Lai, Chen, & Chen, 2011) have shown to produce good results as compared to traditional algorithms such as simulated annealing (Geman & Geman, 1984). As in local methods, refinement is also done to smoothen the disparity map.

Post-processing of the computed disparities is crucial to get rid of mismatches. This is done in the refinement step where disparities are refined to reduce noise by filtering inconsistent pixels. Comparing left-to-right and right-to-left disparity maps (left-right consistency check) is applied to detect occluded areas (Fua, 1993). Noise reduction approaches such as median filter and Gaussian convolution can be applied. The median filter approach cleans up mismatches. In Gaussian convolution weights of neighboring pixels are given by a Gaussian distribution. Surface fitting can be applied to fill holes due to occlusions or by distributing neighboring disparity estimates (Hirschmuller & Scharstein, 2009). Another post-processing technique is associating confidence with per-pixel depth estimates by looking at the curvature of the correlation surface although it's useful when applied in later processing stages.

2.2. Review of Convolutional Neural Networks

Convolutional neural networks are networks that are widely used for processing images and time-series data. They consist of neurons organized in three dimensions, the spatial dimensionality of input image i.e. height and width, and depth. A summing operation applied to inputs to the neurons results in linear output. A neuron's output f is modeled as a function of its input x using activations. The most commonly used activations are hyperbolic tangent given as $f(x) = \tanh(x)$ and sigmoidal given as $f(x) = (1 + e^{-x})$. These saturating activations take a longer time during network training with gradient descent than the non-saturating rectified linear units (ReLUs) with a function $f(x) = \max(0, x)$ (Krizhevsky, Sutskever, & Hinton, 2012). **Figure 2-2** below shows some of the most commonly used activation functions.

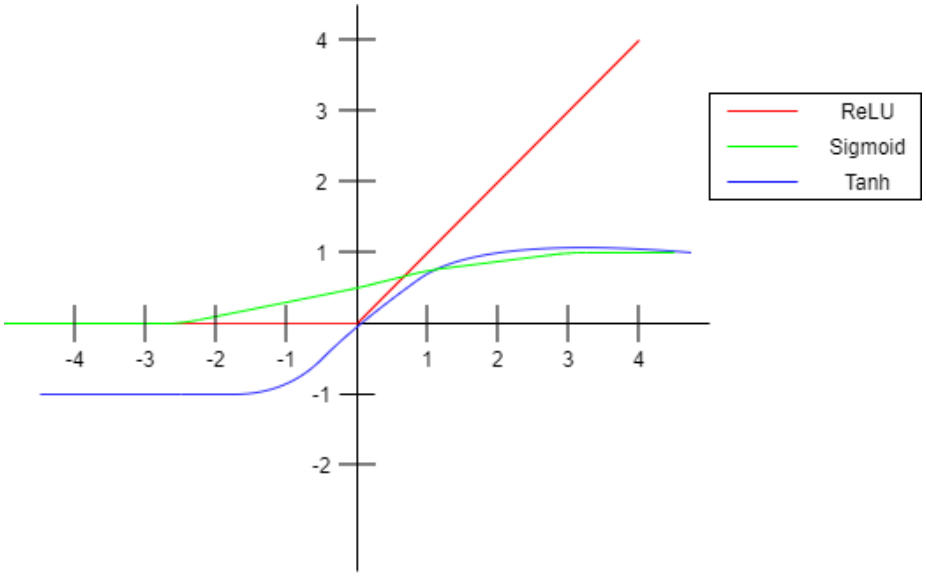
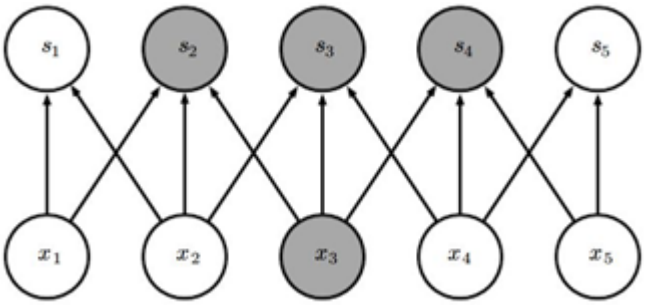


Figure 2-2: Most commonly used activation functions in CNNs

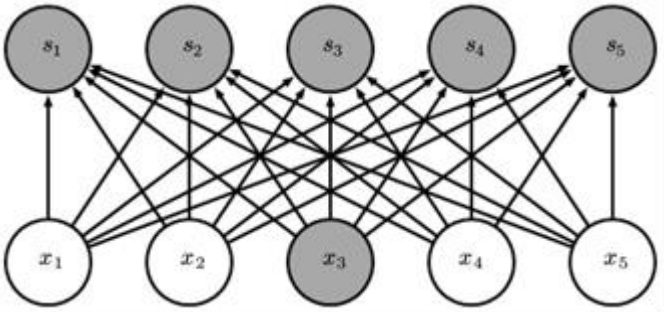
CNN's consist of convolutional, pooling, and fully connected layers. Convolutional layers consist of layers of learnable filters that convolve across the height and width of the input volume and compute dot products between filter elements and input generating a 2D activation map. For example, the kth output feature map y_k in a given convolutional layer is computed as shown by equation (2.3) below.

$$y_k = f(w_k * x) \tag{2.3}$$

where $f(.)$ is the non-linear activation function, x is the input image, w_k is the convolutional filter of kth feature map, and $*$ denotes the 2D convolution operator. Using filters with a smaller dimension than the input reduces the number of connections and number of parameters to be computed resulting in sparse connectivity (**Figure 2-3** below). The same set of weights are learned for each location in the input in a given layer. This means that the parameters are shared during convolutions. Fully connected layers lack these two properties. As shown in **Figure 2-4**, units in a fully connected layer are connected to subsequent layers, and parameters are not shared but instead used only once (Bengio, Goodfellow, & Courville, 2015). Another property of convolution is equivariance that enables features occurring at different locations to be detected effectively. This means that a transformation on the input before applying a function is not changed and still exists even after the function is applied (Bengio et al., 2015).



(a)



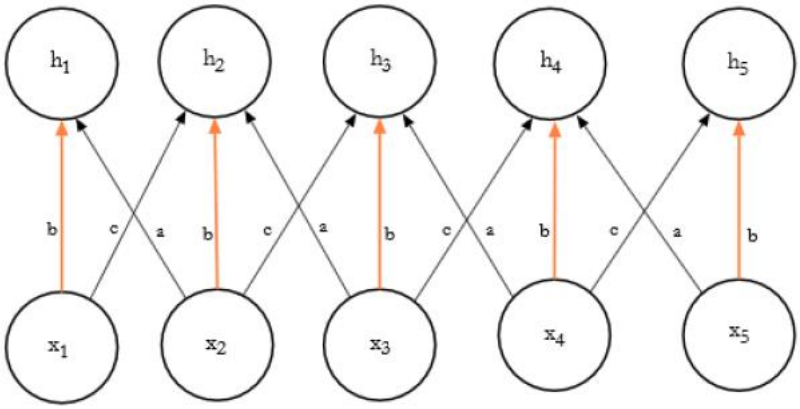
(b)

Figure 2-3: Sparse Connectivity (a) Convolution with a 3x3 kernel, only 3 outputs are affected by $\mathbf{x_3}$ (b)Using matrix multiplication no sparse connectivity and all outputs are affected by $\mathbf{x_3}$. (Adapted from Bengio, Goodfellow, & Courville, 2015)

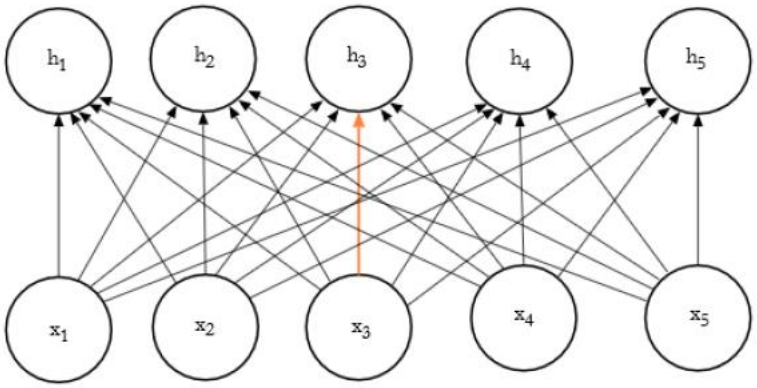
Pooling layers are used for down-sampling along the spatial dimensionality of the input image. It is done through max-pooling and average pooling. Max-pooling takes the maximum over an input region while average pooling returns average over the window considered. Pooling with a stride $S > 1$ down-samples the input image by a factor of S. The effect of pooling by a big factor is that it reduces the spatial dimension of the output resulting in loss of crucial information.

Training CNNs involves feeding the network with inputs x and generating output y through forward propagation. A backpropagation is done to compute gradients of the layer outputs. An optimization procedure is done to update the gradients during backpropagation. Stochastic gradient descent (SGD) is commonly applied. SGD consists of learning rate and momentum. The learning rate controls the rate at which the learning takes place by accelerating training. However, the selection of the value learning rate is quite sensitive. Choosing a larger value may cause the network to start diverging rather than converging. On the other hand, a smaller value may lead to more training time and the network might get stuck at local minima. A common technique is to reduce the learning rate during the training process (Nair & Hinton, 2010). Momentum accelerates the learning with the SGD approach by using the moving average of the gradient instead of the current real value (Bengio et al., 2015). Network training with few

samples leads to overfitting where it loses its ability to generalize to new data. Techniques for preventing overfitting have been proposed such as data augmentation and drop out techniques. Data augmentation involves transforming the dataset through scaling, rotations, flipping, and color transformations. Dropout is also used to prevent overfitting by dropping out some units in the network (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). Layers in a CNN are arranged hierarchically with lower layers extracting low-level features such as edges. Deeper layers extract complex and data specific features and therefore have large receptive fields (Lecun et al., 2015).



(a)



(b)

Figure 2-4: Parameter Sharing (a) Convolution with a 3x3 kernel, due to parameter sharing the central element of the kernel is used in all input locations. (b) In Fully Connected using the central element of the weight matrix, the parameter is used only once. (Adapted from Bengio et al., 2015)

2.3. Related work on stereo matching using CNN

Žbontar & Le Cun, (2016) proposed to compute matching cost by multi-layer representations learned by a CNN (MC-CNN). They train two networks that differ in complexity to predict how well two patches match and initialize the matching cost computation. One of the networks, the fast architecture is a Siamese network that computes a cosine similarity score by comparing

features vectors from both patches. The other network is a slow network with a fully-connected layer that replaces the dot product in the fast architecture. They implement cost aggregation and semi-global matching post-processing to refine the cost and left-right consistency check to eliminate errors in occluded areas. The fast architecture was 80 times faster than the slow one. It also obtained the lowest error rate on the KITTI stereo dataset (A Geiger, Lenz, Stiller, & Urtasun, 2013). **Figure 2-5** shows the architecture of the accurate CNN proposed by (Žbontar & Lecun, 2016).

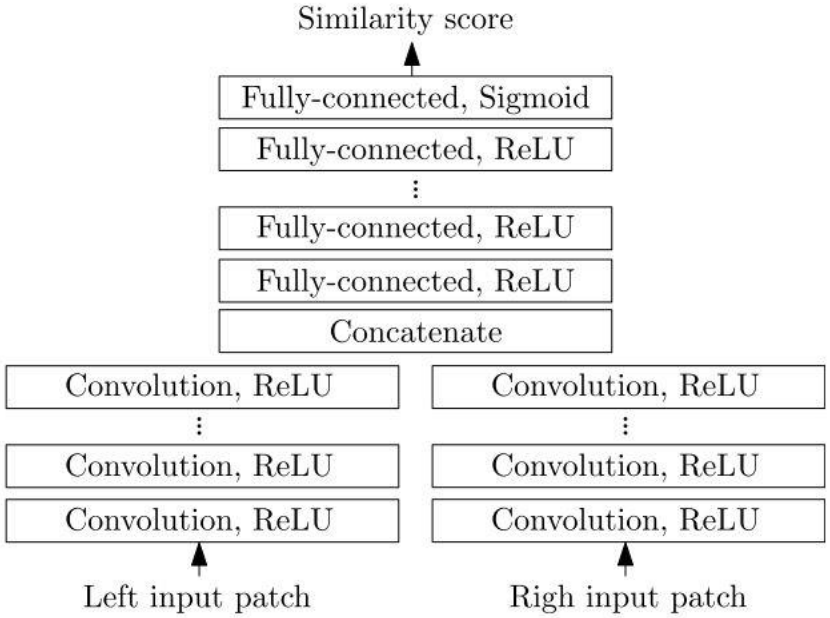
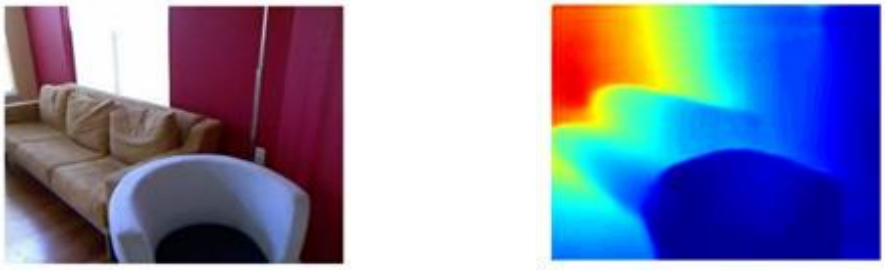


Figure 2-5: Architecture of the accurate network by (Žbontar & Lecun, 2016)

Likewise, Luo, Schwing, & Urtasun, (2016) presented a method (Content-CNN) that employs a Siamese network architecture that exploits a product layer to compute the inner product between the two representations. The network is trained as a multi-class classification problem where possible disparities are treated as classes. Compared to existing approaches, their network presented a better matching performance on challenging datasets in less than a second. Patch-based methods achieve good accuracy but they require exhaustive matching of patches and hence slow. They also don't use the whole image context (Ilg et al., 2017).

