

PAD-Net: A Perception-Aided Single Image Dehazing Network

Yu Liu¹ and Guanlong Zhao²

¹Department of Electrical and Computer Engineering, Texas A&M University

²Department of Computer Science and Engineering, Texas A&M University

{yliu129, gzhao}@tamu.edu

Abstract

In this work, we investigate the possibility of replacing the ℓ_2 loss with perceptually derived loss functions (SSIM, MS-SSIM, etc.) in training an end-to-end dehazing neural network. Objective experimental results suggest that by merely changing the loss function we can obtain significantly higher PSNR and SSIM scores on the SOTS set in the RESIDE dataset, compared with a state-of-the-art end-to-end dehazing neural network (AOD-Net) that uses the ℓ_2 loss. The best PSNR we obtained was 23.50 (4.2% relative improvement), and the best SSIM we obtained was 0.8747 (2.3% relative improvement.)

1. Introduction

Due to the existence of air pollution, dust, mist, and fumes, images taken in an outdoor environment will often contain complicated, non-linear and data-dependent noises, known as haze, which challenges many high-level computer vision tasks such as object detection and recognition. Taking autonomous-driving as an example, hazy or foggy weather will obscure the vision of on-board cameras and create a loss of contrast in the subject with light scattering through the haze particles, adding superior difficulties for self-driving tasks. Thus, dehazing is a highly desirable image restoration technique to enhance better results of computational photography and computer vision tasks.

Early approaches of dehazing often require additional information such as scene depth to be given or captured from comparing multiple different images of the same scene [1, 2, 3]. While these methods can effectively enhance the visibility of hazy images, their tractability is limited since the required additional information or multiple images are not always available in practice.

To address this problem, a single-image dehazing system, which aims at restoring the underlying clean image from a observed hazy image, is more feasible for real application and has received an increased interests in recent years. Traditional single-image dehazing methods exploit

natural image prior and perform statical analysis [4, 5, 6, 7]. More recently, dehazing algorithms based on neural networks [8, 9, 10] have shown state-of-the-art performance, among which the AOD-Net [10] has the ability to train an end-to-end system while outperforming the others on multiple evaluation metrics. AOD-Net minimizes the ℓ_2 norm of the difference between the haze and clean images. However, the ℓ_2 norm suffers from a handful of known limitations that may leave the dehazed image output of the AOD-Net away from the optimal quality, especially considering about its correlation with human perception of image quality[11]. On the one hand, ℓ_2 norm implicitly assumes a white Gaussian noise, which is an oversimplified case that is not valid in general dehazing cases. On the other hand, ℓ_2 treats the impact of noise independently to the local characteristics, such as structural information, luminance and contrast, of an image. However, according to [12], the sensitivity of the Human Visual System (HVS) to noise depends on the local properties and structure of a vision.

Alternatively, the structural similarity index (SSIM), is widely employed as a metric to evaluate image processing algorithms from a more perceptual point of view. Besides, it also possesses a differentiable property and can be used as a cost function. Therefore, in this work, inspired by [13], we propose to use loss functions that match with human perception (e.g., SSIM [12], MS-SSIM [14]) as training objectives of a dehazing neural network developed based on the AOD-Net [10]. We call this Perception-Aided Single Image Dehazing Network: *PAD-Net*. We hypothesize that even without changing the neural network architecture, the PAD-Net will lead to better dehazing performance than its baseline AOD-Net.

2. Related work

In this section, we briefly summarize the single-image dehazing methods that have been proposed in previous works and compare their advantages and deficiencies. Then, we proposed our perceptual guided end-to-end dehazing network that boosts the learning performance compared to the baseline AOD-Net.

The atmospheric scattering model has been widely used in previous haze removal work [15, 16, 17]:

$$I(x) = J(x)t(x) + A(1 - t(x)), \quad (1)$$

where x indexes pixels in the observed hazy image, $I(x)$ is observed hazing image, and $J(x)$ is the clean image to be recovered. The parameter A denotes the global atmospheric light, and $t(x)$ is the transmission matrix defines as:

$$t(x) = e^{-\beta d(x)}, \quad (2)$$

where β is the scattering coefficient of the atmosphere, and $d(x)$ represents the distance between the object and the camera.

The key to a successful haze removal algorithm is to recover the transmission matrix $t(x)$, on which the majority of the dehazing methods have focused through either physically grounded priors or data-driven approaches.

