High Dynamic Range Imaging of Dynamic Scenes using Deep Learning

Uma Arunachalam
Carnegie Mellon University
uarunach@andrew.cmu.edu

Abstract

Merging low dynamic range (LDR) images to produce a high dynamic range (HDR) image usually involves conventional weighting schemes and operate under the assumption that the input images are perfectly aligned. This work focuses on images that are misaligned and uses learning based methods to model the weights for merging. I align the images using optical flow against a reference image and use a 4-layer convolution neural network to predict the HDR image. I also experiment with different end to end network approaches for merging while ensuring differentiability. Qualitative and quantitative results demonstrate the effective of this approach and how learning based methods are skilled at understanding artifacts introduced in the image due to misalignments and faster at inference.

1. Introduction

Standard cameras are limited in dynamic range, as a result of which many parts of the image are under/over exposed. HDR images are traditionally produced from a set of such LDR images captured at different exposures using weighting schemes that estimate the contribution of each input LDR image towards a pixel of the output. These methods usually assume static scenes with no camera motion and can produce different ghosting artifacts even with little camera motion during capture. These processes also require significant time and compute requirements for merging as well. Thus the main motivation behind this approach is to relax the requirements on capture process while producing a reasonable output of dynamic scenes, as well reduce processing time by using learning based techniques. Thus, when accounting for misalignment it gives room for performing merging using a hand held camera instead of using a tripod, a software tool for automatic capture. This can even be useful for self driving applications in which images are captured under different lighting conditions while the vehicle is in motion. Reducing processing time can also help such applications as well as mobile deployment of these methods. Deep learning based methods have been increasing used in the recent past for various applications ranging from colorization to complex photographic enhancements. The public availability of datasets has also accelerated research in this area, helping networks learn rather than just memorize to training data. Thus many approaches involve replacing the traditional methods used in the pipeline with a deep learning black box, that enhances performance due to implicit learning while training.

For this application, my intuition behind why modelling weighting schemes using neural networks helps in removing artifacts due to misalignment is that they reject pixels with motion and extrapolate for merging. When merging using conventional weighting schemes such as photon, tent, uniform, optical flow methods with inherent limitations will affect the merge process. Thus this process of understanding motion and how to predict its intensity for the resulting HDR image is complex, and modelling the deep learning (DL) black box to understand it can give interesting results when trained well. This work is inspired from [6], in which they prove the effectiveness of these deep learning methods.

This method takes 3 LDR images and uses a 2 stage approach to produce the HDR image. Stage 1 involves estimating how the first and the third image warp with the second image as baseline reference. Based on the estimated flow vectors, the aligned images are fed as input to the next stage. From this part of the pipeline, an end to end, fully differentiable network is used to either predict the HDR image itself or the weights required for blending of the input. The tone mapped version of the predicted output is further used in calculating the cost function for minimization and backpropagation.

The whole report is structured as follows: Section 2 describes related work for this application, followed by approach and experimental setup. Section 5 highlights the results obtained and Section 6 covers conclusion and future work. It is followed by Section 7 in which I discuss some experiments I tried that did not result in decent outputs.

2. Related Work

Optical Flow Applications of optical flow range from alignment estimation to motion estimation. Various approaches

Figure 1. Comparison against RAFT(d) and CeLiu(c) for input images (a) and (b) before exposure adjustment. Results are warped against (a) as reference, and the pink box indicates incorrect motion estimation in (d)