End-to-end network architectures on large datasets to predict disparity have been recently proposed. Laina et al. (2016) proposed a fully convolutional network architecture to estimate depth maps from a single image. They investigate popular architectures AlexNet (Krizhevsky et al., 2012), VGG-16 (Simonyan & Zisserman, 2014), and ResNet-50 (He, Zhang, Ren, & Sun, 2016) trained on ILSVRC (Russakovsky et al., 2015) as the encoder using their pre-trained weights. The decoder comprises of un-pooling and convolutional layers that up-sample feature

maps from the contractive part to high-resolution outputs. They apply a data augmentation technique to increase the training samples. The network was evaluated on an indoor NYU depth dataset (Silberman, Hoiem, Kohli, & Fergus, 2012) and an outdoor Make3D dataset (Saxena, Sun, & Ng, 2009) achieving better results and showed high generalization ability. However, networks that estimate depth from single images do not exploit stereopsis hence they have poor generalization ability to unseen datasets (Ummenhofer et al., 2017). **Figure 2-6** shows an NYU dataset image and its corresponding disparity map estimated by (Laina et al., 2016).



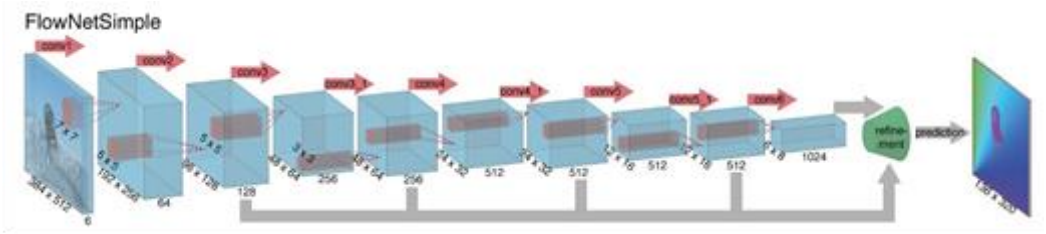
(a) (b)

Figure 2-6: (a) NYU dataset RGB image and (b) Corresponding depth map estimated by the network used in (Laina et al., 2016)

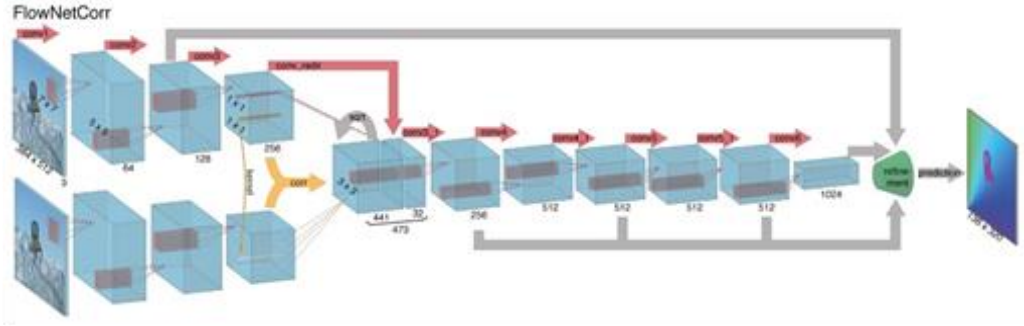
Kendall et al. (2017) proposed an end-to-end architecture (GC-Net) that incorporates geometry and contextual information to estimate disparity from stereo images. They use 2D convolutional operations using 5×5 filters followed by residual blocks of 3×3 filters. The left and right stereo images are fed through these layers to obtain unary features. These unary features are used to compute stereo matching costs by forming a cost volume that enables incorporation of context. They extend the idea of encoder-decoder by using 3-D transposed convolutions in the decoder that make predictions with the same resolution as input. By incorporating context, the method achieves improved accuracy on the challenging KITTI datasets (A Geiger et al., 2013).

Fischer et al., (2015) proposed a network to compute optical flow estimation using CNN. They employ an encoder-decoder architecture that allows concatenation of feature maps from the encoder part in the decoder. The encoder extracts high-level features from the input images while the decoder up convolves the coarse feature maps to full resolution of input images. They propose two network architectures that differ in the complexity as shown in **Figure 2-7** below. The generic network named FlowNetSimple consists of encoder that extracts high-level features from stacked stereo image pairs and a decoder that up samples coarse feature maps to full resolution of input images. This enables fine prediction resulting in accurate disparity maps. The other architecture, FlownetCorr consists of two parallel streams that extract features from left and right image pairs independently. The feature vectors are then compared by a correlation layer

generating a four-dimensional result. For refinement, a decoder that 'up-convolves' the resulting map and concatenates it with feature maps from the encoder is implemented resulting in a fine prediction. The networks are trained on the synthetic Flying chairs dataset and tested on Sintel and KITTI benchmark datasets.



(a)



(b)

Figure 2-7: Architecture of (a) FlowNetS and (b) FlowNetCorr by (Fischer et al., 2015)

Following the idea of FlowNet (Fischer et al., 2015), Mayer et al., (2016) propose a method to estimate disparity and scene flow jointly using CNN. They created a large synthetic dataset with ground truth for training and evaluation. The joint network is achieved by fine-tuning disparity and scene flow pre-trained networks. They performed data augmentation by introducing spatial – rotation, translation scaling, and chromatic – color, contrast, brightness transformations for all images to obtain diversity. Data augmentation techniques such as rotation and translations are not recommended for disparity since they break the epipolar constraint.

2.4. Transfer learning

CNN layers learn general features such as edges and color blobs in the first layers regardless of the cost function and input images. The features are referred to as general since they are independent of dataset and task. Layers on top learn dataset and task-specific features that enable accurate predictions to be made. Transfer learning involves training a base network on a given dataset and task and transferring the features learned to a target dataset and task. This is quite

important especially when the target dataset is small to prevent the network from overfitting (Yosinski, Clune, Bengio, & Lipson, 2014). This strategy requires that the features be suitable to both datasets otherwise the process will fail.

The approach in transfer learning is to apply a pre-trained model directly to a target dataset without parameter tuning. This only works when the dataset the network was trained on is almost similar to the target dataset (Bengio, 2012). Alternatively, the network can be initialized with pre-trained weights and update them by fine-tuning on the new target dataset. This can be done by freezing the base network and training the parameters of the last layers or train all the parameters of the network. However, the choice of whether to fine-tune the last layers or train all parameters of the network depends on the target sample size and the depth of the network (Yosinski et al., 2014). A small target dataset may result in overfitting when fine-tuned on a deeper network. In this case, it is recommended to fine-tune only the last layers and freeze the base network. With the availability of a large target dataset, the whole network can be fine-tuned for improved performance.

The availability of pre-trained models allows the concept of transfer learning to be implemented in many tasks. For depth estimation from stereo images, various pre-trained models are available publicly and can be used for fine-tuning in datasets different from the one they are trained on such as aerial imagery.

3. METHODOLOGY

This chapter describes the set of experiments conducted towards the main objective of generating a digital surface model from UAV images. **Figure 3-1** below shows an overview of the methodology followed in the study. We design a CNN model and train it end-to-end using a synthetic dataset. We directly test the trained model on real-world datasets to test its generalization ability. We then optimize its parameters and train it on real-world datasets. We test our final model and recover scenes from the disparity images. Lastly, we compare our results with conventional methods.

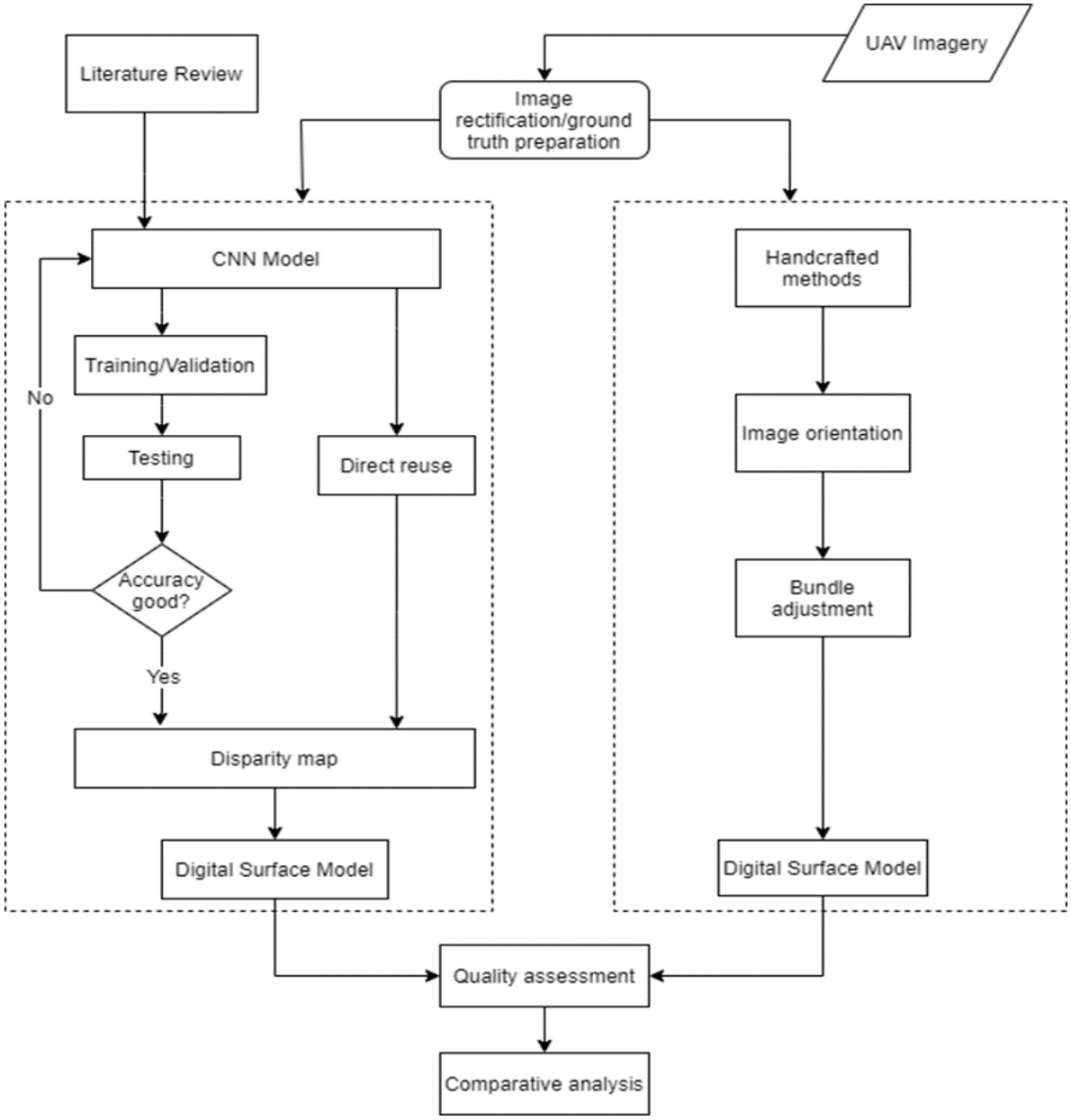


Figure 3-1: Overview of methodology

3.1. CNN architecture

We design a CNN architecture based on the model proposed by (Chang & Chen, 2018). Our model consists of four main modules; feature extraction, spatial pyramid pooling, cost volume, and regularization. Because of the nature of our dataset comprising of diverse features, we chose this architecture with spatial pyramid pooling that enables us to capture and aggregate context information. **Figure 3-2** below shows the general architecture of our CNN. **Table 3.1** describes the final network architecture.