Conventional single image dehazing methods commonly exploit natural image priors and perform static analysis. For example, [4, 5] demonstrate that the dark channel prior (DCP) is informative to calculate the transmission matrix. [6] proposed a color attenuation prior and created a linear model for scene depth of the hazy image to allow for an efficient supervised parameter learning method, and [7] proposed a non-local prior based on the observation that pixels in a given cluster are often non-local and each color cluster in the clear image became a haze-line in RGB space.

More recently, CNNs have been applied to the haze removal application after demonstrated successes in many other computer vision tasks. [9] exploits a multi-scale CNN (MSCNN) that predicts a coarse-scale holistic transmission map of the entire image and refines it locally. [8] proposed the *DehazeNet*, a trainable transmission matrix estimator, and recovers the clean image combined with estimated global atmosphere light. Both these methods learn the transmission matrix from the CNN first and recover the haze-free image with separately calculated atmospheric light. Moreover, [18] proposed a complete end-to-end dehazing network names AOD-Net which takes the hazy image as input and directly generates clean image output.

In this project, we adopt the transformed atmospheric scattering model and the convolutional network architecture proposed in [10] and aim at improving its performance by utilizing perceptually motivated loss functions.

3. Proposed work

In this section, the proposed *PAD-Net* is explained. We first introduce the transformed atmospheric scattering model and the dehazing network architecture design based on it, which we adopt the work in [10] to facilitate an end-to-end single image dehazing. Then, we discuss the

perceptually-motivated loss functions that will be explored in our project.

3.1. End-to-end Dehazing Network Design

Based on the atmospheric scattering model (1), the clean image generated by our network can be formulated as:

$$J(x) = K(x)I(x) - K(x) + b, \text{ where} \quad (3)$$

$$K(x) = \frac{\frac{1}{t(x)}(I(x) - A) + (A - b)}{I(x) - 1},$$

where b is the constant bias whose default value is set to 1. Here, the core idea is to unify the two parameters in (1) $t(x)$ and A into one formula, i.e. $K(x)$, and directly minimize the reconstruction errors in the image pixel domain. Since $K(x)$ is dependent on the input $I(x)$, we in fact build an *input-adaptive* deep model, and train the model by minimizing the reconstruction errors between its output $J(x)$ and ground truth clean image.

Therefore, the proposed deep neural network is composed of two major parts: a K -estimation module to estimate $K(x)$ in (3) with five convolutional layers, and a clean image generation modules that follows to produce the recovery clean image via element-wise calculation. The entire network diagram of the PAD-Net is visualized in Fig. 1.

As depicted in Fig. 1, the five convolutional layers are implemented with different filter sizes to capture multi-scale features of the input hazy image and are concatenated with intermediates layers in order to compensate the information loss during convolutions, which is inspired by [8, 9]. Output images (i.e., $J(x)$ in (3)) from the network is then compared with the ground truth clean image at the loss layer to compute the error function for back propagation. This end-to-end dehazing network can be easily embedded with other deep models as a stage in high-level computer vision tasks such as object detection, classification as well.

One thing to mention is that, the PAD-Net, inherited from the AOD-Net [10], is a light-weighted network which has only three convolutional filters. In fact, if we analyze the atmospheric scattering model ((1) and (2)), we can find that there are only three unknown parameters, β , A and $d(x)$, in the model. And in our adopted benchmark, RE-SIDE dataset [18], the β and A are pair-wised selected constants, and the depth map $d(x)$ are either calculated from a depth dataset such as NYU2 [19] or estimated with convolutional neural network [20]. Therefore, the complexity of the hazy model is relatively low. Given this observation, in our work, we keep the setting of filter numbers in the AOD-Net to facilitate a quick training while obtaining good learning results.

3.2. Perceptual Loss Functions

At the loss layer, different error functions will be investigated to optimize the image dehazing results and the results