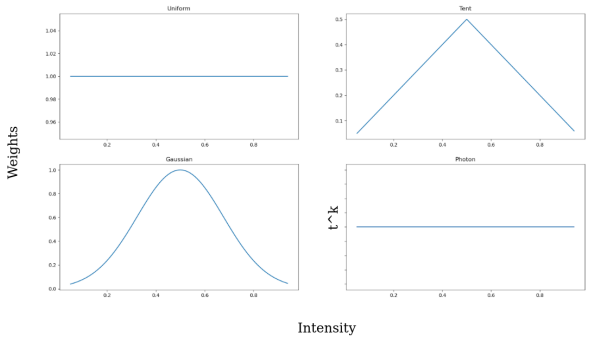

Figure 2. Blending weights for traditional merging

have been proposed and improved upon in literature for the estimation of optical flow in video sequences; Starting from Horn and Schunchk [5], which exploits brightness patterns of pixels with motion in a purely geometry based approach, to recent methods in deep learning based on fine to coarse methods using CNNs [1]. CeLiu's optical flow implementation is a simple ready to use framework [2]. It uses an iterative re-weighted least squares technique under temporal matching constraints to estimate flow. FlowNet proposes variations of an end to end trainable network: simple and correlation based CNN to predict dense 2-channel flow vector for each pixel. However I spent a lot of time in trying to install FlowNet to no avail and hence did not experiment with it. I also evaluated another deep learning based approach called Recurrent All Pairs Field Transforms (RAFT) [9], which consists of a feature encoder that extracts features per pixel for both images to be aligned, a correlation volume generation for computing visual similarity and performs iterative update to recurrently update optical flow estimates from correlation volumes and current estimates. The idea of using deep learning for optical flow was an experiment to see if a combined framework embedded with late fusion of the flow output and the actual input could help in predicting the HDR image. However, the results of RAFT, using model trained on the things dataset, on input images used for training (as shown in Figure 1), was not great and hence did not proceed with this approach. A comparison against the CeLiu's classical method is also shown.

HDR Merging Traditional methods for merging include using different set of blending weights [7] such as Uniform, Photon, Tent and Gaussian methods, as shown in Figure 2. HDRNet [4] proposes an image enhancement technique with light computations aimed at performance on devices with limited compute abilities. They use a neural network architecture to learn local, global and context dependent changes to approximate the desired transformation of input. Their method can even take lower resolution inputs at run time and apply learned affine transformations in bilateral space. While their method generalizes for various inputs, I wanted to experiment specifically with HDR merging from a few LDR images. One such method [6] that requires 3 inputs with a reference image (can be generalized for merging operations only) uses neural networks, the universal function approximators, to model the merge process after aligning the input using flow. Another learning based method [10] for merging that uses a non-flow based deep network for dynamic scenes uses a translation network (Unet [8]) with a symmetric encoder decoder architecture. Thus they implicitly account for flow while performing merging. [3] uses a convolution neural network (CNN) to predict HDR image from a single exposure image. They also use a hybrid dynamic range auto-encoder framework. The encoder produces a latent space representation of the LDR input and the decoder reconstructs from this latent space a HDR image in log domain.

3. Approach

As shown in Figure 3 and similar to [6], I follow a 2-step process of aligning the images and merging them using a CNN. It takes 3 images of any scene (Z_1, Z_2, Z_3) and produces a HDR image H which is based on Z_2 image as reference. Z_1, Z_2 are images taken at lower and higher exposures in comparison to Z_2 respectively. After alignment, the 3 images (I_1, I_2, I_3), where $I_2 = Z_2$, are combined in the merge process using a 4-layer neural network. The tone mapped version of the predicted input is further used for optimization. I experimented with RAFT [9] for optical flow to see if using a deep learning method for optical flow as well can produce better results, however the pre-trained model of RAFT required further training to produce decent