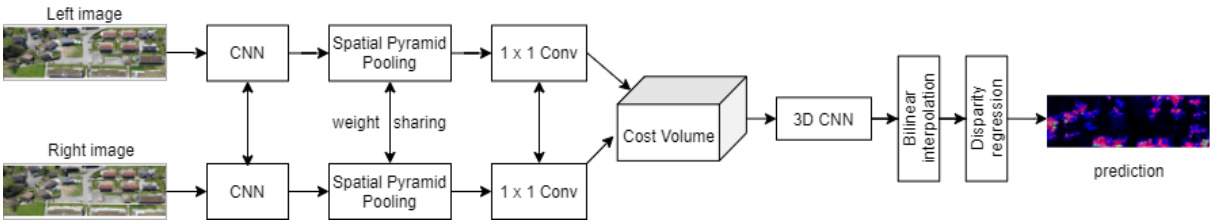


Figure 3-2: General architecture of the designed CNN (Adapted from Chang & Chen, 2018).

3.1.1. Feature extraction

The input left and right stereo pairs of size $H \times W \times 3$ are passed through small cascaded 3×3 filters that downsample them by a factor of two (see **Figure 3-3**). They are followed by residual blocks of ResNet structure as implemented in (He et al., 2016). To capture a larger receptive field, we apply a dilated convolution at the last two blocks of feature extraction. The resulting feature maps are $\frac{1}{4}$ of the input image size. Weights in the parallel CNN and spatial pooling pyramid branches are shared.

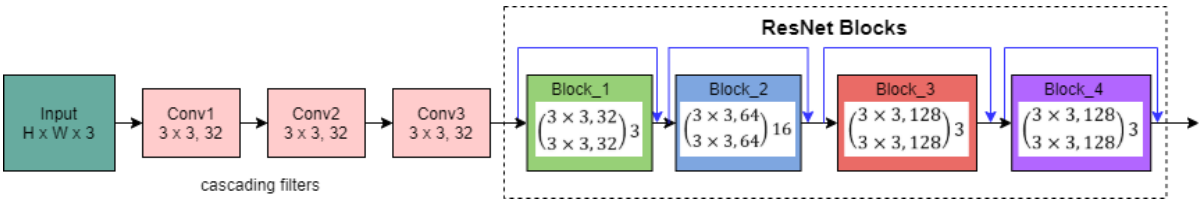


Figure 3-3: Feature extraction module consisting of cascaded filters and ResNet blocks.

A spatial pooling pyramid (SPP) module is applied to the feature maps to capture context information. We design four average pooling blocks of 64×64, 32×32, 32×32, 16×16 followed by 1×1 convolutions. This was motivated by the work in Zhao, Shi, Qi, Wang, & Jia (2017) who observed that average pooling works well than max-pooling while pooling with pyramid parsing outperforms global pooling. The pooled feature maps are then upsampled using bilinear or cubic interpolation to the input feature map size. Feature maps from the four levels are then concatenated to form the final output.

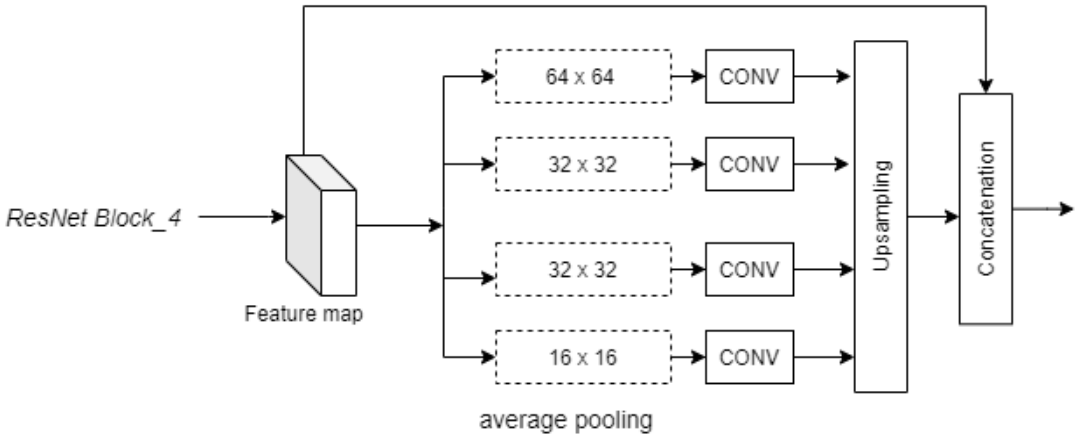


Figure 3-4: Spatial pyramid pooling module with four levels of pyramid parsing

3.1.2. Cost volume

The 3-dimensional feature maps of the left and right images from the SPP module are aggregated to form a 4-dimensional cost volume. The resulting output is a volume of a dimension $H \times W \times D \times N$ where H is the height, W is the width, D is the disparity, and N is the number of filters.

3.1.3. 3D CNN

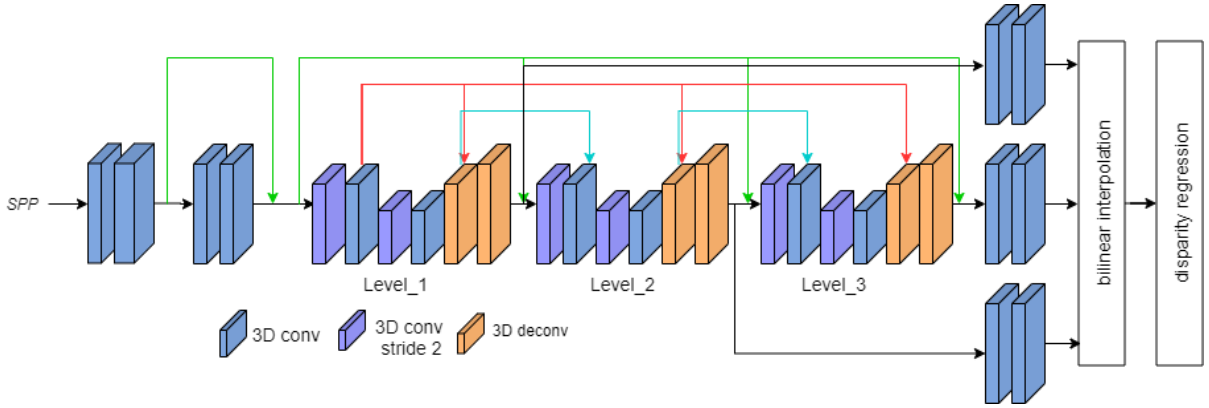


Figure 3-5: Stacked 3D CNN

As shown in **Figure 3-5** above, we use an encoder-decoder network architecture to learn more context features. It aggregates features along spatial and disparity dimensions of the cost volume. As in (Chang & Chen, 2018), we stack three 3D CNNs each generating a disparity map through the disparity regression module. Each of the 3D CNN has a loss. The three losses are summed to get the total loss used during training. The final disparity map is obtained from the three refined disparity maps. In our final implementation, we use the $Smooth\ L_1 Loss$ also known as $Hüber\ Loss$ to train the network. $Hüber\ Loss$ is less sensitive to outliers than mean squared error $MSE\ Loss$, and prevents exploding gradients (Girshick, 2015). The loss function is computed as shown in equation 3.1.

$$L(d,\hat{d}) = \frac{1}{N}\sum_{i=1}^{N} smoothL_1\ (d_i - \hat{d}_i) \tag{3.1}$$

where N is the number of labelled pixels, d is the ground truth disparity, and $\hat{d}$ is the predicted disparity. The disparity regression module computes the disparity of each pixel from the predicted cost C_d as the sum of each disparity d weighted by its probability using a softmax operation σ. Equation 3.2 below shows the disparity regression.

$$\hat{d} = \sum_{d=0}^{D} d \times \sigma(-C_d) \tag{3.2}$$

3.1.4. Training procedures

We implement a common strategy to train our designed model. First, we train the model on a synthetic dataset from scratch using Adam optimizer for 20 epochs at a learning rate of 0.001. We then fine-tune the trained model using the UAV target dataset. In finetuning we first aim to determine the optimal parameters that we will use to train our model. We conduct a parameter optimization experiment using different parameters that affect the learning and regularization of deep networks. We focus on the parameters that influence the quality of disparity maps and incorporate the KITTI dataset which has a denser ground truth than ours. We tune these parameters over a sample of 400 image pairs (training and validation). The optimal parameters determined are used in our fine-tuning experiment (see 4.2.2). Further, we investigate the effect of varying training set in transfer learning by using different sample sizes of the training set (see 4.2.3).

Table 3.1: Final network architecture

Layer name	Description(shape,filters,dilation,stride)	Output
Input		H x W x 3
Conv1	$3 \times 3, 32$	$\frac{1}{2}$ H x $\frac{1}{2}$W x 32
Conv2	$3 \times 3, 32$	$\frac{1}{2}$ H x $\frac{1}{2}$W x 32
Conv3	$3 \times 3, 32$	$\frac{1}{2}$ H x $\frac{1}{2}$W x 32
Conv_stack_1	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 3$	$\frac{1}{2}$ H x $\frac{1}{2}$W x 32
Conv_stack_2	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 16$	$\frac{1}{4}$ H x $\frac{1}{4}$W x 64

Conv_stack_3	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 3, d = 2$	$\frac{1}{4}$ H x $\frac{1}{4}$W x 128
Conv_stack_4	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 3, d = 2$	$\frac{1}{4}$ H x $\frac{1}{4}$W x 128
SPP1	64×64 *average pooling* $3 \times 3, 32$ Bilinear interpolation	$\frac{1}{4}$ H x $\frac{1}{4}$W x 32
SPP2	32×32 *average pooling* $3 \times 3, 32$ Bilinear interpolation	$\frac{1}{4}$ H x $\frac{1}{4}$W x 32
SPP3	32×32 *average pooling* $3 \times 3, 32$ Bilinear interpolation	$\frac{1}{4}$ H x $\frac{1}{4}$W x 32
SPP4	16×16 *avearge pooling* $3 \times 3, 32$ Bilinear interpolation	$\frac{1}{4}$ H x $\frac{1}{4}$W x 32
concatenation	conv_stack_2, conv_stack_4, SPP1, SPP2, SPP3, SPP4	$\frac{1}{4}$ H x $\frac{1}{4}$W x 32
fusion	$3 \times 3, 128$ $1 \times 1, 32$	$\frac{1}{4}$ H x $\frac{1}{4}$W x 32
Cost volume	Concatenated left and right 3D volumes	$\frac{1}{4}$ H x $\frac{1}{4}$W x $\frac{1}{4}D$ X 32
3Dconv0	$3 \times 3 \times 3, 32$ $3 \times 3 \times 3, 32$	$\frac{1}{4}$ H x $\frac{1}{4}$W x $\frac{1}{4}D$ X 32
3Dconv1	$3 \times 3 \times 3, 32$ $3 \times 3 \times 3, 32$	$\frac{1}{4}$ H x $\frac{1}{4}$W x $\frac{1}{4}D$ X 32
3Dstack1_1	$\begin{bmatrix} 3\times3\times3, 64 \\ 3\times3\times3, 64 \end{bmatrix}, stride = 2$	$\frac{1}{8}$ H x $\frac{1}{8}$W x $\frac{1}{8}D$ X 64
3Dstack1_2	$3 \times 3 \times 3, 64$ $3 \times 3 \times 3, 64$	$\frac{1}{16}$ H x $\frac{1}{16}$W x $\frac{1}{16}D$ X 64
3Dstack1_3	$Deconv\ [3 \times 3 \times 3, 64], stride = 2$ add 3Dstack1_1	$\frac{1}{8}$ H x $\frac{1}{8}$W x $\frac{1}{8}D$ X 64
3Dstack1_4	$Deconv\ 3 \times 3 \times 3, 32$ add 3Dconv1	$\frac{1}{4}$ H x $\frac{1}{4}$W x $\frac{1}{4}D$ X 32
3Dstack2_1	$\begin{bmatrix} 3\times3\times3, 64 \\ 3\times3\times3, 64 \end{bmatrix}, stride = 2$ add 3Dstack1_3	$\frac{1}{8}$ H x $\frac{1}{8}$W x $\frac{1}{8}D$ X 64
3Dstack2_2	$3 \times 3 \times 3, 64$ $3 \times 3 \times 3, 64$	$\frac{1}{16}$ H x $\frac{1}{16}$W x $\frac{1}{16}D$ X 64
3Dstack2_3	$Deconv\ [3 \times 3 \times 3, 64], stride = 2$ add 3Dstack1_1	$\frac{1}{8}$ H x $\frac{1}{8}$W x $\frac{1}{8}D$ X 64
3Dstack2_4	$Deconv\ 3 \times 3 \times 3, 32$	$\frac{1}{4}$ H x $\frac{1}{4}$W x $\frac{1}{4}D$ X 32

