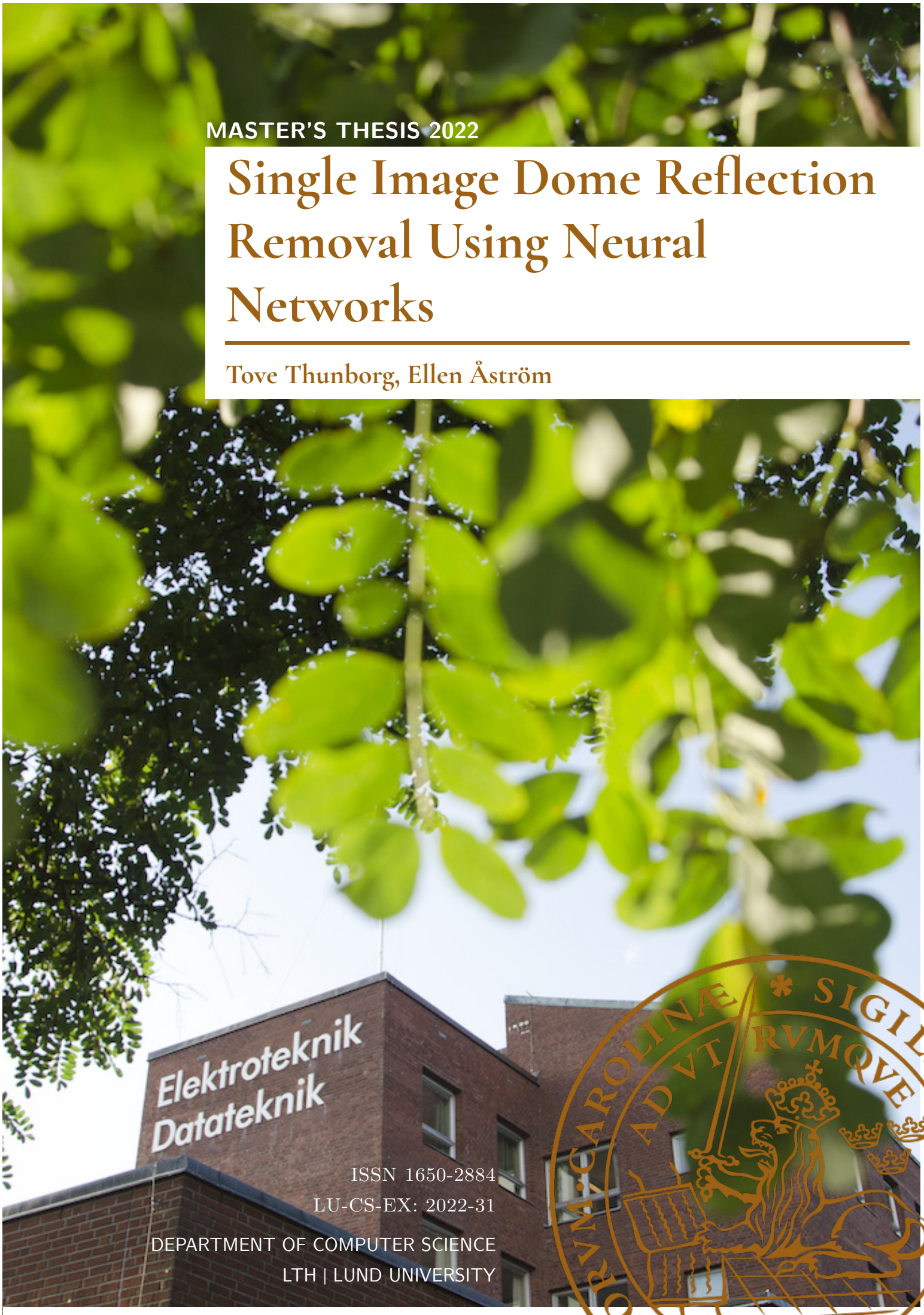


MASTER'S THESIS 2022

Single Image Dome Reflection Removal Using Neural Networks

Tove Thunborg, Ellen Åström

ISSN 1650-2884
LU-CS-EX: 2022-31

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2022-31

Single Image Dome Reflection Removal Using Neural Networks

Filtrering av reflektioner i domekameror med hjälp av neurala nätverk

Tove Thunborg, Ellen Åström

Single Image Dome Reflection Removal Using Neural Networks

Tove Thunborg
tove.thunborg@gmail.com

Ellen Åström
ellen.astrom@outlook.com

June 13, 2022

Master's thesis work carried out at Axis Communications AB.

Supervisors: Victor Lantz, victor.lantz@axis.com
Pierre Nugues, pierre.nugues@cs.lth.se

Examiner: Michael Doggett, michael.doggett@cs.lth.se

Abstract

Surveillance cameras are an important part of protecting people in their everyday life. Some of these cameras carry a protective dome which sometimes creates unwanted image artifacts in the form of circular lens reflections. One could solve this problem mechanically by developing less reflective domes, but this has shown to be quite hard.

Another, perhaps more reliable solution, would be to develop a neural network which can filter out the reflections. Many reflection removal networks already exist. However, none of them have been trained on dome reflections.

In this thesis, we investigate the dome reflection removal performance of four existing reflection removal networks. We fine-tune the networks using our own synthesized dataset, and evaluate the results both quantitatively and qualitatively.

The results show that the Enhanced Reflection Removal Network perform best. Moreover, this fine-tuned network shows a significant improvement in the dome reflection removal ability, compared to the initial pre-trained network.

Keywords: Reflection removal, Dome cameras, Image artifacts, Convolutional neural networks, Generative adversarial networks

Acknowledgements

We would like to thank Pierre Nugues, our supervisor at LTH, for all the help and feedback on both the implementation and the report. Your input has always made the project clearer and easier to tackle.

We would also like to thank Victor Lantz, our supervisior at Axis. Thank you for always answering questions, offering help and giving general supervision. Additionally, a great thanks to the people in our team at Axis that participated in both the evaluation process as well as general discussions of our problem.

Moreover, the computations and data handling were enabled by resources provided by the Swedish National Infrastructure for Computing (SNIC) at Chalmers Centre for Computational Science and Engineering (C3SE) partially funded by the Swedish Research Council through grant agreement no. 2018-05973.

Lastly, we would like to thank Joar Karlgren Gustavsson and Vendela Nigård for telling us about SNIC and helping us to get started.

Contents

Chapter 1
Introduction

This chapter formulates the reflection removal problem and proposes our solutions to it. We present previous work and the utilities used during our implementation.

1.1 Background

Surveillance is an important part of protecting people in their everyday life. By monitoring the citizens' behavior, wrongdoing can both be prevented and discovered. Perhaps the most standard way of surveillance is using video cameras. Today, cameras are installed in almost all public places.

A common camera in surveillance is the dome camera shown in Figure 1.1. As noticed, it is easily recognized by its protective plastic cover, also called dome. This type of camera has the advantage of wide angle images, while the dome makes it more durable. Figure 1.2 shows images taken with a dome camera under normal conditions.

Figure 1.1: A dome camera. Image taken from amazon.com

Reflection-free image in laboratory environment. Reflection-free image in daylight.

Reflection-free image at night. Reflection-free image at night.

Figure 1.2: Four examples of reflection-free images in different environments.

However, the protective transparent dome of these cameras tends to reflect the camera lens back into the image. This phenomena only happens in certain lighting conditions, but can still be disturbing. The reflections generally take the appearance of light circles in the image. Figure 1.3 shows examples of images with such dome reflections. As seen in the figure, the dome reflection decreases the overall image quality and is therefore something one wants to avoid.

One solution to this could be to develop a less reflective dome, but this approach has shown to be rather hard in practice. A possibly more reliable method is to filter out the reflections after the image is taken – a procedure often referred to as *reflection removal*.

Reflection removal is the process of separating a reflection layer from an image and keeping only the background, called transmission layer. As mentioned in Zhang et al. (2018), the original image, $\boldsymbol{I}$, can be approximated as

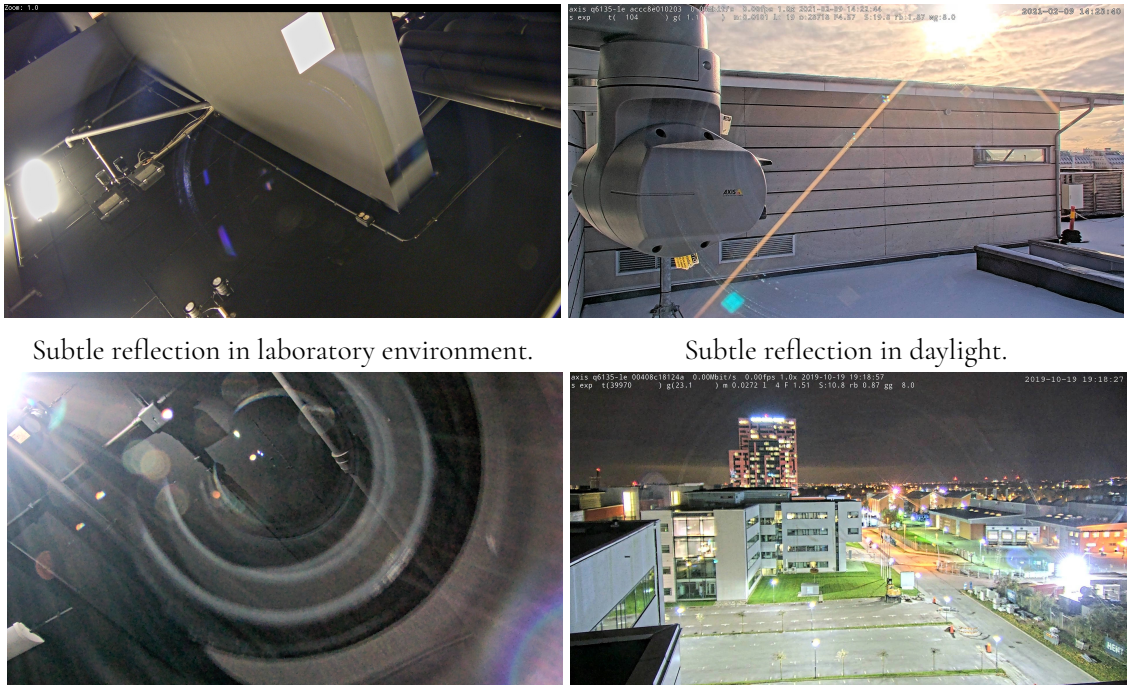
$$\boldsymbol{I} = \boldsymbol{T} + \boldsymbol{R}, \tag{1.1}$$

where $\boldsymbol{T}$ and $\boldsymbol{R}$ represents the transmission and reflection layers, respectively.

Since the number of unknowns exceeds the number of equations in Eq. 1.1, reflection removal is an ill-posed problem. There exists a large number of possible $\boldsymbol{T}$ and $\boldsymbol{R}$ for each image, and there are no known priors or additional constraints (Zhang et al., 2018).

To add additional constraints, one possible approach is to use multiple images of the same scene captured in various ways or at different angles. However, since surveillance cameras have specific settings and are stationary mounted, this is not possible in our case.

A better approach would be *single image reflection removal*, where only one image is utilized in the removal process. The most recent solutions are built on approximating the transmis-



Subtle reflection in laboratory environment. Subtle reflection in daylight.

Strong reflection in laboratory environment. Subtle reflection at night.

Figure 1.3: Four examples of images containing dome reflections in different environments.

sion layer using neural networks, where no priors nor constraints are needed. These kinds of neural networks rely on images with reflections and their corresponding ground truth – an identical image without the reflection.

However, one of the great challenges in single image reflection removal is the retrieval of a reflection image and its corresponding ground truth. This is hard since a reflection appears due to a reflective surface being present in the image. If the surface is removed, more details than just the reflection would differ between the images. However, assume it was possible to create such image pairs. Then one could proceed to train a neural network to input a reflection image and output the filtered image without reflection.

1.2 Contributions

Amanlou et al. (2022) systematically reviewed recent research in single image reflection removal. The neural networks presented have not been trained on any specific type of reflection. In contrary, our project will solely focus on reflections from dome cameras.

In this thesis, we fine-tune four neural networks and evaluate their dome reflection removal results to determine the best performing one. To fine-tune, we create a synthetic dataset of blended images. Our blended images are generated by adding a transmission and reflection layer together. Our reflection layers are processed images taken in a laboratory environment against dark background. To evaluate, we use 38 images containing real-world dome reflections and 445 synthetic images. Our contributions can be summarized:

- From transmission and reflection layers, we create a dataset containing blended im-

ages. The reflection layers are based on images taken in a laboratory environment. Our dataset is used for both training and evaluation.

- We gather 38 images containing real dome reflections, which are used for evaluation.
- We train four neural networks built for reflection removal on our synthetic dataset. Thereafter, we evaluate their performance on both synthetic and real-world data.
- We determine the best performing network and use it to remove reflections from images taken with dome cameras. The network, ERRNet, performs best during evaluation on both synthetic and real-world data.

1.2.1 Individual Contribution

To the highest extent possible, we have collaboratively worked on this thesis. We have actively participated in all parts, including research, implementation and writing.

Regarding implementation, the work was divided into training and evaluation. Tove mainly focused on training and testing the networks, while Ellen wrote code for evaluation.

During the phase of writing, we worked individually on different parts, although reviewing was done collectively. We estimate that we have contributed equally.

1.3 Previous Work

Through recent years, multiple different algorithms for reflection removal have been proposed (Wan et al., 2017). Each algorithm has its own advantages and disadvantages, and this has led to varying results. In this chapter, we discuss previous work within the field.

1.3.1 Multiple-Image Methods

Some reflection removal algorithms require multiple image input. This is a way to add priors and constraints, making the problem formulation less ill-posed. The most elementary solutions position the camera in different viewpoints, as done by Be'ery and Yeredor (2008). Later implementations, like Gai et al. (2012) and Guo et al. (2014), used the same technique.

Other multiple image methods use images with and without an added feature. One example of such a feature is polarizing filters, as introduced by Schechner et al. (1999). The same approach has later been implemented by both Diamant and Schechner (2008) and Kong et al. (2014).

Furthermore, Agrawal et al. (2005) and Agrawal et al. (2006), instead made use of flash and no-flash image pairs. A third approach is to use different focuses as features, which was done by Schechner et al. (1998).

In the case of a stationary mounted dome camera, neither of these features are available. This makes these methods ineffective in our type of application. Instead, our focus will be single image reflection removal.

1.3.2 Single-Image Methods

Single image reflection removal requires only one image. Here, the early solutions are methods built on optimizing different priors of one of the layers.

