Deep Neural Network Architectures and Learning Methodologies for Classification and Application in 3D Reconstruction

Timothy Forbes

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Computer Science at

Concordia University

Montreal, Quebec, Canada

December 2018

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: Timothy Forbes

Entitled: Deep Neural Network Architectures and Learning Methodologies for Classification and Application in 3D Reconstruction

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final Examining Committee:

_____ Chair
T. Glatard

_____ Examiner
A. Krzyzak

_____ Examiner
T. Popa

_____ Co-Supervisor
S. P. Mudur

_____ Co-Supervisor
C. Poullis

Approved by _____
Chair of Department

_____2019 _____
Dean of Faculty

ABSTRACT

Deep Neural Network Architectures and Learning Methodologies for Classification
and Application in 3D Reconstruction

Timothy Forbes

In this work we explore two different scenarios of 3D reconstruction. The first, urban scenes, is approached using a deep learning network trained to identify structurally important classes within aerial imagery of cities. The network was trained using data taken from ISPRS benchmark dataset of the city of Vaihingen. Using the segmented maps generated by the network we can proceed to more accurately reconstruct the scenes by a process of clustering and then class specific model generation. The second scenario is that of underwater scenes. We use two separate networks to first identify caustics and then remove them from a scene. Data was generated synthetically as real world datasets for this subject are extremely hard to produce. Using the generated caustic free image we can then reconstruct the scene with more precision and accuracy through a process of structure from motion. We investigate different deep learning architectures and parameters for both scenarios. Our results are evaluated to be efficient and effective by comparing them with online benchmarks and alternative reconstruction attempts. We conclude by discussing the limitations of problem specific datasets and our potential research into the generation of datasets through the use of Generative-Adverserial-Networks.

ACKNOWLEDGEMENTS

I would like to thank Charalambos Poullis for convincing me to pursue my studies and continue on with this Masters. After a fulfilling two years I have come to appreciate everything I have learned regardless of the trouble I went through learning it. I would also like to thank Sudhir Mudur and Poullis for their amazing help and presence as my supervisors. They were always there for me and while my communication skills were often lacking, their patience and understanding was not.

CONTRIBUTION OF AUTHORS

I find it necessary to mention Mark Goldsmith whose previous work in our lab helped guide decisions taken in a portion of the work detailed in this thesis.

Contents

List of Figures

List of Tables

1 Introduction

The reconstruction of 3D models of objects or scenes has always been one of the main objectives in computer vision. Typically, classification has been used to assist the process by identifying the class of each object prior to its reconstruction. Many successful applications which follow this paradigm have already been proposed and it has been shown that there is a direct dependency between the accuracy of the classification and the quality of the reconstruction. This is because if prior information about the object is known i.e. its class, then assumptions can be made which can simplify and further improve its reconstruction.

In this thesis, we investigate the use of deep neural network architectures and learning methodologies for addressing the problem of classification and thereby improving the quality and accuracy of reconstruction. We present two different case studies with distinct characteristics and requirements.

Firstly, we focus on *supervised learning* for the classification and reconstruction of large-scale urban areas for which *training data is available*. We present a novel network architecture which (i) reduces the number of parameters in the network and (ii) eliminates the need for post-processing. We also propose a number of reconstruction algorithms for each class.

Secondly, we focus on *supervised learning* for the classification of caustics and reconstruction of underwater scenes for which *training data is very hard to create and to the best of our knowledge, not presently available from any external source*. We present how synthetic data can be used to train a novel, fast, two-stage network architecture for identifying and removing caustics in real underwater videos. We show that removing caustics using the proposed technique prior to reconstructing achieves better results in terms of accuracy for shallow underwater objects.

1.1 Case Study #1: Large-scale Urban Areas

For large-scale urban areas, data is typically found in the form of topographic images of the site with corresponding ground truth. These images are semantically segmented into a set of classes of geospatial features e.g. buildings, roads, trees, etc. Given a semantically segmented image and its corresponding depth map one can use the segmentation to drive the reconstruction of a virtual scene representing the urban area. Thus, there is a clear dependency between the accuracy of the semantic segmentation and the final reconstruction.

In order to achieve high-accuracy semantic segmentation a Convolutional Neural Network (CNN) is used. To train the network we use the ISPRS (International Society for Photogrammetry and Remote Sensing) benchmark dataset [1] and in particular a dataset of an urban area from a historical city in Germany (Vaihingen). The data is in the form of high resolution orthophotos and a depth map generated using Structure-from-Motion(SfM) and Multi-View Stereo(MVS) techniques. The geospatial feature classes present in the data are buildings, roads, trees, low vegetation, cars, and clutter which are then converted into virtual representations of the urban area.

1.2 Case Study #2: Underwater Scenes

In the case of underwater scenes the automated processing traditionally involves extracting and describing distinctive features in each of the images, matching them, and then applying a combination of Structure-from-Motion(SfM)-based and Multi-View Stereo(MVS)-based techniques. In the context of underwater archaeology, this has been shown to produce good results [2] especially in deep waters i.e. $> 40m$, where no natural light reaches the site. However, in shallow waters i.e. $< 10m$, these techniques almost completely fail because of natural light coming in from above the water surface causing caustic effects on the seabed. Fast changes in illumination caused by the rapid motion of the water surface adversely affect feature detection and matching. As a result, all subsequent processing results in erroneous results. Real ground truth for caustics is not available and is extremely labour intensive to generate. One method of training, in this work, is done by synthesizing a shallow underwater environment and then rendering images with and without caustics to use as a training dataset. The network is then applied on the real data for caustic classification and removal and subsequent 3D underwater object reconstruction.

1.3 Technical Contributions

Out contributions are:

- A complete framework for geospatial feature classification and reconstruction of large-scale urban areas. We present the design and development of a novel network architecture for supervised learning which reduces the number of parameters and eliminates the need for post-processing of the semantic segmentation results.
- We propose a pipeline framework for automatically processing the semantic segmentation result and generating 3D models according to the class of each object in the image.
- The design and development of a two-stage network architecture: (i) SalienceNet, a small and easily trainable CNN architecture for the classification of caustics, and (ii) DeepCaustics, a CNN architecture for the removal of caustics. Although, the networks are trained using synthetic data, we show that real data can be processed quickly and successfully by this trained network to generate higher accuracy reconstruction of underwater objects.

1.4 Thesis Organization

The current chapter establishes a basis behind the research done in this thesis. Chapter 2, provides a comprehensive set of definitions and explanations for the terminology and concepts used within the work, and serves to clarify a number of terms unambiguously. In Chapter 3 we explore various techniques that address the same or related problems addressed in this thesis. Chapter 4 presents a pipeline centered around a novel deep neural network architecture

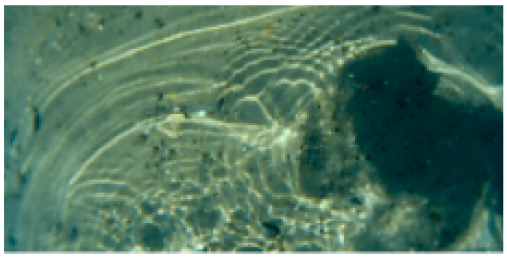
proposed for semantic labeling and subsequent reconstruction of urban scenes, labelled as Case Study #1. Chapter 5, Case Study #2, presents a two-stage, lightweight, fast, dual deep neural network architecture used for segmenting and removing caustics from underwater images. In Chapter 6 we summarize the proposed techniques and methodologies, discuss the challenges faced We identify potential research directions for future work in Chapter 7.

2 Fundamentals

Below we provide a brief summary of the most important terminology and concepts relating to deep learning and in particular the network architectures and learning methodologies employed in this thesis.

2.1 Keywords

Caustics are complex physical phenomena resulting from the projection of light rays being reflected or refracted by a curved surface as shown in Figure 1a.



(a) Ocean floor disrupted by caustics

(b) Image is classified into the plane class.

Figure 1

Image classification involves classifying an image into a finite set of predefined classes. An example is shown in Figure 1b where an image containing a plane is classified/mapped onto its textual label/class.

Image generation refers to creating an image given a vector of inputs. This is also commonly referred to as the decoder in some convolutional neural networks.

Semantic segmentation merges image classification and generation to produce a labeled image where each pixel in the image represents a class. This is also commonly referred to simply as classification. An example is shown in Figure 2 where the image is mapped into a semantic segmentation containing six classes.

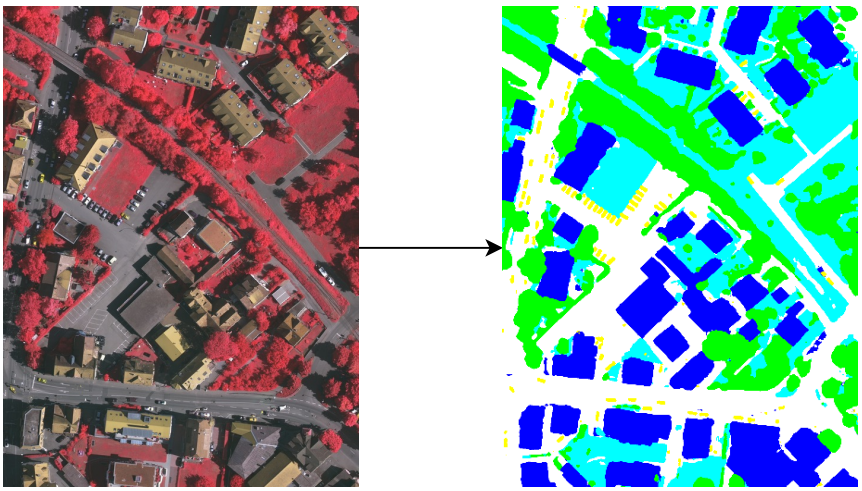


Figure 2: Each pixel in the generated image has been classified into six different classes (one class not present in the image).

Ground Truth is the desired output of the network, the correct answer that the network must achieve. In most cases the ground truth has to be created with considerable manual effort.

2.2 Concepts

Deep learning is a form of machine learning that consists of multiple layers of non-linear processes with modifiable parameters. Information is fed forward through the layers achieving multiple hierarchies of abstraction. Deep learning refers to large neural networks that employ supervised or unsupervised learning for their training.

Supervised learning works by producing an error value as a result of a comparison between the output of the network and the desired output, or, ground truth. This error value is then back-propagated through the network in order to modify the networks parameters such that the error is minimized on the next pass. In a simplistic case, as seen in Figure 3, the error can be interpreted as a convex curve. Each point on the curve represents an error produced by the network as a function of its parameters. Using the negative gradient of the error with respect to the parameters allows a modification of the parameters in the direction of the minimum. This is commonly referred to as **gradient descent** and is a fundamental aspect of neural network training [3; 4].

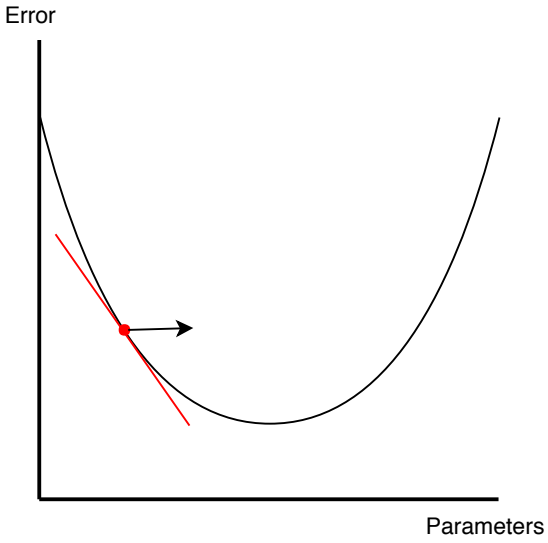


Figure 3: The red line represents the gradient and the arrow represents the direction in which to modify the parameters in order to minimize error.

Unsupervised learning techniques do not have any ground truth to derive an error. Error values are generated using a variety of different ways including probability distribution divergence [5] and pattern analysis [6].

CNNs or Convolutional Neural Networks are a type of neural network consisting mainly of convolution layers, hence its name. A **convolution layer** operates by traversing an image's pixels using a kernel of a predetermined size. The **kernel** is a block of weights, or parameters, that are multiplied to an image's pixels and produce a value representing a pixel in a new image of some higher abstraction. Say we have an image I and a kernel W of size 3×3 then the operation to produce the new image I^* would be as follows:

$$I^*_{i,j} = \sum_{x=0}^{3} \sum_{y=0}^{3} W_{x,y} I_{i+x,j+y} \tag{1}$$

Often, as in many other neural networks a bias is added to the result.

Deconvolution layers also, and more correctly, known as transposed convolution layers, are present in image generation. Normally a convolution layer will reduce the spatial dimension and increase abstraction. Deconvolution layers tend to decrease abstraction into higher image dimension. This layer essentially performs the inverse of the convolution producing multiple pixel values given a single input pixel [7].

ReLU or rectified linear units are a method of introducing non-linearity into the feed-forward process of learning. It outputs the input value if it is greater than zero or zero, as can be seen in (2) [8]. This activation function is the one

most used in this work and generally follows a convolution layer.

$$f(x) = max(x, 0) \tag{2}$$

In a CNN, there is also commonly a **pooling layer**, which reduces an area of pixels into an average (3) or into the maximum (4). Given a pooling size of 2×2:

$$I^*_{i,j} = \frac{1}{4} \sum_{x=0}^{2} \sum_{y=0}^{2} I_{i+x,j+y} \tag{3}$$

$$I^*_{i,j} = max(I_{i,j}, I_{i+1,j}, I_{i,j+1}, I_{i+1,j+1}) \tag{4}$$

Batch Normalization is a process of normalizing the output of the preceding layer. It subtracts the batch mean and divides by the batch standard deviation. This allows for separation of learning between layers [9].

GANs or Generative-Adversarial-Networks consist of two networks competing against each other. One network, the discriminator, learns to identify real images from fake images. The other network, the generator, learns to generate images that look real. As the discriminator gets better at determining the false generated images the generator gets better at producing more realistic ones. Ideally the system converges when the discriminator can only tell with 50% certainty that a generated image is fake [5]. A common analogy is that of a counterfeiter making fake currency and a policeman identifying them. As the policeman learns to identify the counterfeit bills the counterfeiter learns to make them more realistic. Effectively, the generator learns to implicitly match the distribution of the real images. Upon convergence the generator can be used to produce realistic images within the distribution of the real images.

3 Related Works

Below we provide a brief overview of the state-of-the-art in the areas of (i) semantic segmentation, (ii) caustic removal, and (iii) urban and underwater scene and reconstruction.

3.1 Semantic Segmentation

Both in urban semantic segmentation and caustic labeling the goal is to achieve semantic labeling in order to proceed onto the next parts of the pipeline. The problem of semantic labeling has been explored through a variety of methods. Prior to deep learning, semantic labeling relied on hand engineered features. One of these methods proposed the generation of features that were classified into unary potentials then fed into conditional random fields (CRF), localizing the label and segmenting object instances [10].

Deep learning, over the past few years, has proved to be very effective at object recognition due to its ability to learn important features. Convolutional neural networks, first introduced by Yann Lecun in 1989, provide an effective method of extracting important features [11]. Girshick et al. use convulutional layers as a step within their system of segmentation [12]. One of the first cases of solely using deep learning in semantic segmentation was done by Long et al. in their work: "Fully Convolutional Networks for Semantic Segmentation". They employ a network that uses convolutional layers to abstract high level information which can then be used to infer a label map [13]. However, applying deep learning to semantic segmentation becomes a challenge as localized information is often lost in favor of high-level information. Chen et al. [14; 15] apply a CRF or a discriminatively trained domain transform to the model's output to preserve edge information and to smooth semantic segmentation. Noh et al. [16] perform deconvolution to reach the original input resolution allowing the network to learn localization through deconvolution kernels.