	add 3Dconv1	
3Dstack3_1	$\begin{bmatrix} 3\times3\times3,64 \\ 3\times3\times3,64 \end{bmatrix}, stride = 2$ add 3Dstack2_3	$\frac{1}{8}$ H x $\frac{1}{8}$W x $\frac{1}{8}D$ X 64
3Dstack3_2	$3\times3\times3,64$ $3\times3\times3,64$	$\frac{1}{16}$ H x $\frac{1}{16}$W x $\frac{1}{16}D$ X 64
3Dstack3_3	$Deconv\ [3\times3\times3,64], stride = 2$ add 3Dstack1_1	$\frac{1}{8}$ H x $\frac{1}{8}$W x $\frac{1}{8}D$ X 64
3Dstack3_4	$Deconv\ 3\times3\times3,32$ add 3Dconv1	$\frac{1}{4}$ H x $\frac{1}{4}$W x $\frac{1}{4}D$ X 32
Output_1	$3\times3\times3,32$ $3\times3\times3,1$	$\frac{1}{4}$ H x $\frac{1}{4}$W x $\frac{1}{4}D$ X 1
Output_2	$3\times3\times3,32$ $3\times3\times3,1$ Add output_1	$\frac{1}{4}$ H x $\frac{1}{4}$W x $\frac{1}{4}D$ X 1
Output_3	$3\times3\times3,32$ $3\times3\times3,1$ Add output_2	$\frac{1}{4}$ H x $\frac{1}{4}$W x $\frac{1}{4}D$ X 1
Disparity regression	Upsampling regression	D x H x W H x W

3.2. Point cloud generation

Disparity maps obtained from the trained network are 2D grayscale images. To obtain a 3D point cloud, the disparity maps are reprojected using the camera parameters. The general equation that describes the mapping of 3D information from 2D images is as shown by the equation below.

$$x = PX \qquad (3.3)$$

where x is a 2D homogeneous image point defined by pixels, $\boldsymbol{X}$ a 3D world point on the terrain, and P the camera matrix.

Figure 3-6 shows camera geometry that describes the relationship between an image point x and the corresponding point $\boldsymbol{X}$ on the terrain.

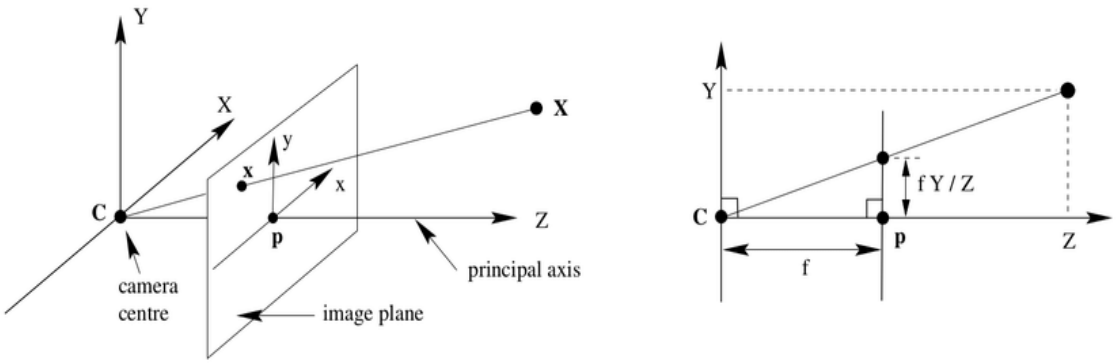


Figure 3-6: Camera Geometry (Adapted from Hartley & Zisserman, 2003)

From **Figure 3-6** above the general equation 3.3 can be expanded in matrix notation as shown in equation 3.4.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} p1 & p2 & p3 & p4 \\ p5 & p6 & p7 & p8 \\ p9 & p10 & p11 & p12 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.4}$$

with x a 3×1 matrix, X a 4×1 matrix, and a camera matrix P a 3×4 matrix. The camera projective matrix P can be further written as;

$$P = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.4}$$

Equation 3.4 above can be decomposed into two matrices as;

$$P = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.5}$$

where f is the focal length of the camera in pixels, p_x and p_y are the coordinates of the principal point and accounts for different camera and image origins. This can be written as shown in equation 3.6 below.

$$P = K\,[\,I \mid 0\,] \tag{3.6}$$

However, there are three different coordinate systems – camera, image, and world coordinate systems and therefore, a transformation between them is needed. To align the camera and world systems, a 3D rotation and translation is required. The projection matrix now becomes;

$$P = K\,[\,R \mid t\,] \tag{3.7}$$

where K is a 3×3 intrinsic calibration matrix, R is a 3×3 extrinsic rotation matrix that describes the orientation of the camera coordinate system and t is a 3×1 extrinsic translation vector that relates image and object coordinate systems. The above projective matrix, equation 3.7 together with equation 2.1(describes how to recover depth from disparity) are used to project a 2D disparity image to a 3D point cloud. The point cloud is then triangulated to obtain a digital surface model (DSM). We use LAStools (Rapidlasso GmbH, 2020) to remove noise in the cloud and create DSM.

3.3. Conventional methods

Traditional methods rely on set parameters that are selected by the user and are therefore slow and computationally expensive. However, these methods have successfully been implemented in commercial software and generate good results. Most of the implementations are based on the semi-global matching algorithm. For comparison purposes in this study, we use Pix4Dmapper to generate point cloud and DSM. The general pipeline implemented in Pix4D is shown in **Figure 3-7**. The three main steps i.e. *initial processing*, *point cloud and mesh*, and *DSM and Ortho-mosaic* are described below.

Initial processing

In the first step, the user is allowed to select the processing options. The main options are the image size to be used in extracting key points and what will be included in the quality report. For precise results, *Full* image size should be selected but takes a longer time than *Rapid* that has a lower image scale. Original image size under *Custom* is recommended. Matching allows users to select images to be matched optimizing for grid or corridor flight paths and terrestrial images. Our images are captured in a grid path and we use *full* image size. Calibration option allows users to set the number of key points to be extracted, calibration method, and optimization for internal and external parameters. We applied the defaults with the rematch set to *automatic* to add more matches after initial processing for improved quality of reconstruction.

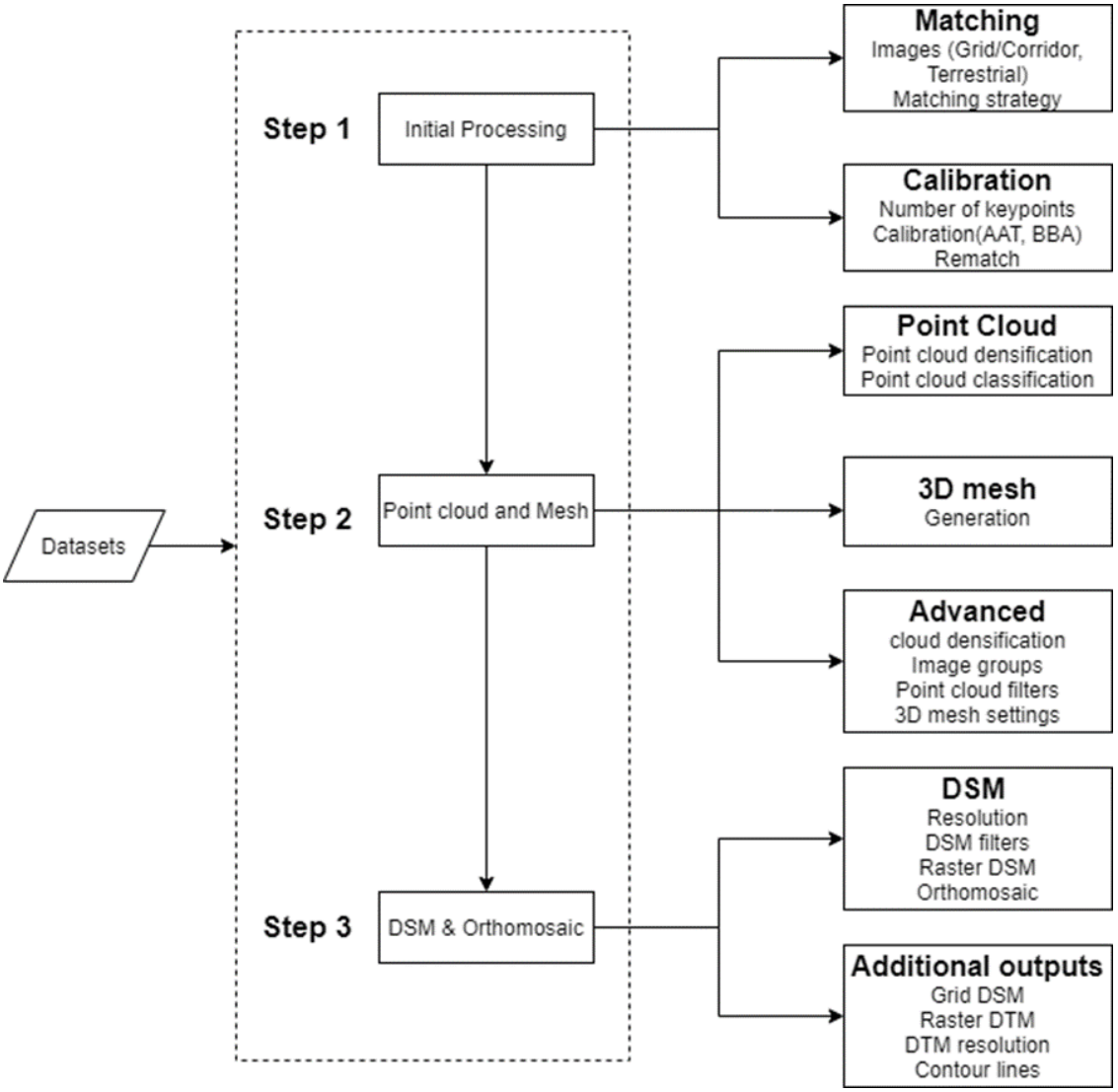


Figure 3-7: General pipeline in Pix4D showing the main parameters to be selected by the user in each step. (Note: Outputs from each step are exported to be used in next step or saved).

Point cloud and Mesh

The second step involves setting parameters to improve the point density for better accuracy of DSM and ortho-mosaic. The user is allowed to select parameters for point cloud densification. We used the recommended image scale – Half image size, the density of point cloud – Optimal, and 3 as the minimum number of matches per 3D point. We didn't apply classification to the generated point cloud. We choose to merge the point cloud into one file and export it as a las file for further analysis.

DSM and Ortho-mosaic

In the last step, the user selects the parameters for the desired output i.e. DSM or Ortho-mosaic such as resolution, filters, and output file format. Here, we set the recommended resolution of 1 ground sampling distance (GSD) i.e. a DSM of 5cm resolution. To remove erroneous points, we

apply noise filtering and smoothen the resulting DSM using surface smoothing. For further analysis and comparison, we need a raster DSM and therefore we generate a GeoTIFF file. To obtain a raster file, an interpolation algorithm is applied. Two interpolation techniques are provided and differ in computation time and quality of results. Inverse distance weighting (IDW) which applies a weighted average of all points is suitable for scenes with buildings while triangulation which is based on Delaunay triangulation and faster than IDW is recommended for flat areas. Lastly, we generate a merged DSM GeoTIFF file that is exported for further analysis.

4. EXPERIMENTS

4.1. Data description

We use synthetic datasets for training the network from scratch and real-world datasets for fine-tuning. The synthetic dataset SceneFlow (Mayer et al., 2016) is a collection of more than 39,000 stereo frames of size 960×540 pixels. It consists of three different datasets FlyingThings3D, Monkaa, and Driving datasets created using a modified version of the Blender suite. The modification is done in such a way that in addition to generating stereo images, it also generates three additional passes per frame for each view. This results in 3D positions of all surface points providing a dense and complete ground truth even in occluded areas. FlyingThings3D is a collection of everyday objects flying along randomized 3D trajectories. It is created by translating and rotating 3D objects downloaded from ShapeNet (Savva, Chang, & Hanrahan, 2015) database and randomly texturizing the object's material. Monkaa is created by rendering scenes from the short film Monkaa while the Driving dataset consists of dynamic street scenes resembling KITTI datasets captured from a moving car. It is created by rendering car models and detailed tree models.