These priors could, for instance, be related to the image gradient distribution. It is a well-known fact that the gradient of an image is sparse. Thus, the decomposition of the two layers, transmission and reflection, can be based on minimizing the total number of corners and edges. Levin et al. (2004) used this edge minimization approach. Later, Levin and Weiss (2007) used the same optimization criteria, but with the difference of the end-user interactively classifying parts of the gradients as either belonging to the transmission or reflection layer.

Other methods used the plausible assumption that the camera is focused on the transmission part of the image. Thus, the reflection layer should be more blurry. Sharp edges can therefore be marked as part of the background, while smoother ones are assumed to belong to the reflection. Both Yan et al. (2013) and Wan et al. (2016) used this premise when creating their models.

Lastly, an effect related to the physical properties of the reflective material, is that thick glass tends to create a double reflection. This phenomena is called the *ghosting effect*. The first reflection is created when the light hits the reflective surface, and the second when it leaves it. Therefore, there are two reflections, with a shift in between. Shih et al. (2015) based their algorithm on this assumption.

Building a model based on optimizing one of these assumptions makes the models less flexible. They perform well for the cases they are optimized for, but the results are poor for other types of reflections. In the first optimization case, Zhang et al. (2018) argue that the image gradient distribution is a low-level feature. The results will therefore be limited when a high-level understanding of the image is necessary. Moreover, the assumption regarding blurry reflection layers, breaks down when the reflection is sharp, and the third as soon as the ghost effect is not clear enough.

Instead, the most recent models have been built using deep learning techniques. This has led to better results on natural images which are more versatile. The underlying assumption is that the reflection and transmission layers have different distributions, and that machine intelligence can learn to tell the difference. This idea was first implemented by Fan et al. (2017b), in a model called CEILNet. Since then, numerous others have built similar neural networks.

1.4 Utilities

Training a convolutional deep learning model is a computationally heavy task, which requires high-performing GPUs. We performed both training and testing using the GPU cluster Alvis. Alvis is a resource provided by the Swedish National Infrastructure for Computing (SNIC) at Chalmers University, Göteborg. The cluster contains a variation of GPUs, but the majority of times, we trained using a NVIDIA A40. This specific GPU has 8GB VRAM, 64GB system memory and 16 CPU cores.

All scripts are based on Python 3.8. Apart from this, the pre-implemented networks required different packages. See their respective GitHub repositories for further details.

Chapter 2
Dataset

In this chapter, we describe the generation of synthetic data and the gathering of real-world data. Furthermore, we present the most commonly used evaluation metrics in single image reflection removal.

2.1 Real and Synthetic Dataset

One great challenge with single image reflection removal is to retrieve a dataset of images containing reflection and the corresponding ground truth, without reflection. The problem originates in the fact that light refraction varies between materials. Due to this, a photo taken through a plastic dome will be unaligned compared to a photo taken with the same camera, but without dome. A combination of such images can therefore not be considered a pair consisting of a reflection image and its ground truth background. Note that some reflection removal networks are able to handle unaligned data, but this does not hold for the majority.

Instead, we synthesize the images used for training by blending a transmission and reflection layer together. The transmission layer will then act as the target for the neural network. In terms of images containing real reflections, they are only used for evaluation.

2.1.1 Final Synthetic Dataset

We use laboratory reflection images and background images to create synthetic reflections. All these images are RGB, with dimensions 1080×1920 and are gathered from four Axis dome cameras. The different camera models we use are called P5655, Q6075, Q6135, and Q6315. In total, we collected 1488 background images from Axis's database and we took 2060 reflection images in a laboratory environment. We did the latter by placing the specific camera in a dark room and, while aiming a light source at it, took pictures towards a dark wall. Figure 2.1 further explains the set up. By shifting the camera's position and the light source's position and strength, we obtain a diverse set of reflection images.

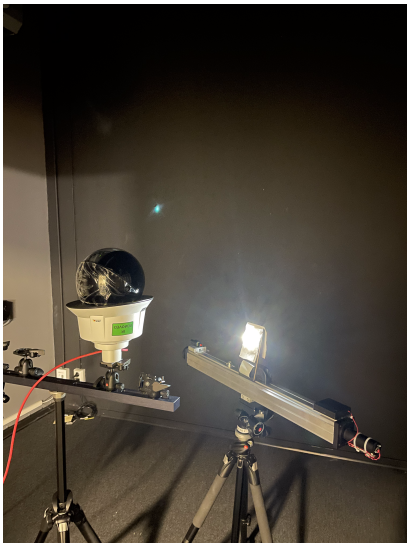


Figure 2.1: Photo of the laboratory environment, where we created the reflection layers.

To create the final dataset, we process the images before blending. Previous single image reflection removal networks have all used synthetic data and similar code for generation. We start from the synthesizing code written by Zhang et al. (2018) and make changes to better fit the requirements for our data. After some trial and error, we find the generated images comparable to real reflection images taken by Axis's dome cameras.

To create a more realistically looking reflection layer, the reflection image is pre-processed using a Gaussian blur and segmentation. The size of the Gaussian blur is a random odd integer between 3 and 17, and the variance is randomly chosen between 0.2 and 4. Moreover, the segmentation threshold is based on which camera and what lab session the reflection layer is created. Due to the large variation in brightness, for camera Q6075, the images are also separated according to a "light" and "dark" category. Table 2.1 shows the segmentation thresholds, lab session, and number of images per camera.

Table 2.1: Camera type, segmentation thresholds, lab session and number of images of that session.

Camera category	Threshold	Lab Session	Nbr of images
Q6135	0.25	1st	92
Q6135	0.22	2nd	391
Q6075 light	0.4	1st	205
Q6075 dark	0.3	1st	362
P5655	0.25	1st	277
Q6315	0.55	1st	39
Q6315	0.37	2nd	85
Q6315	0.3	3rd	609

To allow the generation of a larger, fixed size dataset, while decreasing the risk of overfitting, we augment the images. The data augmentation consists of mirroring and random cropping of both reflection and background layer. We choose the final dataset to contain

4000 blended train images and 445 blended test images.

To create the final dataset, we pair a random background image with a random reflection image. Next, the images are both randomly mirrored – the background only in the y-axis, and the reflection in either the y-axis, x-axis or both. Additionally, we crop the layers to the same dimensions.

However, due to the properties of the reflection layer, we restrict the cropping. One observation of the reflection layer is that the dome reflection is always centered in the x direction, but not necessarily in y. Therefore, to let the dataset have similar properties to real-world images, we keep the reflection centered in the x-direction even after cropping. Moreover, the crop of the reflection image also has to be more restricted to a certain area. This is because we want to keep the main part of the reflection after the images are cropped.

Another fact which influences the cropping is that some of the reflection images contain a water stamp in the upper part of the image. Therefore, the reflection images are split into two categories: layers with and without water stamp.

- If the image does not contain water stamp, we randomly crop it within the range 80-280 pixels on both sides in the x-direction. In the y-direction, we crop it randomly, but not necessarily equally, on both sides within the range 4-200 pixels.
- On the other hand, if the image contains water stamp, we crop it randomly within the range 160-360 pixels on both sides in the x-direction. In the y-direction, we once again crop it randomly, but not necessarily equally, on both sides but now within the range 80-120 pixels.

Once the reflection is cropped, the corresponding (random) background image is cropped according to the dimensions of the reflection image. Thus, this crop is not restricted in any way except dimension-wise. Figure 2.2 shows two examples of a blended images and their ground truths.

Original reflection image. | Processed reflection image. | Background image, i.e. ground truth. | Blended background and reflection.

Figure 2.2: Two cropped blended images with corresponding background and reflection images.

2.1.2 Real-World Data

In terms of real-world data, we gather 38 images from Axis's database, chosen with the intention to cover different types of reflections. For instance, these images cover both strong

and subtle reflections, outdoor and indoor environment, natural and artificial lighting, and differently colored reflections.

2.2 Evaluation Metrics

According to Amanlou et al. (2022), reviewing earlier single image reflection removal papers, there are two frequently used metrics for evaluation. These are the *peak signal-to-noise ratio* (PSNR) and the *structural similarity index measure* (SSIM).

2.2.1 PSNR

The PSNR metric is a global comparison on pixel level between images. PSNR is calculated in decibels (dB) and can be described through the formula:

$$\text{PSNR} = 10 \log_{10} \left(\frac{M^2}{MSE} \right),$$

where M is the maximum possible pixel value and MSE is the mean squared error between the images.

In the extreme case where the images are identical, the MSE becomes zero and the PSNR goes to infinity. In general, the higher the PSNR is, the higher is also the image quality (i.e. similarity between two images). Nevertheless, if the PSNR is higher than 40 dB, the human eye generally has a hard time to observe a distinct difference between the quality in the images. Therefore, this threshold is often used as a guideline of what should be considered high image quality (Chervyakov et al., 2020).

2.2.2 SSIM

As mentioned, the PSNR returns a quality score based on the global variations in two images. However, this is not necessarily how humans would compare and interpret differences in images. The human mind would rather assess the image quality based on the ability to mentally recover information.

SSIM is a method which tries to interpret image quality more similarly to humans. To do this, SSIM takes local luminance, contrast and structural features into account in images, rather than only comparing images globally through the pixel-wise difference.

For instance, even if the contrast between an image and its corresponding reference varies, this does not mean that the human brain would consider the quality to differ greatly. If the compared image instead is blurred, the human mind will recognize a larger decrease in image quality. The MSE could return similar scores for both these artifacts, despite the human brain clearly interpreting the former as higher quality.

As mentioned, the SSIM score is based on the luminance (l), contrast (c), and structure (s). Since these components can be varying within an image, the SSIM is calculated locally within a certain window. The final SSIM score becomes the mean of all local scores. One such local score is calculated according to

$$\text{SSIM}(x, y) = [l(x, y)]^{\alpha} \times [c(x, y)]^{\beta} \times [s(x, y)]^{\gamma},$$

where α, β, and γ are positive parameters defining the relative importance of each component. Moreover, l, c and s are defined as

$$
\begin{aligned}
l(x,y) &= \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}, \\
c(x,y) &= \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}, \\
s(x,y) &= \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3},
\end{aligned}
$$

where μ_x, μ_y, σ_x, σ_y, and σ_{xy} are the local means, standard deviations, and cross-covariance for images x and y. Lastly, C_1, C_2 and C_3 are small constants used to avoid instability when the denominator is close to zero (Wang et al., 2004).

Chapter 3
Architectures

In this chapter, we describe the four networks we evaluate: ERRNet, IBCLN, DADNet, and RAGNet. In order to fully understand the networks, we first present some base architectures. We assume the reader to have basic knowledge of convolutional neural networks. Therefore, we did not include details on this.

3.1 Base Architectures

3.1.1 VGG-19

The VGG-19 network (Simonyan and Zisserman, 2015) is a commonly used pre-trained network for image classification. The network, shown in Figure 3.1, consists of 19 weight layers which are mainly convolutional layers with a kernel size of 3×3. The VGG-19 network stands out as it achieved state-of-the-art results in image classification when it was introduced. This holds despite decreasing the number of network parameters significantly compared to earlier classification networks. Moreover, the pre-trained VGG-19 network generally performs well on a variety of different datasets.

3.1.2 Long Short-Term Memory

Long-short term memory (LSTM) is an architecture to process sequences. It is similar to recurrent neural networks but includes more information. In addition, it can solve backpropagation problems.

Backpropagation is a central algorithm in neural networks. Starting from the output of the loss function, this algorithm makes it possible to propagate backwards through the hidden layers and compute the contribution every node had on the error (Chollet, 2017).

A known problem with backpropagation occurs when propagating back to the deeper layers. Going through the layers, the errors have a tendency of vanishing or exploding exponen-

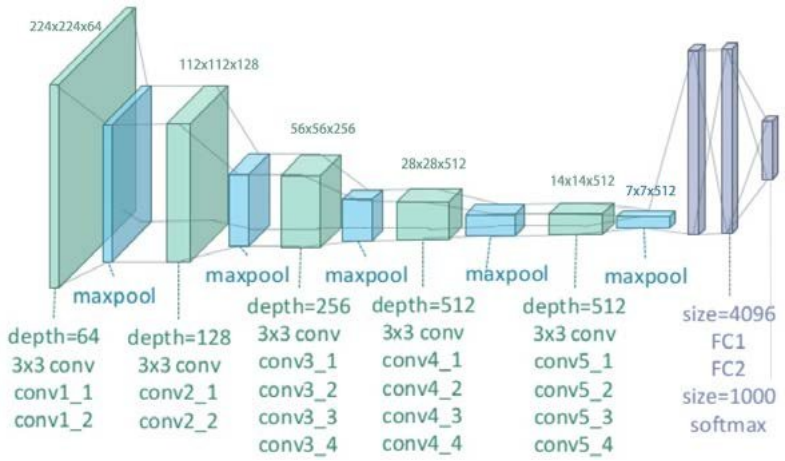


Figure 3.1: General architecture of VGG-19 (Zheng et al., 2018).

tially, giving name to the *vanishing and exploding gradient problem*. While vanishing gradients makes the network unsure of in which direction to adjust the weights, exploding gradients makes the learning procedure unstable (Goodfellow et al., 2016).

LSTM networks introduce a secondary flow of information, parallel with the backpropagated errors. As explained in Chollet (2017), this can be compared to a conveyor belt of information running concurrently with the backpropagation algorithm. At any time, information can be put on this belt and stay intact, which allows relevant information to be extracted when needed. Thus, LSTM networks prevent information from disappearing or expanding during backpropagation.

3.1.3 Encoder-Decoder Networks

An encoder-decoder network consists of two parts: an *encoder* and a *decoder*. As explained by Ye and Sung (2019), the encoder takes an input and maps it onto a feature space. In the next step, the decoder will take this feature map and decode it into an output. The two parts work together and the layers are trained jointly to learn to extract helpful features and generate an optimal output.

Figure 3.2 gives an example of an encoder-decoder architecture proposed by Badrinarayanan et al. (2015) for image segmentation. As one can see, the encoding layers iteratively perform convolution and pooling to obtain lower-level feature maps of the input. During the phase of decoding, these will be translated into an output using up-sampling. To avoid losing too much information, the encoder and decoder layers are connected, as illustrated by the arrows in the figure.

When handling image artifacts, the encoder part will ideally extract the representation of these artifacts in the feature maps and remove them. The decoding layers will then be able to reconstruct the image without the unwanted parts. Stepping through the decoder layers, the generated image will gradually become more detailed until reaching its original size. Thus, the encoder-decoder network learns to map a flawed image to an improved version (Mao et al., 2016).