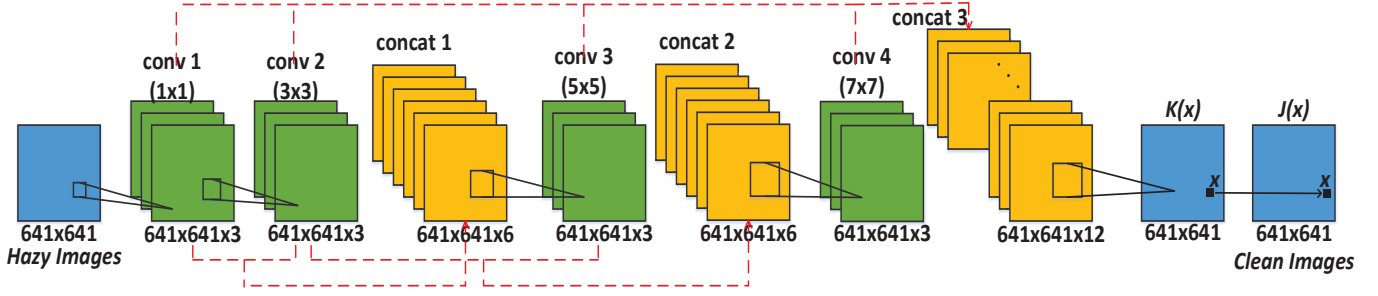


Figure 1. The network diagram of PAD-Net

will be compared. In the following sections we introduce the error metrics that will be examined in our project. We show their key features and how to compute their derivative for backpropagation steps. These loss functions will be implemented in the loss layer individually or jointly, specified in Section 4.

1. The ℓ_2 error

The ℓ_2 norm of the error is generally chosen as the loss function for image dehazing [10] given its simplicity and convexity. The ℓ_2 norm penalizes large error, but is more tolerant to small error, regardless of the underlying structure in the image. As a result, it sometimes produces visible splotchy artifacts on the restored images. The HVS, on the other hand, is more sensitive to luminance and color variations in texture-less regions [21]. The loss function for a patch P can be written as:

$$\mathcal{L}^{\ell_2}(P) = \frac{1}{N} \sum_{p \in P} (x(p) - y(p))^2, \quad (4)$$

where N is the number of pixels in the patch, p is the index of the pixel, and $x(p)$ and $y(p)$ are the pixel values of the generated image and the ground truth image respectively. Since $\partial \mathcal{L}^{\ell_2}(P) / \partial q = 0, \forall q \neq p$, for each pixel p in the patch, the derivate can be denoted as:

$$\partial \mathcal{L}^{\ell_2}(P) / \partial x(p) = x(p) - y(p). \quad (5)$$

Note that, even though $\mathcal{L}^{\ell_2}(P)$ is a function of the patch as a whole, the derivatives are back-propagated for each pixel in the patch.

2. The ℓ_1 error

The ℓ_1 error is studied as an attempt to reduce the artifacts introduced by the ℓ_2 and bring different convergence properties. Unlike the ℓ_2 norm, the ℓ_1 norm does not over-penalize large errors. The error function of ℓ_1 is:

$$\mathcal{L}^{\ell_1}(P) = \frac{1}{N} \sum_{p \in P} |x(p) - y(p)|. \quad (6)$$

The derivatives of the ℓ_1 is also simple. Similar to ℓ_2 norm, the derivatives of a certain pixel in a patch only depends on the difference between its own value and the ground truth value at the same location and do not rely on other pixels in the same patch.

$$\partial \mathcal{L}^{\ell_1}(P) / \partial x(p) = \text{sign}(x(p) - y(p)). \quad (7)$$

The derivative of $\mathcal{L}^{\ell_1}(P)$ is not defined at 0. However, if the error is 0, we do not need to update the weight. So here we use the convention that $\text{sign}(0) = 0$.

3. SSIM

Considering that image dehazing is a real-world application that reproduces visually clear and pleasing images, a perceptually motivated metric such as SSIM is worth studying. SSIM is a perception-based model that considers image degradation as perceived change in structural information, while also incorporating important perceptual phenomena, including both luminance masking and contrast masking terms. Inheriting the definition of $x(p)$ and $y(p)$ in (4), and let μ_x , σ_x^2 , and σ_{xy} be the mean of x , the variance of x , and the covariance of x and y , approximately, μ_x and σ_x can be viewed as estimates of the luminance and contrast of x , and μ_{xy} measures the structural similarity of x and y in terms of the tendency that they vary together. Then, the SSIM for pixel p is defined as:

$$\begin{aligned} SSIM(P) &= \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \cdot \frac{2\sigma_{xy} + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \\ &= l(p) \cdot cs(p). \end{aligned} \quad (8)$$

where the means and standard deviations are computed with a Gaussian filter G_{σ_G} with standard deviation σ_G . $l(p)$ and $cs(p)$ measure the comparisons of the luminance, and a combined contrast with structure similarity between x and y at the pixel p , respectively. The loss

function for SSIM can be then defined as:

$$\mathcal{L}^{SSIM}(P) = \frac{1}{N} \sum_{p \in P} 1 - SSIM(p). \quad (9)$$

Note that (8) indicates that the computation of $SSIM(p)$ requires looking at a neighboring of pixel p since it involves the mean and standard deviations of the Gaussian filter G_{σ_G} on the pixel. This means that $\mathcal{L}^{SSIM}(P)$, as well as its derivatives, cannot be calculated in some boundary region of P .