Real-world datasets consist of the KITTI benchmark dataset and a UAV dataset. KITTI 2015 scene flow benchmark dataset (Menze & Geiger, 2015) was created to provide dynamic objects and ground truth for the evaluation of scene flow models. This is after the challenging KITTI 2012 benchmark dataset (Geiger, Lenz, & Urtasun, 2012) used for evaluation failed to provide scene flow ground truth for moving objects. They leveraged the KITTI raw data (Geiger et al., 2013) to create a realistic scene flow dataset with independently moving objects and ground truth. It comprises of 200 training and 200 test scenes. To generate ground truth disparity maps, the vehicles are equipped with a rotating laser scanner.

The UAV dataset consists of nadir images taken from a UAV system equipped with an RGB camera. The images were captured in Rwanda and Europe. They are acquired with an 80% forward and 60% side overlaps at 2~3 cm GSD. We sample 4,000 images from these datasets for our experiments. We use 3,500 image pairs for training and 500 image pairs for testing. A summary of the datasets used in the study is given in **Table 4.1** below.

Table 4.1: Description of datasets used in the study

Dataset	Description	Status	Size
Scene Flow	A synthetic dataset consisting of images with non-dynamic scenes and dense ground truth	Available	Over 39,000 image pairs
KITTI 2015	Close-range images of road, vehicles, and other objects. Laser ground truth	Available	400(200 image pairs for training and 200 image pairs for testing)
UAV dataset	Nadir UAV images taken with RGB camera at 2~3 cm GSD with 80% forward and 60% side overlaps.	Available	4,000(3,500 training image pairs and 500 image pairs for testing)

4.1.1. Data preparation

The images are first rectified to align the epipolar lines in the left and the right images. This reduces the computation of disparity to one dimensional hence reducing training time.

In aerial imagery, there are no publicly available benchmark datasets with ground truth that can be used to train stereo matching networks. The acquisition of labeled data is expensive and requires conversions and transformations such as camera model registration, coordinate system conversions, and projections. For this study, we use an improved variant of semi-global matching (SGM) algorithm by (Guo, Xu, & Zheng, 2016) that uses fast census transform which is faster than the traditional census transform. SGM fast census transform differs from the traditional SGM algorithm in terms of speed and noise sensitivity. Apart from being faster than traditional SGM, the fast transform is less sensitive to noise than baseline SGM. It also generates denser and more regular disparity maps with fewer artifacts than SGM. **Figure 4-1** below shows disparity maps obtained from a UAV stereo pair using the above algorithm.

Figure 4-1: UAV stereo pairs and corresponding disparity maps generated by SGM (Guo et al., 2016)

4.1.2. Software and Implementation

We implement the model in a PyTorch environment. A graphics processing unit is required to speed up computation and reduce training time. We utilize an NVIDIA GeForce RTX 2080i GPU card for faster processing. MATLAB R2019a (MathWorks, 2019) is used to rectify stereo image pairs while Pix4Dmapper (Pix4D, 2020) is used to generate products for comparison.

4.2. CNN Experiments

4.2.1. Direct testing

In the first experiment, we aim to test the generalization ability of pre-trained CNN models to target datasets. In our case, the model is trained on a synthetic dataset and tested on aerial imagery. We first train our CNN model on SceneFlow synthetic dataset for 30 epochs using Adam optimizer and betas $\beta_1 = 0.9$ and $\beta_2 = 0.999$ at 0.001 learning rate. We directly apply the trained model on our target dataset for testing.

4.2.2. Parameter optimization

We carry out a preliminary experiment to determine the optimal parameters to be used in the implementation of the model. Since our UAV dataset has a sparse ground truth, we incorporate the KITTI 2015 benchmark dataset which has a denser ground truth in the preliminary experiment. In the first experiment, we use 200 KITTI training images and sample 200 UAV images for parameter tuning. From a total of 400 images, we use 80% for training and 20% for validation. In the second experiment, we sample a total of 400 UAV images. As in the first experiment, we use 80% for training and 20% for validation. For testing, we use the 200 KITTI testing images and sample 100 images from the UAV dataset.

In this experiment, we use the pre-trained model from the experiment in 4.2.1 above. We tune the parameters over the 400 real-world images in the two experiments. The parameters that result in low validation errors in the two experiments are selected for final implementation.
The parameters used in fine-tuning with both KITTI 2015 and UAV dataset are shown in the table below.

Table 4.2: Parameters used in parameter optimization experiment

Parameter	Values
Maximum disparity	192, 256, 320
Learning rate	0.01, 0.001, 0.0001
Betas	$\beta_1 = 0.9, \beta_2 = 0.999$
Optimizer	$Adam, SGD$
Momentum	0.9
Epochs	300
Loss	$SmoothL_1(Hüber), MSELoss$

A combination of optimal parameters determined in the parameter optimization experiment are used to train the network. After training, the network is evaluated on all the testing dataset. The selected parameters for final implementation are shown below in **Table 4.3**.

Table 4.3: Final implementation parameters

Parameter	Value(s)
Learning rate	0.001, 0.0001
Maximum disparity	256
Optimizer	$Adam\ optimizer$
Betas	$\beta_1 = 0.9, \beta_2 = 0.999$
Maximum epochs	400
Loss	$Hüber\ Loss$

4.2.3. Final implementation

Using the parameters determined in experiment 4.2.2, we implement the network on our target UAV dataset. We adopt a training strategy used in most deep learning experiments whereby the network is first trained on a synthetic dataset with dense ground truth and finetuned on the target dataset. We implement our fine-tuning using the pre-trained model from the experiment above and fine-tune on our target UAV dataset. We conduct two experiments as follows;

First, to evaluate the effect of increasing the sample size in transfer learning, we divide our training dataset into four samples. We set the sample sizes to 200, 600, 1200, and 3000 image pairs. For all sample sizes, we use the parameters in **Table 4.3** for training. The trained models are tested on our testing dataset. Second, we train the model on the whole dataset and increase the number of epochs to 400 at a learning rate of 0.001. We change the learning rate for the last 100 epochs to 0.0001 to decrease the learning process. We set the maximum disparity to 256 since the dataset contains low objects on terrain and tall objects such as houses. The trained model is tested on the testing dataset and the disparity maps obtained used to generate a point cloud.

4.2.4. Comparison of CNN and traditional methods

We conduct a comparison between the CNN method and traditional hand-crafted methods. From our experiments in section 4.2.3, we select the strategy that gives the best disparity map i.e. training the network with the whole UAV dataset. We reproject the disparity maps to obtain point cloud and DSM as explained in section 3.2. For the traditional methods, we sample images from our dataset and use Pix4Dmapper (Pix4D, 2020) to generate point clouds and digital surface model (described in section 3.3). We assess the quality of the products and conduct an in-depth comparative analysis of the two methods.

4.3. Results and Analysis

4.3.1. Direct testing on target dataset

We present the results of directly applying a pre-trained model to a target dataset. As shown in **Figure 4-2** below, deep learning models lack generalization ability and cannot guarantee transfer learning from models trained on synthetic datasets to a target dataset. This is especially challenging when the datasets differ in features such as in our case of aerial imagery. As noted by Yosinski et al. (2014), transfer learning is only possible when the features in the two datasets are similar. In our case, the features in UAV imagery are quite different from the features in the SceneFlow synthetic dataset, and therefore the network performs poorly.

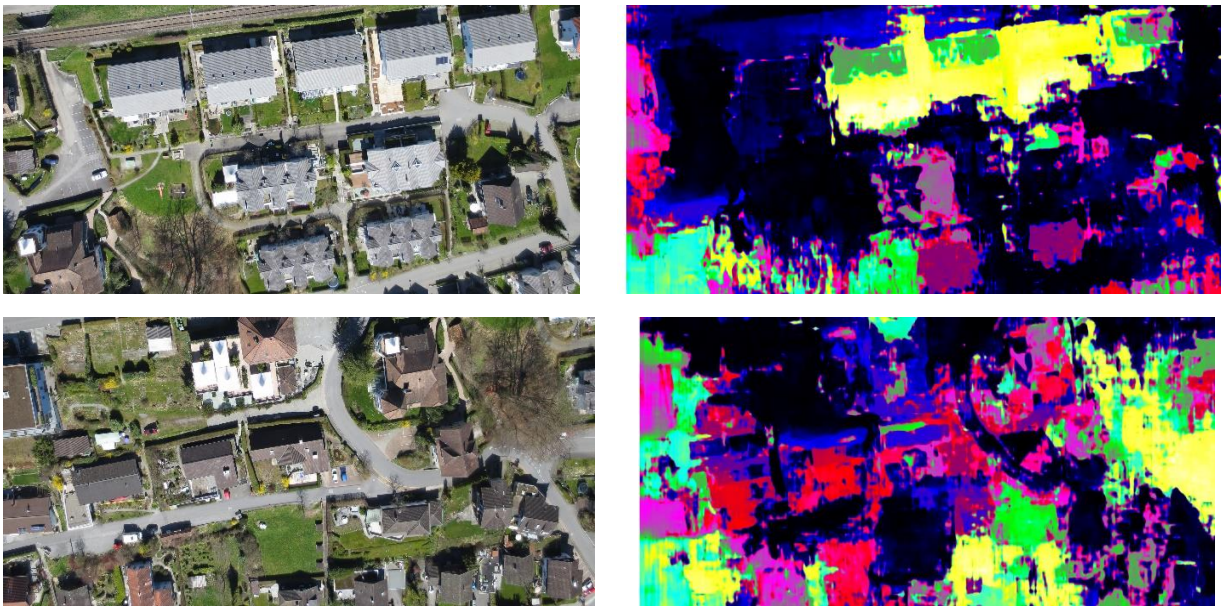


Figure 4-2: Disparity maps generated by directly testing a pre-trained CNN two image pairs.

4.3.2. CNN optimization

In determining the optimal hyperparameters for our model, we conducted a preliminary experiment to determine the parameters to use in our final implementation. **Table 4.4**, **Table 4.5**, and **Table 4.6** present the different accuracies obtained from the experiment using the UAV dataset. For each maximum disparity value, we try different values of learning rate, optimizer, and loss. These are the parameters that affect the learning process in deep networks. The number of epochs was kept constant i.e. 300. The momentum and the *betas* values for *Adam optimizer* were as described in **Table 4.3**.

Table 4.4: Accuracies obtained for a maximum disparity of 192.

Max disparity	Learning rate	Optimizer	Loss	Accuracy
192	0.01	Adam	SmoothL1	96.82
		SGD	MSE	96.54
192	0.001	Adam	SmoothL1	97.10
		SGD	MSE	96.94
192	0.0001	Adam	SmoothL1	97.13
		SGD	MSE	97.02

Table 4.5: Accuracies obtained for a maximum disparity of 256

Max disparity	Learning rate	Optimizer	Loss	Accuracy
256	0.01	Adam	SmoothL1	97.11
		SGD	MSE	96.93
256	0.001	Adam	SmoothL1	97.37
		SGD	MSE	97.08
256	0.0001	Adam	SmoothL1	97.24
		SGD	MSE	97.13

Table 4.6: Accuracy obtained for a maximum disparity of 320

Max disparity	Learning rate	Optimizer	Loss	Accuracy
320	0.01	Adam	SmoothL1	96.41
		SGD	MSE	96.18
320	0.001	Adam	SmoothL1	96.72
		SGD	MSE	96.59
320	0.0001	Adam	SmoothL1	96.86
		SGD	MSE	96.77

We observe that when the maximum disparity is set to 320, the accuracy decreases as compared to disparity values of 192 and 256. A value of 0.001 for learning rate and a maximum disparity of 256 gives the best accuracy as compared to 0.01 and 0.0001. Although a value of 0.0001 has good accuracy, it may take longer for larger sample sizes given that the sample size used is only 400 image pairs. Thus, it may cause the network to take more time to converge. **Figure 4-3** below shows the accuracies obtained by using different learning rates.