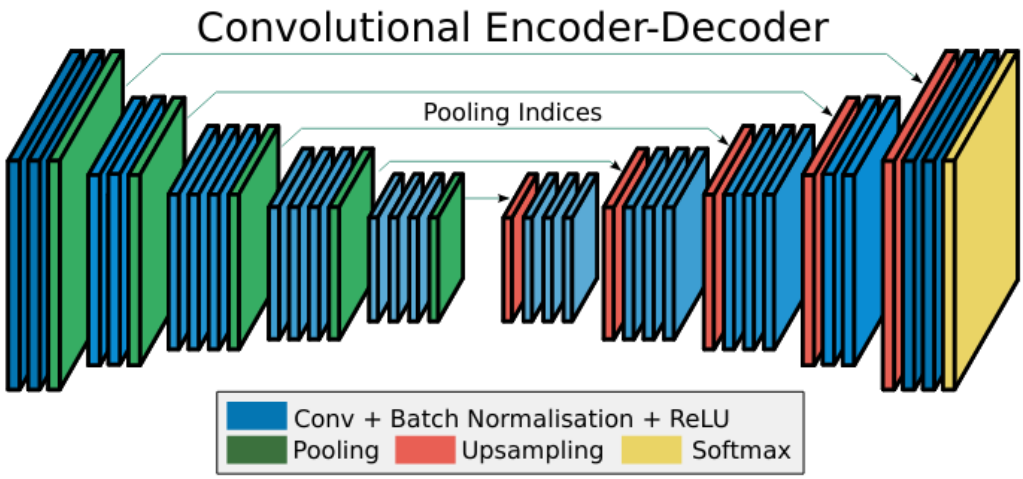


Figure 3.2: An example of an encoder-decoder architecture used in a convolutional neural network. (Badrinarayanan et al., 2015)

3.1.4 Generative Adversarial Networks

A collection of images displaying some object or scene can be assumed to represent a certain distribution. Imagine that one would like to generate fake but realistic images of a similar scene or object. A plausible approach would be to try to mimic the distribution to which these images belong. This type of image generation can be done using *generative adversarial networks* (GANs) (Goodfellow et al., 2014).

These networks work through the use of a generator, G, and discriminator, D, competing against each other. The general idea of this network is to train G to generate fake, but realistic images, and D to distinguish between generated and real images. To achieve this, the discriminator is trained with both fake and real images to output the probability of the image being real. At the same time, the generator wants to fool the discriminator by trying to produce realistic images from a latent space vector $\mathbf{z}$.

Let us consider the generator G, the latent space vector $\mathbf{z}$, and the set of real images X. Then, the generator G will try to generate images which could be mistaken as some $\mathbf{x} \in X$. This means the aim of the generator becomes forming a distribution $p_{\mathbf{z}}(\mathbf{z})$ which after mapping through G mimics the real image distribution $p_{data}(\mathbf{x})$.

On the other hand, the discriminator is trained to maximize the probability of assigning the correct label to real and fake images. Thus, for some real image $\mathbf{x} \in X$, the ideal behavior of D is to output $D(\mathbf{x}) = 1$. Furthermore, for some fake image $\hat{\mathbf{x}} \in G(\mathbf{z})$, the ideal behavior of D is to output $D(\hat{\mathbf{x}}) = 0$.

This competition between D and G can be viewed as a minimax game. To obtain this behavior by D and G, the minimax value function is defined:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]. \tag{3.1}$$

Let us assume the discriminator is behaving optimally, namely, correctly classifying generated

fake images as fake, and real images as real. Then Equation 3.1 becomes

$$\begin{aligned} V(D,G) &= \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \\ &= \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log(1)] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - 0)] \\ &= \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[0] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[0] = 0, \end{aligned}$$

which is the maximization of Eq. 3.1.

In practice, both the generator and discriminator will get increasingly better at their respective tasks throughout the training process. Eventually, the generator will be able to mimic the distribution of real images. This makes the distribution of real and fake data indistinguishable for the discriminator. Therefore, the optimal behavior of the discriminator would be to classify the probability of an image being real or fake equally. Namely, a 50% chance of either. When this state is reached, then

$$\begin{aligned} V(D,G) &= \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))], \\ &= \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log(0.5)] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - 0.5)], \\ &= -\log 4, \end{aligned}$$

which is the global optimum for the minimax game. When reaching this state, parameters of both the discriminator and generator will have converged. Figure 3.3 shows a converging adversarial process.

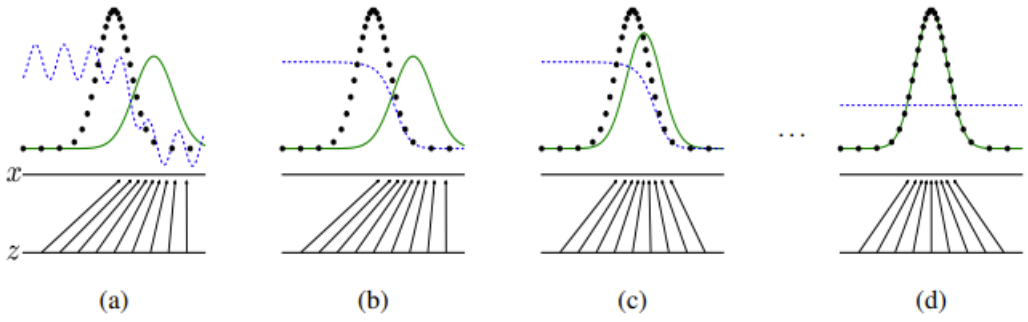


Figure 3.3: The evolution of a general GAN. The upward arrows represents the mapping $G(\mathbf{z})$. The dotted black curve shows the real data distribution. The green curve shows the distribution estimated by G. Furthermore, the blue curve shows the discriminator's outputted probability. As seen in (a), the discriminator is quite certain of a data point either being real or fake – those points at left are according to D quite likely to be real, whereas points at the right are quite likely to be fake. In (b) and (c), we see the evolution of D and G as they are both getting better at their respective task. Finally in (d), convergence is reached – the generator now perfectly mimics the real data distribution and the discriminator cannot distinguish between real and fake data points (Goodfellow et al., 2014).

3.2 Neural Networks for Single Image Reflection Removal

As mentioned in Section 1.3, there already exist multiple neural networks for single image reflection removal. In this thesis, we trained four of these pre-implemented networks on our synthetic dataset.

In order to be able to train and thereafter evaluate, the networks have to include code and instructions for both training and testing. To avoid having to start training from scratch, we have a requirement that there exists a saved pre-trained model which can be used for fine-tuning. Since there is no consensus about which neural network to consider state-of-the-art in single image reflection removal, we assume the most recent ones are the best. We choose the networks RAGNet, DADNet, IBCLN and ERRNet. At the time of writing, these are the latest published networks under the task Reflection Removal at www.paperswithcode.com, which also fulfilled our criteria.

In this section, we present the networks in their publication order. Note that the networks sometimes use similar structures and loss functions, but generally different notations.

3.2.1 ERRNet

Wei et al. (2019) introduced the Enhanced Reflection Removal Network (ERRNet). The term enhanced reflection removal refers to the fact that ERRNet uses channel-wise context and multi-scale spatial context modules. Unlike the three other networks, ERRNet can handle both aligned and unaligned image pairs.

Since ERRNet can handle both aligned and unaligned data, different loss functions are used in these two cases. Our data only consists of aligned image pairs and thus only these loss functions are relevant. In total, ERRNet makes use of three losses for the aligned data: a pixel loss, feature loss and adversarial loss. Figure 3.4 shows the general architecture of ERRNet (Wei et al., 2019).

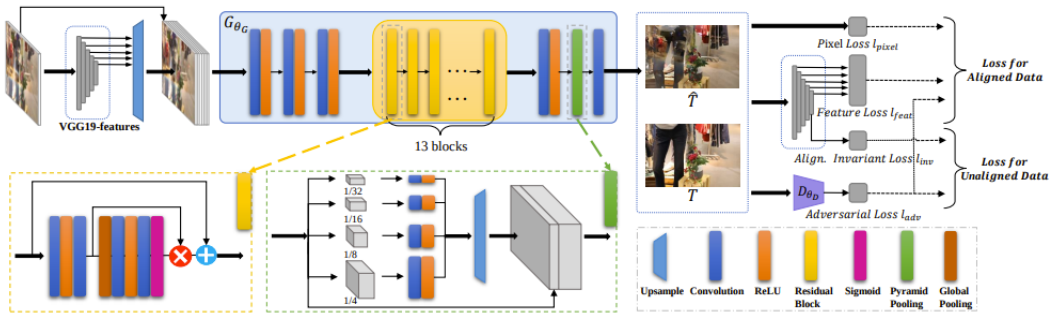


Figure 3.4: The general architecture of ERRNet (Wei et al., 2019).

Channel-Wise Context Modules

In a classic convolutional network, each of the learned filters only operates locally. This leads to an impaired ability to exploit contextual information outside of the feature map. The

assumption in ERRNet is that information about the reflection and transmission layer can be found in the feature map channels (Wei et al., 2019).

Consequently, ERRNet uses a network proposed in Hu et al. (2017), called the squeeze and excitation network. In ERRNet, Figure 3.4, this part represents the lower left block framed in yellow. This method works by squeezing the spatial information into scalar channel descriptors. These channel descriptors are formed via a global average pooling operation, which sums and averages each value in the corresponding feature map. Once the descriptors are created, they are passed through a smaller, so called, excitation net to determine their relative importance. This allows for the network to suppress less important feature maps and promote others.

Assume the input $\mathbf{X} \in \mathbb{R}^{H' \times W' \times C'}$ is sent through the feature transformation $\mathbf{F}_{tr}$. Further let $\mathbf{U} \in \mathbb{R}^{H \times W \times C}$ denote the output feature maps of this transformation. Then the predicted channel-wise importance can be determined as seen in Figure 3.5. Here, the feature maps $\mathbf{U}$ are inputted to the transformation $\mathbf{F}_{sq}$. The output of $\mathbf{F}_{sq}$ are the feature maps reduced to one dimension, obtained via global average pooling. Thus, the output of $\mathbf{F}_{sq}$ is a vector of dimension $1 \times 1 \times \boldsymbol{C}$. This vector is then passed through the excitation block: $\mathbf{F}_{ex}(\cdot, \mathbf{W})$, where $\mathbf{W}$ are weights learned throughout the training. The aim of the weights $\mathbf{W}$ is to suppress less important feature maps and promote others. The last step is to combine the weights with the input $\mathbf{U}$ scaled through $\mathbf{F}_{scale}(\cdot, \cdot)$, outputting $\tilde{\mathbf{X}}$. Here, $\tilde{\mathbf{X}}$ has the exact same dimensions as $\mathbf{U}$, that is $\mathbb{R}^{H \times W \times C}$. However, unlike $\mathbf{U}$, $\tilde{\mathbf{X}}$ has a notion of the relative importance of its feature maps (Hu et al., 2017).

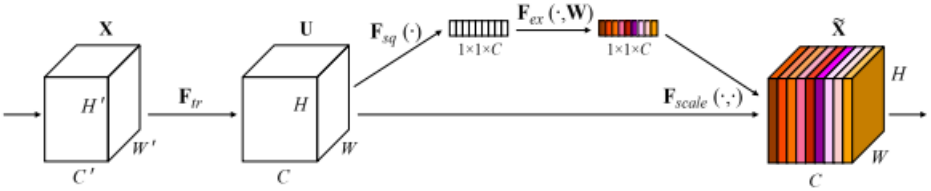


Figure 3.5: General architecture of the squeeze and excitation network (Hu et al., 2017).

Multi-Scale Spatial Context Module

In addition to the channel-wise context module, ERRNet uses a multi-scale spatial context module for each channel. This module is responsible for extracting information through the spatial dimensions instead of through the channels. In Wei et al. (2019), this is done through something called pyramid pooling (see the lower middle block framed in green in Figure 3.4).

Pyramid pooling was first introduced in He et al. (2014). In semantic segmentation, pyramid pooling has shown to be effective in representing the global scene. Unlike classic pooling, this approach uses several pooling operations for the input feature map. Moreover, the pyramid pooling layer is often situated at the tail of the network, as the last step before the final output. Figure 3.6 shows the general principle of pyramid pooling.

As stated in Wei et al. (2019), ERRNet uses four different pooling operations at the end of the network. These layers output feature maps with spatial dimensions scaled according to the lower, middle block with green frame in Figure 3.4. Once the four different pooling

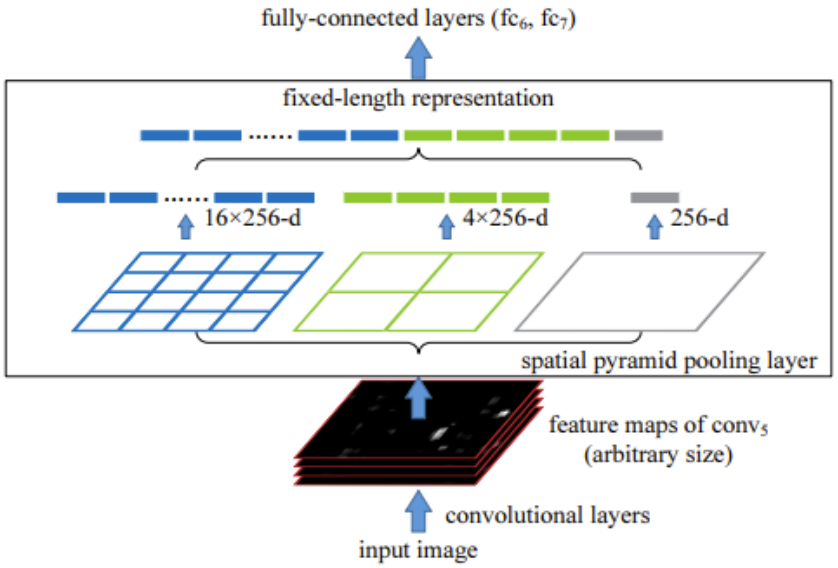


Figure 3.6: General principle of a network structure containing spatial pyramid pooling. Here 256 is the filter number of the conv_5 layer, and conv_5 is the last convolutional layer (He et al., 2014).

operations are calculated, they are proportionally concatenated as visualized in Figure 3.6. The last step involves upsampling to the desired output size.

Pixel Loss

An indication of a good prediction of the transmission layer would be if the pixel value difference between the prediction and the ground truth is small. ERRNet does not only use the plain pixel-wise difference but also the pixel-wise gradient difference (Wei et al., 2019). As explained in Fan et al. (2017a), the gradient will detect edges in the image. This could be a useful feature in reflection removal since the reflection layer likely contains detectable edges.