However, the convolutional nature of the network studies in this work allows us to write the loss as:

$$\mathcal{L}^{SSIM}(P) = 1 - SSIM(\tilde{p}). \quad (10)$$

where \tilde{p} is the center pixel of patch P . Again, this is because, even though the network learns the weights maximizing SSIM for the central pixel, the learned kernels are then applied to all the pixels in the image. Note that the error can still be back-propagated to all the pixels within the support of G_{σ_G} as they contribute to the computation of (10). More Formally, the derivatives of $\mathcal{L}^{SSIM}(P)$ at pixel p can be computed as:

$$\begin{aligned} \frac{\partial \mathcal{L}^{SSIM}(P)}{\partial x(q)} &= -\frac{\partial}{\partial x(q)} SSIM(\tilde{p}) \\ &= -\left(\frac{\partial l(\tilde{p})}{\partial x(q)} \cdot cs(\tilde{p} + l(\tilde{p})) \cdot \frac{\partial cs(\tilde{p})}{\partial x(q)} \right), \end{aligned} \quad (11)$$

where $l(\tilde{p})$ and $cs(\tilde{p})$ are the first and second term of SSIM (i.e. (8)) and their derivatives are:

$$\frac{\partial l(\tilde{p})}{\partial x(q)} = 2 \cdot G_{\sigma_G}(q - \tilde{p}) \cdot \left(\frac{\mu_y - \mu_x \cdot l(\tilde{p})}{\mu_x^2 + \mu_y^2 + C_1} \right), \quad (12)$$

and

$$\begin{aligned} \frac{\partial cs(\tilde{p})}{\partial x(q)} &= \frac{2}{\mu_x^2 + \mu_y^2 + C_2} \cdot G_{\sigma_G}(q - \tilde{p}) \cdot [(y(q) - \mu_y) \\ &\quad - cs(\tilde{p}) \cdot (x(1) - \mu_x)], \end{aligned} \quad (13)$$

where $G_{\sigma_G}(q - \tilde{p})$ is the Gaussian coefficient associated with pixel q .

4. MS-SSIM

The choice of σ_G would impact training performance of SSIM. Specifically, the network with smaller σ_G loses

the ability to preserve the local structure and reintroduce splotchy artifacts in flat regions, while the network with large σ_G tends to keep the noises in the proximity of edges. Our work, we adopt the idea of multi-scale SSIM from [13] in which M different values of σ_G will be used rather than directly fine tune its value to optimize the performance of SSIM. Given a dyadic pyramid of M levels, MS-SSIM is defined as:

$$\mathcal{L}^{MS-SSIM}(P) = l_M^\alpha(p) \cdot \prod_{j=1}^M cs_j^{\beta_j}(P). \quad (14)$$

where l_M and cs_j are the terms design in (8) at scale M and j , respectively. For convenience, we set $\alpha = \beta_j = 1$ for $j = 1, \dots, M$. Similarly to (10), we can approximate the loss for patch P with the loss computed at its center pixel \tilde{p} :

$$\mathcal{L}^{MS-SSIM}(P) = 1 - MS-SSIM(\tilde{p}). \quad (15)$$

The, the derivatives of the MS-SSIM loss function can be written as:

$$\begin{aligned} \frac{\partial \mathcal{L}^{MS-SSIM}(P)}{\partial x(q)} &= \left(\frac{\partial l_M(\tilde{p})}{\partial x(q)} + l_M(\tilde{p}) \cdot \sum_{i=0}^M \frac{1}{cs_i(\tilde{p})} \frac{\partial cs_i(\tilde{p})}{\partial x(q)} \right) \cdot \prod_{j=1}^M cs_j(\tilde{p}), \end{aligned} \quad (16)$$

To speed up the training, in stead of computing M levels of pyramid P , we adopt the approach proposed in [13] and uses M different values for σ_G , each one being half of the previous, on the full-resolution patch. In the loss function defined in this work, $\sigma_G^i = 0.5, 1, 2, 4, 8$.