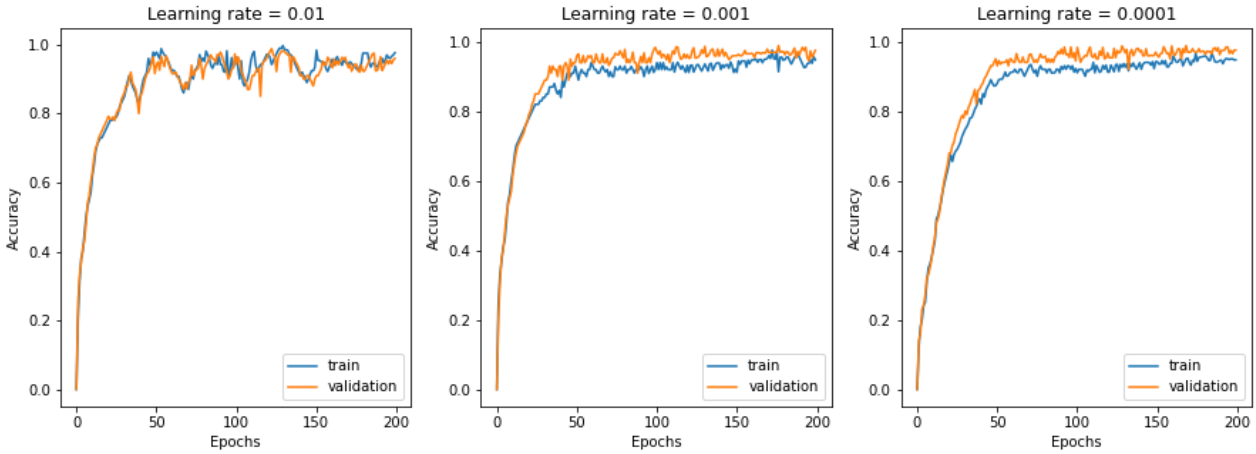


Figure 4-3: Accuracy obtained by different values of the learning rates.

From **Figure 4-3** above, a large learning rate causes oscillations and is not stable as compared to smaller values that enable the model to learn well. As the learning rate gets smaller, the model learns slowly and therefore small learning rates are not suitable especially when the sample size is large. With our model configuration, the results above show that the model performs well with a learning rate of 0.001.

4.3.3. Varying sample size

We conducted experiments to determine the effect of the size of the dataset used in transfer learning. We set different sample sizes of our UAV dataset and used each sample to train our model. We then tested each trained model on our testing dataset. The results are shown in **Table 4.7**.

Table 4.7: Test accuracy of different sample sizes

Sample Size	**200**	**400**	**800**	**1600**	**3000**
Accuracy (%)	96.48	95.21	96.78	97.23	97.89
Improvement (%)	**-**	**-1.27**	**1.57**	**0.45**	**0.66**

When using a small sample size i.e. 200 image pairs for fine-tuning a deep CNN model, the performance is poor. We observed that increasing the size of the training dataset increases accuracy. This is not the case when using 400 pairs as the performance dropped from 96.48% to 95.21%. However, as we increased the sample size to 1600 image pairs, the accuracy improved significantly from 95.21% to 97.23%. Training with the whole training dataset i.e. 3000 image pairs leads to a 97.89% accuracy. The results show that for deeper CNN models to learn features effectively, they require a large training dataset.

4.3.4. Final experiment

In the final training experiment, we used all the UAV dataset containing 3000 image pairs and 3000 labels to train our CNN model. The trained model was tested on our testing dataset. The disparity maps obtained from the testing set are shown below.

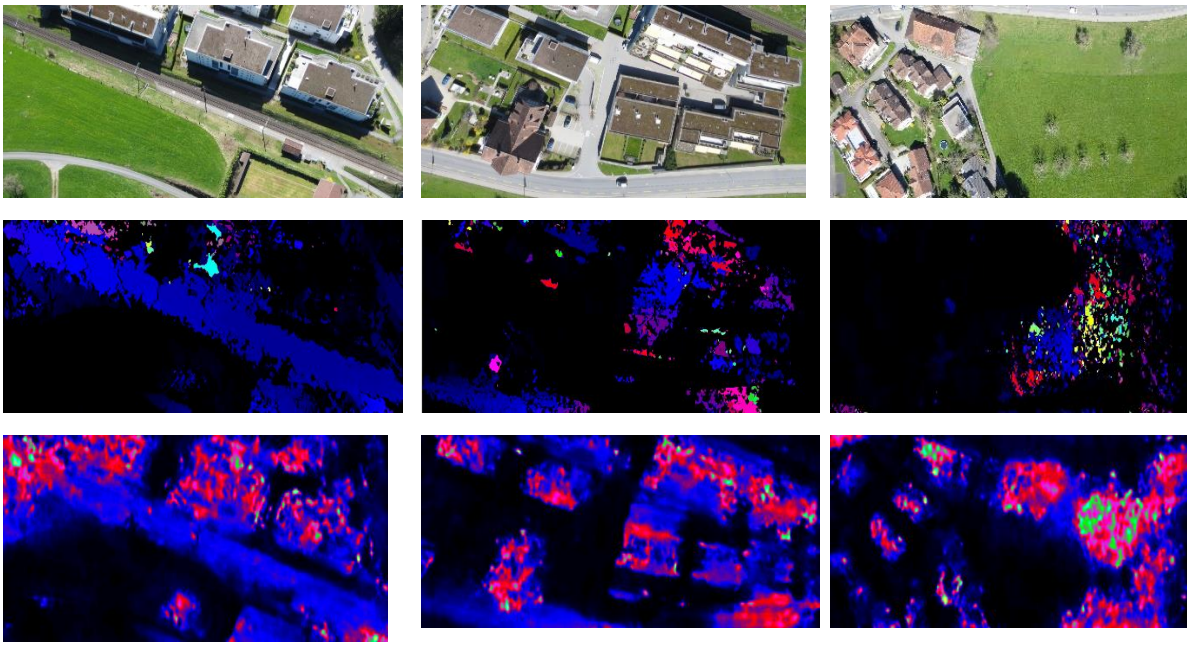


Figure 4-4: Disparity estimation results: Top row: UAV testing stereo images, Second row: SGM method, Last row: Finetuned CNN.

Figure 4-4 shows disparity maps generated using the SGM and fine-tuned CNN model. The result generated by SGM is sparse as compared to the CNN maps. In the SGM method, objects such as buildings are not distinguishable as in pre-trained and fine-tuned CNN results. This is clear especially in occluded areas and along the edges of objects. Additionally, the SGM maps have clutter and artifacts while the fine-tuned CNN maps are smooth and denser. There is significant improvement between the CNN pre-trained on synthetic datasets and the CNN fine-tuned on aerial imagery. However, looking closely at the results in the third column, the disparity map from the fine-tuned CNN is poor, especially on the buildings. This is because our training dataset consisted of scenes with more vegetation and few buildings in most areas which causes the network to overfit. This can be reduced by adding more data during training and incorporating data from diverse scenes.

4.3.5. 3D point cloud generation

We reprojected the disparity maps generated by our model to recover the scene. The results are presented below.

Figure 4-5: 3D scenes recovered from dense disparity maps: Left: Stereo images and Right: Dense 3D point cloud

From the above results, the dense disparity maps generated a high-density point cloud that reveals fine details of objects although some errors are evident. Errors due to occlusions are visible in buildings and tall objects. This is due to shadows caused by occlusions that lead to matching errors. Gaps are also visible in areas with vegetation and flat areas. This is as a result of matching failure in vegetation. Fine details in vegetation are difficult to capture using deep learning because of occlusions and hence no penetration. CNN methods leverage stereo information and errors such as those caused by occlusions are difficult to avoid when using stereo imagery. With multi-view geometry, such errors can be eliminated.

4.4. Comparative analysis

4.4.1. Point cloud evaluation

We evaluate the results of our model and the conventional methods for 30 image pairs. The point cloud obtained from our pre-trained and finetuned models is filtered to remove noise i.e. invalid disparities. The clouds were registered in Cloud Compare (Daniel, 2015) and Iterative Closest Point (ICP) algorithm (Besl & McKay, 1992) used to improve the registration. Afterward, we computed the distance between the two clouds – modified Hausdorff distance i.e. the distance between two nearest neighbor points. The results are as shown in **Table 4.8** below.

Table 4.8: Accuracy assessment of point clouds generated by our model compared to conventional methods.

Method	Amount of points	Mean	STD
Pre-trained	5,343,851	0.2063	0.2717
Finetuned	3,145,612	0.1181	0.1425

We segmented a building surface from the two clouds and carried out a comparison against a segment from the reference cloud. The accuracy is higher as compared to the whole point cloud. The model performs well on building surfaces and edges. Our fine-tuned model outperforms the pre-trained one. Both models, however, perform better on building roofs. See **Table 4.9** below.

Table 4.9: Accuracy assessment of point cloud from a building segment

Method	Amount of points	Mean	STD
Pre-trained	12,543	0.1178	0.0825
Finetuned	9,871	0.0653	0.0317

We conducted a comparison between our method and conventional method based on mean, standard deviation, amount of point cloud generated, and runtime (see **Table 4.10**). We set the maximum distance for distance computation to 5. We noted that deep learning methods if pre-trained are quite fast than conventional methods. For a testing dataset of 30 image pairs, our CNN model takes a maximum of 2 minutes to generate a dense point cloud while Pix4D takes a minimum of 45 minutes on the same. Thus, in terms of speed, deep learning methods are 20 times faster than conventional methods. Conventional methods on the other hand generate point clouds with a higher point density than deep learning methods. The low point density in deep

learning methods is due to matching errors caused by variations such as illumination, occlusions, and texture in images. This leads to many invalid points that are removed through filtering.

Table 4.10: Comparison between deep learning and conventional methods

Method	**Mean**	**STD**	**Amount of points**	**Point density (m^2)**	**Runtime (mins)**
Conventional	2.0862	3.5022	3,797,651	24	45
CNN	1.8916	2.1505	3,145,612	20	2

The large errors in the RMSE of the conventional methods can be attributed to a lack of ground control points (GCPs) to improve the accuracy of the products. A study by (Oniga, Breaban, & Statescu, 2018) recommended a density of 1 GCP per 0.02 Ha for high accuracy products in UAS photogrammetry. With the availability of GCPs, the accuracy of conventional methods is expected to be few centimetres.

4.4.2. Digital surface models

We generated digital surface models from the point clouds and visualized them in a GIS software for analysis.

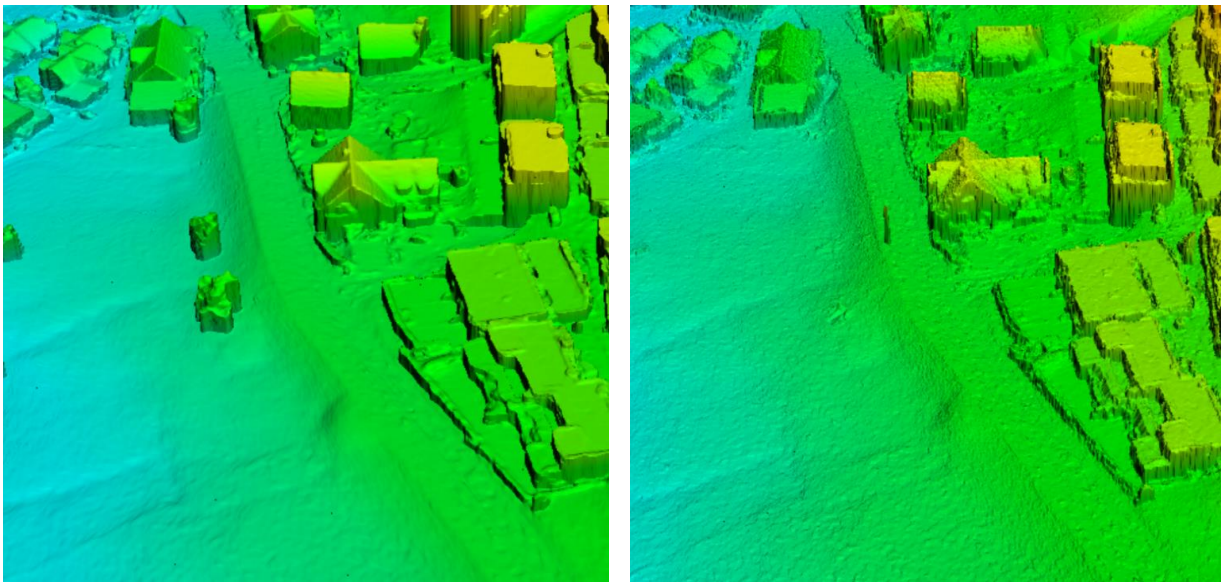


Figure 4-6: Digital surface models from CNN (Left) and conventional methods (Right)

DSM generated by our method contains smooth and sharp building edges that resolve details of objects and structures. On the other hand, objects surfaces and edges in the DSM from conventional methods are fairly sharp and rough especially in tall buildings although, in small objects, edges are preserved. In our DSM, chimneys and dormers can be seen clearly while in the

other method they cannot be well differentiated. This might be due to the interpolation technique used in conventional methods since triangulation doesn't perform well in areas with tall buildings. Furthermore, the lack of GCPs for georeferencing in conventional methods has been found to affect the quality of DSM. Martínez-Carricondo et al. (2018) noted that the presence of well-distributed GCPs leads to accurate photogrammetric products. Further, they demonstrated that increasing the number of GCPs improved the quality of digital surface models. Thus, the quality of the DSM was affected by a lack of ground control points. On the right, trees along the road have been smoothed out while in our method (left image), they have been preserved. Our results show that deep learning methods are competitive and have potential in UAS photogrammetry. Moreover, they are very fast as compared to conventional methods.