Let T denote the transmission layer and $\hat{T}$ denote the predicted transmission, then the pixel-wise intensity difference is penalized via the pixel loss function

$$l_{\text{pixel}} = \alpha \|\hat{T} - T\|_2^2 + \beta \left(\left\| \nabla_x \hat{T} - \nabla_x T \right\|_1 + \left\| \nabla_y \hat{T} - \nabla_y T \right\|_1 \right).$$

Here, ∇_x and ∇_y are the gradient operator along the $\mathbf{x}$- and $\mathbf{y}$-direction, respectively. In practice, $\alpha = 0.2$ and $\beta = 0.4$ (Wei et al., 2019).

Feature Loss

The idea behind the feature loss is to investigate features on different decoder layers. This enables a comparison of the output and ground truth in different resolutions. Thus, both high and low level information can be compared between the predicted transmission and the target transmission image (Zhang et al., 2018).

In the ERRNet implementation, the comparison between the generated transmission and the true transmission is done at different levels in the pre-trained VGG19 network, see

Simonyan and Zisserman (2015). The loss function is described as

$$l_{\text{feat}} = \sum_{l} \lambda_l \left\| \phi_l(T) - \phi_l(\hat{T}) \right\|_1 .$$

In this function, ϕ_l is the feature from the l-th layer of VGG-19 and $\{\lambda_l\}$ are balancing weights. The layers considered in VGG-19 are conv2_2, conv3_2, conv4_2, and conv5_2, displayed in Figure 3.1 (Wei et al., 2019).

Adversarial Loss

By the use of an adversarial loss, one can ensure the generated transmission layer looks realistic. ERRNet will, from a blended image, aim to approximate a realistically looking transmission layer by the use of a generative adversarial network. As always in adversarial processes, we make use of a generator G_{θ_G} and discriminator D_{θ_D} (Wei et al., 2019).

In the ERRNet implementation, the discriminator is relativistic, as introduced in Arjovsky et al. (2017). The relativistic term means that D_{θ_D} will not output the probability of a transmission layer being real. Instead, the main goal of D_{θ_D} is to output the probability that the given real data is more realistic than a randomly sampled fake data. The goal of the overall network is to make the generator generate realistically looking transmission layers. Therefore, as stated by Wei et al. (2019), the generator loss is equivalent to the whole adversarial loss according to Equation (3.2).

Let T denote the transmission layer, and $\hat{T}$ a randomly sampled predicted transmission layer. Then, the adversarial loss can be expressed as

$$l_{adv} = l_{adv}^{G} = -\log\left(D_{\theta_D}(\hat{T}, T)\right) + \log\left(1 - D_{\theta_D}(T, \hat{T})\right). \tag{3.2}$$

In this function, $\hat{T}$ is generated via G_{θ_G} and $D_{\theta_D}(x, y) = \sigma\left(C(x) - C(y)\right)$ with $\sigma(\cdot)$ being the sigmoid function. As explained in Arjovsky et al. (2017), $C(\cdot)$ can be interpreted as how realistic the input data is. A negative number means that the input data looks fake, and a positive number indicates the input data looks real. Passing this through the sigmoid function will output a probability between 0 and 1, as we expect the discriminator to do.

During training, the goal of the generator is to minimize the loss in Eq. (3.2). To do this, the generator has to get increasingly better at generating realistic transmission layers. The procedure would not converge unless the discriminator and generator network simultaneously got better. Once convergence is reached

$$\sigma\left(C(T) - C(\hat{T})\right) = \sigma\left(C(\hat{T}) - C(T)\right) = \sigma(0) = 0.5$$

holds, indicating the discriminator can no longer distinguish between real and fake data.

Overall Loss

Summing all of the previous losses together, the loss function for aligned data becomes

$$l_{aligned} = w_1 l_{pixel} + w_2 l_{feat} + w_3 l_{adv},$$

where $w_1 = 1, w_2 = 0.1$, and $w_3 = 0.01$.

3.2.2 IBCLN

Li et al. (2019) proposed an Iterative Boost Convolutional LSTM Network (IBCLN). The name refers to the fact that this network iteratively predicts the transmission and reflection layer by the use of cascaded sub-networks. Furthermore, the information between the sub-networks is passed on via LSTM networks.

As explained in Li et al. (2019), the assumption in IBCLN is that the transmission layer is a dominant feature in the image, whereas the reflection layer has weaker features. In the ultimate case, the network has auxiliary information in the form of ground truth transmission layer, which is the goal to predict. Obviously, this is not available, since if it was, the problem with reflection removal would not exist. Nevertheless, if one could use an approximation of the transmission layer as auxiliary information, this would provide guidance for the network.

In IBCLN, a reflection image is assumed to be described through the equation $I = \alpha T + R$, where α is a constant, usually within the range 0.8-1. Here, α is used to model the slight attenuation of light as it passes through the reflective surface. This means that if one could make a prediction, $\hat{T}$, of the dominant layer T, one would also be able to predict the residual reflection $\hat{R} = I - \alpha\hat{T}$. To do this, IBCLN makes use of two sub-networks – one which predicts the dominant transmission layer and another which predicts the weaker reflection layer (Li et al., 2019).

The two sub-networks share the same architecture. First is an encoder, consisting of convolutional layers extracting the image features. This is followed by a convolutional LSTM unit, which in turn is connected to a convolutional decoder network. Figure 3.7 gives an overview of such sub-network block.

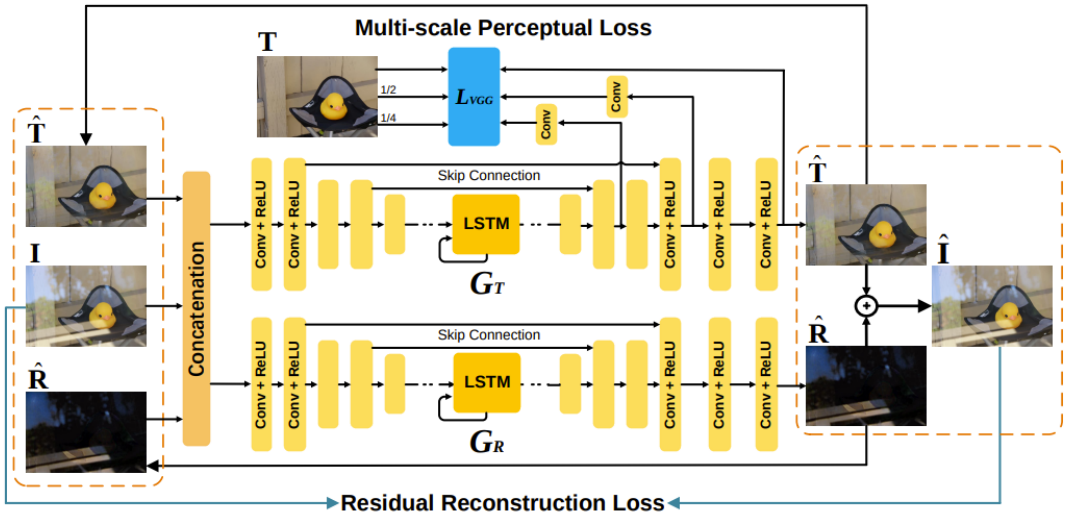


Figure 3.7: The general architecture of one prediction iteration in IBCLN (Li et al., 2019).

The idea in IBCLN is to use several connected sub-networks to make iterative predictions of reflection and transmission layers. As a first step, an initial transmission layer, T_0, and residual reflection layer, R_0, are introduced. These initial layers are always assumed to be the same. That is, T_0 is assumed to be the image I and R_0 is assumed to be 0.1 for all entries.

These initial layers are used as input in the first sub-network. Then, the output of this

sub-network is passed on as input to the next sub-network, and so on. Thus, in each step, the network has the approximations of both layers available. This cycle continues until convergence is reached. See Figure 3.8 for a visualization of the cascaded prediction process in IBCLN (Li et al., 2019).

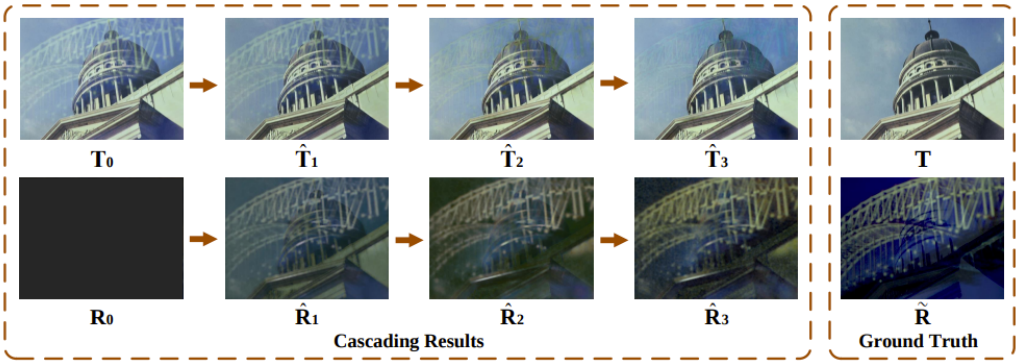


Figure 3.8: The cascaded prediction steps used in IBCLN (Li et al., 2019).

Figure 3.9 shows the general architecture of IBCLN. Between each prediction of T and R, the two sub-networks G_T and G_R are seen. Here, each G_T and G_R resembles the sub-network structure in Figure 3.7.

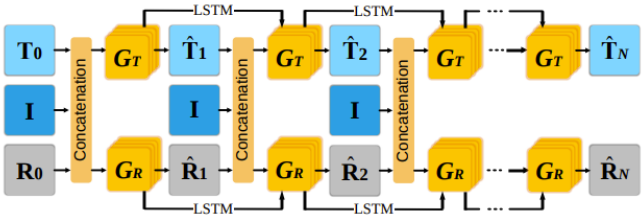


Figure 3.9: The general architecture of IBCLN (Li et al., 2019).

As always when training neural networks, the loss functions are of great importance. In IBCLN, four different loss functions are used, the so-called: residual reconstruction loss, multi-scale perceptual loss, pixel loss and adversarial loss (Li et al., 2019).

Residual Reconstruction Loss

One important aspect of the generation of transmission and reflection layer is that the combination of both layers should resemble the original image. To make sure the network accounts for this, IBCLN uses a so called residual reconstruction loss.

Consider the image $\hat{I} = \alpha\hat{T} + \hat{R}$, being the reconstructed image from the predicted reflection layer $\hat{R}$, the predicted transmission layer $\hat{T}$ and a constant α (usually within the range 0.8-1). Then, the residual reconstruction loss is described as

$$\mathcal{L}_{\text{residual}} = \sum_{I \in \mathcal{D}} \sum_{t=1}^{N} \mathcal{L}_{MSE}\left(I, \hat{I}_t\right),$$

where $\mathcal{L}_{MSE}$ is the mean squared error, t is the time step of the two sub-networks, $\mathcal{D}$ denotes the dataset and N is the final time step when $\hat{T}$ converges (Li et al., 2019).

Multi-Scale Perceptual Loss

The multi-scale loss in IBCLN closely resembles the feature loss in ERRNet. As in ERRNet, the comparison between the generated transmission and the true transmission is done at different decoder levels by the use of the pre-trained VGG19 network. At the top of Figure 3.7, the multi-scale perceptual loss is visualized.

The loss function is described as

$$\mathcal{L}_{MP} = \sum_{T,T^3,T^5 \in \mathcal{D}} \Big(\mathcal{L}_{VGG}(T, \hat{T}) + \gamma_3 \mathcal{L}_{VGG}\left(T^3, \hat{T}^3\right) + \gamma_5 \mathcal{L}_{VGG}\left(T^5, \hat{T}^5\right) \Big),$$

where $\hat{T}$, $\hat{T}^3$, and $\hat{T}^5$ corresponds to the outputs of conv1_2, conv3_2, and conv5_2 for time step N in VGG-19, see Figure 3.1. Moreover, T, T^3, and T^5 denote the ground truth transmission layer, at the same scale as the corresponding approximated transmission. Lastly, $\gamma_3 = 0.8$ and $\gamma_5 = 0.6$ (Li et al., 2019).

Pixel Loss

In IBCLN, there is an additional pixel level loss to make sure the output transmission and reflection layers are as close as possible to their respective ground truth. This loss is described through

$$\mathcal{L}_{\text{pixel}} = \sum_{T \in \mathcal{D}} \sum_{t=1}^{N} \left[\mathcal{L}_{MSE}\left(T, \hat{T}_t\right) + \mathcal{L}_{MSE}\left(\widetilde{R}, \hat{R}_t\right) \right].$$

In this loss, $\widetilde{R} = I - \alpha \cdot T$ is the residual reflection, and $\hat{T}_t$ and $\hat{R}_t$ are the transmission and reflection outputs at time step t (Li et al., 2019).

Adversarial Loss

As in ERRNet, IBCLN uses an adversarial loss to improve the realism of the predicted transmission layer. This loss is similarly connected to the ability of discriminator D. By the use of the adversarial loss introduced in Zhang et al. (2018), IBCLN defines

$$\mathcal{L}_{adv} = \sum_{T \in \mathcal{D}} -\log D(T, \hat{T}),$$

where I and $\mathcal{D}$ denote the original image and dataset, respectively. Here, $D(T, \hat{T})$ outputs the probability that the prediction $\hat{T}$ is a real transmission image given the input transmission T.

Overall Loss

The overall loss is obtained by summing all of the previous losses together, and multiplying with a respective weight, according to

$$L = \lambda_1 \mathcal{L}_{\text{residual}} + \lambda_2 \mathcal{L}_{MP} + \lambda_3 \mathcal{L}_{\text{pixel}} + \lambda_4 \mathcal{L}_{adv}.$$

As stated in Li et al. (2019), the weights in IBCLN have empirically been set to $\lambda_1 = 2, \lambda_2 = 1, \lambda_3 = 2, \lambda_4 = 0.01$.

3.2.3 DADNet

The name DADNet refers to the term Deep Adversarial Decomposition Network. Compared to the other networks, DADNet does not have the sole objective to filter out reflections in images. Instead, the ambition is broader – to separate different kinds of single mixed images. This network can thus be used in various tasks, such as image deraining, photo reflection removal and image shadow removal (Zou et al., 2020).

To output the layer separation, DADNet uses an adversarial training process. A generative network G is trained to minimize the distance between the separated outputs and their corresponding ground truth. To make sure the output layers are well-separated, a discriminator D_C is added to the network. As always in adversarial training, the generator and discriminator are competing in a minimax game. In DADNet, convergence is reached when the discriminator is unable to distinguish between the quality of the layer separation performed by the generator and the quality of two real, actually separate layers (Zou et al., 2020).