4. Experimental Setup

4.1. Systems

In our experiment, the following error functions are applied to the loss layer of the proposed *PAD-Net*.

- **Baseline:** using ℓ_2 loss alone
- **L1:** using ℓ_1 loss alone
- **SSIM:** using SSIM loss alone
- **MS-SSIM:** using MS-SSIM loss alone
- **MS-SSIM+L2:** using a weighted sum of MS-SSIM and ℓ_2 as the loss function:

$$\mathcal{L}^{MSSSIM-L2} = \alpha \cdot \mathcal{L}^{MSSSIM} + (1 - \alpha) \cdot G_{\sigma_G^M} \cdot \mathcal{L}^{\ell_2}, \quad (17)$$

A point-wise multiplication between $G_{\sigma_G^M}$ and \mathcal{L}^{ℓ_2} is added for the ℓ_2 loss function term because MS-SSIM propagates the error at pixel q based on its contribution to MS-SSIM of the central pixel \tilde{q} , as determined by the Gaussian weights.

- **MS-SSIM+L1**: using a weighted sum of MS-SSIM and ℓ_1 as the loss function:

$$\mathcal{L}^{MSSSIM-L1} = \alpha \cdot \mathcal{L}^{MSSSIM} + (1 - \alpha) \cdot G_{\sigma_G^M} \cdot \mathcal{L}^{\ell_1}, \quad (18)$$

Similarly, the ℓ_1 loss is also weighted by the Gaussian filter $G_{\sigma_G^M}$.

4.2. Data and evaluation metrics

The benchmark dataset for this project is the RESIDE dataset [18], which has a large number of pictures for training the neural network and evaluating the dehazing performance. The ITS (indoor images) and OTS (outdoor images) sets from RESIDE provide a total number of over 400,000 images for training. Due to the limits of time and computational resources, we randomly sampled 10,000 images from ITS and OTS as the training data for this work. Among the 10,000 images, 2,790 are from IST and the rest 7,210 are from OTS, and they roughly takes the same percentage of number of images in the corresponding dataset, respectively. We also randomly sampled another 1,000 non-overlapping set of images as the validation data. All proposed systems were evaluated on the held-out SOTS subset, which contains 1,000 synthetic haze images (500 indoor and 500 outdoor images).

We used PSNR and SSIM as objective measurements of dehazing performance. Due to the limited scope of this project, we will not be able to run subjective evaluations on the dehazing results.

4.3. Implementation details

We built our neural network and loss functions using PyCaffe [22] because it has been proven to be flexible enough for research purposes while being able to support fast prototyping. The neural network architecture of AOD-Net was defined by following [10], and we referred to an open-source pre-trained model [23] published by the original authors to make sure that our implementation was as close to the original AOD-Net as possible. Unless otherwise noted, the base learning rate and mini-batch size of the systems were set to 0.01 and 8, respectively. The networks were initialized using Gaussian random variables. We used a momentum of 0.9 and a weight decay of 0.0001, following [10]. We also clipped the L2 norm of the gradient to be within [-0.1, 0.1] to stabilize the network training process, as suggested by [24]. All systems were trained on a Nvidia GTX 1070 GPU for around 14 epochs, which empirically ensures convergence.

Systems	PSNR		
	Indoor	Outdoor	All
AOD-Net	21.01	24.08	22.55
L2	20.73	25.58	23.15
L1	20.27	25.83	23.05
SSIM	19.64	26.65	23.15
MS-SSIM	19.54	26.87	23.20
MS-SSIM+L1	20.16	26.20	23.18
MS-SSIM+L2	20.45	26.38	23.41

Table 1. PSNR results without fine-tuning

Systems	SSIM		
	Indoor	Outdoor	All
AOD-Net	0.8372	0.8726	0.8549
L2	0.8235	0.9090	0.8663
L1	0.8045	0.9111	0.8578
SSIM	0.7940	0.8999	0.8469
MS-SSIM	0.8038	0.8989	0.8513
MS-SSIM+L1	0.8138	0.9184	0.8661
MS-SSIM+L2	0.8285	0.9177	0.8731