Other works have the deep network perform the segmentation by preserving low-level information for the network's segmentation process. SegNet [17], the network that inspired our work, consists of encoders and decoders that share pooling indices in order to preserve lower level information. Several other works apply this concept with slight variations [18; 19]. Pohlen et al. keep a single residual stream with information at the original resolution [20]. One network consists of holding previous pixel-specific layer activations within vectorized columns [21]. Another uses visual and geometric cues during unpooling [22]. Lin et al. use separate network paths to capture all available information from earlier layers [23]. A few high-performing networks base their work on the idea that a convolution has less contextual reach than assumed and they employ larger kernels [24], global image information [25], and different pooled feature maps [26; 27]. A major contribution to this approach in semantic segmentation was that of dilated con-

volutions. High resolution images are generally very heavy to process and depending on the kernel size there maybe be many operations that do not benefit to the abstraction of higher, larger context level features. Dilated convolutions introduced by Yu and Koltun allows the aggregation of multi-scaled contextual information without loosing resolution [28].

The previously mentioned works have a tendency to perform pixel-to-pixel level classification, this means that the classification can be done on a variable sized input to the model. Another approach is that of patch-to-pixel which occasionally performs well [29; 30; 31] but comes at the cost of processing time and neighboring contextual patch information.

Variants of the aforementioned network architectures have been employed in the context of semantic labeling of geospatial features each with unique advantages and trade-offs [32].

3.2 Caustic Removal

Although a plethora of work has already been reported on caustics generation, only a handful of techniques have been proposed for caustics removal; the majority within the context of image enhancement.

Trabes and Jordan [33] present a technique for tuning a sunlight-deflickering filter for moving scenes underwater. They propose a continuous parameter optimization inside a basic filter, which employs feedback in order to improve the performance. As reported in their paper there is a high sensitivity of the filter's performance to badly optimized parameters and in particular, the segmentation parameter which is part of the objective function in the optimization.

A different approach was presented by Gracias et al. [34]. The authors present a mathematical solution which involves calculating the temporal median between images within a sequence. A strong assumption of this work, is the fact that feature matching (a Harris corner detection variant established by Gracias and Santos-Victor [35]) is employed for the formation of the sequence which makes this approach very susceptible to the light variations in the images and in particular caustics effects. This work has since been extended in Shihavuddin et al. [36] by proposing an online sunflicker removal method which treats caustics as a dynamic texture. As reported in the paper this only works if the seabed or bottom surface is flat. Similar approaches have also been proposed for general cases of dehazing and descattering of images [37; 38; 39].

Schechner and Karpel [40] propose a method based on processing a number of consecutive frames. These frames are analyzed by a non-linear algorithm which preserves consistent image components while filtering out fluctuations. Their proposed method however does not take into account the camera motion which almost always leads to registra-

tion inaccuracies.

In order to avoid registration inaccuracies Swirski and Schenchner [41] present a method for removing caustics using a stereo-rig. The stereo cameras provide depth maps which can then be registered together using ICP (Iterative Closest Point; a technique for aligning multiple geometries to one another). This again makes a strong assumption on the rigidity of the scene which is seldom the case in underwater.

Trabes and Jordan, propose an approach which employs optical flow techniques and curvature predictions of pixel traces during the motion [42]. As with the aforementioned technique there are strong assumptions on the small motion and color constancy between consecutive frames.

3.3 Reconstruction

Reconstructing caustic scenes has been rarely attempted. Some tried using structure from motion techniques with poor results [43; 44]. Although the scene is static and the camera is moving smoothly the variation in illumination throws off the feature matching algorithm.

Urban reconstruction, on the other hand, has been an active research area since the early 80s hence it is no surprise that a vast body of work exists. A comprehensive survey of state-of-the-art can be found in Musialski et al. [45] where techniques proposed over the past few years are categorized according to the objective, type, and scale of data.

There are techniques which use symmetries and regularities in the geometry. Zhou et al. [46] proposed an automated system which given the *exact bounding volume of a building* can simplify the geometry based on dual contouring while retaining important features. Using this technique the authors were able to simplify the original geometry considerably. On a similar line of research, Verdie et al. [47] proposed a method which produces excellent reconstructed models from pointcloud data which can also produce models at different levels-of-detail. Although tghbese techniques produce impressive results, they do require considerable user interaction during the pre-processing stage typically in the form of carefully identifying the objects' points.

Other techniques aim for full-automation and are therefore entirely data-driven. One such example is the work of Poullis et al. [48] where pointcloud data is converted automatically to polygonal 3D models. This technique is applicable directly on the raw pointcloud data without requiring any user interaction. Later, Poullis [49] extended the work to include a fast boundary refinement algorithm based on graph-cuts which was used to refine the boundaries. Overall, these techniques scale well with vast amounts of data however this comes at the cost of increasing the difficulty of enforcing symmetry constraints such as the Manhattan world assumption. In other words, larger areas can be

processed but the generated models are noisier than the previous approach of using regularities. A solution to this side-effect was proposed by Arikan et al. [50] where a system for generating polyhedral models from semi-dense unstructured point-clouds was developed. Planar surfaces were first extracted automatically based on prior semantic information, and later refined manually by an operator.

Finally, a rather different approach was proposed by Xiao et al. [51] where the authors used inverse constructive solid geometry to address the reconstruction problem. Rather than using boolean operations on simple primitives to generate a complex structure, they start off with a point cloud representing the indoor area of a structure and decompose that into layers which are then grouped into higher-order elements. This works very well for highly regularized scenes such as indoor spaces however it does not produce useful results for large-scale outdoor areas.

4 Case Study #1: Urban Scene Classification and Reconstruction

4.1 Overview

For reconstruction of urban scenes our first step is to semantically segment imagery into labelled data using a deep learning network with a distinct architecture which we have named as SegNeXt. The imagery input into SegNeXt is two geo-registered remote sensing images; an orthophoto in the form of IR-RG (InfraRed-Red,Green) and a corresponding depth map. The images are classified using the proposed SegNeXt which produces smooth predictions while retaining high frequency details. The SegNeXt predictions are then used as a proxy in grouping the 3D points into disjoint and contiguous clusters. Finally, to reconstruct the scene each cluster is processed depending on its classification: for buildings the boundaries are extracted and 3D polygonal models are extruded, trees are replaced by procedural models and their placement is determined by a Voronoi tessellation of the cluster, cars are replaced by simplified CAD models, and roads, low vegetation, and clutter are replaced by a simplified mesh of the terrain. Figure 4 shows a diagram of the pipeline of the proposed framework. This work has been published in the 15th Conference on Computer and Robot Vision [52].

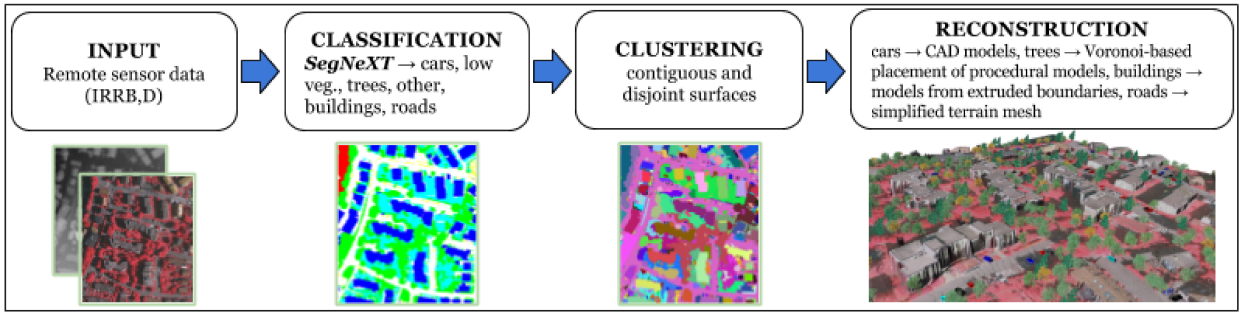


Figure 4: System pipeline.

4.2 Urban Classification

4.2.1 Dataset

The training and testing of the network is performed on data provided by the International Society for Photogrammetry and Remote Sensing (ISPRS) as part of a benchmark competition on urban object detection and 3D building reconstruction [1]. Multiple datasets are available for each competition i.e. 2D semantic labeling, 3D semantic labeling, 3D reconstruction, etc. In this case our objective is 2D semantic labeling thus we used the Vaihingen dataset which consists of a total of 33 pairs of IRRG (infrared, red, green) images and their associated depth maps. Each image is an

orthophoto of the historical city Vaihingen, Germany, has an average resolution 2000×2500, and a sampling density of $9cms$. For each IRRG-depth image pair, a ground-truth image is also provided showing the manually assigned per-pixel classification into six classes: (a) buildings (blue), (b) roads (white), (c) trees (green), (d) red (clutter), (e) low vegetation/natural ground (cyan), (f) cars (yellow). The 'clutter' class contains areas for which a class could not be assigned e.g. water, vertical walls, areas where computing the depth (SfM+MVS) has failed, etc. The 'low vegetation/natural ground' class contains areas on the ground covered by vegetation other than trees such as low bushes, grass, etc.

4.2.2 Network Architecture

A number of network architectures have already been reported for semantic labeling. State-of-the-art performance is generally associated with how deep and wide the networks are. However this introduces a significant drawback since the deeper/wider the network, the larger the number of parameters that need to be optimized during the training phase; and although training time is often overlooked it is an important factor when assessing the overall applicability and deployment of a network architecture.

Furthermore, when dealing with deep network architectures the resolution of the input data deteriorates as the data progresses through to the deeper layers. This often materializes as non-smooth and noisy predictions during the final upsampling stages in the network. A common way of addressing this is to smooth the predictions using a conditional random field based post-processing approach.

In our work, we propose a distinct network architecture which uniquely combines the strength of convolutional autoencoders with feed-forward links in generating smooth predictions and reducing the number of learning parameters, with the effectiveness which cardinality-enabled residual-based building blocks have shown in improving prediction accuracy and outperforming deeper/wider network architectures with less learning parameters, to address the aforementioned limitations of existing state-of-the-art. The closest related work in terms of network architecture is with SegNet presented in [17] where the concept of feeding forward *pooling indices* from the encoders to the decoders was introduced, and the ResNeXT building blocks presented in [53] where the concept of *cardinality* was introduced and was shown that increasing cardinality was more effective than deeper/wider network architectures. Hence, to summarize, the topology of the proposed network resembles that of SegNet with the main differences that the feed forward connections are between feature maps (as opposed to pooling indices) and each encoder or decoder consists of a ResNeXT block.

The main structure of our network can be seen in Figure 5 in comparison with SegNet and ResNeXT architectures.

The network consists of a set of encoders followed by a corresponding set of decoders. Information in the form of *feature maps* - as opposed to SegNet's pooling indices - is fed forward to each decoder from its respective encoder. This results in retaining high-frequency information therefore improving boundary delineation which in turn results in smoother predictions.

The internal architecture of each encoder is that of a ResNeXT block. These blocks consist of convolutions applied across a group of feature maps that have been evenly split and are then concatenated back together. For example, a block inputs a set of 128 feature maps that are then sliced into 4 sets (i.e. cardinality is 4) of 32 feature maps, a different convolution kernel is applied to each set and the resulting feature maps are concatenated back into the 128. This operation is known as the split-transform-merge technique [53]. All other convolutions are followed by a batch normalization [9] and a ReLU activation layer.

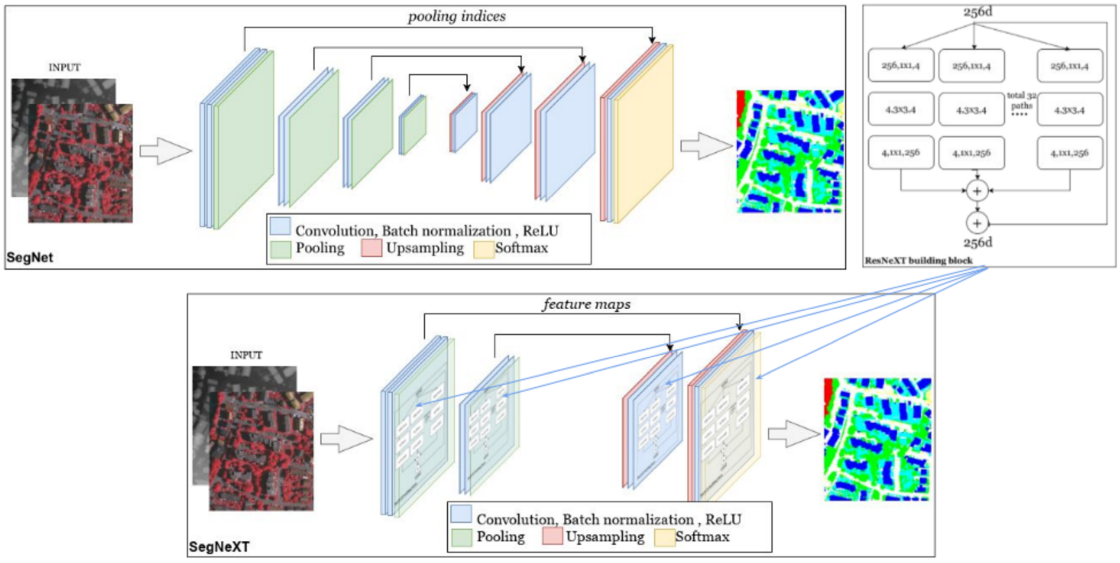


Figure 5: Top: SegNet architecture, ResNeXT block. Bottom: Our SegNeXt architecture.

4.2.3 Training

The network is trained on all of the available data with their corresponding ground truth. We decided to forego validation in order to maximize the available data for training. The training is performed for 2000 epochs on a single nVidia GTX 1070. We used the Keras (with Theano) API for the development and training/testing of the network, and the code is available as open source. Our network trains using Keras' generator method which establishes one epoch after a certain stepsize, 64 in our case. This means each epoch consists of 64 groups containing the 32-batch samples. We trained for 11 days for 2000 epochs with each epoch requiring 500 seconds to complete.

Data Input

Our network takes in patches of 150×150 that are selected from the 16 available training images. We select random points within each image and construct a patch around each point. This decision is based on the observations that (a) given the high resolution of the images random sampling results in patches with large content variability, and (b) due to this large content variability (i.e. 16×5 million patches per $2K \times 2.5K$ image) the overall accuracy on the training images can be used as a proxy (i.e. almost the same) for the overall accuracy on the validation/testing images.

Each patch is represented as a 3-dimensional tensor containing the IRRG data and the corresponding depth map. We select 2 patches from each of the 16 image-pairs for each batch resulting in a total batch size of 32 different patches of $150 \times 150 \times 4$. Class balancing (near equal training samples from different classes) was also tested but did not show any improvement in learning which could be attributed to the large amount of data used.

4.2.4 Validation

The proposed network was validated against the additional 17 image pairs available for testing for which ISPRS did not make the ground truth publicly available. In order to generate our test images we employ a sliding window approach that evaluates each patch at intervals of 10 pixels in the diagonal in order to minimize context based errors. Our results are then averaged into one final image.

