Single Image Reflection Removal

Jin Huang
huang86@stanford.edu

June 11, 2019

1 Introduction

Reflection Removal is an unsolved problem. Users often capture a photo with reflection, for example, when they take scenery photos in a running car or capture outdoor scene inside a building. A solution is demanded to remove the reflection or at least make it not that obvious. But training data are very difficult to collect, usually mixed with large amount of synthetic data, which could be on a different distribution. I then take advantage of t-SNE to informally prove my concern and then focus more on data captured in real scenes.
I use an end-to-end neural network to solve the problem. The input are images with reflection. The output are the separated reflection layer and transmission layer. I show that using synthesized data helps but could produce unexpected results. I further discuss the optimal solution of the problem.

2 Prior Research

[2] produced CEILNet, which was the state-of-the-art and cited by many papers. [1] outperforms CEILNet by adding two perceptual losses : a feature loss from a visual perception network, and an adversarial loss that encodes characteristics of images in the transmission layers. [4] took advantage of dilated convolutions to do fast image processing on multiple topics.

3 Dataset

It's very difficult to gather large training set for the problem, as we usually don't have a ground truth when taking a photo with reflection. The dataset for the problem are in common synthesized or captured in controlled environment.
Thus I leverage with database provided in [1] with 5000 synthetic images and 110 real image pairs.
I also get the access to the benchmark dataset in [3], which includes real images of 40 scenes in a controlled lab environment by imaging pairs of daily objects and postcards, as well as scenes in natural outdoor environments with three

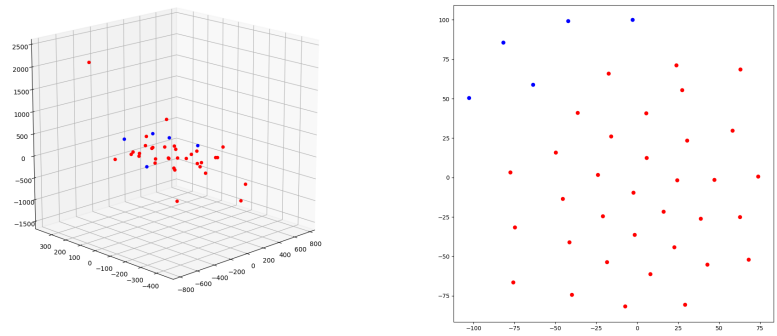


Figure 1: t-SNE plots on reflection layers of test images. Blue points are the reflection layers generated by baseline model on test images provided by [1]. Red points are the expected reflection layers generated by my first trained model on the new test set by [3].

different pieces of glasses. But I can only take one sample from a scene into the training set, as all the images in one scene are very similar. I get 198 real image pairs by combining the two dataset and removing the similar data. I put 180 pairs into training set and 18 into dev set. I use the 46 natural outdoor images without ground truth in the benchmark dataset as the test set. So I'm not able to get the perceptual metric results, like PSNR and SSIM, on the test set. The test set evaluation result is only based on user study.

4 Problem Analysis

I set up the trained model from [1] as the baseline. I find that the test images provided by [1] do not look like images captured from real scenes, more like synthetic ones, which are with thick and solid reflections. Also, the baseline model does not perform as well on my test set from benchmark dataset [3].
I think the two datasets are with very different distributions. But it's very hard to indicate different distributions on high dimensional data. So, I think about PCA. And I finally use a better method, t-Distributed Stochastic Neighbor Embedding, to plot the distribution in Figure 1. As PCA is a linear algorithm, it will not be able to interpret the complex polynomial relationship between features while t-SNE is made to capture exactly that.[5]
The input images have too many noises, which are not able to indicate the difference on the two test sets. Instead I plot with the reflection layers that are with simpler shapes and tightly correlated to the problem without noise. The