In some areas of a mixed image, the linearity $I = T + R$ does not hold. Therefore, the predicted layer separation might not be realistic. DADNet tries to solve this problem using two (Markovian) discriminator networks, D_{M1} and D_{M2}, which input a respective patch of the two predicted and separated layers and classifies it as real or fake. Figure 3.10 shows the general architecture of DADNet.

To invoke the behavior described in this section, DADNet makes use of three losses: a so called crossroad loss, separation critic loss and something (in our project) referred to as Markovian loss (Zou et al., 2020). Note that DADNet uses reverse notation for prediction and ground truth, compared to the other networks. Namely, in DADNet x denotes the prediction and $\hat{x}$ the ground truth.

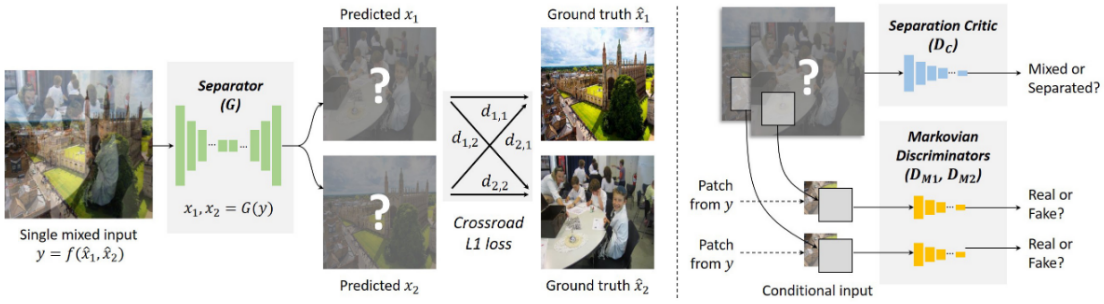


Figure 3.10: The general architecture of DADNet (Zou et al., 2020).

Crossroad Loss

Assume $\hat{x}_1$ and $\hat{x}_2$ represent the true transmission and reflection layer, and that $y = f(\hat{x}_1, \hat{x}_2)$ is their mixture. Here, the function $f(\cdot)$ is unknown and can be either linear or non-linear. Given the mixture image y, the generator G aims to predict the separate layers x_1 and x_2, according to

$$x_1, x_2 = G(y).$$

In DADNet, x_1 and x_2 are not necessarily output in a certain order. Therefore, the standard pixel-wise difference between the output and ground truth can not be directly computed. Instead, the crossroad loss is defined as:

$$l_{\text{cross}}\left(\left(x_1, x_2\right), \left(\hat{x}_1, \hat{x}_2\right)\right) = \min\left\{\|x_1 - \hat{x}_1\|_1 + \|x_2 - \hat{x}_2\|_1,\ \|x_1 - \hat{x}_2\|_1 + \|x_2 - \hat{x}_1\|_1\right\}.$$

With the use of this loss, the order of outputs does not matter as the minimum of the above will always correspond to the correct pairing between output and ground truth. Thus, the crossroad loss can be described through:

$$\mathcal{L}_{\text{cross}} = E_{\hat{x}_i \sim p_i(\hat{x}_i)}\left\{l_{\text{cross}}\left(\left(x_1, x_2\right), \left(\hat{x}_1, \hat{x}_2\right)\right)\right\},$$

where $p_i\left(\hat{x}_i\right)$ represents the distribution of the image data, and $i \in \{1, 2\}$ (Zou et al., 2020).

Separation Critic Loss

As stated in Zou et al. (2020), a separation critic loss is used in DADNet. The separation critic loss aims to make the two output layers as well-separated as possible. This is an adversarial training process where a generator G tries to minimize the distance between the outputs and ground truth. On the other hand, a discriminator D_C tries to maximize this distance. Thus, the discriminator is supposed to output the probability of the predicted layers being well-separated. Namely, output something closer to 1 if the layers are well-separated and something closer to 0 otherwise.

Let (x_1, x_2) be the predicted layer separation and $(\hat{x}_1, \hat{x}_2)$ be the ground truth separation. Then the loss function is summarized:

$$\begin{aligned}\mathcal{L}_{\text{critic}}\left(G, D_C\right) = {} & E_{\hat{x}_i \sim p_i(\hat{x}_i)}\left\{\log D_C\left(\hat{x}_1, \hat{x}_2\right)\right\} + E_{x_i \sim p_i(x_i)}\left\{\log\left(1 - D_C\left(x_1, x_2\right)\right)\right\} \\ & + E_{\hat{x}_i \sim p_i(\hat{x}_i)}\left\{\log\left(1 - D_C\left(x_1', x_2'\right)\right)\right\}.\end{aligned}$$

It is desirable that the discriminator learns to recognize a blend of two images as something which is not well-separated. Hence (x_1', x_2') is considered in the last term, corresponding to the linear mix of the ground truth layers, according to

$$x_1' = \alpha\hat{x}_1 + (1 - \alpha)\hat{x}_2, \quad x_2' = (1 - \alpha)\hat{x}_1 + \alpha\hat{x}_2,$$

where α is a random weight.

Markovian Loss

As mentioned, DADNet also makes use of two discriminator networks, D_{M1} and D_{M2}. These networks input a respective $N \times N$ patch of the two predicted layers and classifies it as real or fake. The discriminators will output the probability of the image patch being real. Optimal behavior for the discriminator is to classify patches of ground truth layers as real, and patches of generated layers as fake. For the discriminator to minimize the loss during these cases, the loss function is defined as:

$$\begin{aligned}\mathcal{L}_{Mi}\left(G, D_{Mi}\right) = {} & E_{(\hat{x}_i, y) \sim p_i(\hat{x}_i, y)}\left\{\log D_{Mi}\left(\hat{x}_i \mid y\right)\right\} \\ & + E_{(x_i, y) \sim p_i(x_i, y)}\left\{\log\left(1 - D_{Mi}\left(x_i \mid y\right)\right)\right\},\end{aligned}$$

where $i = 1, 2$ (Zou et al., 2020).

Overall Loss

The overall objective function in DADNet becomes

$$\mathcal{L}(G, D_C, D_{Mi}) = \mathcal{L}_{\text{cross}}(G) + \beta_C \mathcal{L}_{critic}(G, D_C) + \beta_M \sum_{i=1,2} \mathcal{L}_{Mi}(G, D_{Mi}),$$

where β_C and β_M are positive weights (Zou et al., 2020).

3.2.4 RAGNet

The name RAGNet relates to the Reflection Aware Guidance used in the network. The overall structure of RAGNet is an encoder-decoder network.

During filtering, the transmission layer prediction is split into two stages. In the first stage, the reflection layer is estimated using the simple U-net segmentation, explained in Ronneberger et al. (2015). In the second stage, the initially predicted reflection is sent through the encoder part of the network. After this, the encoded reflection is sent to a decoder network, where the transmission is predicted. The prediction is made using, so called, reflection aware guidance modules (RAG modules) – hence, the name RAGNet (Li et al., 2020).

Let $\hat{\boldsymbol{R}}$ and $\hat{\boldsymbol{T}}$ denote the reflection and transmission layer predictions, respectively. In areas with high intensity reflection, the linearity $\hat{\boldsymbol{T}} = \boldsymbol{I} - \hat{\boldsymbol{R}}$ usually does not hold. Subtracting the estimated reflection layer $\hat{\boldsymbol{R}}$ from the image $\boldsymbol{I}$ will in these cases lead to darker areas than desired in the predicted transmission $\hat{\boldsymbol{T}}$. A seen in Figure 3.11, the transmission prediction $\boldsymbol{I} - \hat{\boldsymbol{R}}$ does not look accurate.

Figure 3.11: Shows the darker areas in the predicted transmission $\hat{\boldsymbol{T}} = \boldsymbol{I} - \hat{\boldsymbol{R}}$. These can appear when linearly subtracting the reflection layer $\hat{\boldsymbol{R}}$ from an image with strong reflection (Li et al., 2020).

To account for the non-linearity in these heavy reflection areas, RAGNet makes use of the RAG module. In the RAG module, a mask $\boldsymbol{M}$ is introduced to further exploit the reflection layer. The prediction of the transmission layer is then based on this mask, which describes the properties of the reflection layer. Thus, the term reflection-aware guidance comes from the fact that the transmission layer is predicted through guidance of the estimated reflection and its properties.

Areas where the mask is close to zero indicates the linearity $\boldsymbol{I} = \boldsymbol{R} + \boldsymbol{T}$ does not hold. Therefore, an ordinary subtraction is not going to recover the transmission layer. Instead,

image inpainting (a task of reconstructing missing regions in an image) would be a better approach in these areas (Li et al., 2020).

Similarly to IBCLN, RAGNet makes use of a reconstruction loss, perceptual loss and adversarial loss. Regarding these losses, they mostly differ in notation between these networks, even though there are some smaller differences in the way they are calculated. In addition to these losses, an exclusion and mask loss is used in RAGNet (Li et al., 2020). Figure 3.12 illustrates the second stage of RAGNet.

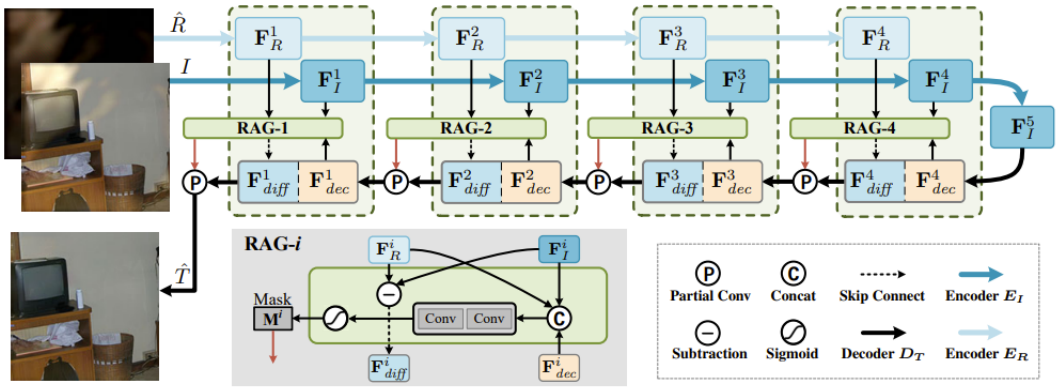


Figure 3.12: The general architecture of the second stage of RAGNet, where the first stage reflection estimation $\hat{R}$ is used as input (Li et al., 2020).

Reconstruction Loss

In the case of synthetic images, the ground truths T and R are available. Assume the network outputs the predictions $\hat{T}$ and $\hat{R}$, then the pixel-wise difference between the outputs and the corresponding ground-truths T and R can be calculated through

$$\mathcal{L}_{\text{rec}} = \sum_{Y \in \{T,R\}} \|\hat{Y} - Y\|_1.$$

In RAGNet, this is called the reconstruction loss.

Perceptual Loss

Similarly to ERRNet and IBCLN, RAGNet also uses a perceptual loss. Let us assume that ϕ represents the pre-trained VGG-19 model. The goal of the perceptual loss function is to minimize the ℓ_1 difference between $\phi(\hat{T}), \phi(\hat{R})$ and $\phi(T), \phi(R)$ in conv1_2, conv2_2, conv3_2, conv4_2, and conv5_2 layers. Accordingly, the loss function becomes

$$\mathcal{L}_{\text{percep}} = \sum_{Y \in \{T,R\}} \sum_{l} \kappa_l \left\| \phi_l(\hat{Y}) - \phi_l(Y) \right\|_1,$$

where l indicates the index of the convolutional layer in VGG-19. Moreover, $\{\kappa_l\}$ are weights used to balance different layers (Li et al., 2020).

Adversarial Loss

Just like IBCLN, RAGNet uses a discriminator network to calculate the adversarial loss. According to Li et al. (2020), the discriminator consists of 4 layers and the whole RAGNet is considered to be the generator G. The adversarial loss function is defined as

$$\mathcal{L}_{\text{adv}} = -\mathbb{E}_I \log D(I, G(I)),$$

where I is the input image.

Exclusion Loss

The exclusion loss used in RAGNet is directly taken from Zhang et al. (2018). The loss works in the gradient domain, where the edges of an image are clearer. Based on the assumption that the edges of both layers are unlikely to correlate, the number of shared edges is minimized.

The loss function is defined as

$$\mathcal{L}_{\text{excl}} = \frac{1}{N+1} \sum_{n=0}^{N} \sqrt{\left\| \Psi\left(T^{\downarrow n}, R^{\downarrow n}\right) \right\|_F},$$

where $\Psi(T, R) = \tanh\left(\lambda_T |\nabla T|\right) \circ \tanh\left(\lambda_R |\nabla R|\right)$, with normalization factors λ_T and λ_R. Moreover, ∇T and ∇R are the gradients of T and R, and $\| \cdot \|_F$ denotes the Frobenius norm. Lastly, $T^{\downarrow n}$ and $R^{\downarrow n}$ denote the n times down-sampled versions of T and R, where $T^{\downarrow 0}$ and $R^{\downarrow 0}$ are the original inputs. In this implementation, $N = 2$, $\lambda_T = \frac{1}{2}$, and $\lambda_R = \frac{\|\nabla T\|_1}{\|\nabla R\|_1}$.

Mask Loss

As mentioned, RAGNet utilizes a mask to invoke guidance for the network in areas of heavy reflection. In these areas, linearity between the image, reflection and transmission layers will not hold, i.e. $T \neq I - R$. If linearity does not hold, the corresponding feature maps will neither be linear (Li et al., 2020).

Let the feature maps for the original image and reflection layer be denoted $\mathbf{F}_{\text{I}}$ and $\mathbf{F}_{\text{R}}$, respectively. Further let the difference between these be $\mathbf{F}_{\text{diff}} = \mathbf{F}_{\text{I}} - \mathbf{F}_{\text{R}}$. As one can imagine when observing Figure 3.11, $\mathbf{F}_{\text{diff}}$ will not be useful in extracting features of the transmission layer, in the case of heavy reflection. In fact, the case of heavy reflection removal rather mimics an inpainting problem. This is a consequence of that the reflection layer is intense, and thus, the transmission layer will not be recovered from the linear relationship $T = I - R$.