The network's performance is measured in terms of Precision (P), Recall (R), and F_1 score which are defined as,

$$P = \frac{tp}{tp + fp} \qquad R = \frac{tp}{tp + fn} \qquad F_1 = 2 \times \frac{P \times R}{P + R} \tag{5}$$

where tp indicates the true positives, fp indicates the false positives, and fn indicates the false negatives.

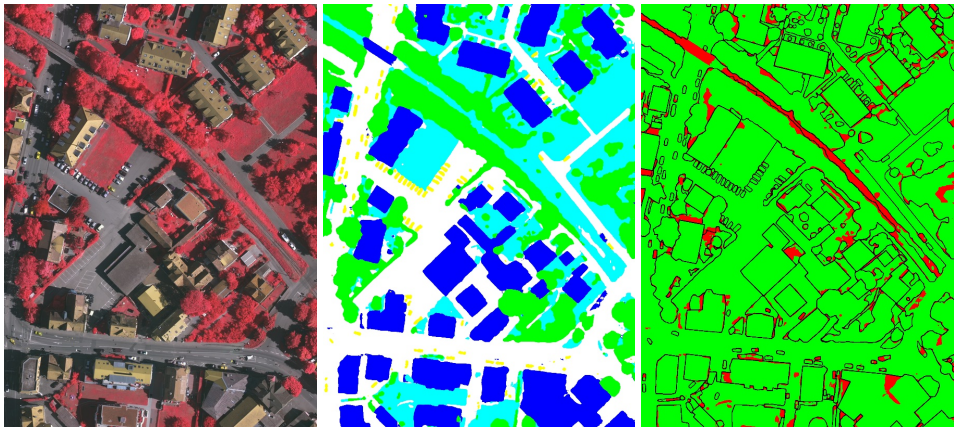
Table 1 shows the evaluation of the overall classification results for the 17 test images and as it can be seen the overall accuracy is 89.2%. At the moment of writing the highest overall performance is 91.2% from a deep fully-convolutional neural network (FCN) ensemble followed by post-processing using a fully connected CRF (F-CRF) for further improving the results. Close inspection of our evaluation results indicate that there is about a $1 - 2\%$ variation between our classification (buildings, trees, roads, and clutter) and the state-of-the-art, and a highest difference of about 4% for the car class. We can only assume that the ensemble consists of networks tuned at different scales although no publication is available to corroborate this. Despite the lower overall accuracy on the entire test dataset, there are cases where the overall accuracy of our network outperforms the state-of-the-art, such as test images V-0004 shown in Figure 14b. This can be attributed to the fact that our network is under-performing (compared to state-of-the-art) in

classifying cars so in the presence of many cars in the test image the overall accuracy drops; similarly, in cases where not many cars are present in the test images the overall accuracy increases and surpasses other competing networks. Figures 6 and 7 show the results for two of the 17 test images, namely V-0003 and V-0012.

Furthermore, we have also tested variations of the proposed network architecture. In particular, we have experimented with (a) data augmentation, (b) atrous convolution [54], (c) CRF-based post-processing. There were no improvements in the overall accuracy which was in fact lower in the range of $84 - 89\%$. For the CRF-based post-processing the results were almost identical i.e. 89.1% which verifies the claim that deep autoencoders with feed-forward links between feature maps produce smooth, non-noisy results.

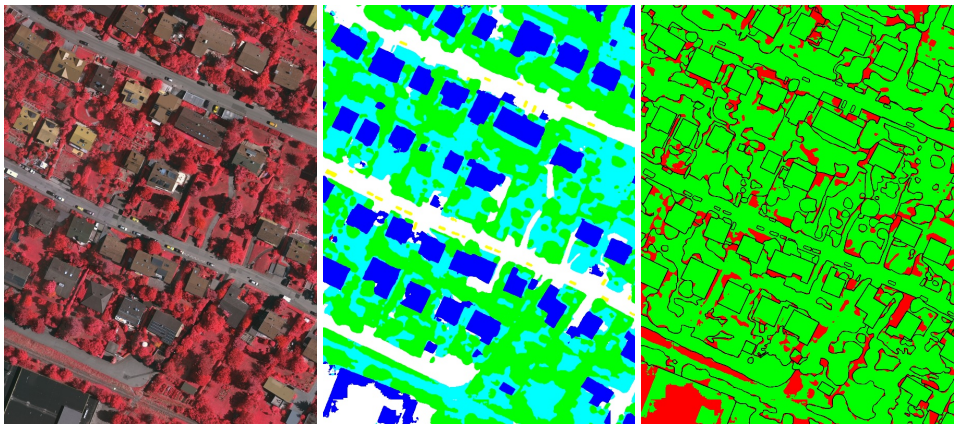
ref. $\rightarrow$ pred. $\downarrow$	roads	building	low veg	tree	car	clutter
roads	0.937	0.023	0.031	0.007	0.002	0.000
building	0.048	0.931	0.018	0.003	0.000	0.000
low veg.	0.047	0.015	0.800	0.138	0.000	0.000
tree	0.010	0.003	0.077	0.911	0.000	0.000
car	0.209	0.056	0.006	0.003	0.726	0.000
clutter	0.379	0.345	0.016	0.003	0.054	0.204
Precision	0.896	0.949	0.847	0.865	0.862	0.979
Recall	0.937	0.931	0.800	0.911	0.726	0.204
F1	0.916	0.940	0.823	0.887	0.788	0.338

Table 1: The overall evaluation of the classification results for the 17 test images for which ground truth was not provided. The network performance statistics were computed by and provided by the ISPRS Working Group II/4 organizers as part of their urban classification benchmark. All shown values are percentages.



(a) (b) (c)

Figure 6: The evaluation result for one of the 17 test images. Resolution 1887×2557 (a) Satellite image of the urban area V-0003. (b) Generated label map. (c) Red/green image, indicating wrongly classified pixels. The railroad is classified incorrectly as a low vegetation instead of clutter.



(a) (b) (c)

Figure 7: The evaluation result for another of the 17 test images. (a) Satellite image of the urban area V-0012. (b) Generated label map. (c) Red/green image, indicating wrongly classified pixels.

4.2.5 Network Comparison

Results on this dataset are posted to the public on the ISPRS benchmark website. From there we could gather information about higher performing networks, if the authors have made that information available with their submission. Unfortunately not many of the highest performing networks give detailed information, although a brief abstract of their work is sometimes found. As mentioned earlier, at the time of submission, the highest performing network achieved accuracy of 91.2%. This network, labelled NLPR3, is defined as an ensemble of fully-convolutional neural networks followed by fully connected conditional random fields. The use of ensemble networks generally means that the size of the whole system is significantly larger thus also means more training time and memory.

We evaluate other networks that made their work available and that performed better on the dataset compared to our network. We do this in an attempt to further understand how improvements can be made or to justify and explain why other networks had higher accuracy.

Liu et al. present the second best performing network in table 2. Their network comprises of a similar encoder-decoder structure. However they implement a cascade of multi-scale contexts appended to their encoder. They do this in order to distinguish confusing man-made objects present in the scene. Their work is well explained a performs extremely well given they do not use and depth data. Their model is appended onto existing encoder and so its size is not much greater than any of the current state of the art networks. Unfortunately their work was published later than our submission and could not be used as insight to any valuable decision [55].

Wang et al. also perform exceptional results with their GSN submission considering they forgo the use of DSM in their training. Their model consists of using a gated convolutional network that allows more effective information to

be passed from feature maps of different sizes. The only downside of their work is the size of their network as it bases off of ResNet and hence requires a lot of memory and training time [56]. Marmanis et. al achieve similar performance to GSN with their DLR_10 submission. They make use of DSM and nDSM and implement an explicit edge detection in the intial phases of their network. They also make use of an ensemble network meaning their system is very large and requires a lot of memory to run in parrallel or time to run sequentially [57].

The RIT submission explores networks that consist of fusing branches of the convolutional streams and then concatenating those features for further processing to obtain the final output result. Some of their networks perform well and require little training parameters however their most performing one, the composite fusion network, requires larger branches of network streams meaning more parameters and longer computation time. They also make use of DSM and nDSM in their network [58].

Finally the INR submission makes use of DSM in a similar network to SegNet but instead of upsampling and convolving they unpsample every encoder result and then contantenate everything into a final output. This results in a network with fewer parameters [59].

It is important to note that the ground truth data was done by hand and as such it is prone to human error which is commonly said as $1\% - 5\%$. This simply means that many different networks could perform better or worse given a different dataset, and given different annotations for the ground truth.

Institution	Road	Building	Low Veg	Tree	Car	Overall	
WAS4	91.9	94.2	82.2	88.6	74.9	89.4	
INR	91.1	94.7	83.4	89.3	71.2	89.5	Paper
RIT_7	91.7	95.2	83.5	89.2	82.8	89.9	Paper
WUH_C5	91.9	95.9	82.8	87.6	74.4	89.9	
BUAA_2	91.6	94.7	84	89.3	86	89.9	
UCal5	92.2	94.8	83.2	89.5	85.7	90.1	
DLR_10	92.3	95.2	84.1	90	79.3	90.3	Paper
GSN3	92.2	95.1	83.7	89.9	82.4	90.3	Paper
HUSTW3	92.1	95.3	85.6	90.5	78.3	90.7	
BKHN_4	92.7	95.1	84.7	89.8	86.6	90.7	
CASIA2	93.2	96	84.7	89.9	86.7	91.1	Paper
NLPR3	93	95.6	85.6	90.3	84.5	91.2	

Table 2: Results of highest performing systems submitted by each institution that performed better that our network. The last column dictates whether the institution attached a paper to their submission.

4.3 Network Design

4.3.1 Structure

Our initial attempts at urban classification consisted of building a network that would identify a single pixel on each forward pass. The reasoning was that as a patch is convolved the data is abstracted from surrounding pixels accumulating into a very condensed array of information that would give an accurate classification of that single pixel. While this proved somewhat true final results were not very good in boundary delineation. We assume this is because a boundary can delineate two classes within one pixel difference, hence two very similar condensed arrays could represent two different classes.

We then attempted a cascade method which consisted of models that were unique to each class. Since each model would produce a probability of a pixel being a specific class or not we could then find the class with the highest probability. Our system consisted of first running the network on input patches of size 55 through a vegetation class network. We would then produce the output from that network and concatenate it to the input and pass it through the tree class network. We proceeded the same way for cars, buildings, and finally ground. While this improved boundary delineation and each model converged faster it still took hours for testing of a single image. We also determined that cascading also ended up accumulating errors.

Our next attempts consisted of simple convolutions followed by deconvolutions. This produced faster results but boundary delineation was not good. Further research pushed the idea to have links between each encoder and decoder block. Around this time ResNeXt [53] was a popular well performing network. Applying it to our network improved performance without increasing parameter size and convergence time. This architecture proved effective and yielded good results so we settled on that.

4.3.2 Parameters

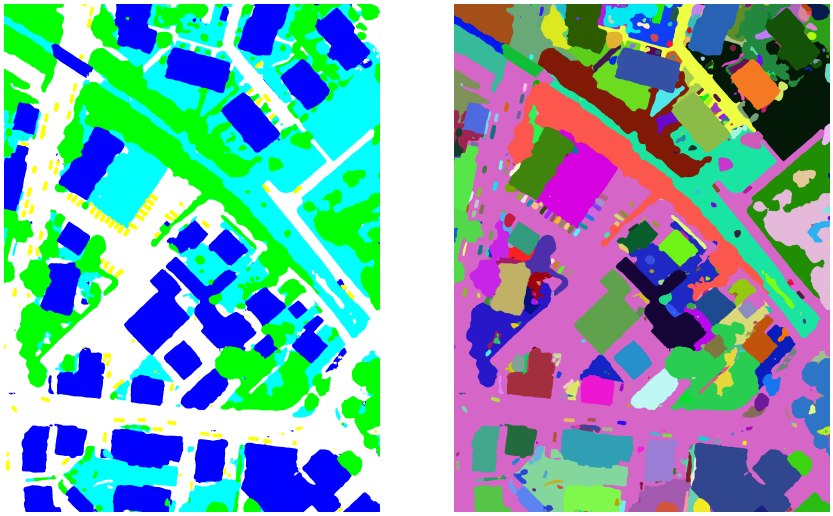
Having such a large network with so many parameters proved extremely difficult. Parameters defining convolution kernel sizes were chosen based on similar models created by other researchers. Slight changes were made but no significant improvements were noted and so some parameters were set for symmetry and ease of presentation. Patch size was determined by testing different models going from 55×55 to 256×256, at different intervals. As patch size increased our batch size had to decrease (due to our hardware memory limitation) during training and testing. Once we reached a patch size of 150 we noted that convergence times were significantly greater and did not justify the slight increase in performance. We also noted that as batch size decreased classes with smaller pixel coverage (such as

cars) performed worse. We later assumed that as batch size decreases variation in pixels does too, meaning that each backward pass generalizes less to less consistent pixels. Class balancing may have been a suitable course of action at this stage if we had thought of it.

Parameters were iterated through randomly, and occasionally manually tested based on educated assumptions. Through an empirical study of the tested parameters we reached a model that performed well and we were able to move onto reconstruction of the urban scenes.

4.4 Urban Reconstruction

The result of SegNeXt is a per-pixel classification into one of the six classes i.e. cars, buildings, trees, roads, low vegetation, and clutter. Next, the 2D per-pixel classification image is used as a proxy to cluster the 3D points in the depth map. This results in a cluster map shown in Figure 8b containing disjoint, contiguous regions. Based on the cluster's classification, an automatic per-class reconstruction is performed to generate the 3D models for each class. Finally, the 3D models are fused together to create a complete virtual representation of the entire site.



(a) (b)

Figure 8: (a) SegNeXt predictions for test image V-0003. (b) Agglomerate clustering of neighbouring points in (a) of the same class.

4.4.1 Buildings

Perhaps one of the most important aspects in urban reconstruction is the accurate modeling of buildings where large depth discontinuities appear as jagged edges in the captured remote sensing data. Typically, a refinement and smoothing operation is performed as a first step prior to further processing.

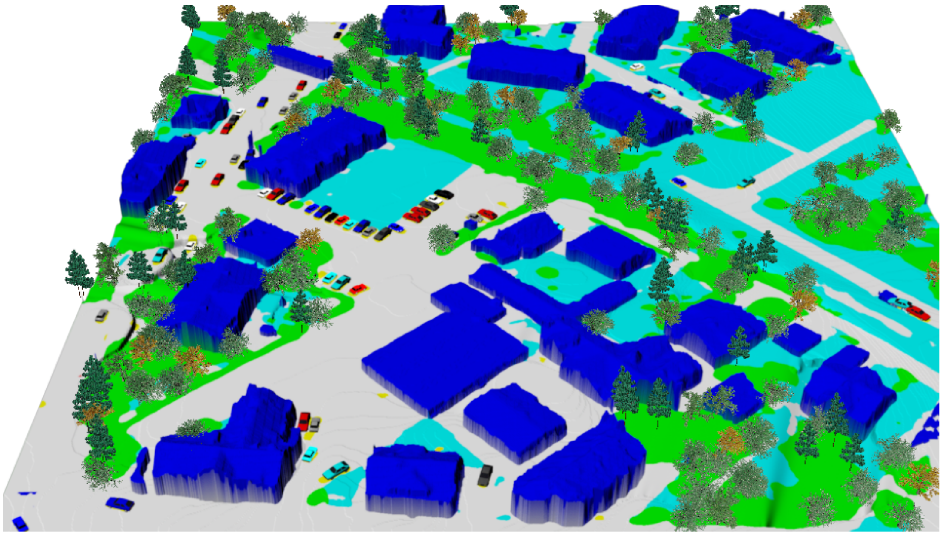


Figure 9: Buildings boundaries are extruded and roof points are triangulated. SegNeXt classification image used as texture.

