Low-Light Light Field (LF) Restoration

Snehal Singh Tomar
Aqil K H
Subhankar Chakraborty

June 12, 2021

The Story So Far...

- A LF camera offers unique advantages such as post-capture refocusing & aperture control, but low-light conditions severely limit these capabilities
- We need to decode raw LFs captured using lenslet based plenoptic cameras
- This problem has been successfully addressed in [1], and its MATLAB code is publicly available
- Restoring LFs captured in low-light is not possible with single-frame low-light enhancement techniques designed for smartphones and DSLR cameras

[1] Dansereau, Donald G., Oscar Pizarro, and Stefan B. Williams. "Decoding, calibration and rectification for lenselet-based plenoptic cameras." Proceedings of the IEEE conference on computer vision and pattern recognition. 2013.

Stage I

Decoding LFs using Python

PlenoptiCam

- PlenoptiCam [2] is an open-source software for scientific light field computation
- Completely Python based and comes with a GUI
- It has the ability to calibrate an image taken by a plenoptic camera and extract sub-aperture images or synthetically focused photographs
- Our Implementation using plenopticam can be found here.

[2] Hahne, Christopher, and Amar Aggoun. "PlenoptiCam v1.0: A light-field imaging framework." arXiv preprint arXiv:2010.11687 (2020).

Plenopticam V/S MATLAB LF toolbox

IMG_1219.lfr from the L3F Dataset - Decoded and Post-Processed using *Plenopticam*[2]

IMG_1219.lfr from the L3F Dataset - Decoded using *MATLAB LF Toolbox*[1]

Plenopticam V/S MATLAB LF toolbox

IMG_1468.lfr from the L3F Dataset - Decoded and Post-Processed using *Plenopticam*[2]

IMG_1468.lfr from the L3F Dataset - Decoded using *MATLAB LF Toolbox*[1]

Plenopticam V/S MATLAB LF Toolbox

IMG_1544.lfr from the L3F Dataset - Decoded and Post-Processed using *Plenopticam*[2]

IMG_1544.lfr from the L3F Dataset - Decoded using *MATLAB LF Toolbox*[1]

Stage II

Rectifying Low-Light LFs, decoded using *Plenoticam* based Python Implementation

L3FNet: Introduction

- L3FNet [3] is a deep neural network for Low-Light Light Field (L3F) restoration
- It not only performs visual enhancement of each LF view but also preserves the epipolar geometry across views
- This is achieved by adopting a two-stage architecture
  - Stage-I looks at all the LF views to encode the LF geometry
  - This encoded information is then used in Stage-II to reconstruct each LF view
- Four LFs of different scenes, with different light settings varying from optimal to extreme low-light are captured using the Lytro Illum Camera to generate a dataset

[3] Lamba, Mohit, Kranthi Kumar Rachavarapu, and Kaushik Mitra. "Harnessing multi-view perspective of light fields for low-light imaging." IEEE Transactions on Image Processing 30 (2020): 1501-1513.