Li et al. (2020) conclude that different reflection areas in the image have to be treated differently depending on their respective intensity. This means that if $\mathbf{F}_{\text{diff}}$ cannot recover the information, it is an indication that inpainting should be used in these areas. Therefore, $\mathbf{F}_{\text{diff}}$ is not useless in these areas but can rather work as a complement to the decoder feature, $\mathbf{F}_{dec}$. To take advantage of this information, RAGNet utilizes a mask $\mathbf{M} = \left[\mathbf{M}_{diff}, \mathbf{M}_{dec}\right]$, where $[\cdot, \cdot]$ denotes the concatenation operation, and $\mathbf{M}_{\text{diff}}$ and $\mathbf{M}_{dec}$ are masks corresponding to $\mathbf{F}_{\text{diff}}$ and $\mathbf{F}_{dec}$, respectively. Figure 3.13 shows an example of a learned mask, $\mathbf{M}$.

As covered by Liu et al. (2018), inpainting can be solved through partial convolutions, by the use of a mask M. In RAGNet, the mask $\mathbf{M}_{\text{diff}}$ is used to determine which areas in the

image that are linear, and which are not. Moreover, the partial convolution is defined as

$$\mathbf{F}' = \begin{cases} (\mathbf{W} * (\mathbf{F} \circ \mathbf{M})) \circ \frac{1}{\overline{\mathbf{M}}_{3\times3}} + \mathbf{b}, & \overline{\mathbf{M}}_{3\times3} > 0 \\ 0, & \text{otherwise} \end{cases},$$

where $\mathbf{W}$ and $\mathbf{b}$ denote the weight and bias of the partial convolution, respectively. Moreover, $\mathbf{F} = \left[\mathbf{F}_{diff}, \mathbf{F}_{dec}\right]$, whereas $*$ and $\circ$ represents convolution and entry-wise product. Lastly, $\overline{\mathbf{M}}_{3\times3}$ contains the average values of $\mathbf{M}$ in 3×3 neighborhood regions, which are calculated through a 3×3 average pooling operation (Li et al., 2020).

When solving the inpainting problem, the goal of $\mathbf{M}_{diff}$ is to be zero in areas with intense reflection and closer to one within areas with weak reflections. This will act as a guide of linearity to the decoder feature $\mathbf{F}_{dec}$. To make the mask behave this way, the mask loss function is defined as

$$\mathcal{L}_{\text{mask}}^{\text{diff}} = \sum_{i=1}^{4} \left\| \mathbf{M}_{\text{diff}}^{i}[R > \varphi] \right\|_1,$$

where i refers to the i-th layer in Figure 3.12, and $\| \cdot \|_1$ denotes the ℓ_1 norm. Moreover, $\mathbf{M}_{\text{diff}}^{i}[R > \varphi]$ represents the part of $\mathbf{M}_{\text{diff}}^{i}$ where the reflection layer R is greater than the threshold $\varphi = 0.3$.

As the careful reader might notice, this loss function is minimized when $\mathbf{M}_{\text{diff}} = \mathbf{0}$, leading to a trivial and unwanted solution. In weaker reflection areas, both $\mathbf{F}_{diff}$ and $\mathbf{F}_{dec}$ are reliable, and we want the partial convolution to take both of these into account. Therefore, in weaker reflection areas, M should be closer to 1. To account for this, an additional loss is constructed as

$$\mathcal{L}_{\text{mask}}^{\text{reg}} = \sum_{i=1}^{4} \left\| \mathbf{M}^{i}[R < \xi] - 1 \right\|_1,$$

where $\xi = 0.01$ is the threshold for areas with fewer reflections. Thus,

$$\mathcal{L}_{\text{mask}} = \mathcal{L}_{\text{mask}}^{\text{diff}} + \mathcal{L}_{\text{mask}}^{\text{reg}}$$

will be the total mask loss function (Li et al., 2020).

Overall Loss

The overall loss used in RAGNet is the weighted sum of all formerly mentioned losses. This is formulated as

$$L = \lambda_1 \mathcal{L}_{\text{rec}} + \lambda_2 \mathcal{L}_{percep} + \lambda_3 \mathcal{L}_{\text{excl}} + \lambda_4 \mathcal{L}_{adv} + \lambda_5 \mathcal{L}_{\text{mask}},$$

where $\lambda_1 = \lambda_2 = \lambda_5 = 1$, $\lambda_3 = 0.2$ and $\lambda_4 = 0.01$ (Li et al., 2020).

3.3 Training Details

To find the best network for dome reflection removal, we train each of the four presented networks using our final synthetic dataset. This dataset consists of 4000 images. The training

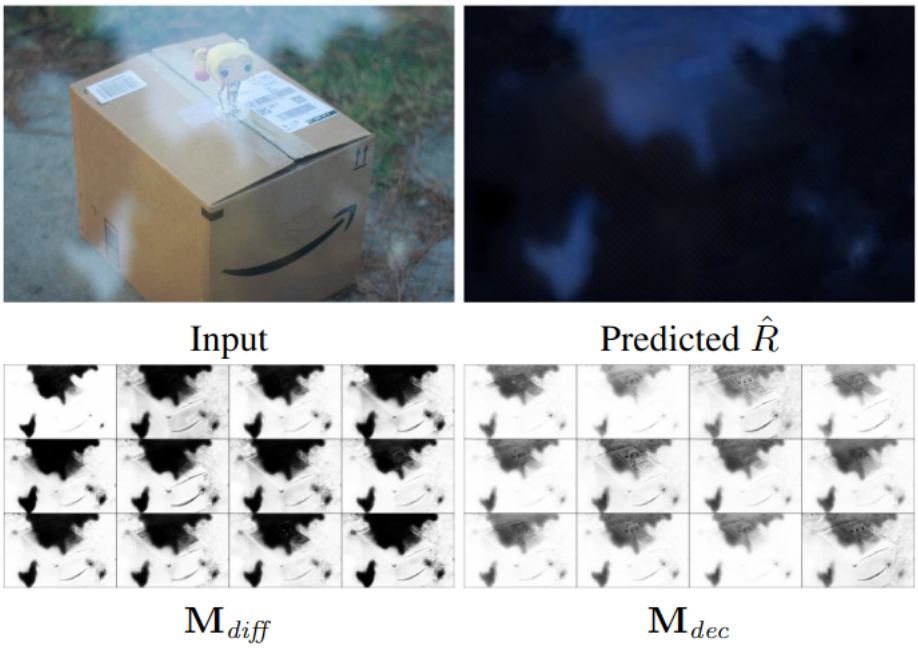


Figure 3.13: Shows an example of 12 randomly selected channels for $\mathbf{M}_{diff}$ and $\mathbf{M}_{dec}$. As seen, the values close to zero in $\mathbf{M}_{diff}$ indicates the areas of heavy reflection in the reflection layer. (Li et al., 2020).

is based on pre-trained models of the networks. In order to speed up training and better suit the structure of the networks, we down-sample our dataset to images of size 256×256 pixels.

During training, all network parameters are kept unchanged according to the developers' implementation. Instead, our main changes in implementation are related to pre-processing of data. In order to analyze the networks' performances equally, we remove all cropping, resizing and data augmentation inside the networks. Instead, we perform these operations restrictively before passing the images into the network.

Chapter 4
Evaluation

In this chapter, we first present the process which led to the decision of our final synthetic dataset. Secondly, we evaluate the performance of the networks on both synthetic and real-world data. Lastly, we present the best performing network.

4.1 Synthetic Dataset Evaluation

In previous single image reflection removal nets, the majority of works use synthetic data and similar code for generating this data (Amanlou et al., 2022). Our data generation originates from the synthesizing code written by Zhang et al. (2018). However, to better suit our specific dataset, we make some changes. See Chapter 2 for the specific details.

We create our dataset iteratively. To be able to evaluate the different versions of it, we fine-tune IBCLN using each version. By fine-tuning, we mean training using our own dataset, but starting from a pre-trained version of the network. Based on the results of fine-tuning, we modify the dataset to keep the background intact and remove the reflections. See below Section 4.1.1.

For fine-tuning, we use IBCLN mainly for its simplicity in setup and training process. Moreover, to speed up training and better suit the network structure, we use down-sampled images of size 256×256 pixels. After fine-tuning, we evaluate the performance qualitatively on real-world images and quantitatively on synthetic ones. The quantitative comparison is based on PSNR and SSIM.

4.1.1 Comparison of Synthetic Datasets

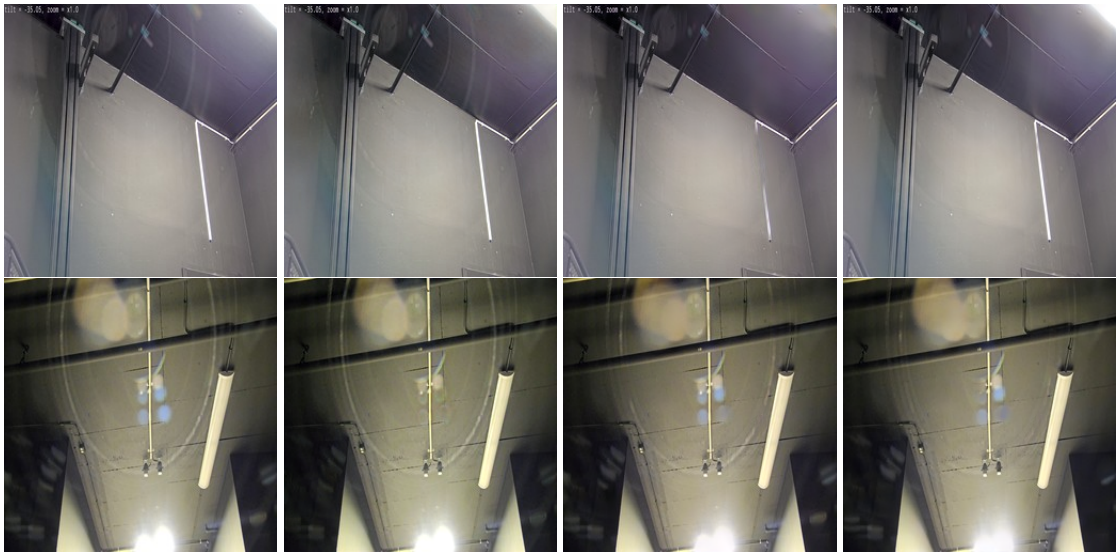
A first version of the synthetic dataset is referred to as the precursor. We create the precursor without too much restrictions neither on the properties of the reflection layer nor on augmentation.

After fine-tuning IBCLN using the precursor dataset, we test the network on some synthetic and real example images. Studying the filtered results, we observe two main things. Light sources, flares, and bright lines are often dimmed out of the images during filtering, as considered part of the reflection layer. Secondly, often the reflection from the lens is still intact or only partially removed.

Based on these observations, we improve our synthesizing process by adding reflection layer images containing less light sources. We also adjust the code to ensure the lens will be clearly visible in the images. Thereafter, we generate our final dataset.

Figure 4.1 shows a comparison between the performance of the pre-trained version of IBCLN, the precursor dataset, and the final dataset. We choose these examples to emphasize the differences in performance. To further illustrate the distinctions, Figure 4.2 shows the detected reflections of the same images.

Looking at Figure 4.1 and 4.2, the pre-trained version struggles to detect and remove the majority of the reflection. The precursor performs better. However, in the first image, the precursor partially removes the white cord in the background. Instead, in the second image, parts of the circular lens reflection remains intact. In contrary to the precursor, the final version leaves the white cord intact in the first image and removes more of the circular reflection in the second one.



Real-world image Pre-trained Precursor Final

Figure 4.1: Filtered results of real reflection images by IBCLN, fine-tuned on different synthetic datasets.

To further strengthen the assumption that the final dataset is the best, we make a quantitative evaluation based on the metrics PSNR and SSIM. Table 4.1 shows the quantitative comparison of the performance of IBCLN fine-tuned using the final dataset, the precursor, and the original pre-trained network. Here, the test set is synthetically generated irrespective of the compared datasets.

For both metrics, a higher value implies a more similar estimated transmission image. However, for SSIM the highest possible value is 1 whereas the PSNR is unbounded. To calculate SSIM, we use the default settings of the function *structural_similarity* from *skimage.metrics* in Python. For PSNR we use the function *peak_signal_noise_ratio* from *skimage.metrics*.

real-world image | Pre-trained | Precursor | Final

Figure 4.2: Detected reflection on real reflection images by IBCLN, fine-tuned on different synthetic datasets. The detected reflection layer is marked pink, while green represents background.

Table 4.1: Quantitative comparison using mean, median and standard deviation of the evaluation metrics PSNR and SSIM. The evaluation was made on a synthetic test set consisting of 173 images. The best results are marked in bold.

	PSNR			SSIM		
Network	**Mean**	**Median**	**Std**	**Mean**	**Median**	**Std**
Pre-trained	25.801	26.167	3.945	0.945	0.960	0.0485
Precursor	**29.259**	29.354	**3.731**	**0.966**	0.974	**0.0327**
Final	29.227	**29.757**	3.948	**0.966**	**0.976**	0.0333

Studying Table 4.1, the results are ambiguous. For example, the mean value of PSNR implies that the precursor is the best dataset. Additionally, the mean of SSIM is the exact same value.

To further examine this result, we perform a t-test of the mean PSNR scores for the precursor and final dataset. Using a paired t-test, the p-value is

$$p = 0.8189 > 0.01,$$

which implies the difference is not significant. Thus, we cannot conclude that there is a performance difference between the precursor and final dataset.

However, observing Figures 4.1 and 4.2, the filtered results of real-world images are clearly better using the final dataset. Therefore, despite the ambiguous results in Table 4.1, we choose this dataset to be our final version.

4.2 Network Evaluation

After we decide the final dataset, the next step is to determine the best performing network from Section 3.2. We first fine-tune all four pre-trained nets using our final dataset. After this, we evaluate each network performance on both synthetic and real-world data.

Our final synthetic test set contains 445 randomly chosen images. Here, there exists a target for the filtered result – the transmission image. Thus, the synthetic evaluation is based on PSNR and SSIM.

Apart from this, we gather 38 real-world images containing dome reflection. For these images, no ground truth exists, which is why we use a human evaluation process.

4.2.1 Synthetic Image Evaluation

The network evaluation using synthetic test data is based on the metrics PSNR and SSIM. Filtering the 445 blended images from our final synthetic test set, we can compare each image to the original transmission image. Doing this for the four networks, we summarize the result using mean, median, and standard deviation in Table 4.2.

Table 4.2: Quantitative comparison using mean, median, and standard deviation of the evaluation metrics PSNR and SSIM. The evaluation is made on our final synthetic test set consisting of 445 images. The best results are marked in bold.