The main characteristic of the proposed SegNeXt architecture is the fact that its predictions are smooth and do not require a post-processing step of CRF-based smoothing. As it can be seen from the results SegNeXt produced smooth and non-noisy results which were extruded to generate 3D models with smooth boundaries. The 3D points corresponding to clusters classified as buildings are triangulated using a Delaunay triangulation. Figure 9 shows examples of reconstructed buildings. Although the accuracy for buildings is in the high ninetieth percentile for all test images, there are a few cases of misclassified clusters which cannot be identified correctly without additional information.

4.4.2 Cars

Clusters classified as car are removed from the geometry and replaced by simplified car CAD models. Principal Component Analysis (PCA) is performed to determine the dominant orientation of the cluster. Using the dominant orientation the CAD model is correctly placed and oriented with respect to the cluster. Cases where two cars are parked in very close proximity may lead to misclassification. An example is shown in Figure 11. To identify these cases and resolve them we examine the ratio of the cluster's eigenvalues, and its overall area measured in pixels.

From the classifications of the 16 training images we compute the average area of a car; $area_{avg}^{cars}$, and the average ratio of the eigenvalues; $r_{avg}^{cars} = \frac{1}{N}\sum_{i=0}^{N}(\frac{\lambda_{max}}{\lambda_{min}})_{C_i}$ where $0 \leq i \leq N$ and N is the number of all car clusters C_i. Our experiments show that $area_{avg} \approx 2900$ and $r_{avg} \approx 7$ for a car in a remote sensing image with sampling density of $9cms$. Car clusters with $area^{C_i}$ significantly greater than $area_{avg}$ are further processed to determine whether they represent cars in a line ($r^{C_i} > r_{avg}$) or side-by-side ($r^{C_i} < r_{avg}$). Thus, if T cars appear in line then the car i is placed

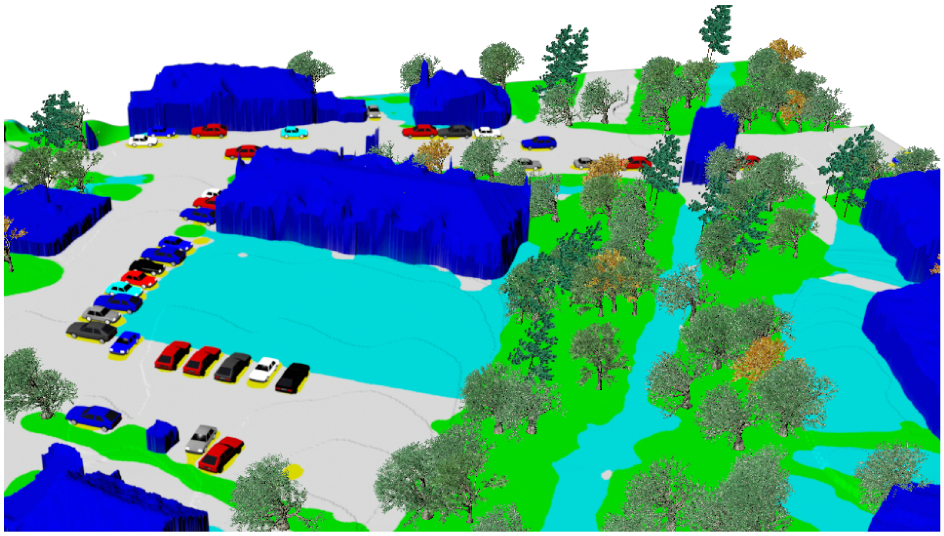


Figure 10: Close-up of a larger area which includes a parking lot demonstrating the cars' modeling and placement.

at location $i - 0.5 \times T \times (r_{avg}cos(\theta), -r_{avg}sin(\theta), 0)$ where θ is the angle formed between the dominant orientation of the current cluster's eigenvectors in terms of the world's X-Y axes. If the T cars are side-by-side the car i is placed at location $i - 0.5 \times T \times (-r_{avg}sin(\theta), r_{avg}cos(\theta), 0)$. If a cluster $(\frac{area_{avg}}{2}) < area^{C_i} < area_{avg}$ then a car is still placed to account for potential errors in classification otherwise the cluster is ignored.

To increase realism a variety of 10 CAD car models has been used. An example resolution is shown in Figure 11. Finally, since the depth map contains the depth values for the roof of the car, we use the average depth value of the hole-filled area corresponding to the car as the ground value in order to place the car. Figure 10 shows a close-up of a larger area including a parking lot where the cars have been replaced.

4.4.3 Trees

Clusters classified as trees are removed from the geometry and are replaced by procedurally generated models. Trees do not have a uniform shape or density which makes it impossible to identify the number of trunks and their precise location without additional information e.g. last pulse from LiDAR. In this work we address this problem by subdividing each classified tree cluster using the Voronoi tessellation. A procedurally generated tree is placed on the Voronoi center which results in a more evenly distributed placement of the trees. An example of tree clusters is shown in Figure 9 and a close up of the procedurally generated models in Figure 12. The Voronoi diagram for the same tree clusters is shown in Figure 13. Similarly with the cars, the depth values of a tree correspond to the top branches of the tree and not to the ground, therefore we use the depth value of the nearest classified road or low vegetation point for the placement of the tree trunk. To increase realism a variety of 8 procedurally generated models have been used.

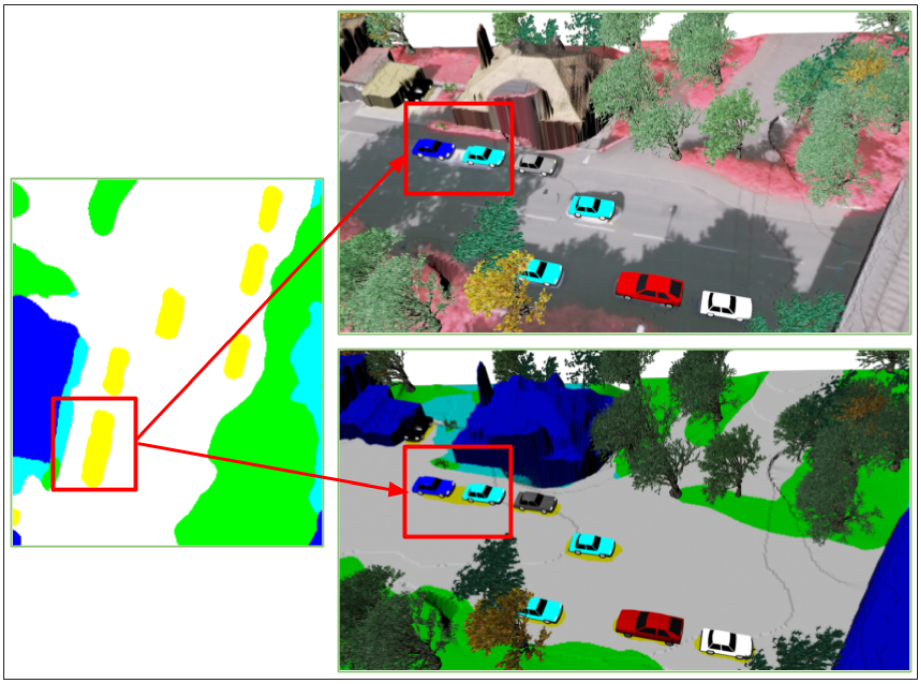


Figure 11: (left) A closeup of a SegNeXt classification result where two cars in a row are classified as one merged cluster. (right) The ratio of eigenvalues and the area are used to identify and appropriately handle special cases such as the one shown on the left where cars are parked in line and in very close proximity. This causes the classification to merge the two clusters into one.

4.4.4 Roads, Low vegetation and Clutter

Points contained in road and low vegetation are grouped together and are converted into a mesh using nearest neighbour triangulation. Holes resulting from the removal of buildings, trees, cars and clutter are filled-in using neighbourhood information. Finally, the dense mesh is further reduced by simplification. Points classified as clutter are ignored as they do not represent any constant representation e.g. vertical walls, railroads, areas where SfM+MVS failed, etc.

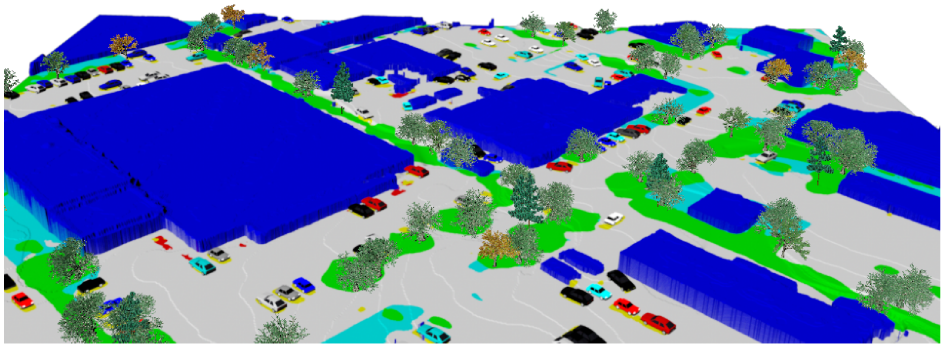
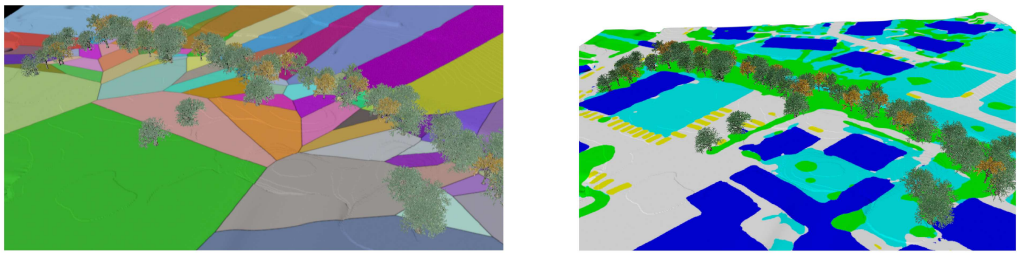


Figure 12: Classified tree clusters are removed from the geometry and are replaced by procedurally generated models.



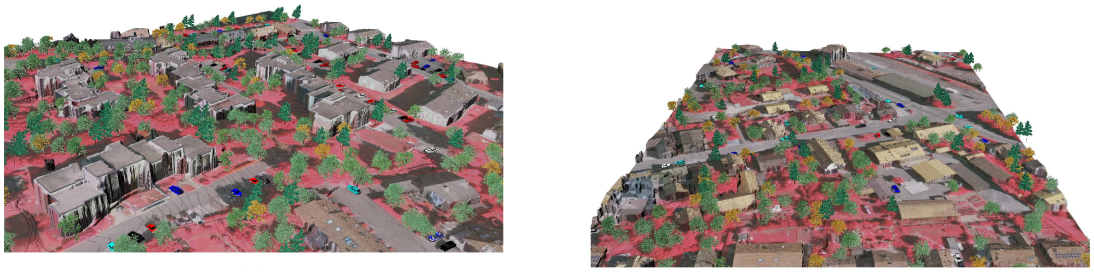
(a) (b)

Figure 13: (a) The Voronoi diagram for the tree cluster in (b). Voronoi clusters are color-coded and centers where the tree trunks are positions are marked as black dots.

4.5 Experimental Results

We tested the proposed framework on all 17 test images of the benchmark dataset. The average time of processing the semantic labels and converting them to 3D models is 10 minutes.

Figure 14a shows the results from V-0000. The provided orthophoto IRRG images are used to texture the models; no facade information is available. The final result of V-0004 is shown in Figure 14b. In Figure 15 we show the result for V-0013.



(a) (b)

Figure 14: Final results for V-0000 (a) and V-0004 (b).

Figure 15: Final result for V-0013.

5 Case Study #2: Caustics Removal and Underwater Scene Reconstruction

5.1 Overview

In order to improve reconstruction quality and accuracy of underwater scenes the first step is to identify caustics and remove them. Caustic detection and removal from real images was attempted using a Convolutional Neural Network with an architecture similar to that of a Convolutional Autoencoder reported in [60], where the first layers perform convolution, and subsequent layers perform de-convolution. CNNs are a natural choice when dealing with images; however, unlike traditional convolutional networks, which rarely attempt pixel level classification, we require an architecture that produces images of sizes matching input image sizes, hence the need for de-convolution layers. In order to perform the removal of caustics on a video we required the network to perform on each frame as fast as possible and hence be very small. SfM can then be used on the resulting images to produce a higher quality and more accurate reconstruction.

Images with caustic are fed into a first network called SalienceNet which outputs a salience map corresponding to the probability of a pixel being a caustic. This salience map is then concatenated to the original image and fed into another network, DeepCaustics, where the caustics are removed from image producing a caustic free image that can be used for more accurate reconstruction. This section of work has been published in the IEEE Journal of Oceanic Engineering [61].

5.2 SalienceNet

5.2.1 Dataset

Since there is no ground truth data available for real world underwater caustics, we decided on supervised learning using synthetic data for ground truth. Using sample underwater video images with caustics we created a set of synthetic data using 3D objects for underwater seabeds, multiple lighting and global illumination for rendering caustic effects with the virtual camera located below the water surface as in real world shallow water imaging. In addition to the rendered caustics images, we rendered the masks containing confidence values of a caustic occurring at a pixel and also corresponding caustic-free images. The dataset was created with Arnold Renderer using photon mapping [62] in Autodesk Maya 2017. The masks containing the confidence values are used as ground truth to the SalienceNet. The input to SalienceNet is a rendered RGB image containing caustics of an underwater scene. The network operates on a batch of 32 images of size 400×400. Each pixel of the output image takes a value in the range of $[0, 1]$, corresponding

to the confidence of caustics occurring at that pixel, hence the output is a single channel black and white image.

5.2.2 Network Architecture

After extensive experimentation, we have concluded that the network architecture with the optimal performance relative to size consists of four hidden layers; the first two consisting of 3 and 5 convolution filters respectively, and the last two consisting of 5 and 1 de-convolution filters respectively. The filter sizes are 5×5, 3×3, 3×3, 5×5 in each layer respectively. This results in a total of $2 \times (5 \times 3 \times 3) + (4 \times 5 \times 5)$ weight parameters and $8 + 6$ bias parameters, for a total of 204 parameters to be learned. Such a small number of parameters means that each forward pass on the network during testing will be extremely fast. This means that a whole video sequence can be processed almost in real-time. Each convolution/deconvolution in the network is followed with a ReLU activation unit [63]. Initially, sigmoid activation units were used in the last layer to ensure that the final output is in the range $[0, 1]$ however, our experiments have shown that ReLU units perform better [they still map the output in the range $[0, 1]$ provided the input data falls within the manifold learned] and, in addition computing the gradients becomes more stable during back-propagation i.e. no 'squashing' leading to vanishing gradients. Adding more units and/or more layers has also been tested, but with no improvements important enough to justify longer processing times. Larger filter sizes were also tested, but yielded blurry results. In order to get reasonable results with an initial layer consisting of larger filters, more layers with decreasing filter size were needed, but this required a reduction in the size of the images in the data set due to memory constraints, and added no significant advantages. A diagram of the network's architecture, chosen based on all the experimental evaluations and considerations described above, is shown in Figure 16

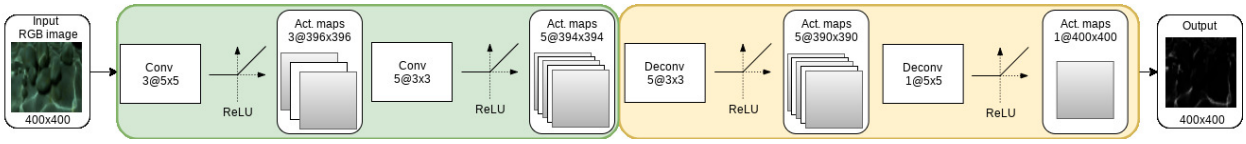


Figure 16: SalienceNet: a 4-layer CNN consisting of 2 convolutional layers followed by 2 deconvolutional layers.