We use a modified version of the below architecture

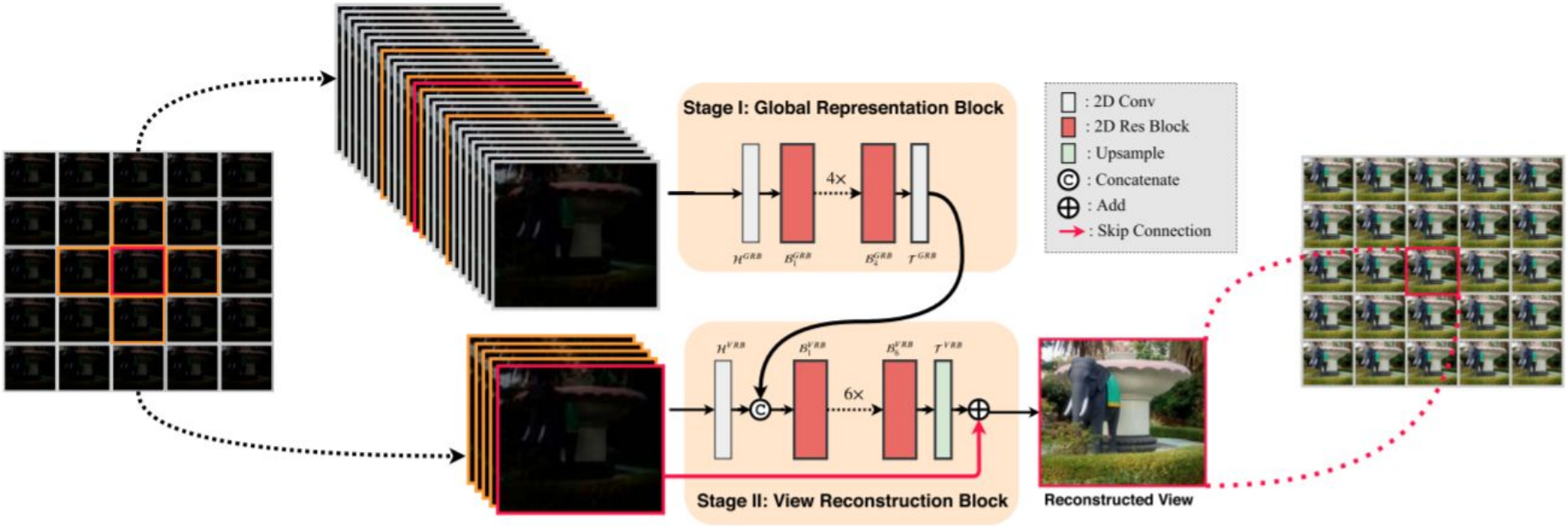

Fig. 3. L3Fnet architecture: The proposed architecture consists of a Global Representation Block (Stage-I) and a View Reconstruction block (Stage-II). Stage-I operates on the full low-light LF to obtain a latent representation that encodes the LF geometry. This latent representation is then used by Stage-II to restore each LF view.

Previous Results

- The training data contains 18 pairs of well lit and their corresponding L3Fs
- The test set contains 8 pairs of well lit and their corresponding L3Fs
- A view from the ground-truth image from each scene is compared to the corresponding view in the rectified L3F image
- The image on the left is the rectified image after training L3FNet for 8000 iterations, and the one on the right is the ground-truth image
- **Note: All images so far were the saved JPEGs using MATLAB**

Restored Image

Ground-Truth Image

Restored Image

Ground-Truth Image

Experiment Ideas

1. Perform rectification on L3F images with preprocessing without saving them as JPEG
2. Perform rectification on L3F images after removing all preprocessing without saving them as JPEG
3. Perform rectification on L3F images saved as JPEG after being decoded by MATLAB
4. Perform rectification on images which have not even been demosaiced
5. Try with different combinations of the loss functions

Implementation Details

- We are using a variant of L3FNet
- We restore only the central 3×3 views
- Combinations of losses used
  - L1 loss
  - L1 loss and gradient loss
- The model did not train well upon incorporating the perceptual loss
- Trained for 20,000 iterations on GTX 1080 or GTX 1080 Ti

Decoded L3F View

Decoded L3F View

Experiment 1

- Post-processing (auto white balance, color correction, contrast equalization) performed on both the low-light as well as the well-lit decoded LFs
- The decoded LFs are saved as NumPy arrays instead of JPEG images to preserve the complete data
- Loss used is the L1 loss
- Quantitative Results
  - PSNR: 17.34235191345215
  - SSIM: 0.4914553463459015

Restored Image

Ground-Truth Image

Restored Image

Ground-Truth Image

Experiment 2

- Post-processing (auto white balance, color correction, contrast equalization) performed only on the well-lit decoded LFs
- The decoded LFs are saved as NumPy arrays instead of JPEG images to preserve the complete data
- Loss used is the L1 loss
- Quantitative Results
  - PSNR: 17.882638931274414
  - SSIM: 0.521033525466919

Restored Image

Ground-Truth Image

Restored Image

Ground-Truth Image

Experiment 3

- Post-processing (auto white balance, color correction, contrast equalization) performed on both the low-light as well as the well-lit decoded LFs
- The decoded LFs are saved as JPEG images
- Loss used is the L1 loss
- Quantitative Results
  - PSNR: 17.960960388183594
  - SSIM: 0.5644765496253967

Restored Image

Ground-Truth Image

Restored Image

Ground-Truth Image

Decoded L3F View

Decoded L3F View

Experiment 4

- Post-processing (auto white balance, color correction, contrast equalization) performed only on the well-lit decoded LFs
- The decoded LFs are saved as NumPy arrays instead of JPEG images to preserve the complete data
- Loss used is the L1 loss along with the Perceptual loss
- Quantitative Results
  - PSNR: 5.234211444854736
  - SSIM: 0.12238209694623947

Experiment 5

- Post-processing (auto white balance, color correction, contrast equalization) performed on both the low-light as well as the well-lit decoded LFs
- The decoded LFs are saved as JPEG images
- Loss used is the L1 loss along with the Perceptual loss
- Quantitative Results
  - PSNR: 5.286728382110596
  - SSIM: 0.08754804730415344

Experiment 6

- Post-processing (auto white balance, color correction, contrast equalization) performed on both the low-light as well as the well-lit decoded LFs
- The decoded LFs are saved as NumPy arrays instead of JPEG images to preserve the complete data
- Loss used is the L1 loss along with the Gradient loss
- Quantitative Results
  - PSNR: 17.932992935180664
  - SSIM: 0.5531979203224182