	PSNR			SSIM		
Network	**Mean**	**Median**	**Std**	**Mean**	**Median**	**Std**
DADNet	30.749	30.596	3.314	0.968	0.974	0.0273
ERRNet	**32.926**	**32.916**	3.346	**0.978**	0.982	**0.0225**
IBCLN	31.640	31.478	**3.235**	0.976	**0.983**	0.0256
RAGNet	31.711	31.813	3.403	0.972	0.979	0.0295

In Table 4.2, ERRNet has the highest mean and median for PSNR and the highest mean for SSIM. Regarding median of SSIM, IBCLN performs best. Moreover, Figure 4.3 shows filtering results of all networks, on a few synthetic images. To better see the details of the images, the reader is advised to zoom in.

4.2.2 Real-World Image Evaluation

For real reflection images, no original transmission image is available. Therefore, we cannot use PSNR and SSIM to assess the reflection removal performance. Instead, we ask 7 people at Axis to grade 38 filtered real-world images. The participants are asked to primarily focus on the questions:

> *How reasonable does the filtered image look overall? Would you consider the quality better than the unfiltered image?*

when grading according to:

1: Large decrease in quality

2: Small decrease in quality

3: No difference in quality

4: Small increase in quality

5: Large increase in quality.

Due to a small grading scale, everyone is further asked to pinpoint the best performing network for each image. See Appendix A for additional information on the human evaluation instructions. Table 4.3 shows the results. It is clear that ERRNet got the highest score. In addition, a selection of filtered real-world images can be seen in Figure 4.4.

Table 4.3: Evaluation based on 38 real-world images containing reflection. For each network, the result of every image is graded by all participants according to a scale from 1-5, with 5 being the best. For each image, the participants also marked the best network.

Network	DADNet	ERRNet	IBCLN	RAGNet
Mean of Grades	3.261	**4.238**	3.681	3.921
Times Ranked Best	1	**189**	15	36

4.3 Further Evaluation – ERRNet

4.3.1 T-Test

In the synthetic evaluation, ERRNet has the highest mean for both PSNR and SSIM, see Table 4.2. The differences are although not large. Thus, there might not be a statistically significant difference between the values.

To further examine this, we perform paired t-tests comparing the mean value distributions of ERRNet against the other networks. Tables 4.4 and 4.5 outline the PSNR and SSIM results, respectively. In the tables, μ_{diff} is calculated as

$$\mu_{diff_i} = \text{mean}_{ERRNet} - \text{mean}_{Network_i},$$

while μ_{lower} and μ_{upper} represent the lower and upper quantile.

Table 4.4: T-tests for PSNR. The mean values of PSNR from Table 4.2 have been compared between ERRNet and the other networks.

Network	p-value	t-value	μ_{diff}	μ_{lower}	μ_{upper}
DADNet	<0.0001	21.610	2.177	1.978	2.374
IBCLN	<0.0001	14.095	1.286	1.106	1.465
RAGNet	<0.0001	10.945	1.215	0.997	1.433

The t-tests conclude that there is a statistically significant difference in the mean values between ERRNet and the other networks. This holds for both PSNR and SSIM. Thus, in regards to synthetic data, the dome reflection removal performance of ERRNet is significantly better compared to the other networks.

Table 4.5: T-tests for SSIM. The mean values of SSIM from Table 4.2 have been compared between ERRNet and the other networks.

Network	p-value	t-value	μ_{diff}	μ_{lower}	μ_{upper}
DADNet	<0.0001	21.553	0.0106	0.00962	0.0115
IBCLN	<0.0001	4.174	0.00222	0.00117	0.00326
RAGNet	<0.0001	8.111	0.00580	0.00439	0.00720

4.3.2 Comparison with Pre-Trained Version

To conclude that fine-tuning ERRNet actually has an impact, we compare our fine-tuned network to the pre-trained one. The results on synthetic data are shown in Table 4.6. To further illustrate the differences, Figure 4.5 contains filtering results of real-world images.

Table 4.6: Quantitative comparison using mean, median and standard deviation of the evaluation metrics PSNR and SSIM. The evaluation was made on our final synthetic test set consisting of 445 images. The best results are marked in bold.

	PSNR			SSIM		
Network	**Mean**	**Median**	**Std**	**Mean**	**Median**	**Std**
Pre-trained	25.573	25.518	3.337	0.954	0.967	0.0440
Fine-tuned	**32.926**	**32.916**	**3.346**	**0.978**	**0.982**	**0.0225**

When studying Table 4.6, it is clear that fine-tuning highly increases the results. Regarding the real example images in Figure 4.5, the filtering results looks considerably worse for the pre-trained model compared to the one fine-tuned with our dataset. Our fine-tuning of ERRNet clearly has a positive impact on the results.

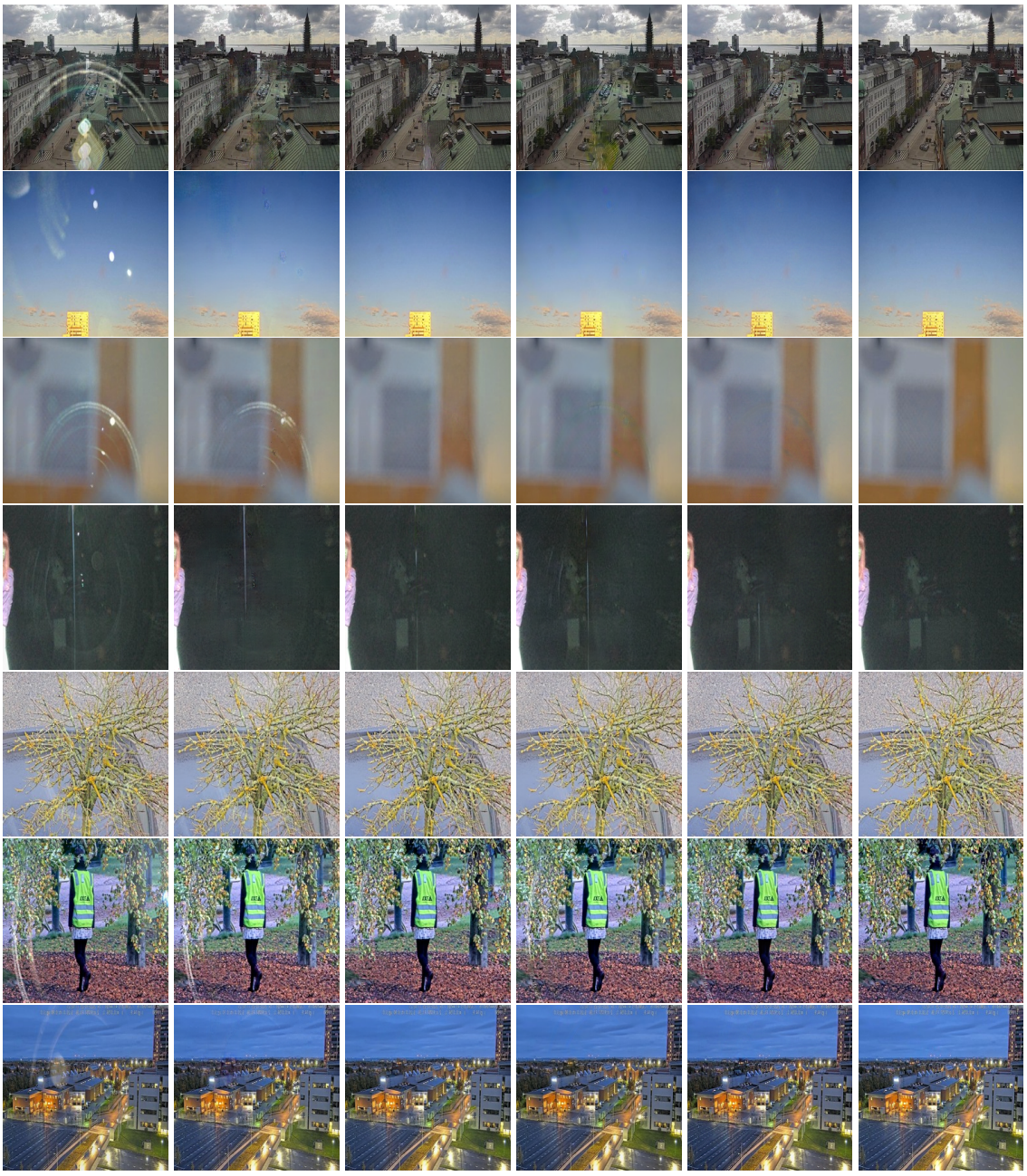


Figure 4.3: Visual comparison of seven images from our final synthetic test dataset. The figure displays input images, transmission images, and filtered results from the four networks.

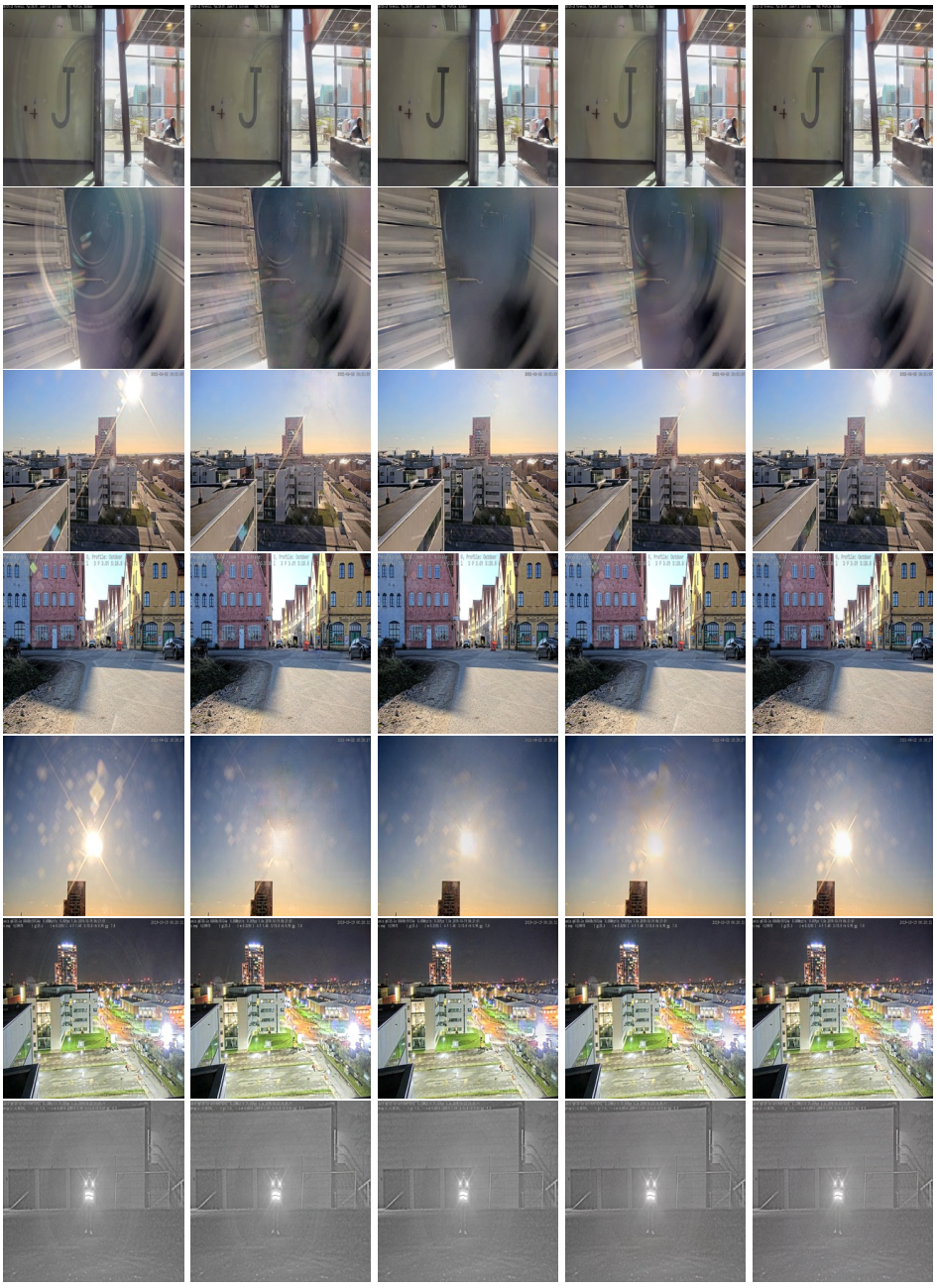


Figure 4.4: Visual comparison of seven images from our real-world test set. The figure displays input images and filtered results from the four networks.

Real-world image | Pre-trained ERRNet | Fine-tuned ERRNet

Figure 4.5: Filtered results of real-world reflection images. The figure displays input images and filter results for the pre-trained and fine-tuned version of ERRNet, respectively.

Chapter 5
Discussion

In this chapter, we discuss our results and present proposals for further work. Thereafter, we present our conclusions.

5.1 Best Network

In almost all aspects of our synthetic reflection removal evaluation, ERRNet performed the best. This includes the highest mean and median for PSNR, and the highest mean for SSIM during the synthetic evaluation. Although, IBCLN obtained the highest median of SSIM.

Since the difference between the mean values were not that large, we performed a t-test. The t-test concluded that the mean value difference of PSNR and SSIM is statistically significant between ERRNet and the other networks, in all cases. This means we can conclude that ERRNet performs significantly better than the other networks on our synthetic test data.

Even if the results from the synthetic evaluation points in favor of ERRNet, it is also important that the network can perform well on real-world images. In the evaluation of real-world images, ERRNet got the highest mean value and was most frequently ranked best. Thus, ERRNet was considered to be the best network.

5.1.1 ERRNet vs. RAGNet

One general observation is that ERRNet often removes more reflection than the other networks. However, sometimes ERRNet removes too much of the background. This is the opposite problem compared to RAGNet, which we noticed has a tendency to not filter out a sufficient amount of the reflection. Nevertheless, RAGNet often retains the background properties better than ERRNet, which seems more prone to blurring the images. For instance, compared to ERRNet, RAGNet rarely removes light sources in the images.

Considering these observations, RAGNet could be more interesting to investigate in some applications. Especially under circumstances where it is important that the background part remains untouched, for example in surveillance. Although according to our evaluation, ERRNet is clearly better.

Perhaps the best network would be a combination of ERRNet and RAGNet. Generally, RAGNet does not remove parts that should be kept in the image. This might be due to the reflection aware guidance used in RAGNet – it has a better notion of where the reflection is and its properties in certain areas of the image. On the other hand, when ERRNet finds the reflection, it is efficiently removed. Thus, combining these attributes of the two networks might lead to a better result.

5.2 Real-World Image Evaluation

The human evaluation on real-world images was consistent. ERRNet was by far ranked best most frequently.