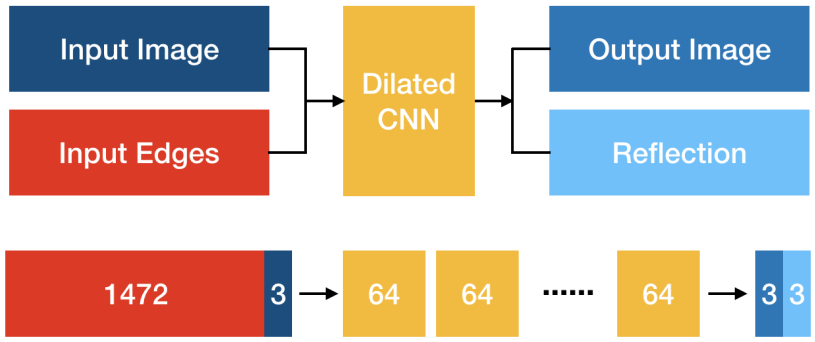


Figure 2: Model Structure and Dimension Transformation

3D plots still cannot clearly illustrate the difference. While, the 2D t-SNE plots informally prove my guess.
In my approach, to achieve better performance on real world images, I drop the synthetic training data and focus on only real data, which are very few. It may still work, as the VGG-19 pre-trained model contains low-level and high-level features from ImageNet dataset that contains large amounts of images.

5 Method

Given an input image $I \in [0,1]^{w \times h \times 3}$ with reflection, the approach is to decompose I to a transmission layer I_T and a reflection layer I_R using a network $f(I;\theta)$, where θ is the parameters to train.
I use a fully convolutional network. The general structure shown in Figure 2 is inherited from CEILNet solution. The input image first feeds VGG-19 pre-trained model to get low-level and high-level features, which are edge patterns. And the feature data are concatenated with the input image to get the final input data, which has 1475 channels in total.
The data dimension is then reduced to 64 by a 1 x 1 convolution and keeps the same till the output layer. The network uses dilated convolutions to aggregate the low-level and high-level image information instead of transforming data dimension. The dilation rate varies from 1 to 64. To compute information for both transmission layer and reflection layer, the network keeps two paths and finally outputs two images.
To train the model, I use a L1 loss on reflection layer, a feature loss that combines features from the pre-trained VGG-19 model, an adversarial loss from a GAN discriminator and an exclusion loss in the gradient domain to separate the reflection and ground truth, similar as [1].

$$\mathcal{L} = \alpha_1 \mathcal{L}_{L1} + \alpha_2 \mathcal{L}_{feature} + \alpha_3 \mathcal{L}_{adversarial} + \alpha_4 \mathcal{L}_{exclusion}$$

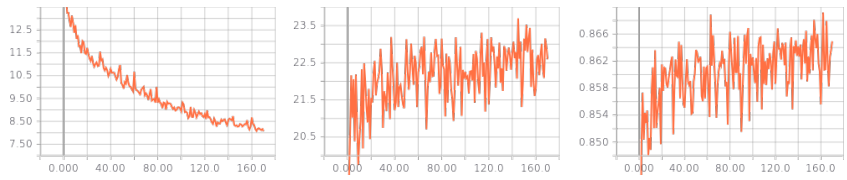


Figure 3: Train with parameters initialized from baseline model, using Wasserstein GAN with gradient penalty

6 Experiments

I set up the TensorFlow environment with my Nvidia RTX 2070 on a Windows PC and run the training with batch size 1 and learning rate 10^{-3}. The GPU has limited memory that only accepts about 512*512 input in training. A inter-cubic down-sampling is involved.

I first use a random crop method similar as 10-crop to preprocess the training dataset and train with only real images from scratch. It fails as the cropped area is comparable small that it does not contains reflection in many cases, which produces a blank or feature-less reflection layer. Basically, it fails to separate the reflection layer from the input images. It also proves that adding more training data with no reflection does not work. The problem needs labeled data with detailed label information.