Table 2. SSIM results without fine-tuning

5. Results

5.1. Compare different loss functions

In the first set of experiments, we would like to investigate that which loss function provides the best objective dehazing performance. For **SSIM**, the standard deviation of the Gaussian filter was set to $\sigma_G = 5$. C_1 and C_2 in (8) were 0.01 and 0.03, respectively. For **MS-SSIM**, the Gaussian filters were constructed by setting $\sigma_G^i = \{0.5, 1, 2, 4, 8\}$. The **MS-SSIM+L1** loss function used $\alpha = 0.025$, and **MS-SSIM+L2** used $\alpha = 0.1$, following [13]. The results are summarized in Table 1 and Table 2. The AOD-Net’s results were copied from the RESIDE dataset paper [18], which was trained using the whole OTS and ITS sets. To provide a fair comparison, we also used the L2 norm loss function to train on our training set. Overall, the system **MS-SSIM+L2** achieved the best performance on both PSNR and SSIM, outperformed AOD-Net by 3.8% and 2.1%, respectively. Although AOD-Net performed well on indoor images, we found that when the number of training samples was the same, **MS-SSIM+L2** achieved a similar PSNR (20.45) compared with **L2** (20.73) and a higher SSIM (0.8285 v.s. 0.8235) in the indoor case. In terms of performance on outdoor images, the proposed loss functions generally worked better than the AOD-Net.

5.2. Fine-tuning MS-SSIM+L2

Based on the results above, we performed further fine-tuning on system **MS-SSIM+L2**, expecting to see bet-

α	PSNR		
	Indoor	Outdoor	All
0.1	20.68	26.18	23.43
0.3	20.47	26.49	23.48
0.5	20.46	26.39	23.43
0.7	20.32	26.67	23.50
0.9	20.50	26.34	23.42

Table 3. PSNR results with fine-tuning on MS-SSIM+L2

α	SSIM		
	Indoor	Outdoor	All
0.1	0.8229	0.9266	0.8747
0.3	0.8197	0.9248	0.8722
0.5	0.8116	0.9226	0.8671
0.7	0.8140	0.9211	0.8676
0.9	0.8165	0.9204	0.8685

Table 4. SSIM results with fine-tuning on MS-SSIM+L2

ter performance. We first analyzed the learning curve of the MISSIM-L2 system before fine-tuning, as illustrated in Fig. 2. It can be seen from the plot that the learning curve converged quickly and the learning errors fluctuate from iterations to iterations, which indicated we may need a smaller learning rate and a larger mini-batch size. Given that, we used weights from a pre-trained AOD-Net model [23] to initialize our network. During training, we applied a smaller learning rate (0.002) and a larger mini-batch size (16). We also tested on different α values to adjust the contribution of MS-SSIM and L2 to the fused loss function. All the other hyper-parameter were kept unchanged as in Section 4.3. The results are summarized in Tables 3 and 4. For PSNR, setting α to 0.7 yielded the best performance (23.50). For SSIM, the highest score (0.8747) was achieved when setting α to 0.1. We also plot the learning curve after fine tuning in Fig. 3 and it demonstrates that after fine tuning, the error starts at a small value at the beginning of the training due to the pre-train process. And the learning curve is now smoother and flatter as a result of the better choice of the learning rate and mini-batch size.

6. Discussion

From the results in Section 5 we can see that the proposed loss functions work well on outdoor images, but not as good on indoor testing samples. Part of the reason was that our training set contains limited number of outdoor samples, and the reported AOD-Net used around 40 times more indoor samples than our implementation for training. As we noted in Section 5.1, when we used the same training set to train on the L2 norm and the proposed loss functions, we achieved similar (PSNR) or better (SSIM) performance

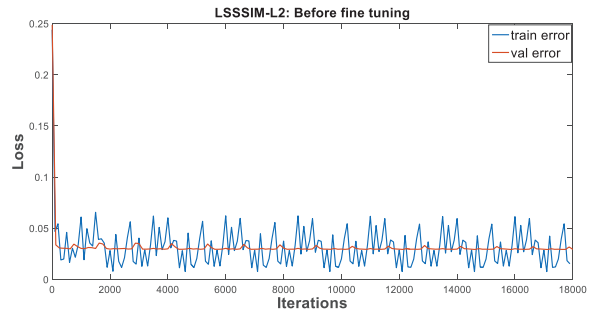


Figure 2. A sample learning curve of MS-SSIM+L2 before fine-tuning, the training loss curve was sampled every 10 iterations.