Thus, the network models the following operation,

$$ReLU(\Psi^{-1} * ReLU(\Psi^{-1} * ReLU(\Psi \star ReLU(\Psi \star X)))) \rightarrow \Phi^{p} \tag{6}$$

where p is a pixel and Φ is its saliency value. In the above equation X denotes the input data, Ψ denotes a convolution kernel, $\star$ denotes the convolution operation, Ψ^{-1} denotes a de-convolution kernel, $*$ denotes the de-convolution operation, and $ReLU(.)$ is the rectified linear unit activation function.

As previously mentioned, the output is a gray-scale saliency map with pixel values ranging from $[0, 1]$; the larger

the saliency value the higher the confidence of caustics occurring at that pixel.

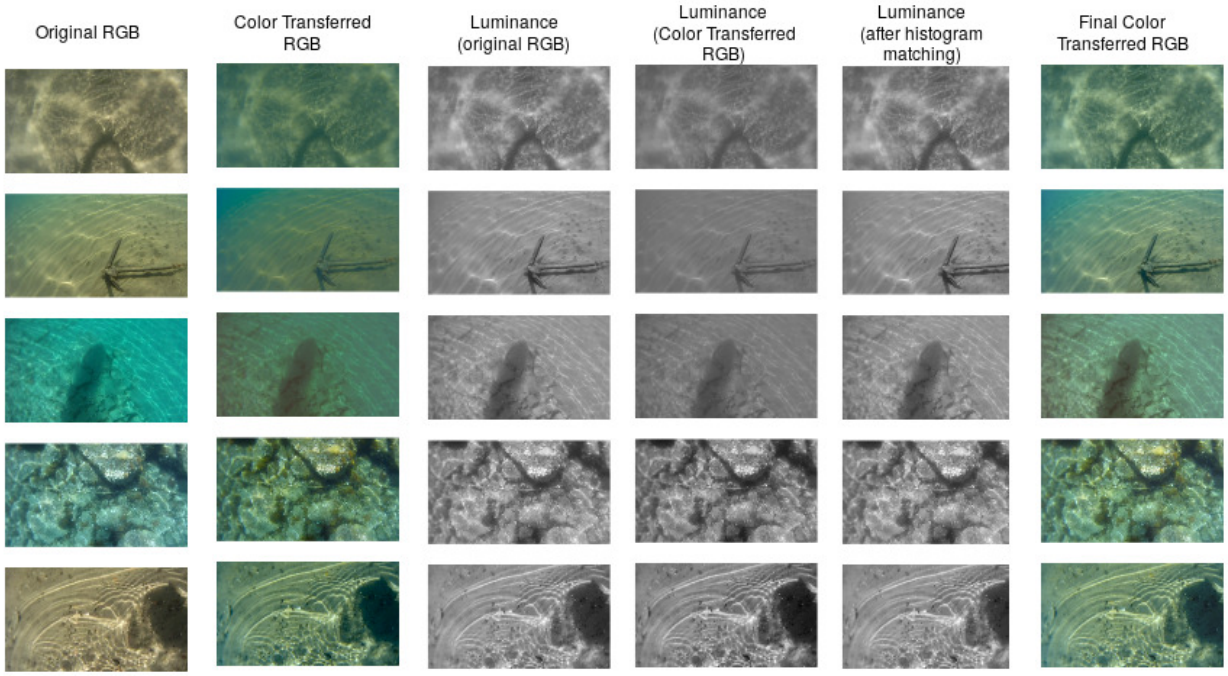


Figure 17: Color transfer [64] and histogram matching [65] of the luminance channel are applied to the real images in order to make the real images fall within the manifold learned from the training images and remove possible bias to colors. The images showing the luminance are shown as grayscale images.

5.2.3 Training

The network was implemented and trained using Theano API [66] on a set of 500 synthetic images. 60 synthetic images were reserved for validation. Figure 18 shows example pairs of input and ground truth images used for training the SalienceNet. The left column shows the rendered RGB image used as input and the center column shows the rendered caustics masks using global illumination and final gathering. Indirect lighting results in almost all pixels having a non-zero brightness value. The right column shows the thresholded caustics mask used during training as the ground truth saliency map.

Although a synthetic data set was used for training, we were ultimately interested in the network's performance on the real data set which can be considered a form of transfer learning. The network was trained for 1500 epochs at a learning rate of 0.001, which required approximately 9.5 hours of training on an GeForce GTX 1070 8GBVRAM GPU, followed by an additional 5000 epochs at a learning rate of 0.0001. A batch size of 32 was used. We also experimented with dropout [67], but no improvements were noticed, which indicates that overfitting was not a problem during the training process. Figure 19 shows the cost function error, modeled as a Mean-Squared Error (MSE) for 6500 epochs. Mean-Squared Error is the sum of squares of difference in pixel values from the ground truth and our predicted results.

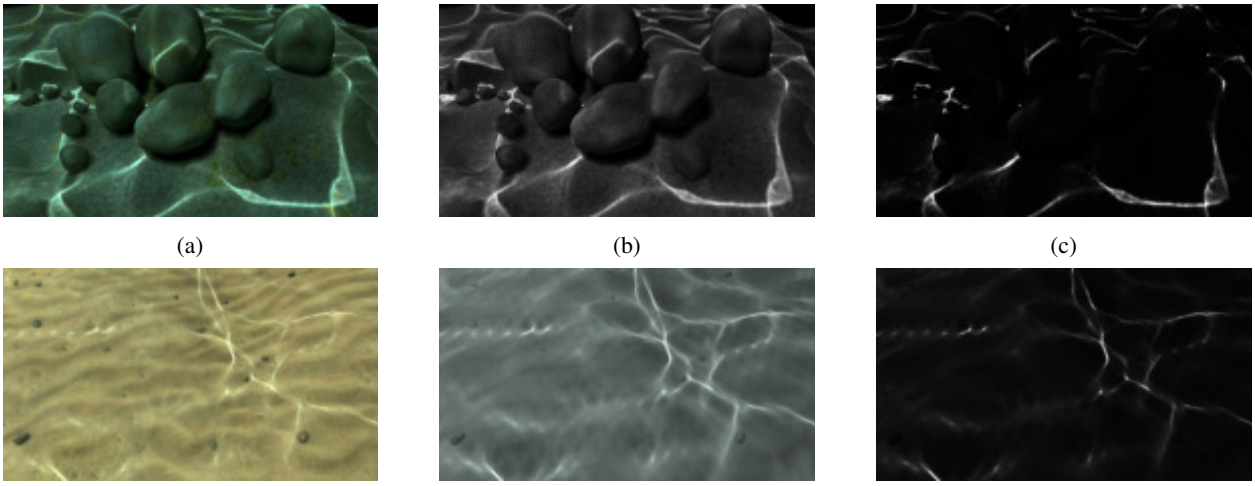


Figure 18: Examples of the training data. Left column: The rendered RGB image used as input to SalienceNet and DeepCaustics. Center column: The rendered caustics masks using global illumination and final gathering. Indirect lighting results in almost all pixels having a non-zero brightness value. Right column: The thresholded caustics mask used during training as the ground truth saliency map.

The formula is as follows,

$$\frac{1}{n}\sum_{j=1}^{n}(p_{ij}^{truth} - p_{ij}^{pred})^2 \tag{7}$$

where p_{ij} is a pixel in the i^{th} row and j^{th} column in the ground truth image or the SalienceNet image result.

5.2.4 Validation

As previously mentioned our main goal is to transfer the learning to real data sets. Towards this goal of transfer learning, due to the color variance of the water, seabed and caustics, the real images are first preprocessed such that the color space manifold matches that of the training set. Given a real image the preprocessing involves performing color transfer described by Reinhard et al. [64], converting from RGB color space to Luv, performing histogram matching as done by Coltuc et al. [65] on the luminance channel L, and converting back to RGB color space. Figure 17 shows an example of this process.

Following the training of SalienceNet using synthetic data, we processed a number of varied real world underwater videos through this network and obtained very good results. Although we cannot directly assess the accuracy of the SalienceNet in removing caustics given the lack of ground truth on the real images, we do evaluate its performance in terms of the reconstruction metrics described in Section 5.5. In other words, the metrics used to evaluate the reconstruction serve as a proxy for determining the performance of the networks. Five images from distinctive videos taken underwater are shown in the left column Figure 20 and the resulting saliency maps produced by SalienceNet are

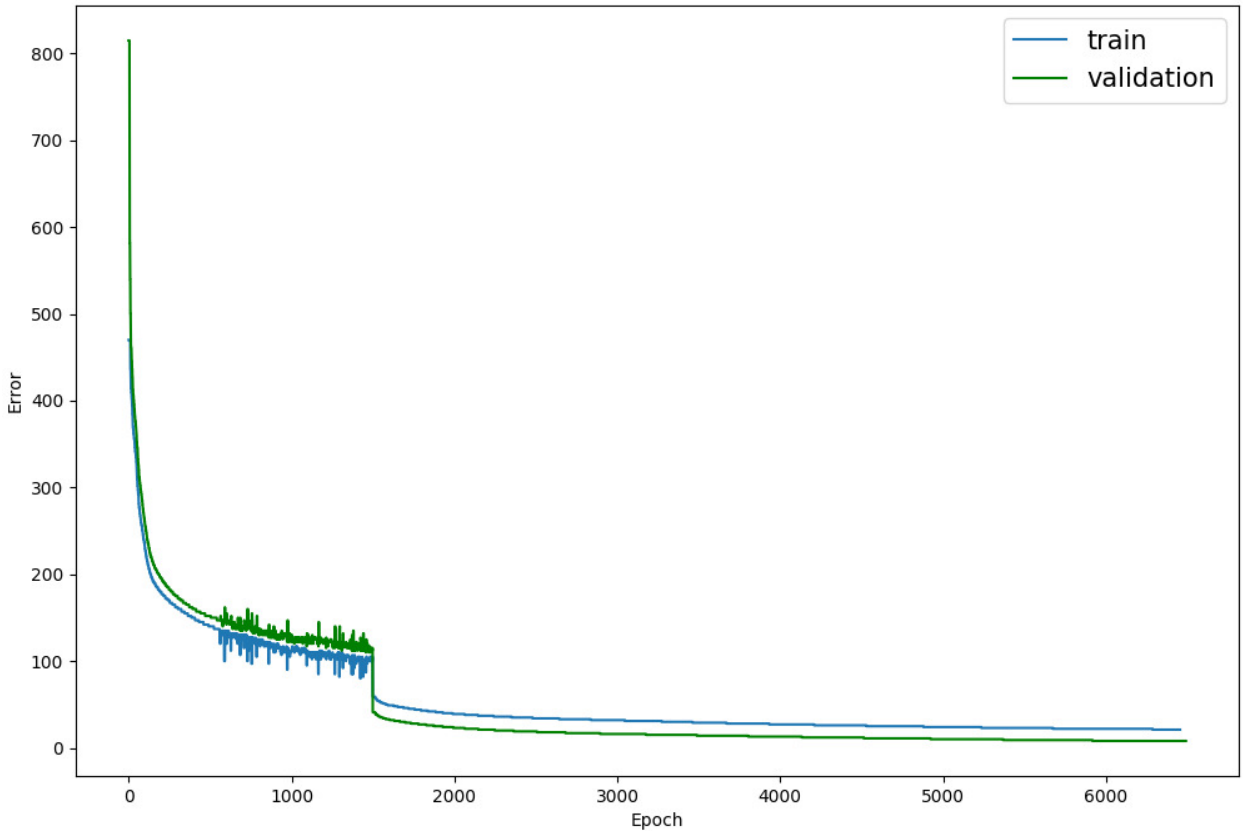


Figure 19: SalienceNet Cost Function Error (training and validation curves are almost identical): First 1500 epochs at a learning rate of 0.001 and the next 5000 epochs at a learning rate of 0.0001. The cost function used is Mean-Squared Error (MSE).

shown in the right column.

5.3 DeepCaustics

5.3.1 Dataset

The input to DeepCaustics is the pair of an image containing caustics and the saliency map generated by SalienceNet. The two are first coupled together into a 4-channel RGBA format where the fourth channel contains the saliency value for the corresponding pixel. The ground truth used for training is a rendered caustic-free image corresponding to the synthetic input images. An example is shown in Figure 21. The network operates on a batch of 16 images of size 400×400. The output of the network is a caustic-free RGB image corresponding to the input.

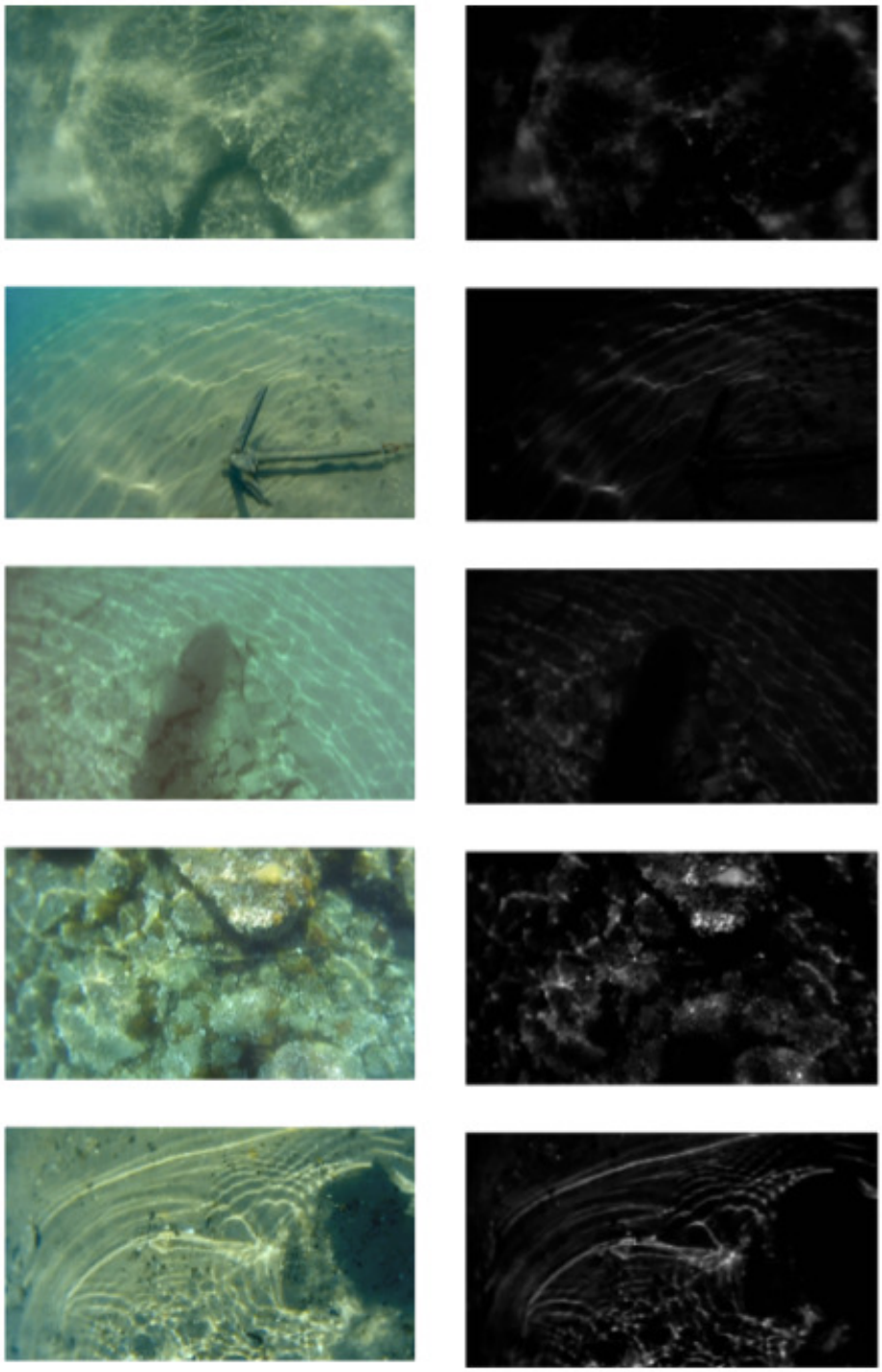


Figure 20: (right) The saliency maps generated by SalienceNet for the five real images (left) from distinctive videos taken underwater containing caustics of varying frequencies and shape.