5. DISCUSSIONS

We present discussions drawn from the experiments conducted and the results obtained.

5.1. CNN features

We conducted a literature review on CNN models applied in stereo matching that provided insights on the architectures suitable for end-to-end disparity regression. Deep networks have been found to be suitable in many tasks including stereo matching because of their ability to integrate low and high-level features (Zeiler & Fergus, 2014). Although they may suffer from accuracy degradation during training, residual connections have enabled these deep networks (residual networks) to register increased accuracies with fewer parameters. They are less complex as compared to plain networks such as VGG-16 and AlexNet. In this study, we, therefore, implemented a residual network and noted that organizing it in blocks enables the extraction of different features effectively. We also applied a dilated convolution at the last layers of the feature extraction module to ensure features are extracted across a large receptive field.

The use of encoder-decoder architecture results in smooth and dense disparity maps without applying post-processing techniques. However, the design of the architecture may be influenced by the input data to be used. Most of the CNN architectures available are designed to use close-range images and therefore the use of aerial imagery requires design considerations that enable incorporation of different features from the whole image. The introduction of spatial pyramid by (Lazebnik, Schmid, & Ponce, 2006) enables the pooling of features from fine regions to local features enabling the aggregation of features from the whole image. A similar technique was implemented in (He, Zhang, Ren, & Sun, 2014) for classification and semantic segmentation. In Liu, Rabinovich, & Berg (2015) pyramid pooling was also used to incorporate global context information. The same technique is applied in Chang & Chen (2018) using four blocks of fixed size. In this study, we designed the blocks in a way that allows the capturing of low and high objects present in our dataset. This improves correspondence computation in occluded areas and regions with illumination and texture variations. We noted that these features coupled with 3D CNN improved disparity estimation significantly. Undertaking such considerations in the selection of deep learning models is beneficial when adapting networks designed for computer vision tasks to different domains such as aerial imagery.

5.2. Parameter optimization

We investigated the influence of different parameters on the performance of deep neural networks for stereo matching tasks. Key parameters that affect the quality of disparity maps were found to be - loss, the maximum number of disparities, optimizer, and learning rate. We noted that deep CNN models require a moderate learning rate to be able to learn features effectively. A high or low learning rate would, therefore, lead to an increase in training loss and thus reduce the accuracy. In our parameter optimization experiments, we used a constant sample size of the data. However, the performance of the model is influenced by the amount of data used as noted when training with a large sample size in our final implementation. Therefore, in the determination of optimal parameters for training a deep neural network, it would be interesting to vary the sample size while varying the parameter values. This would result in the best values for the desired accuracy.

We incorporated a close-range dataset with denser ground-truth than our dataset in our optimization experiment. It emerged that when using both datasets for parameter search, the validation loss was deteriorating as the number of epochs increased. This may have been caused by the variations in the dataset features. This implies that CNN models tend to learn well when the features or scenes in the data are more or less the same.

5.3. Size and quality of dataset

In this study, we used SGM (Guo et al., 2016) to generate ground truth for our training. The quality of the ground truth influences the quality of disparity maps generated by the trained network. The SGM generated labels were poor and sparse and this had an effect on the performance of our model. For further work, it would be beneficial to use ground truth generated by methods such as laser (LiDAR), which is denser and would result in improved accuracy and quality of maps generated. Another aspect is the size of the training data used in our study. As stated before, deep CNN models require a large amount of data with dense ground truth for training. In our study, we used a total of 4000 image pairs which is a small dataset compared to available synthetic datasets used for training deep neural networks. Although we implemented a common strategy in deep learning experiments where we first trained the network with a large synthetic dataset and finetuned with our target dataset, the size of the finetuning target dataset matters. However, we noted that increasing the sample size during training increases the performance of the model.

In all our experiments, the input images were cropped or downsampled to a smaller size during training. This is due to memory constraints since large image sizes demand more memory

and computation time. It's not practical then to use large image sizes given the amount of data and number of epochs required for a neural network to be trained fully. However, during testing, the network can take a larger image size depending on the available memory card. In this study, we had access to limited resources (GPU) and thus we used a small batch size in all experiments. Radiuk (2018) investigated the effect of batch size in convolutional neural networks during training and testing. The author found that large batch sizes improved the accuracy although too large batch sizes led to poor generalization. It would be interesting to try a larger batch size with the availability of multiple GPUs. Despite the aforementioned limitations, the performance of the model was good in terms of the quality of disparity maps.

5.4. Evaluation and comparative analysis

In our study, we conducted a comparison of our method and conventional methods based on the final products. However, due to a lack of ground control points, we could not carry out georeferencing in conventional methods for improved accuracy and results. We envisage that with proper data collection including the establishment of ground control points in the study area, the evaluation would be accurate. A better approach would be comparing our method with Light Detection and Ranging (LiDAR) methods. This can be explored in the future to see how deep learning methods perform compared to LiDAR methods.

6. CONCLUSION AND RECOMMENDATIONS

6.1. Conclusion

The main objective of this study was to investigate deep learning approaches based on convolutional neural networks for 3D reconstruction from UAV imagery. We tested various state-of-the-art CNN models used in computer vision to determine the suitable model to adapt to our domain. We developed a methodology to recover 3D scenes and generate a digital surface model from UAV imagery using the adapted model. To achieve this, we optimized the parameters of the designed CNN model and trained it end-to-end for disparity regression. The generated disparity maps were used to recover the 3D scenes and create surface models. Lastly, we carried out a comparative analysis of our method and conventional methods.

Our results show that deep learning methods outperform conventional methods by far in terms of speed during testing. The products generated by CNN methods are comparable to those of conventional methods. Therefore, we postulate that deep learning methods have the potential to perform better than traditional methods. Answers to the questions posed at the beginning of the study are given below.

6.1.1. Answers to research questions

Specific objective 1

i. What are the existing networks that have been applied in stereo matching? And what network architecture is suitable?

In section 2.3, we described some of the architectures that have been implemented in stereo matching tasks. However, we noted that most of these networks have exploited close-range imagery for segmentation and recognition tasks in computer vision. Some of the networks compute the matching cost and require cost aggregation and post-processing techniques to refine the output. End-to-end architectures with encoder and decoder structure are, however, the best suited for disparity regression.

Specific objective 2

ii. Do pre-trained CNN models have generalization ability that enables transfer learning to target dataset?

We noted that pre-trained models have poor generalization. This can be attributed to the differences in features of the dataset used in training and the target dataset. As Bengio (2012) stated, features are only transferable when the two datasets are similar. In our case, the two datasets were quite different and thus the network performed poorly. The model performance, however, was improved by training the network using the target dataset.

iii. What parameters are to be considered when adapting a CNN model for a specific task?

In section 5.2, we described the parameters that affect the performance of convolutional neural networks as applied in stereo matching. They include loss, number of maximum disparities, optimizer, and learning rate. In addition, we demonstrated that the training set sample size is very crucial in transfer learning. Different tasks will require consideration of different parameters and thus they are task-specific. Further, we noted that CNN features such as spatial pooling pyramids improved disparity estimation by incorporating contextual features.

Specific objective 3

iv. What is the quality of point clouds and DSM?

The method developed in this study generated accurate results. As shown in our analysis, our method generated point clouds with a high-density that enables full recovery of buildings and objects. Fine details of objects and structures in the scene are captured accurately. However, errors due to noise are visible in occluded areas and flat areas with vegetation. The mean and standard deviation obtained for segments of the cloud taken from building surfaces were low (**Table 4.9**). This shows how accurate our method performs well on objects surfaces. Building edges and other object features are clearly defined in the digital surface model. Bumps and holes can be seen in some areas due to gaps and noise in point clouds.

Specific objective 4

v. What is the performance of deep learning approaches compared to handcrafted methods?

We carried out a comparative analysis of our method and conventional methods. Our method outperforms conventional methods in terms of runtime. Conventional methods, on the other hand, are quite slow but generate high-quality results. As described in section 4.4, our method is accurate on buildings and objects although errors are present in untextured and occluded areas. By visual analysis, we noted that our surface models were smooth and complete enabling fine details in the scene to be visible. Overall, our method performed well in terms of speed while generating almost the same results with conventional methods in terms of quality.

6.1.2. Limitations

The following limitations were met during the study.

- Lack of dense ground truth that ultimately affected the quality of the results obtained.
- Lack of computation resources such as multiple graphics processing units that inhibited the implementation of robust models.

- Due to a lack of stereo imagery, our image processing involved rectification and computations using different software. This may have introduced errors and distortions that may have affected the quality of our final results.

6.2. Recommendations

We recommend the following for further research work:

- Explore ways of collecting aerial imagery using a stereo image setup. This will minimize the distortions and errors that may be introduced during image processing.
- Explore techniques of generating dense and sufficient ground truth from aerial imagery for improved results.
- Investigate the application of multi-view geometry techniques in CNN methods.
- Incorporating LiDAR data or other high-quality data such as CAD models in the comparison metrics.

LIST OF REFERENCES

Bandrova, T., Zlatanova, S., & Konecny, M. (2012). Three-Dimensional Maps for Disaster Management. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *1*(September), 245–250. https://doi.org/10.5194/isprsannals-I-4-245-2012

Bengio, Y. (2012). Deep Learning of Representations for Unsupervised and Transfer Learning. In *JMLR W&CP: Workshop on Unsupervised and Transfer Learning* (Vol. 27, pp. 17–37).

Bengio, Y., Goodfellow, I., & Courville, A. (2015). *Deep Learning [pre-pub version]*. MIT Press.

Besl, P. ., & McKay, N. D. (1992). A Method for Registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *14*, 239–256. https://doi.org/http://dx.doi.org/10.1109/34.121791

Boykov, Y. Y., & Jolly, M.-P. (2001). Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images. In *International Conference on Computer Vision* (Vol. I, p. 105). Retrieved from http://www.csd.uwo.ca/faculty/yuri/Papers/iccv01.pdf

Chai, D. (2017). A Probabilistic Framework for Building Extraction from Airborne Color Image and DSM. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *10*(3), 948–959. https://doi.org/10.1109/JSTARS.2016.2616446

Chang, J. R., & Chen, Y. S. (2018). Pyramid Stereo Matching Network. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 5410–5418. https://doi.org/10.1109/CVPR.2018.00567

Daniel, G. . (2015). Cloud Compare - 3D Point Cloud and Mesh Processing Software. Retrieved from https://www.danielgm.net/cc/

Eigen, D., Puhrsch, C., & Fergus, R. (2014). Depth map prediction from a single image using a multi-scale deep network. *Advances in Neural Information Processing Systems*, *3*(January), 2366–2374.

Fischer, P., Ilg, E., Philip, H., Hazırbas, C., Golkov, V., Cremers, D., & Brox, T. (2015). FlowNet: Learning Optical Flow with Convolutional Networks. In *IEEE International Conference on Computer Vision (ICCV)*. Santiago, Chile: IEEE. https://doi.org/10.1109/ICCV.2015.316

Fua, P. (1993). A parallel stereo algorithm that produces dense depth maps and preserves image features. *Machine Vision and Applications*, *6*(1), 35–49. https://doi.org/10.1007/BF01212430

Fusiello, A., Roberto, V., & Trucco, E. (1997). Efficient stereo with multiple windowing. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, (July), 858–863. https://doi.org/10.1109/cvpr.1997.609428

Garg, R., Vijay Kumar, B. G., Carneiro, G., & Reid, I. (2016). Unsupervised CNN for single view depth estimation: Geometry to the rescue. In *Proceedings of the European Conference on Computer Vision (ECCV)*. https://doi.org/10.1007/978-3-319-46484-8_45

Geiger, A, Lenz, P., Stiller, C., & Urtasun, R. (2013). Vision meets Robotics: The KITTI Dataset. *The International Journal of Robotics Research (IJRR)*, *32(11)*(September), 1231–1237.