Restored Image

Ground-Truth Image

Restored Image

Ground-Truth Image

Experiment 7

- Post-processing (auto white balance, color correction, contrast equalization) performed only on the well-lit decoded LFs
- The decoded LFs are saved as NumPy arrays instead of JPEG images to preserve the complete data
- Loss used is the L1 loss along with the Gradient loss
- Quantitative Results
  - PSNR: 8.891327857971191
  - SSIM: 0.35679861903190613

Restored Image

Ground-Truth Image

Restored Image

Ground-Truth Image

Experiment Metrics (Stage II)

Experiment	Post Processing Applied To..	Decoded Images saved as..	Loss function used during training	PSNR(dB)	SSIM
Expt. 1	Inputs, GTs	.npy files	L1	17.3424	0.4915
Expt. 2	GTs	.npy files	L1	17.883	0.521
Expt. 3	Inputs, GTs	.jpeg files	L1	**17.96**	**0.564**
Expt. 4	GTs	.npy files	L1 + Perceptual	5.234	0.122
Expt. 5	Inputs, GTs	.npy files	L1 + Gradient	5.287	0.088
Expt. 6	Inputs, GTs	.npy files	L1 + Gradient	17.933	0.553
Expt. 7	GTs	.npy files	L1 + Gradient	8.891	0.357

Observations and Inferences

- The network does not train upon incorporating Perceptual loss.
- Training when L3F decoded images are not post-processed gives better PSNR than those being post-processed
- Training on images saved as JPEG gives marginally better PSNR and SSIM than training on the entire data without any quantization
- Adding just the gradient loss does not seem to help much over using just the L1 loss. Rather, it does not train properly in one case.

Stage III

Rectifying LFs decoded using our Python Implementation without performing Demosaicing and other post processing

Demosaicing Removal

- Changed the source code of Plenopticam to decode LFs without demosaicing
- Demosaicing removed only for the L3Fs and saved as NumPy arrays
- The decoded array is split into 4 channels corresponding to the ‘rggb’ CFA
- The array after passing through the model is upsampled at the end using bilinear-interpolation
- Rest of the model architecture is the same as in the previous experiments
- Trained for 100,000 iterations instead of 20,000 as in the previous cases

Decoded L3F View

Decoded L3F View

Experiment 8

- Demosaicing removed from the L3Fs
- Post-processing (auto white balance, color correction, contrast equalization) performed on the well-lit decoded LFs
- The decoded LFs are saved as NumPy arrays instead of JPEG images to preserve the complete data
- Loss used is a combination of L1, Gradient, DFT and Perceptual loss
- Quantitative Results
    - PSNR: 12.430688858032227
    - SSIM: 0.4996431767940521

Restored Image

Ground-Truth Image

Restored Image

Ground-Truth Image

Experiment 9

- Demosaicing removed from the L3Fs
- Post-processing (auto white balance, color correction, contrast equalization) performed on the well-lit decoded LFs
- The decoded LFs are saved as NumPy arrays instead of JPEG images to preserve the complete data
- Loss used is a combination of L1 and Gradient loss
- Quantitative Results
  - PSNR: 7.4789557456970215
  - SSIM: 0.27252194285392760

Restored Image

Ground-Truth Image

Restored Image

Ground-Truth Image

Experiment Metrics (Stage III)

Experiment	Post Processing Applied To..	Decoded Images saved as..	Loss function used during training	PSNR(dB)	SSIM
Expt. 8	GTs	.npy files	L1+Gradient+DFT +Perceptual	12.43	0.499
Expt. 9	GTs	.npy files	L1+Gradient	7.479	0.273

Inferences

- The perceptual loss function seems to be a better objective function for the rectification of non-demosaiced and decoded L3F images as compared to rectification of demosaiced and decoded L3F images using the L3Fnet-variant. It is actually hard to infer why this is happening.
- The information lost due to JPEG compression does not seem to make a considerable difference. This may be attributed to significant features being learnt by the network irrespective of the slight input compression.

Inferences

- The halo effect observed in rectified non-demosaiced L3Fs may be due to random sampling of pixels taking place during training.
- At this stage, we can’t affirm whether non-demosaiced L3Fs can be better rectified by the L3Fnet-variant or not.
- The fact that Plenopticam decoded L3Fs are harder to train can be attributed to the fact that although they are brighter than their corresponding MATLAB decoded L3Fs, they contain a lot of noise and artefacts which might be throwing the model off.

Future Work

- Debug and make the training using perceptual loss work properly
- Try with different permutations of the input and ground truth, choosing between the MATLAB and Plenopticam decoded LFs to check where the training issue is coming from
- While training for decoded L3Fs without demosaicing, take care of the sampling pattern to maintain the correct CFA configuration
- Explore the prospect of eliminating the python code used for generating decoded images which are being given as training data to the L3Fnet-variant

Questions?