(a) (b) (c)

Figure 21: Examples of the training data for DeepCaustics. (a) The rendered RGB image used as input to DeepCaustics. (b) The saliency map generated by SalienceNet. These two images [(a), (b)] become the input to the DeepCaustics network. (c) The ground truth used for training is a rendered caustic-free image corresponding to (a),(b).

5.3.2 Network Architecture

After extensive experimentation, we have concluded that the network architecture with the optimal performance consists of six hidden layers; the first three consisting of 4, 2, and, 7 convolution filters respectively, the last three consisting of 7, 2, and 3 de-convolution filters respectively. The filter sizes are 3×3, 7×7, 3×3, 3×3, 7×7, 3×3 in each layer respectively. This results in a total of $(4{\times}3{\times}3){+}2{\times}(2{\times}7{\times}7){+}2{\times}(7{\times}3{\times}3){+}(3{\times}3{\times}3)$ weight parameters and $(4 + 2 \times 2 + 2 \times 7 + 3)$ bias parameters, for a total of 410 parameters to be learned. Similarly to SalienceNet, after each [de-] convolution in the network follows a ReLU activation unit [63]. Figure 22 shows the architecture of the DeepCaustics network. An important observation first reported by [68] which was also confirmed during our experimentation is that scaling up the number of activation maps while scaling down the kernel size produces considerably better results.

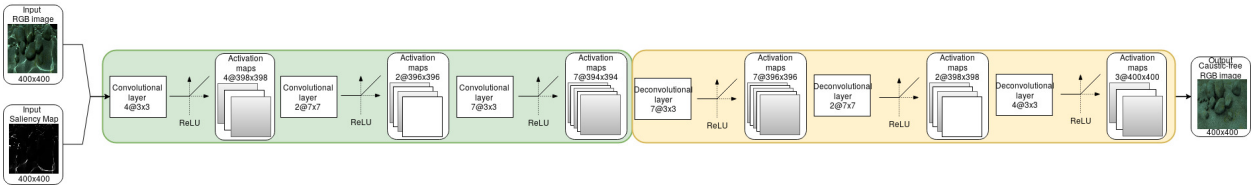


Figure 22: DeepCaustics: a 6-layer CNN consisting of 3 convolutional layers followed by 3 deconvolutional layers. A ReLU activation unit follows each [de-]convolution operation.

5.3.3 Training

The colour transfered pre-processed images along with the saliency values generated by SalienceNet form the input for the trained DeepCaustics. Again, the expected output is a caustic-free image. The network is trained using 180 synthetic image-pairs, and 20 synthetic image-pairs for validation. We trained the DeepCaustics network on the same hardware as for SalienceNet. Figure 23 shows the cost function error, modeled as a structural similarity index (SSIM); a quality measure of one of the images being compared, provided the other image is the ground truth.

SSIM evaluates images accounting for the fact that the human visual system is sensitive to changes in local structure. SSIM is implemented by creating patches from the ground truth images (y) and the predicted images generated from DeepCaustics (x). For each patch we calculate the SSIM:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1) + (2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{8}$$

such that μ is the average value of the patches, σ^2 is the variance, σ_{xy} is the covariance between patches x and y and σ is the standard deviation of the patches.

Our experiments have shown that it performs better than MSE in terms of network training as it has been also discussed by Zhao et al. [69]. The network was trained for 2700 epochs with a learning rate of 0.001. The choice of 2700 epochs is somewhat arbitrary; the learning curve shows that much of the training progress occurs within the first 80 epochs.

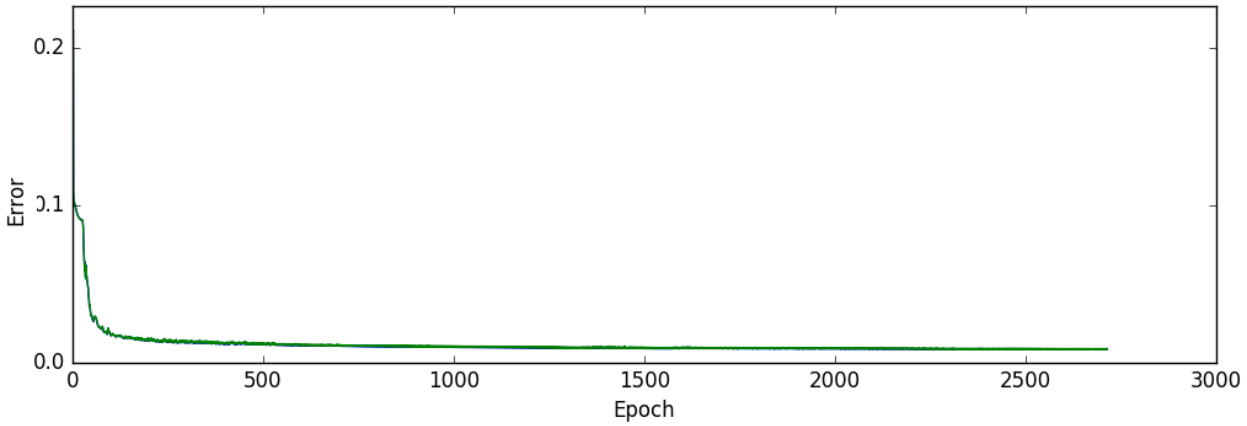


Figure 23: DeepCaustics cost function error. The cost function used is Structural Similarity index (SSIM). SSIM is a quality measure of one of the images being compared, provided the other image is the ground truth. SSIM evaluates images accounting for the fact that the human visual system is sensitive to changes in local structure.

5.3.4 Validation

As previously mentioned this work is motivated by the real world challenges in underwater archaeology. Processing videos for underwater entities in the presence of caustics is very hard. Figure 24 demonstrates that using the proposed approach successfully removes the caustics from the images therefore allowing more accurate further processing towards 3D reconstruction. In the first column we show five sample frames from real world RGB videos containing caustics of varying characteristics. When these images are input to DeepCaustics along with their saliency maps generated by SalienceNet the results obtained are shown in Figure 20. The second column in 24 shows the five RGB images after color transfer and histogram matching. In the third column we show the results generated by

DeepCaustics on the original images i.e. *without* any color transfer and histogram matching operations. As expected the network does not perform well for any images which fall outside the learned manifold. The last column shows the caustic-free images generated by DeepCaustics on the color matched RGB images shown in the second column. The caustics have been successfully removed and therefore further processing with structure-from-motion and multi-view stereo techniques becomes possible.

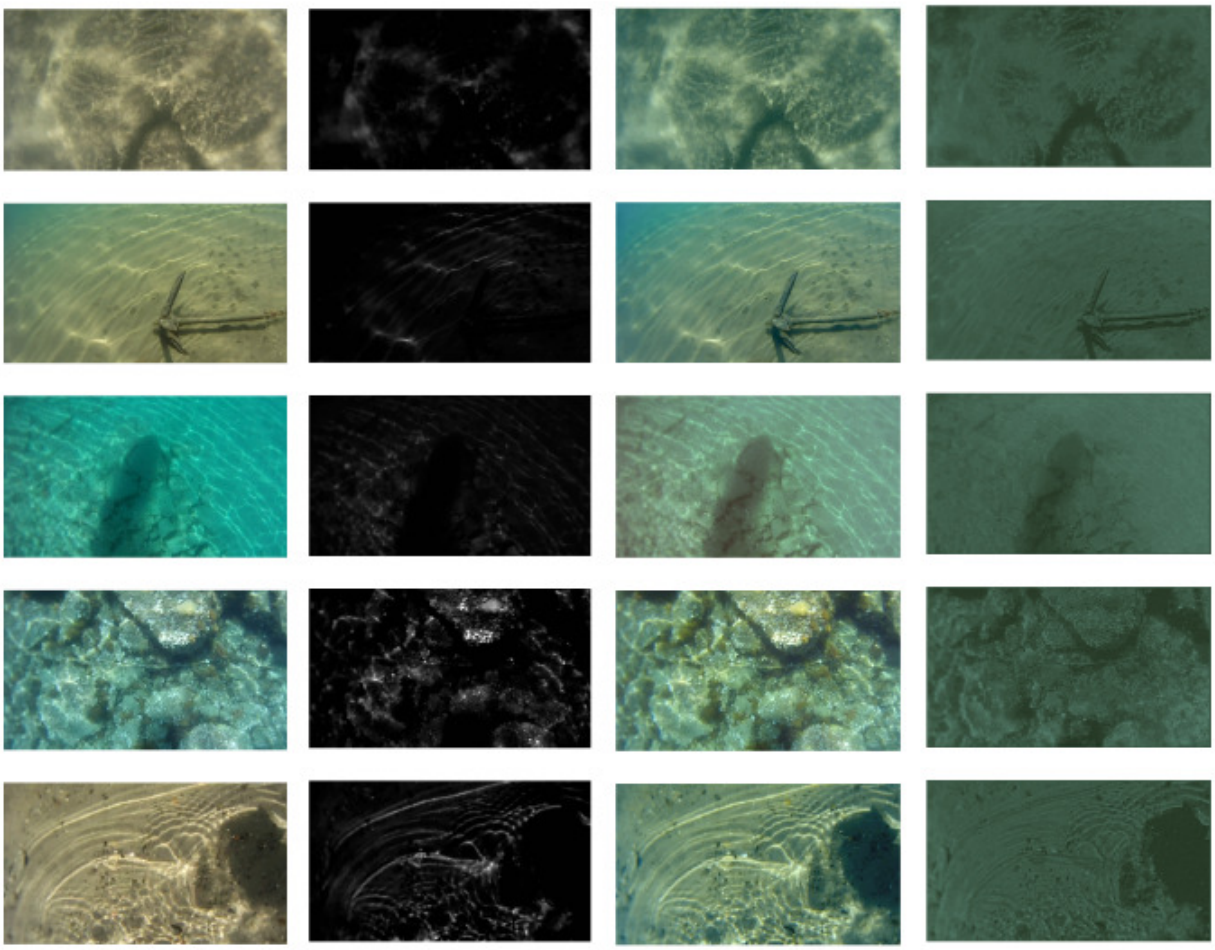
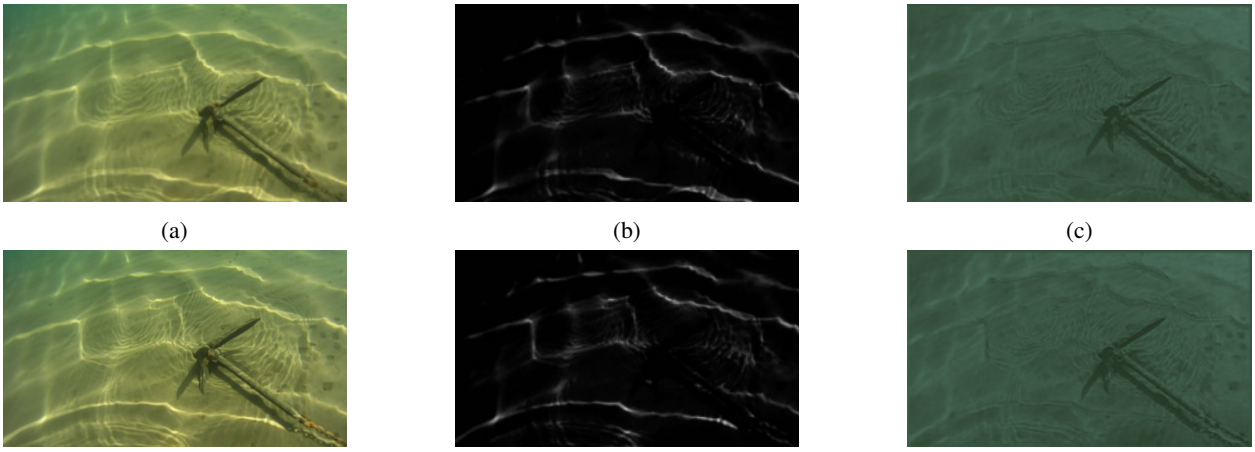


Figure 24: First column: The original RGB images for five selected scenes which contain caustics of varying characteristics. Second column: The RGB images after color transfer and histogram matching. Third column: The results generated by DeepCaustics on the original images *without* color transfer and histogram matching i.e. first column. Fourth column: The caustic-free images generated by DeepCaustics on the color matched RGB images shown in the second column. The caustics have been successfully removed and therefore further processing with structure-from-motion and multi-view stereo techniques is possible.

As said earlier, the images shown in the figures are individual sample frames taken from videos. We actually carry out the process for all frames in the video. Although there is no temporal information being taken into account when processing the video i.e. each frame is individually processed, we have seen that the resulting caustic-free video is smoothly varying and consistent across sequential frames. Figure 25 shows the saliency maps and caustic-free images are consistent between frames 25, 30 although there is considerable camera motion as well as caustic motion between

them.



(a) (b) (c)

(d) (e) (f)

Figure 25: Frames 25 and 30 are processed independently i.e. no temporal information is used. The saliency maps and caustic-free images are consistent although considerable motion [camera, caustics] has occurred between the two frames.

An extreme case of caustics produced by turbulent water motion which causes caustics of varying frequencies which are not well defined in terms of structure is shown in Figure 26. Similarly Figure 27 shows another complex example of caustics appearing on rocks with barnacles. Despite the complexity of the scene the proposed approach is able to correctly classify pixels as caustics/non-caustics and remove them. An example of this is the shadow of the boat being consistently classified as non-caustic throughout the video sequence.

5.3.5 Network Comparison

An in depth evaluation of the technique on other datasets was performed at the Cyprus University of Technology. The evaluation explains how this technique improves the images and provides more robust feature matches between image pairs [70]. Unfortunately it is very hard to compare this method with other techniques as there is no public reference dataset that can be used as a benchmark to our knowledge, and, other techniques are hard to replicate or get a hold of. However we attempt a comparison of different systems by exploring their published work.