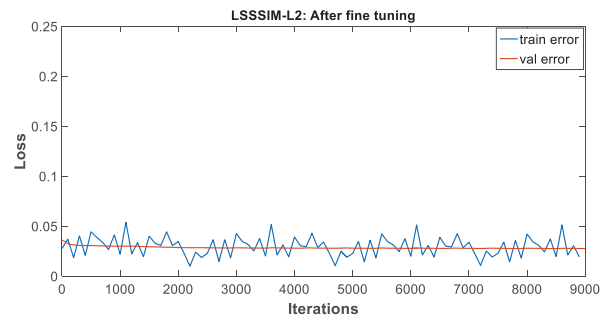


Figure 3. A sample learning curve of MS-SSIM+L2 after fine-tuning with $\alpha = 0.1$, the training loss curve was sampled every 10 iterations.

on indoor testing samples, not to mention that we have significant higher outdoor dehazing performances.

Surprisingly, we found that by directly optimizing the SSIM loss function, we did not obtain the optimal SSIM performance on the test set. After taking a closer look at our implementation, we found that we used a different standard deviation (1.5) for the Gaussian filter in the evaluation SSIM function. When we set the standard deviation as 5 and re-ran the SSIM evaluation, the SSIM score became 0.8597. Though this result is not directly comparable with the other systems, it is larger than when using 1.5 as the standard deviation. We did not use standard deviation 1.5 for training the dehazing model because that will be "cheating" – using a parameter from the evaluation system. Still, it would be interesting to see if setting standard deviation to 1.5 in training could produce optimal SSIM performance on the testing data.

The learning curve of the training process converged very fast, even after fine tuning, as depicted in Fig. 3. We suspected that this was due to the fact that the parameter set used for generating those synthetic images is small, therefore, the neural network might not need too many training iterations to figure out the value of those parameters.

The training costs, in terms of time, for some of the proposed cost functions (SSIM, MS-SSIM) were significant

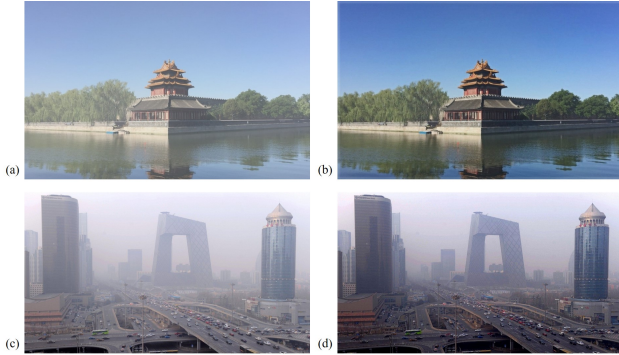


Figure 4. Dehazing examples. (a) A synthetic haze image. (b) Dehazed version of (a). (c) A real haze image. (d) Dehazed version of (c)

higher than using the ℓ_2 norm. For one thing, computing the gradient of those cost functions were of higher time complexity, since the derivate of the ℓ_2 error only involves the computation on the pixel itself while SSIM and MS-SSIM needs to do Gaussian filtering among the patch, as explained in Section 3.2. For another, the ℓ_2 norm implementation was carefully optimized to take full advantages of GPU computing, whereas our implementation was experimental and built on PyCaffe’s python layers, which was recognized for having flexibility for fast prototyping but slow in training. We believe that further implementation optimization can bridge the gap of training cost between the ℓ_2 norm loss and the other loss functions.

Fig. 4 shows some dehaze example using the model trained on the MS-SSIM+L2 loss function ($\alpha = 0.1$). The haze images are from the HTS set in RESIDE. We can see that even though the dehazing neural network was trained only on synthetic data, it can successfully remove haze from both synthetic and real haze images.

7. Conclusion

In this project, we propose to use perception-motivated loss functions to train an end-to-end dehazing neural network. Compared with a baseline system that has the same neural network architecture but uses the conventional ℓ_2 norm (MSE) loss function, we obtained significantly better objective dehazing performance on the SOTS set on the RESIDE dataset. The best PSNR we obtained was 23.50 (4.2% relative improvement), and the best SSIM we obtained was 0.8747 (2.3% relative improvement). For future work, first, we would like to train the proposed system using the full OTS and ITS sets. Second, we will investigate system MS-SSIM+L1 more closely – we did not have time to fine-tune the parameters of this system, but the initial results were still promising. Third, we are planning to conduct perceptual studies to ask human raters to judge the quality of the dehazed images.