Geiger, Andreas, Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? the KITTI vision benchmark suite. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 3354–3361. https://doi.org/10.1109/CVPR.2012.6248074

Geman, S., & Geman, D. (1984). Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-6*(6), 721–741. https://doi.org/10.1109/TPAMI.1984.4767596

Gerke, M., Nex, F., & Jende, P. (2016). Co-Registration of Terrestrial and UAV-Based Images -

Experimental Results. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, *40*(3W4), 11–18. https://doi.org/10.5194/isprsarchives-XL-3-W4-11-2016

Girshick, R. (2015). Fast R-CNN. *Proceedings of the IEEE International Conference on Computer Vision*, *2015 Inter*, 1440–1448. https://doi.org/10.1109/ICCV.2015.169

Godard, C., Mac Aodha, O., & Brostow, G. J. (2017). Unsupervised monocular depth estimation with left-right consistency. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 6602–6611. https://doi.org/10.1109/CVPR.2017.699

Grimson, W. E. L. (1985). Computational Experiments with a Feature Based Stereo Algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-7*(1), 17–34. https://doi.org/10.1109/TPAMI.1985.4767615

Guo, S., Xu, P., & Zheng, Y. (2016). Semi-Global Matching Based Disparity Estimate Using Fast Census Transform. In *2016 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. https://doi.org/10.1109/CISP-BMEI.2016.7852771

Han, X.-F., Laga, H., & Bennamoun, M. (2019). Image-based 3D Object Reconstruction: State-of-the-Art and Trends in the Deep Learning Era. Retrieved from http://arxiv.org/abs/1906.06543

Hannah, M. . (1974). *Computer Matching of Areas in Stereo Images*. Ph.D.Thesis Stanford University. Retrieved from http://www.dtic.mil/dtic/tr/fulltext/u2/p004398.pdf

Hartley, R., & Zisserman, A. (2003). *Multiple View Geometry in Computer Vision* (2nd ed.). New York, NY, USA: Cambridge University Press. https://doi.org/10.5555/861369

He, K., Zhang, X., Ren, S., & Sun, J. (2014). Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision* (pp. 346–361). https://doi.org/10.1007/978-3-319-10578-9_23

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (pp. 770–778). https://doi.org/10.1109/CVPR.2016.90

Hirschmüller, H. (2008). Stereo Processing by Semi-Global Matching and Mutual Information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*(2), 328–341. https://doi.org/10.1109/TPAMI.2007.1166

Hirschmuller, H., & Scharstein, D. (2009). Evaluation of stereo matching costs on images with radiometric differences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(9), 1582–1599. https://doi.org/10.1109/TPAMI.2008.221

Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., & Brox, T. (2017). FlowNet 2.0:Evolution of Optical Flow Estimation with Deep Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*. Honolulu, HI, USA: IEEE. https://doi.org/10.1109/CVPR.2017.179

Jacobsen, K. (2003). DEM generation from satellite data. Retrieved November 19, 2019, from http://pdfs.semanticscholar.org/e066/%0Af8a278f845bf2cb19c9b4e81ec4dde1e1131.pdf

Jang, W., & Ho, Y. (2011). Efficient Disparity Map Estimation Using Occlusion Handling for Various 3D Multimedia Applications. *IEEE Transactions on Consumer Electronics*, *57*(4), 1937–1943. https://doi.org/10.1109/TCE.2011.6131174

Kanade, T., & Okutomi, M. (1994). A Stereo Matching Algorithm With an Adaptive Window: Theory and Experiment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *16*(9), 920–932. https://doi.org/10.1109/34.310690

Kanade, Takeo. (1994). Development of a Video-Rate Stereo Machine. In *Image Understanding Workshop* (pp. 549–557). Monterey, CA: Morgan Kaufmann Publishers. https://doi.org/10.7210/jrsj.15.261

Kendall, A., Martirosyan, H., Dasgupta, S., Henry, P., Kennedy, R., Bachrach, A., & Bry, A. (2017). End-to-End Learning of Geometry and Context for Deep Stereo Regression.

Proceedings of the IEEE International Conference on Computer Vision, *2017-Octob*, 66–75. https://doi.org/10.1109/ICCV.2017.17

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Neural Information Processing Systems*. https://doi.org/10.1201/9781420010749

Kuznietsov, Y., Stückler, J., & Leibe, B. (2017). Semi-supervised deep learning for monocular depth map prediction. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* (pp. 2215–2223). https://doi.org/10.1109/CVPR.2017.238

Laina, I., Rupprecht, C., & Belagiannis, V. (2016). Deeper Depth Prediction with Fully Convolutional Residual Networks. *Computer Vision and Image Understanding*.

Lazebnik, S., Schmid, C., & Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2*, 2169–2178. https://doi.org/10.1109/CVPR.2006.68

Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Liang, C.-K., Cheng, C.-C., Lai, Y.-C., Chen, L.-G., & Chen, H. H. (2011). Hardware-Efficient Belief Propagation. *IEEE Transaction on Circuits and Systems for Video Technology*, *21*(5), 80–87. https://doi.org/10.1109/cvpr.2009.5206819

Liu, W., Rabinovich, A., & Berg, A. C. (2015). ParseNet: Looking Wider to See Better. *ArXiv Preprint ArXiv:1506.04579*, 1–11. Retrieved from http://arxiv.org/abs/1506.04579

Luo, W., Schwing, A. G., & Urtasun, R. (2016). Efficient Deep Learning for Stereo Matching. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 5695–5703. https://doi.org/10.1109/CVPR.2016.614

Mahjourian, R., Wicke, M., & Jun, C. V. (2017). Unsupervised Learning of Depth and Ego-Motion from Monocular Video. *IEEE Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2017.700

Marr, D., & Poggio, T. (1976). Cooperative computation of stereo disparity. *Science*, *194*(4262), 283–287. https://doi.org/10.1126/science.968482

Martínez-Carricondo, P., Agüera-Vega, F., Carvajal-Ramírez, F., Mesas-Carrascosa, F. J., García-Ferrer, A., & Pérez-Porras, F. J. (2018). Assessment of UAV-photogrammetric mapping accuracy based on variation of ground control points. *International Journal of Applied Earth Observation and Geoinformation*, *72*(February), 1–10. https://doi.org/10.1016/j.jag.2018.05.015

MathWorks. (2019). MATLAB. Retrieved October 17, 2019, from https://www.mathworks.com

Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., & Brox, T. (2016). A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-Decem*, 4040–4048. https://doi.org/10.1109/CVPR.2016.438

Menze, M., & Geiger, A. (2015). Object Scene Flow for Autonomous Vehicles. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *07-12-June*, 3061–3070. https://doi.org/10.1109/CVPR.2015.7298925

Nair, V., & Hinton, G. E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. In *27th International Conference on Machine Learning (ICML-10)* (Vol. 33, pp. 807–814). Haifa, Israel. https://doi.org/10.1123/jab.2016-0355

Oniga, E., Breaban, A., & Statescu, F. (2018). Determining the Optimum Number of Ground Control Points for Obtaining High Precision Results Based on UAS Images. In *2nd International Electronic Conference on Remote Sensing*. https://doi.org/10.3390/ecrs-2-05165

Pix4D. (2020). Pix4Dmapper. Retrieved from https://www.pix4d.com/

Pollard, S. B., Mayhew, J. E., & Frisby, J. P. (1985). PMF: A Stereo Correspondence Algorithm using a Disparity Gradient Limit. *Perception*, *14*(4), 449–470.

https://doi.org/10.1068/p140449
Radiuk, P. M. (2018). Impact of Training Set Batch Size on the Performance of Convolutional Neural Networks for Diverse Datasets. *Information Technology and Management Science*, *20*(1), 20–24. https://doi.org/10.1515/itms-2017-0003
Rapidlasso GmbH. (2020). LAStools. Retrieved from https://rapidlasso.com/lastools/
Rottensteiner, F., Trinder, J., Clode, S., & Kubik, K. (2005). Using the Dempster-Shafer method for the fusion of LIDAR data and multi-spectral images for building detection. *Information Fusion*, *6*(4), 283–300. https://doi.org/10.1016/j.inffus.2004.06.004
Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., … Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, *115*(3), 211–252. https://doi.org/10.1007/s11263-015-0816-y
Sadeghi, Y., St-Onge, B., Leblon, B., & Simard, M. (2016). Canopy Height Model (CHM) Derived from a TanDEM-X InSAR DSM and an Airborne Lidar DTM in Boreal Forest. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *9*(1), 381–397. https://doi.org/10.1109/JSTARS.2015.2512230
Savva, M., Chang, A. X., & Hanrahan, P. (2015). Semantically-Enriched 3D Models for Common-sense Knowledge. *CVPR 2015 Workshop on Functionality, Physics, Intentionality and Causality*.
Saxena, A., Sun, M., & Ng, A. Y. (2009). Make3D: Learning 3D scene structure from a single still image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(5), 824–840. https://doi.org/10.1109/TPAMI.2008.132
Scharstein, D., & Szeliski, R. (2002). A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal of Computer Vision*, *47*(1–3), 7–42. https://doi.org/10.1109/SMBV.2001.988771
Sefercik, U. G. (2013). Productivity of terraSAR-X 3D data in urban areas: A case study in trento. *European Journal of Remote Sensing*, *46*(1), 597–612. https://doi.org/10.5721/EuJRS20134635
Shah, J. (1993). Nonlinear Diffusion Model for Discontinuous Disparity and Half-Occlusions in Stereo. *IEEE Computer Vision and Pattern Recognition*, 34–40. https://doi.org/10.1109/cvpr.1993.341004
Silberman, N., Hoiem, D., Kohli, P., & Fergus, R. (2012). Indoor segmentation and support inference from RGBD images Lecture Notes in Computer Science. *ECCV'12: Proceedings of the 12th European Conference on Computer Vision - Volume Part V*, *Part V*(Chapter 54), 746–760. https://doi.org/10.1007/978-3-642-33715-4_54
Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition, 1–14. Retrieved from http://arxiv.org/abs/1409.1556
Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*(1929–1958), 520–525. https://doi.org/10.1109/ICAEES.2016.7888100
Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Berlin: Springer Heidelberg. https://doi.org/10.1002/eji.200526121
Ummenhofer, B., Zhou, H., Uhrig, J., Mayer, N., Ilg, E., Dosovitskiy, A., & Brox, T. (2017). DeMoN: Depth and motion network for learning monocular stereo. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 5622–5631. https://doi.org/10.1109/CVPR.2017.596
Verdie, Y., Yi, K. M., Fua, P., & Lepetit, V. (2015). TILDE: A Temporally Invariant Learned DEtector. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *07-12-June*, 5279–5288. https://doi.org/10.1109/CVPR.2015.7299165
Wang, J., & Li, C. (2007). Acquisition of UAV images and the application in 3D city modeling. *International Symposium on Photoelectronic Detection and Imaging 2007: Image Processing*, *6623*(March 2008), 66230Z. https://doi.org/10.1117/12.791426

Yan, W. Y., Shaker, A., & El-Ashmawy, N. (2015). Urban land cover classification using airborne LiDAR data: A review. *Remote Sensing of Environment*, *158*, 295–310. https://doi.org/10.1016/j.rse.2014.11.001

Yang, G., Manela, J., Happold, M., & Ramanan, D. (2019). Hierarchical Deep Stereo Matching on High-resolution Images. *Computer Vision and Pattern Recognition*.

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? *Advances in Neural Information Processing Systems*, *4*(January), 3320–3328.

Žbontar, J., & Le Cun, Y. (2015). Computing the Stereo Matching Cost with a Convolutional Neural Network. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *07-12-June*(1), 1592–1599. https://doi.org/10.1109/CVPR.2015.7298767

Žbontar, J., & Lecun, Y. (2016). Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. *Journal of Machine Learning Research*, *17*, 1–32.

Zeiler, M., & Fergus, R. (2014). Visualizing and Understanding Convolutional Networks. In *ECCV*. https://doi.org/10.1016/j.ancr.2017.02.001

Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J. (2017). Pyramid Scene Parsing Network. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 6230–6239. https://doi.org/10.1109/CVPR.2017.660

Zhong, Y., Dai, Y., & Li, H. (2017). Self-Supervised Learning for Stereo Matching with Self-Improving Ability. *ArXiv Preprint ArXiv:1709.00930*. Retrieved from http://arxiv.org/abs/1709.00930