In Trabes et al.'s work they establish a sunlight-deflickering filter for scenes that are very similar to those in our work. While they explain how there method works they do not provide very much visual evidence of an effective algorithm. Because of their feedback system their algorithm is stable to fluctuating changes in scene illumination and structure. There method does not require as much resources as other systems yet to perform well it must run consistently to determine its new parameter values. Another limitation to their work is that they require future frames for a section of their algorithm meaning it cannot work on an on-line setting [33].

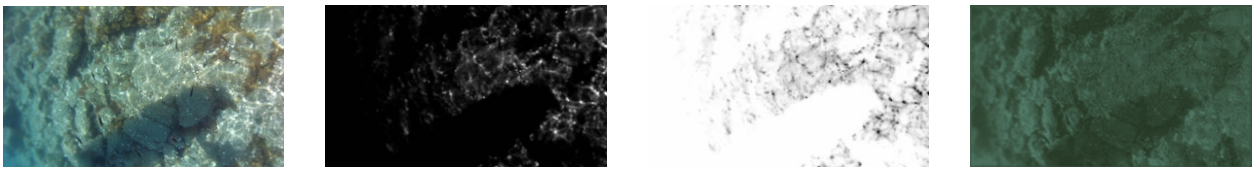
Another method uses multiple frames a adjusts them through the use of temporal information. However, in order to make use of the temporal information they must be aligned and to do so requires a robust feature matching algorithm. Given the intensity of the caustics then the feature matching algorithm could fail and the method would not work [34]. Several techniques that involve temporal information generally have significant disadvantages relative to our work. Schechner's technique unfortunately does not consider changes in camera movement and as such provides images that would not help much for later reconstruction [40].

Shihavuddin's approach relies on previous frames but not future frames. Their mathematical approach is not affected by camera motion and works by predicting the caustics in the coming frame. While their method is effective it requires around 6 seconds per frame, thus proving to be slow [36].

3Deflicker from motion by Swirski method is perhaps the most performing however it requires stereo cameras which prevent it from working on existing monocular video. It also requires expensive and specific technology [41].

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

Figure 26: An extreme example of caustics caused by turbulent water movement. (a) The RGB image. (b) The color transferred images. (c) The saliency map generated by SalienceNet. (d) The caustic-free image generated by DeepCaustics.



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

Figure 27: Another complex example of caustics appearing on rocks barnacles. (a) The original RGB image. (b) SalienceNet's result. (c) Color-inverted saliency map. Note that objects like the shadow of the boat are consistently classified as non-caustics throughout the video sequence. (d) The caustic-free image generated by DeepCaustics. Note the occasional loss of color information due to processing.

5.4 Network Design

5.4.1 Structure

When exploring different structures for our system we had to stay within our most important constraint: size. The smaller the network the faster our forward pass and the more effective our system can be in real-time. This led to a network that would not surpass the 10 layers and where operations consisted of significantly small kernels. A smaller

network also reduces overfitting which is crucial when dealing with transfer learning as in this case.

One critical design choice with regards to our structure was having two different networks instead of a single one. Our main argument is, again, that of working with transferred learning. By sequentially breaking the process we ensure that the learning is transferred correctly onto real-world data in a more controlled manner minimizing a potentially unseen cascading error.

5.4.2 Parameters

In order to find the right parameters for our networks we went through several trials. We ran our networks for 200 epochs and selected the best parameters based on error, prediction results, and number of parameters. Our errors can be seen in Table 3 with bold values representing our selections of each network.

Activation	Kernel	SalienceNet (MSE)	DeepCaustics (SSIM)
3,3	3,3	86.56	0.1628
3,5	5,3	**94.50**	0.1265
3,5,10	15,7,3	103.85	0.1947
4,2,7	3,7,3	115.56	**0.0361**
5,10,20	5,9,5	91.06	0.0359
5,10,20	15,9,5	116.55	0.0251
5,12,24	27,13,5	97.32	0.0245
20,10,15	3,7,5	71.06	0.0120
20,10,15	5,13,7	76.66	0.0179

Table 3: Errors for different network parameters after 200 epochs. Note that SalienceNet and DeepCaustics have different error functions, hence the different ranges of values. Activation column represents the feature maps for each layer with respect to the Kernel values squared for the kernel size. The first row would be a network of two layers, with the first layer consisting of 3 activation maps and a kernel size of 3x3. The second layer would also have 3 activation maps and a kernel size of 3x3.

When selecting our SalienceNet we noticed improvement when increasing layers and layer complexity however the trade off between the error and the number of parameters was too large to ignore. We also noticed that our last two tests, while having low error, over fit towards the artificial data producing erroneous salience images for real images. A similar issue happened with our DeepCaustics network. Our tests removed inefficient parameters but also permitted us to observe how differences in kernel size and how depth affected our results. Again increased complexity provided lower error but this meant more parameters and often over fitting.

The proposed approach has been extensively tested on real videos of underwater scenes containing caustics. The videos contain a large variability in terms of the color, the frequency of the caustics i.e. wave speed and formations, the background i.e. geometry and color [sand, rock, barnacles, etc], and shape i.e. wave interference.

5.5 Reconstruction

In order to evaluate the results of our method we constructed a 3D object from a sequence of thresholded images and a sequence of images generated by DeepCaustics. This was done to demonstrate the effectiveness of our method relative to a simple thresholding of brightness in an image.

(a) (b) (c)

Figure 28: Examples of the images that are used in the 3D reconstruction. (a) The original RGB image. (b) The thresholded image. (c) DeepCaustic generated image placed back into its original colorspace.

The 3D object consists of a block of concrete found underwater. We construct a 3D object using Agisoft's PhotoScan (SfM) from our image sequences, then we extract planes from four different sides of the cube. From these planes we calculate (1) a surface fitting metric evaluating the "smoothness of our reconstruction" and (2) the orthogonality with all adjacent faces. All faces, except opposing ones, should have an orthogonality of 90 degrees to represent how they are perpendicular. Opposing faces should be parallel. The metrics are defined below.

Surface fitting metric

A surface Π is reconstructed and a plane $\Pi_{fitted} = \langle \alpha, \beta, \gamma, \delta \rangle$ with normal $N_{\Pi_{fitted}} = \langle n_x, n_y, n_z \rangle$ is fitted on the resulting $\aleph$ 3D points. The plane fitting is performed using RANSAC and the average error E_{avg} and average $RMSE$ is computed as follows,

$$E_{avg} = \sum_{i=0}^{\aleph} |\delta - (\alpha \times \aleph_i^x + \beta \times \aleph_i^y + \gamma \times \aleph_i^z)|/||\aleph|| \tag{9}$$

$$RMSE = \sqrt{\frac{\sum_{i=0}^{\aleph} \{\delta - (\alpha \times \aleph_i^x + \beta \times \aleph_i^y + \gamma \times \aleph_i^z)\}^2}{||\aleph||}} \tag{10}$$

We reconstruct several different planar surfaces shown in Table 4 and perform plane fitting using RANSAC. This metric is measured as the distance of all points from the fitted surface in terms of the average error E_{avg} and the root-mean-square error $RMSE$.

	Threshold	DeepCaustics
Top	0.8276	0.9690
Front	0.9965	0.9974
Left	0.9787	0.9882
Right	0.9207	0.9564

Table 4: Surface fitting metric for different faces of the 3D reconstructed cube.

	Top	Front	Left	Right
Top	-	90.99	91.86	85.76
Front	56.65	-	92.29	56.28
Left	56.98	57.27	-	31.75
Right	57.26	52.83	35.47	-

Table 5: Orthogonality between different faces of the 3D reconstructed cube. The upper triangle of the table are results from images processed with DeepCaustics. The lower triangle contains results from images where brightness has been thresholded.

Orthogonality metric

A set of perpendicular surfaces are reconstructed and four planes $\Pi_1, \Pi_2, \Pi_3, \Pi_4$ are fitted respectively using RANSAC. The orthogonality metric is defined as the magnitude of the six dimensional vector containing the measured angles between the four planes in terms of the dot product as follows,

$$E_{ortho} = ||\langle N^1_{\Pi_{fitted}} \cdot N^2_{\Pi_{fitted}}, N^1_{\Pi_{fitted}} \cdot N^3_{\Pi_{fitted}}, N^1_{\Pi_{fitted}} \cdot N^4_{\Pi_{fitted}}, ..., N^3_{\Pi_{fitted}} \cdot N^4_{\Pi_{fitted}} \rangle||_{L_2} \quad (11)$$

We reconstructed the concrete block object containing orthogonal planes. For each of the four planes a linear surface is fitted using RANSAC. This metric is measured as the angle formed between the four planes as shown in Table 5.

5.5.1 Discussion

It is somewhat surprising that a small network achieved the success that it did. Although this may be due to some underlying simple properties of the synthetic data set that the network was able to exploit, the fact that good results were achieved on the real data set suggests that this is not a sufficient explanation. Alternatively, it may be that there is some simple property related to caustics, such as brightness or contrast, that is present in both the synthetic and real data set, which the network was able to pick up on, even with relatively few parameters. Small data sets, like our own, lead to over-fitting very early in the learning phase. Having a larger training set would increase the performance of our network as it learns to generalize to a wider set of inputs. Our small set of real test images resulted in biased evaluations as we could not test our system's performance on a wider variety of caustic images. Other examples of learning from small datasets are not unheard of [71]. However, while the size of the data set affects performance our greatest liability is the quality of our dataset. Having real image pairs with and without caustics and with the right

distribution would allow the network to learn more appropriate features applicable to our test set.

6 Summary and Concluding Remarks

6.1 Case Study #1: Urban Scene Classification and Reconstruction

We presented a complete framework for urban reconstruction based on semantic labeling. Our contribution is two-fold: First, we have presented a novel network architecture which uniquely leverages the strengths of deep convolutional autoencoders with feed forward links and cardinality-enabled ResNeXT blocks. In an important result, we have shown that the network is capable of producing smooth results without the need for CRF-based post-processing. The results on benchmark data indicate that the proposed technique can produce comparable and in some cases better classification without the need for excessive computational requirements or training time. Secondly, we have proposed a pipeline for the automatic reconstruction of urban areas based on semantic labeling. An agglomerative clustering is performed on the points based on their class. Each cluster is further processed according to its class and generic objects such as trees and cars are removed and replaced by procedurally generated tree models and car CAD models, respectively. Buildings' boundaries are extracted, extruded and triangulated to generate 3D models. All other classes are triangulated and simplified to form a digital terrain model. Finally, we have extensively tested the proposed framework on all 17 test images and show the realistic virtual environments generated as a result.

6.2 Case Study #2: Caustics Removal and Underwater Scene Reconstruction

In the second case study we reported a deep-learning solution to classification and removal of caustics from shallow underwater images. Given the difficulty to obtain ground truth for the large volumes of pixel data in underwater videos, our solution uses a small synthetic data set created using a standard 3-D modeling software. In creating the synthetic data, we create two ground truth components, a caustics confidence mask and a caustic-free scene. We use the first caustics confidence component to train a CNN, called SalienceNet, which yields a saliency value (confidence of being a caustic) per pixel. Using the saliency as the alpha value in RGBA format and the second caustic-free component as ground truth, we train another CNN, called DeepCaustics, to yield caustic-free versions of input images with caustics. Our tests on a number of real-world underwater videos show that our solution yields good results, which would certainly make further processing of these images more reliable and robust. As can be seen from the results, even with these extremely small networks and small synthetic training data set, we were able to transfer the learning to real-world data quite effectively. Investigating this further would be a challenging and interesting problem for the immediate future, in particular, looking into detail at cases where this fails.

6.3 Concluding Remarks

Through both case studies we were able to explore a deep learning approach to improving accuracy of classification of urban and underwater scenes. With increased accuracy in classification we were in turn able to produce better quality reconstructions. Our networks were based on supervised learning where the algorithm learned to improve its classification using ground truth data.

As with many neural networks it is very hard to learn a very generalized model. Using a very different city in our SegNeXt would yield significantly worse results. Taking underwater imagery in a different color-space led to unusable results. Ideally a network would have a huge variety of data to train on in order to generalize across a large input distribution such that any testing image lies within that distribution. This leads to our main area of future work, discussed in the next chapter.

7 Future Work

7.1 Data Generation

Future work includes investigating the generation aspects of GANs to produce training image pairs of input and ground truth images. Given an input latent vector a GAN network can generate an image lying in the distribution of a given dataset.

7.1.1 Urban Scenes

The ground truth image of a given input image is independent of the distribution different cities can have. Buildings are still buildings in different cities, distribution changes encompass colors, shapes, heights, etc, of buildings. Thus we would generate a ground truth image and then, given that ground truth and some latent noise, we generate an input image within a distribution defined by a new city dataset. This GAN system would allow us to generate training image pairs that our SegNeXt could train on for more generalization. Figure 29 explains the process with a diagram.

7.1.2 Underwater Scenes

Like urban scenes our focus here would be to use a GAN to produce an underwater image. However in this work the input latent vector would be given with explicit differences given the desired output. If an image is to have caustics then a section of the latent vector would hold values greater than 0, if the image does not have caustics then that section would hold only 0s. Essentially this would allow us to produce two identical images, one with caustics if that part of the latent vector is not 0, and one without caustics if the values are 0. We could then use both images for training. More interestingly, we could input a caustic image backwards through the network to produce the latent vector, then zero-out the caustic section, and pass it back forward through the network to produce a caustic free version.