However, when talking to the participants, they sometimes thought it was hard to distinguish between the filtering results. On the other hand, many participants thought that ERRNet was clearly better in some cases. Consequently, the participants might still have answered ERRNet even in the cases where no distinguishable difference could be observed between the networks. When observing Table 4.3, RAGNet is not far behind ERRNet in terms of mean grade. Nevertheless, ERRNet is ranked best 189 times and RAGNet 36 times. Therefore, the over-representation of ERRNet classified as the best network could be an exaggerated result.

Another thing to factor in is that the participants was able to talk to each other about the results. As a consequence, the participants could become biased towards a network which others had perceived being best. For us, this was hard to regulate as the participants were all part of our team and had daily contact. However, we did encourage them to not discuss the results before they handed in the evaluation.

Lastly, the number of images graded in the human evaluation was only 38. In regards to making a statistical conclusion, this would be considered quite few samples. Initially, the aim was to make the participants grade 100 images, but after receiving feedback, this was considered too many to grade within a reasonable amount of time. Therefore, similar images were deleted from the real evaluation test set, ending up at a count of 38 images.

Even though the image count was greatly decreased, not all team members performed the grading. Furthermore, some participants performed the grading but did not finish it for all images. This leads to the conclusion that 38 images were still considered many, in terms of time consumption. One reason why this task was perceived as quite time consuming could be the fact that the network results in many cases were similar. Moreover, the images were small, making it even harder to see the differences.

Regardless of these facts, our results are consistent, for both synthetic and real-world images. It is clear that ERRNet is the best network, giving an improved image quality for both synthetic and real-world data.

5.3 Further Work

Reflection removal is considered a hard problem. Our results on real-world data show a clear increase in image quality. This statement is certified by the mean values from Table 4.3 and by studying the images in Figure 4.4. Thus, our results show that it can be interesting to further investigate if a reflection removal network can be implemented into real dome cameras. However, the results are not perfect, leaving room for improvement.

5.3.1 Synthetic vs. Real-World Data Results

In regards to how well some of the networks performed on synthetic data (examples in Figure 4.3), the networks seem to have converged during training. Most of the networks have great reflection removal abilities on synthetic data. This means that further training on this dataset would not increase the model performance significantly. On the other hand, the results after filtering real-world images are not as convincing. These two facts lead to the conclusion that our dataset might not properly represent real-world data.

In our case, the results showed that our fine-tuned networks have a tendency to filter out light sources, which is not desirable. This behavior could originate from the fact that our reflection layers contained too much light leakage, despite trying to minimize this effect. However, the reason why a dome reflection appears in the first place is a light source in or near the image. Therefore, the presence of a light source is inevitable in the creation of synthetic reflection layers. A result of this is that our networks sometimes include the light leakage as part of the reflection layer.

Another factor that might be contributing to this is the properties of our background images. If a light source is present, it often comes with a reflection. This leads to an under-representation of background images containing light sources. On the other hand, as mentioned, the reflection layers are over-representative in terms of light leakage. This likely contributes to our lacking results on real-world images containing both light source and reflection. In conclusion, this way of creating a synthetic dataset might not be the best to obtain optimal results on real-world images.

5.3.2 Unaligned Image Pairs

One possible improvement could be to complement the dataset with real-world images, taken with and without dome. If this was done, the training data could better represent real-world data. However, removing the reflective surface would lead to more differences in the image than only the reflection. For instance, this could lead to unaligned image pairs. In ERRNet, this would however not be a problem since this network can handle both aligned and unaligned data. The generation of such a complementary dataset would be time consuming, but future work could explore this.

5.3.3 Generative Adversarial Networks

Another idea for further work is to investigate the possibilities of generating a dataset through generative adversarial networks. That is, instead of creating a synthetic reflection image by

blending two layers together, one could train a network to input a background image and add a reflection. During training, the input would be unpaired real-world images with and without reflection, such that the network could learn the underlying structure of these two classes. If successful, one could thereafter input a background image and let the network generate a reflection in this image. Most likely, this reflection image would be more realistic than a blended image, leading to better results on real-world data. Moreover, this could greatly improve the ability to generate bigger and more versatile datasets. This is due to only background images being needed for the generation of new reflection images.

5.4 Summary of Discussion

Even though improvements can be made, our results show that dome reflection removal using neural networks is a very promising approach. The image quality on both synthetic and real-world data is greatly improved. Moreover, the networks seem to understand the main appearance of real dome reflections, despite training only on synthetic data. However, in some cases the networks have problem recognizing the reflection. Nevertheless, the results are quite consistent with regards to which cases the networks struggle with. Therefore, in further work, one suggestion is to complement the training dataset based on these findings.

5.5 Conclusions

In this thesis, we investigated the possibility to remove dome reflections using neural networks. We created a dataset consisting of blended transmission and reflection layers. Using our dataset, we fine-tuned four networks built for reflection removal. Our thesis shows that this is a promising method. According to both our synthetic and real evaluation, the best network was ERRNet.

To improve the results, we propose to focus on generating a more realistic training dataset, as the results are not as good on real-world data. Since ERRNet is able to handle unaligned images, one option is to take images with and without dome, and complement the synthetic dataset. Another option would be to investigate if a generative adversarial network would be able to exploit the underlying structure of reflection and reflection-free images.

To the best of our knowledge, no dome reflection removal networks have been previously proposed. Even though there is room for improvement, our results show a great improvement in image quality for both synthetic and real-world data. Our fine-tuning of ERRNet is without a doubt better than just the pre-trained version. Thus, dome reflection removal using neural networks is clearly a promising method.

References

Agrawal, A., Raskar, R., and Chellappa, R. (2006). Edge suppression by gradient field transformation using cross-projection tensors. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2301–2308.

Agrawal, A. K., Raskar, R., Nayar, S. K., and Li, Y. (2005). Removing photography artifacts using gradient projection and flash-exposure sampling. In *SIGGRAPH 2005*.

Amanlou, A., Suratgar, A. A., Tavoosi, J., Mohammadzadeh, A., and Mosavi, A. (2022). Single-image reflection removal using deep learning: A systematic review. *IEEE Access*, 10:29937–29953.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR.

Badrinarayanan, V., Handa, A., and Cipolla, R. (2015). Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. *CoRR*, abs/1505.07293.

Be'ery, E. and Yeredor, A. (2008). Blind separation of superimposed shifted images using parameterized joint diagonalization. *IEEE Transactions on Image Processing*, 17(3):340–353.

Chervyakov, N., Lyakhov, P., and Nagornov, N. (2020). Analysis of the quantization noise in discrete wavelet transform filters for 3d medical imaging. *Applied Sciences*, 10(4):1223.

Chollet, F. (2017). *Deep Learning with Python*. Manning.

Diamant, Y. and Schechner, Y. Y. (2008). Overcoming visual reverberations. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

Fan, Q., Yang, J., Hua, G., Chen, B., and Wipf, D. (2017a). A generic deep architecture for single image reflection removal and image smoothing.

Fan, Q., Yang, J., Hua, G., Chen, B., and Wipf, D. P. (2017b). A generic deep architecture for single image reflection removal and image smoothing. *CoRR*, abs/1708.03474.

Gai, K., Shi, Z., and Zhang, C. (2012). Blind separation of superimposed moving images using image statistics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(1):19–32.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks.

Guo, X., Cao, X., and Ma, Y. (2014). Robust separation of reflection from multiple images. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2195–2202.

He, K., Zhang, X., Ren, S., and Sun, J. (2014). Spatial pyramid pooling in deep convolutional networks for visual recognition. In *Computer Vision – ECCV 2014*, pages 346–361. Springer International Publishing.

Hu, J., Shen, L., Albanie, S., Sun, G., and Wu, E. (2017). Squeeze-and-excitation networks.

Kong, N., Tai, Y.-W., and Shin, J. S. (2014). A physically-based approach to reflection separation: From physical modeling to constrained optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(2):209–221.

Levin, A. and Weiss, Y. (2007). User assisted separation of reflections from a single image using a sparsity prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(9):1647–1654.

Levin, A., Zomet, A., and Weiss, Y. (2004). Separating reflections from a single image using local features. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I.

Li, C., Yang, Y., He, K., Lin, S., and Hopcroft, J. E. (2019). Single image reflection removal through cascaded refinement.

Li, Y., Liu, M., Yi, Y., Li, Q., Ren, D., and Zuo, W. (2020). Two-stage single image reflection removal with reflection-aware guidance.

Liu, G., Reda, F. A., Shih, K. J., Wang, T.-C., Tao, A., and Catanzaro, B. (2018). Image inpainting for irregular holes using partial convolutions.

Mao, X., Shen, C., and Yang, Y. (2016). Image restoration using convolutional auto-encoders with symmetric skip connections. *CoRR*, abs/1606.08921.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation.

Schechner, Y., Kiryati, N., and Basri, R. (1998). Separation of transparent layers using focus. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pages 1061–1066.

Schechner, Y., Shamir, J., and Kiryati, N. (1999). Polarization-based decorrelation of transparent layers: The inclination angle of an invisible surface. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 814–819 vol.2.

Shih, Y., Krishnan, D., Durand, F., and Freeman, W. T. (2015). Reflection removal using ghosting cues. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3193–3201.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition.

Wan, R., Shi, B., Duan, L.-Y., Tan, A.-H., and Kot, A. C. (2017). Benchmarking single-image reflection removal algorithms. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3942–3950.

Wan, R., Shi, B., Hwee, T. A., and Kot, A. C. (2016). Depth of field guided reflection removal. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 21–25.

Wang, Z., Bovik, A., Sheikh, H., and Simoncelli, E. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612.

Wei, K., Yang, J., Fu, Y., Wipf, D. P., and Huang, H. (2019). Single image reflection removal exploiting misaligned training data and network enhancements. *CoRR*, abs/1904.00637.

Yan, Q., Xu, Y., and Yang, X. (2013). Separation of weak reflection from a single superimposed image using gradient profile sharpness. In *2013 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 937–940.

Ye, J. C. and Sung, W. K. (2019). Understanding geometry of encoder-decoder cnns. *CoRR*, abs/1901.07647.

Zhang, X., Ng, R., and Chen, Q. (2018). Single image reflection separation with perceptual losses.

Zheng, Y., Yang, C., and Merkulov, A. (2018). Breast cancer screening using convolutional neural network and follow-up digital mammography. In Mahalanobis, A., Ashok, A., Tian, L., and Petruccelli, J. C., editors, *Computational Imaging III*, volume 10669, pages 1 – 13. International Society for Optics and Photonics, SPIE.

Zou, Z., Lei, S., Shi, T., Shi, Z., and Ye, J. (2020). Deep adversarial decomposition: A unified framework for separating superimposed images. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12803–12813.

Appendices

Appendix A
Real Image Evaluation

Before the human evaluation on real images, we sent instructions according to Figure A.1 to the participants. The participants ranked and graded the results based on collages containing filtered images from the four networks. Figure A.2 displays an example of such a collage.

SINGLE IMAGE REFLECTION REMOVAL EVALUATION

The evaluation consists of a grading and a ranking.

Ranking: For each image, chose only the best network according to you, and note the network with corresponding name under the column “Best”.

Grading: In addition to this, for each network (columns noted with corresponding network name), grade the images according to the scale:
1: Large decrease in quality
2: Small decrease in quality
3: No difference in quality
4: Small increase in quality
5: Large increase in quality

Primary focus:
How reasonable does the filtered image look overall? Would you consider the quality better than the unfiltered image?

Other things to keep in mind while rating:
- How much of the reflection is filtered? The main objective is to filter out the characteristic rings reflected from the lens, other light defects (flares and so on) are not primarily considered.
- Are background features preserved in the filtered image? For instance, light sources, white lines and other details.
-Is the image blurred after filtering? How much does it affect the image quality?

Figure A.1: The instructions for human evaluation.

Figure A.2: An example of a collage from the human evaluation on real images. The left column contains the filtering results of the networks and the right column contains the original real reflection image.

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2022-06-03

EXAMENSARBETE Single Image Dome Reflection Removal Using Neural Networks
STUDENT Ellen Åström & Tove Thunborg
HANDLEDARE Pierre Nugues (LTH) & Victor Lantz (Axis)
EXAMINATOR Michael Doggett (LTH)

Maskininlärd extrahering av bakgrunder i reflektionsbilder tagna med domekameror

POPULÄRVETENSKAPLIG SAMMANFATTNING **Ellen Åström & Tove Thunborg**

På vissa övervakningskameror sitter ett skyddande hölje, även kallad dome, som i specifika ljusförhållanden skapar reflektioner i bilder. Då övervakning bidrar till ett tryggare samhälle är bildkvaliteten hos dessa kameror avgörande. Detta arbete presenterar ett neuralt nätverk som kan filtrera bort denna typ av reflektioner.

I dagens samhälle finns övervakningskameror i varenda byggnad och gathörn, vilket bidrar till ett tryggt samhälle. Det är därför önskvärt att bildkvaliteten hos dessa kameror ska vara så bra som möjligt. På vissa övervakningskameror sitter ett skyddande plasthölje, en så kallad dome. Problemet med dessa domer är att de har en tendens att skapa cirkulära reflektioner i bilder tagna under vissa ljusförhållanden. Den här typen av reflektioner uppstår när objektivet reflekteras i domen och tillbaka in i bilden. Våra resultat visar att domereflektioner effektivt kan filtreras bort, enligt bilden nedan.


Fundera på hur du själv uppfattar en reflektion i en bild. Visst är det inte så svårt för dig att mentalt separera reflektionen från bakgrunden? Hypotesen är att även datorer kan lära sig denna förmåga genom träning. Vid filtrering av reflektioner i bilder har neurala nätverk visat sig vara speciellt effektiva.

I skrivande stund finns flertalet neurala nätverk som skapats för att filtrera bort reflektioner. Dock har inget av dessa nätverk fokuserat på att filtrera bort just domereflektioner. Det finns därför anledning att tro att vi kan förbättra resultaten på denna typ av bilder. I vårt examensarbete har vi tränat och utvärderat fyra existerande neurala nätverk för att utröna om bildkvaliteten kan förbättras.

För att kunna träna närverken har vi behövt skapa ett eget dataset. Datasetet är syntetiskt genererat genom att slå ihop en bakgrunds- och reflektionsbild. Givet en ihopslagen bild som input, kan man sedan träna nätverken att återskapa tillhörande bakgrundsbild.

Efter att nätverken tränats utvärderade vi resultaten på både syntetiska och verkliga reflektionsbilder. De fyra nätverken presterade olika men ett av dem var onekligen bäst. Resultaten talar sitt tydliga språk: vårt bästa nätverk gav stor förbättring av bildkvaliteten.