Action items	Liu	Zhao
Project idea	60%	40%
System implementation	40%	60%
Experiment and data analysis	40%	60%
Report and presentation	60%	40%

Table 5. Project management

In terms of logistics, the workload division was fair among the two teammates, the details are summarized in Table 5. The project code is open-source, accessible on GitHub at <https://github.com/guanlongzhao/single-image-dehazing>.

References

- [1] K. Tan and J. P. Oakley, “Enhancement of color images in poor visibility conditions,” in *Image Processing, 2000. Proceedings. 2000 International Conference on*, vol. 2, pp. 788–791, IEEE, 2000.
- [2] Y. Y. Schechner, S. G. Narasimhan, and S. K. Nayar, “Instant dehazing of images using polarization,” in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, pp. I–I, IEEE, 2001.
- [3] J. Kopf, B. Neubert, B. Chen, M. Cohen, D. Cohen-Or, O. Deussen, M. Uyttendaele, and D. Lischinski, *Deep photo: Model-based photograph enhancement and viewing*, vol. 27. ACM, 2008.
- [4] K. He, J. Sun, and X. Tang, “Single image haze removal using dark channel prior,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 12, pp. 2341–2353, 2011.
- [5] K. Tang, J. Yang, and J. Wang, “Investigating haze-relevant features in a learning framework for image dehazing,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2995–3000, 2014.
- [6] Q. Zhu, J. Mai, and L. Shao, “A fast single image haze removal algorithm using color attenuation prior,” *IEEE Transactions on Image Processing*, vol. 24, no. 11, pp. 3522–3533, 2015.
- [7] D. Berman, S. Avidan, *et al.*, “Non-local image dehazing,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1674–1682, 2016.
- [8] B. Cai, X. Xu, K. Jia, C. Qing, and D. Tao, “Dehazenet: An end-to-end system for single image haze removal,” *IEEE Transactions on Image Processing*, vol. 25, no. 11, pp. 5187–5198, 2016.
- [9] W. Ren, S. Liu, H. Zhang, J. Pan, X. Cao, and M.-H. Yang, “Single image dehazing via multi-scale convolutional neural networks,” in *European conference on computer vision*, pp. 154–169, Springer, 2016.

- [10] B. Li, X. Peng, Z. Wang, J. Xu, and D. Feng, "Aod-net: All-in-one dehazing network," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4770–4778, 2017.
- [11] L. Zhang, L. Zhang, X. Mou, and D. Zhang, "A comprehensive evaluation of full reference image quality assessment algorithms," in *Image Processing (ICIP), 2012 19th IEEE International Conference on*, pp. 1477–1480, IEEE, 2012.
- [12] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.
- [13] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss functions for image restoration with neural networks," *IEEE Transactions on Computational Imaging*, vol. 3, no. 1, pp. 47–57, 2017.
- [14] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Seventh Asilomar Conference on*, vol. 2, pp. 1398–1402, Ieee, 2003.
- [15] E. J. McCartney, "Optics of the atmosphere: scattering by molecules and particles," *New York, John Wiley and Sons, Inc., 1976. 421 p.*, 1976.
- [16] S. G. Narasimhan and S. K. Nayar, "Chromatic framework for vision in bad weather," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 1, pp. 598–605, IEEE, 2000.
- [17] S. G. Narasimhan and S. K. Nayar, "Vision and the atmosphere," *International Journal of Computer Vision*, vol. 48, no. 3, pp. 233–254, 2002.
- [18] B. Li, W. Ren, D. Fu, D. Tao, D. Feng, W. Zeng, and Z. Wang, "Reside: A benchmark for single image dehazing," *arXiv preprint arXiv:1712.04143*, 2017.
- [19] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgb-d images," in *European Conference on Computer Vision*, pp. 746–760, Springer, 2012.
- [20] F. Liu, C. Shen, G. Lin, and I. Reid, "Learning depth from single monocular images using deep convolutional neural fields," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 10, pp. 2024–2039, 2016.
- [21] S. Winkler and S. Susstrunk, "Visibility of noise in natural images," in *Human Vision and Electronic Imaging IX*, vol. 5292, pp. 121–130, International Society for Optics and Photonics, 2004.
- [22] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 675–678, ACM, 2014.
- [23] B. Li, "Aod-net." <https://github.com/Boyiliee/AOD-Net>. Accessed: 2018-05-01.
- [24] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*, pp. 1310–1318, 2013.