7.2 General

More subtle future work involves different network architectures and techniques for processing the data. Within our underwater caustic removal framework we do not directly make use of the temporal information present in the way caustics change in successive frames of the scene. We would like to upgrade our method to explicitly use such temporal information and see if even better results can be obtained. As deep learning hardware is made more readily accessible

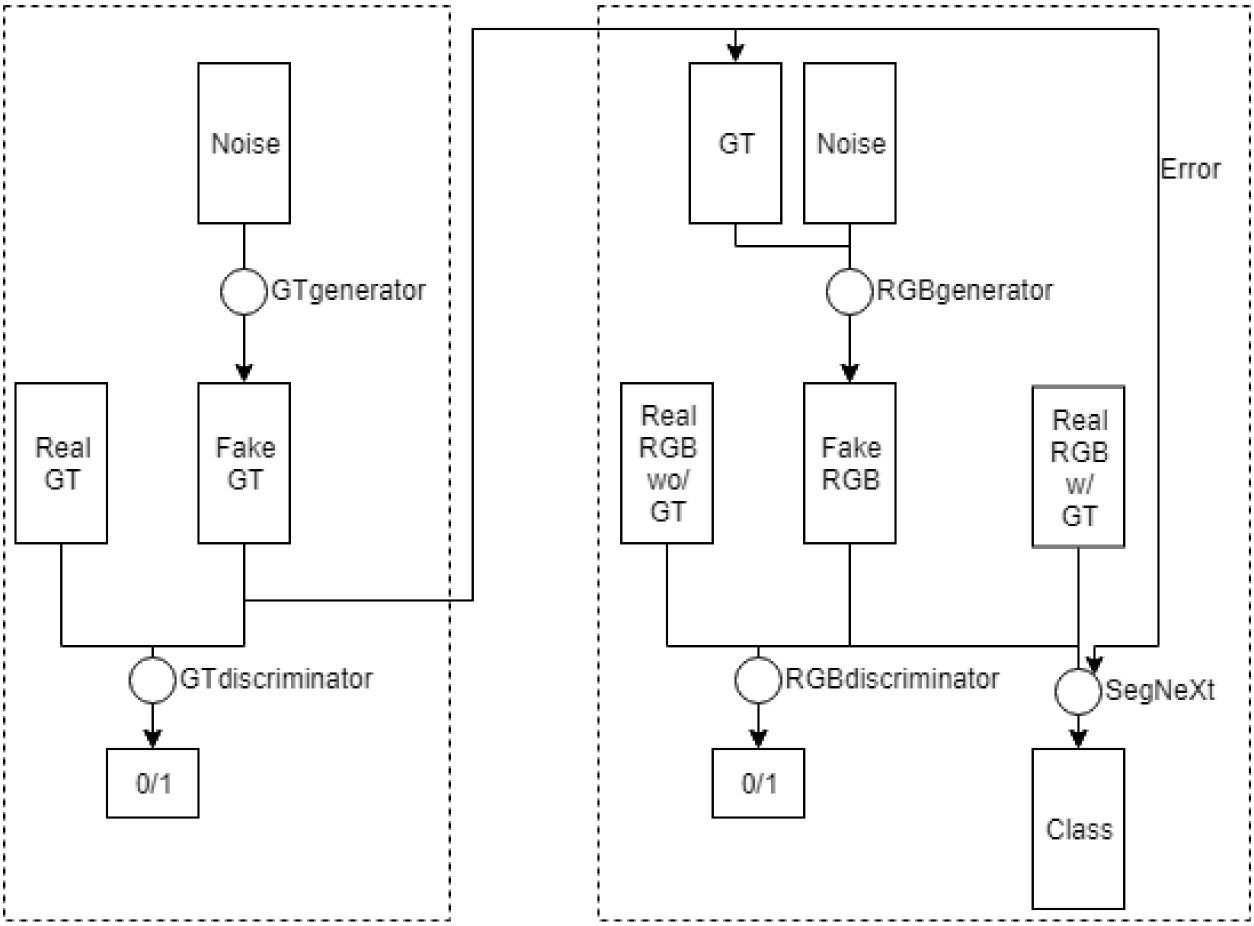


Figure 29: In the left system noise is produced into a "fake ground truth" (Fake GT) using a GAN. This ground truth is then fed into another GAN (right) with noise to produce a real image of a city (Fake RGB). This GAN has a discriminator to ensure the image is produced within the correct distribution of some unlabeled city dataset. There is also a segmentation network to ensure the image is also produced according to the input ground truth. The output of the system ends up being "Fake GT" along with its real image pair "Fake RGB". The segmentation network also implicitely learns to label that distribution of city thus also being a usable output.

we know making our networks larger would be possible without a significant compromise in speed. We would also like to explore the possibility of extending this methodology to other complex phenomena in images that cause similar problems, such as shadows. For urban scenes we would like to explore recently proposed architectures for improved semantic labeling such as dynamic routing capsules [72] in the deep autoencoders which have already achieved state-of-the-art performance, and the exploration of inverse solid geometry for large-scale urban reconstruction which by design resolve the problem of noisy boundaries in the reconstructed 3D models.

References

[1] F. Rottensteiner, G. Sohn, M. Gerke, J. D. Wegner, U. Breitkopf, and J. Jung, "Results of the ISPRS benchmark on urban object detection and 3d building reconstruction," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 93, pp. 256–271, 2014.

[2] P. Drap, D. Merad, J.-M. Boï, J. Seinturier, D. Peloso, C. Reidinger, G. Vannini, M. Nucciotti, and E. Pruno, "Photogrammetry for medieval archaeology: A way to represent and analyse stratigraphy," in *Virtual Systems and Multimedia (VSMM), 2012 18th International Conference on*. IEEE, 2012, pp. 157–164.

[3] J. Kiefer and J. Wolfowitz, "Stochastic Estimation of the Maximum of a Regression Function," *The Annals of Mathematical Statistics*, vol. 23, no. 3, pp. 462–466, Sep. 1952. [Online]. Available: https://projecteuclid.org/euclid.aoms/1177729392

[4] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680. [Online]. Available: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[6] Y. Bengio, A. Courville, and P. Vincent, "Representation Learning: A Review and New Perspectives," *arXiv:1206.5538 [cs]*, Jun. 2012. [Online]. Available: http://arxiv.org/abs/1206.5538

[7] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv:1603.07285 [cs, stat]*, Mar. 2016. [Online]. Available: http://arxiv.org/abs/1603.07285

[8] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10. USA: Omnipress, 2010, pp. 807–814. [Online]. Available: http://dl.acm.org/citation.cfm?id=3104322.3104425

[9] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[10] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context," *Int. Journal of Computer Vision (IJCV)*, Jan. 2009.

[11] Y. Lecun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[13] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," *arXiv:1411.4038 [cs]*, Nov. 2014, arXiv: 1411.4038. [Online]. Available: http://arxiv.org/abs/1411.4038

[14] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs," *arXiv:1412.7062 [cs]*, Dec. 2014, arXiv: 1412.7062. [Online]. Available: http://arxiv.org/abs/1412.7062

[15] L.-C. Chen, J. T. Barron, G. Papandreou, K. Murphy, and A. L. Yuille, "Semantic Image Segmentation with Task-Specific Edge Detection Using CNNs and a Discriminatively Trained Domain Transform," *arXiv:1511.03328 [cs]*, Nov. 2015, arXiv: 1511.03328. [Online]. Available: http://arxiv.org/abs/1511.03328

[16] H. Noh, S. Hong, and B. Han, "Learning Deconvolution Network for Semantic Segmentation," *arXiv:1505.04366 [cs]*, May 2015, arXiv: 1505.04366. [Online]. Available: http://arxiv.org/abs/1505.04366

[17] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *arXiv:1511.00561 [cs]*, Nov. 2015. [Online]. Available: http://arxiv.org/abs/1511.00561

[18] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 11–19.

[19] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[20] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe, "Full-Resolution Residual Networks for Semantic Segmentation in Street Scenes," *arXiv:1611.08323 [cs]*, Nov. 2016, arXiv: 1611.08323. [Online]. Available: http://arxiv.org/abs/1611.08323

[21] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Hypercolumns for Object Segmentation and Fine-grained Localization," *arXiv:1411.5752 [cs]*, Nov. 2014, arXiv: 1411.5752. [Online]. Available: http://arxiv.org/abs/1411.5752

[22] Y. Cheng, R. Cai, Z. Li, X. Zhao, and K. Huang, "Locality-Sensitive Deconvolution Networks with Gated Fusion for RGB-D Indoor Semantic Segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3029–3037.

[23] G. Lin, A. Milan, C. Shen, and I. Reid, "RefineNet: Multi-Path Refinement Networks for High-Resolution Semantic Segmentation," *arXiv:1611.06612 [cs]*, Nov. 2016, arXiv: 1611.06612. [Online]. Available: http://arxiv.org/abs/1611.06612

[24] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun, "Large Kernel Matters – Improve Semantic Segmentation by Global Convolutional Network," *arXiv:1703.02719 [cs]*, Mar. 2017, arXiv: 1703.02719. [Online]. Available: http://arxiv.org/abs/1703.02719

[25] W. Liu, A. Rabinovich, and A. C. Berg, "ParseNet: Looking Wider to See Better," *arXiv:1506.04579 [cs]*, Jun. 2015, arXiv: 1506.04579. [Online]. Available: http://arxiv.org/abs/1506.04579

[26] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs," *arXiv:1606.00915 [cs]*, Jun. 2016. [Online]. Available: http://arxiv.org/abs/1606.00915

[27] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid Scene Parsing Network," *arXiv:1612.01105 [cs]*, Dec. 2016, arXiv: 1612.01105. [Online]. Available: http://arxiv.org/abs/1612.01105

[28] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *CoRR*, vol. abs/1511.07122, 2015. [Online]. Available: http://arxiv.org/abs/1511.07122

[29] R. Alshehhi, P. R. Marpu, W. L. Woon, and M. D. Mura, "Simultaneous extraction of roads and buildings in remote sensing imagery with convolutional neural networks," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 130, pp. 139 – 149, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0924271617300096

[30] K. Nogueira, M. Dalla Mura, J. Chanussot, W. R. Schwartz, and J. A. dos Santos, "Learning to semantically segment high-resolution remote sensing images," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, Dec 2016, pp. 3566–3571.

[31] S. Paisitkriangkrai, J. Sherrah, P. Janney, and A. van den Hengel, "Semantic labeling of aerial and satellite imagery," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, no. 7, pp. 2868–2881, July 2016.

[32] Y. He, S. Mudur, and C. Poullis, "Multi-label Pixelwise Classification for Reconstruction of Large-scale Urban Areas," *arXiv preprint arXiv:1709.07368*, 2017.

[33] E. Trabes and M. A. Jordan, "Self-tuning of a sunlight-deflickering filter for moving scenes underwater," in *Information Processing and Control (RPIC), 2015 XVI Workshop on*. IEEE, 2015, pp. 1–6.

[34] N. Gracias, S. Negahdaripour, L. Neumann, R. Prados, and R. Garcia, "A motion compensated filtering approach to remove sunlight flicker in shallow water images," in *OCEANS 2008*. IEEE, 2008, pp. 1–7.

[35] N. Gracias and J. Santos-Victor, "Underwater video mosaics as visual navigation maps," *Computer Vision and Image Understanding*, vol. 79, no. 1, pp. 66–91, 2000.

[36] A. Shihavuddin, N. Gracias, and R. Garcia, "Online Sunflicker Removal using Dynamic Texture Prediction." in *VISAPP (1)*, 2012, pp. 161–167.

[37] N. Joshi and M. F. Cohen, "Seeing Mt. Rainier: Lucky imaging for multi-image denoising, sharpening, and haze removal," in *Computational Photography (ICCP), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1–8.

[38] K. He, J. Sun, and X. Tang, "Single image haze removal using dark channel prior," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 12, pp. 2341–2353, 2011.

[39] R. Fattal, "Single image dehazing," *ACM transactions on graphics (TOG)*, vol. 27, no. 3, p. 72, 2008.

[40] Y. Y. Schechner and N. Karpel, "Attenuating natural flicker patterns," in *OCEANS'04. MTTS/IEEE TECHNO-OCEAN'04*, vol. 3. IEEE, 2004, pp. 1262–1268.

[41] Y. Swirski and Y. Y. Schechner, "3deflicker from motion," in *Computational Photography (ICCP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1–9.

[42] E. Trabes and M. Jordan, "On-line Filtering of Sunlight Caustic Waves in Underwater Scenes in Motion," in *evaluation in 7th International Scientific Conference on Physics and Control*, 2015, pp. 19–22.

[43] N. Snavely, S. M. Seitz, and R. Szeliski, "Modeling the world from internet photo collections," *International Journal of Computer Vision*, vol. 80, no. 2, pp. 189–210, 2008.

[44] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, "Towards internet-scale multi-view stereo," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 1434–1441.

[45] P. Musialski, P. Wonka, D. G. Aliaga, M. Wimmer, L. v. Gool, and W. Purgathofer, "A survey of urban reconstruction," in *Computer graphics forum*, vol. 32. Wiley Online Library, 2013, pp. 146–177.

[46] Q.-Y. Zhou and U. Neumann, "2.5 D building modeling by discovering global regularities," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 326–333.

[47] Y. Verdie, F. Lafarge, and P. Alliez, "LOD generation for urban scenes," ACM, Tech. Rep., 2015.

[48] C. Poullis and S. You, "Automatic reconstruction of cities from remote sensor data," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 2775–2782.

[49] C. Poullis, "A framework for automatic modeling from point cloud data," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 11, pp. 2563–2575, 2013.

[50] S. Alsisan and N. J. Mitra, "Variation-Factored Encoding of Facade Images." in *Eurographics (Short Papers)*, 2012, pp. 37–40.

[51] J. Xiao and Y. Furukawa, "Reconstructing the world's museums," *IJCV*, vol. 110, no. 3, pp. 243–258, 2014.

[52] T. Forbes and C. Poullis, "Deep autoencoders with aggregated residual transformations for urban reconstruction from remote sensing data," in *2018 15th Conference on Computer and Robot Vision (CRV)*. IEEE, 2018, pp. 23–30.

[53] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," *arXiv preprint arXiv:1611.05431*, 2016.

[54] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[55] Y. Liu, B. Fan, L. Wang, J. Bai, S. Xiang, and C. Pan, "Semantic labeling in very high resolution images via a self-cascaded convolutional neural network," *CoRR*, vol. abs/1807.11236, 2018. [Online]. Available: http://arxiv.org/abs/1807.11236

[56] H. Wang, Y. Wang, Q. Zhang, S. Xiang, and C. Pan, "Gated convolutional neural network for semantic segmentation in high-resolution images," *Remote Sensing*, vol. 9, no. 5, 2017. [Online]. Available: http://www.mdpi.com/2072-4292/9/5/446

[57] D. Marmanis, K. Schindler, J. D. Wegner, S. Galliani, M. Datcu, and U. Stilla, "Classification with an edge: Improving semantic image segmentation with boundary detection," *CoRR*, vol. abs/1612.01337, 2016. [Online]. Available: http://arxiv.org/abs/1612.01337

[58] S. Piramanayagam, E. Saber, W. Schwartzkopf, and F. W. Koehler, "Supervised classification of multisensor remotely sensed images using a deep learning framework," *Remote Sensing*, vol. 10, no. 9, 2018. [Online]. Available: http://www.mdpi.com/2072-4292/10/9/1429

[59] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, "High-resolution semantic labeling with convolutional neural networks," *CoRR*, vol. abs/1611.01962, 2016. [Online]. Available: http://arxiv.org/abs/1611.01962

[60] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *Artificial Neural Networks and Machine Learning–ICANN 2011*. Springer, 2011, pp. 52–59.

[61] T. Forbes, M. Goldsmith, S. Mudur, and C. Poullis, "Deepcaustics: Classification and removal of caustics from underwater imagery," *IEEE Journal of Oceanic Engineering*, no. 99, pp. 1–11, 2018.

[62] H. W. Jensen, *Realistic image synthesis using photon mapping*. Ak Peters Natick, 2001, vol. 364.

[63] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 807–814.

[64] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley, "Color transfer between images," *IEEE Computer graphics and applications*, vol. 21, no. 5, pp. 34–41, 2001.

[65] D. Coltuc, P. Bolon, and J.-M. Chassery, "Exact histogram specification," *IEEE Transactions on Image Processing*, vol. 15, no. 5, pp. 1143–1152, 2006.

[66] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016. [Online]. Available: http://arxiv.org/abs/1605.02688

[67] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[68] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *CoRR*, vol. abs/1409.1556, 2014.

[69] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss Functions for Neural Networks for Image Processing," *arXiv preprint arXiv:1511.08861*, 2015.

[70] P. Agrafiotis, D. Skarlatos, T. Forbes, C. Poullis, M. Skamantzari, and A. Georgopoulos, "Underwater photogrammetry in very shallow waters: Main challenges and caustics effect removall," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLII-2, pp. 15–22, 2018. [Online]. Available: https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLII-2/15/2018/

[71] P. Ren, Y. Dong, S. Lin, X. Tong, and B. Guo, "Image based relighting using neural networks," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, p. 111, 2015.

[72] S. Sabour, N. Frosst, and G. E. Hinton, “Dynamic Routing Between Capsules,” *arXiv:1710.09829 [cs]*, Oct. 2017, arXiv: 1710.09829. [Online]. Available: http://arxiv.org/abs/1710.09829