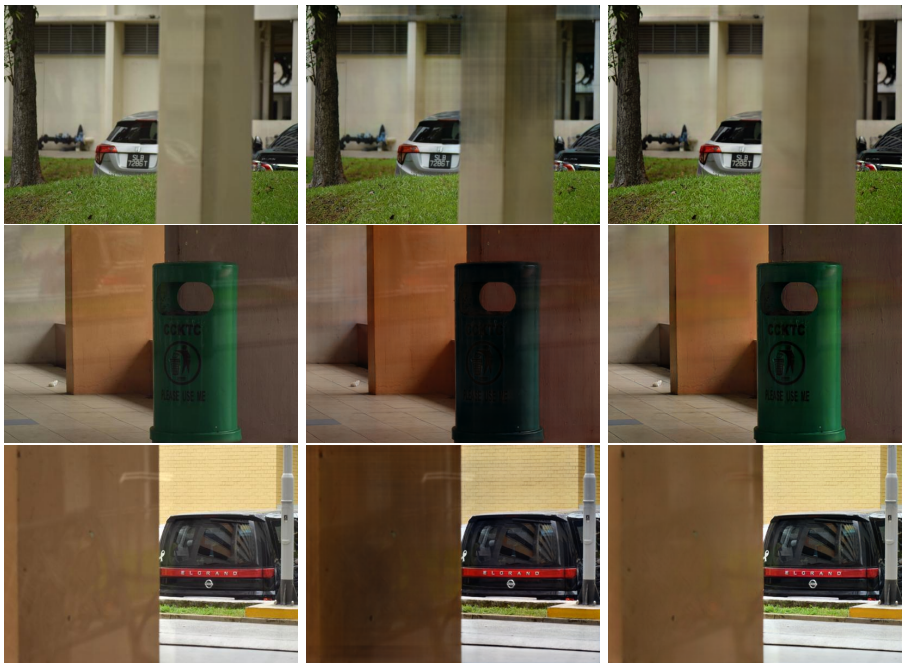
Then, I try to increase the crop area and random crop the input images. The training process looks all right. But the performance on dev set and test set is bad, which means it overfits the training set.

Next, I start to leverage with the baseline model, which equals that I leverage with synthetic training data. I train with real images starting with the parameters initialized by the baseline model, which outperforms the baseline model on test set, where I set $\alpha_1 = 1$, $\alpha_2 = 0.2$, $\alpha_3 = 0.01$, $\alpha_4 = 0.1$.

Finally, I use a different discriminator, Wasserstein GAN with gradient penalty. Together with the random crop preprocessing, it slightly improve the PSNR and SSIM on dev set and performance on test set. I set $\alpha_1 = 0.6$, $\alpha_2 = 0.2$, $\alpha_3 = 0.001$, $\alpha_4 = 0.3$. I increase the weight on exclusion loss to more aggressively separate the transmission and reflection and decrease the impact of L1 loss on reflection layer because it's impossible to estimate reflection precisely based on so few input images.

7 Results and Discussion

The baseline model has a hypothesis that the reflection is thick and always tries to extract more pixels from input images. It could produce good results on the case that the reflection is thick and solid indeed like the synthetic images they trained with. While, IMHO in real world it's more valuable to focus on or at least first solve the case that the reflection is blurry and thin. The photos in



(a) Input (b) Baseline Result (c) My Result

my cell phone that I want to fix are all like that.
However, it's hard to get even thousands of dependent real images with ground truth in pair. Synthetic images still help on providing diversity to prevent overfit. As to my approach, if I keep training with my small training set, it would finally overfit the training set. I do early-stop without waiting it to converge. So, improving the synthetic method to better simulate the real images is also a good way to improve the result.
Frankly my new model is not able to work on the photos in my cell phone, neither the baseline model. I find the ones I want to fix are all with even thinner and more blurry reflection only in a very small area. The models either do not recognize the reflection or try to extract larger area around the small reflection area.
To better solve the problem, the way to prepare and organize data is probably more critical than ML/DL technologies. Gathering a reflection image dataset that has similar size of ImageNet is too difficult. Thus, it's probably more realistic to solve the reflection removal problem focusing on certain scenarios based on training data on the specific distribution.

8 Contributions

The code to train the model and predict the transmission layer is at `https://github.com/goldhuang/ReflectionRemoval`. The models are uploaded to `https://www.dropbox.com/sh/ov76rmiy5rl2ypg/AAAP4S79v4no_LXNi7jEXZy_a?dl=0`

9 References

[1] Single Image Reflection Separation with Perceptual Losses
`https://arxiv.org/pdf/1806.05376.pdf`
`https://github.com/ceciliavision/perceptual-reflection-removal`

[2] A Generic Deep Architecture for Single Image Reflection Removal and Image Smoothing
`https://arxiv.org/pdf/1708.03474.pdf`

[3] Benchmarking single-image reflection removal algorithms.
`http://www.jdl.ac.cn/doc/2011/2018111200074838_wan_benchmarking_single-image_reflection_iccv_2017_paper.pdf`

[4] Fast Image Processing with Fully-Convolutional Networks.
`https://arxiv.org/pdf/1709.00643.pdf`

[5] Introduction to t-SNE.
`https://www.datacamp.com/community/tutorials/introduction-t-sne`

[6] Improved Training of Wasserstein GANs.
`https://arxiv.org/pdf/1704.00028.pdf`