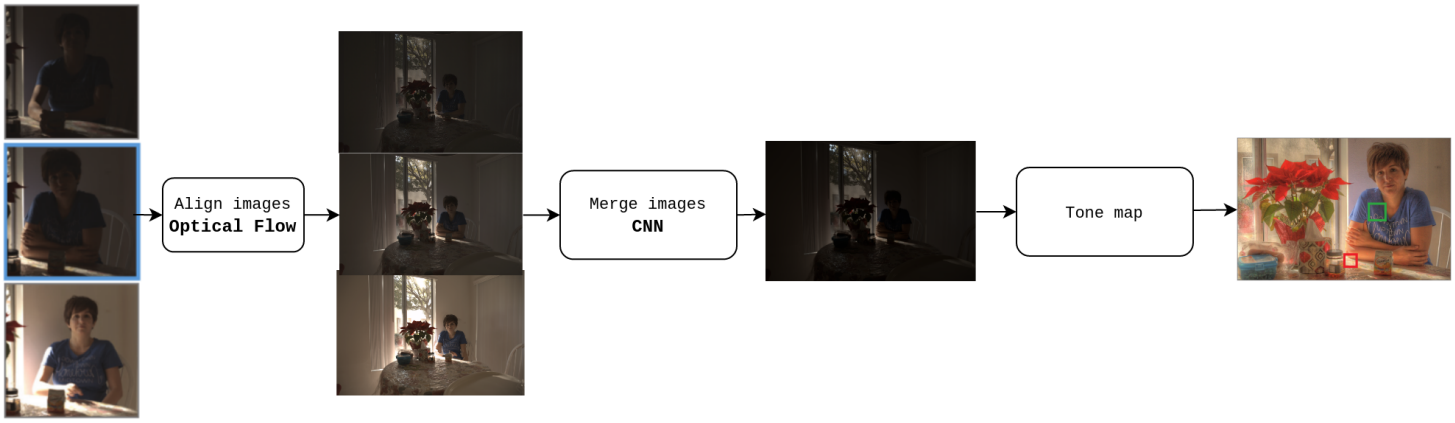

Figure 3. Approach overview: Stage 1: Alignment using Optical Flow, Stage 2: Merging using CNN

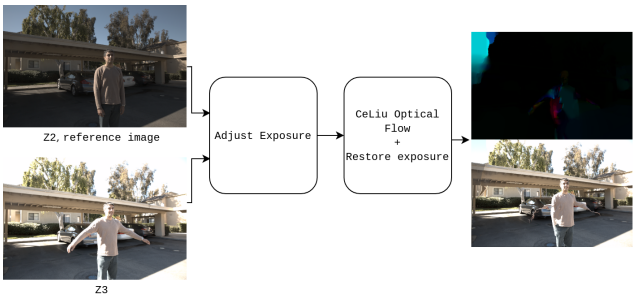

Figure 4. Optical flow pipeline

alignment results and hence did not proceed with this approach. The network used for merging is the same as that used in [6].

3.1. Alignment

CeLiu's method [2] of optical flow explores the brightness constancy assumption. It minimizes the observed image intensities of point in the scene displaced by set of flow vectors along the x and y direction of the image using an iterative re-weighted least squares approach. Thus this method produces a set of flow vectors u and v which represent the translation warp required to align against a reference image. Since the images are assumed to be captured under the same lighting conditions for optical flow, the input images must be adjusted in exposure before computing flow. To adjust exposure between the medium and high exposure inputs $(Z_2 \& Z_3)$ to retain the exposure of the high exposure image, I normalize as shown in Equation 1, where t_2, t_3 are the exposures of the second and the third image used. Z_{23}, the reference image at high exposure, is fed with Z_2 to estimate flow vectors that warp Z_3 against Z_{23}. The output is also clipped to retain values between $[0, 1]$. After warping based on these flow vectors, the aligned image is obtained.

Figure 4 shows the sample output obtained for a medium and high exposure pair, the flow vectors along the x and y direction are combined to be displayed as a RGB image. As can be seen, the aligned image has a few artifacts that occurs as a result of warping. A similar procedure is followed for low and medium exposure images. After the images are aligned, its original exposure is restored to be provided as input (I_1, I_2, I_3) to the next stage.

$$Z_{23} = clip(Z_2(t_3/t_2)^{1/\gamma}) \tag{1}$$

3.2. Deep Merging

This part of the pipeline can be summarized as shown in Figure 5. The HDR merge process can be formulated as:

$$H = g(I, \overline{\mathrm{H}}), \tag{2}$$

where g is the function that maps the aligned input LDR images I and their exposure adjusted counterparts $\overline{\mathrm{H}}$ in the HDR domain with the desired output HDR image H. $\overline{\mathrm{H}}$ represents a set of images H_1, H_2, H_3 such that $H_i = I_i^{\gamma}\ t_i$, where i corresponds to the i^{th} image and t_i, its corresponding exposure. I experiment with two different architectures that use the same network model. It is modelled as a fully convolutional neural network since images must be predicted as output and use filters of decreasing sizes (ranging from 7 to 1), as shown in Figure 6. All layers but the last one is succeeded by a Rectified Linear Unit as activation layer. The last layer is activated by a sigmoid function to ensure a $[0, 1]$ output range. Since the same output size also has to be ensured as input, no striding is performed while estimating the output from each layer, thus no down sampling or upsampling is performed.

The two architectures differ in the number of output layers. In the first architecture, called direct method, I directly estimate the HDR image as output. Thus this model outputs a 3 channel image corresponding to each channel of the output HDR image. In the second architecture, called weight

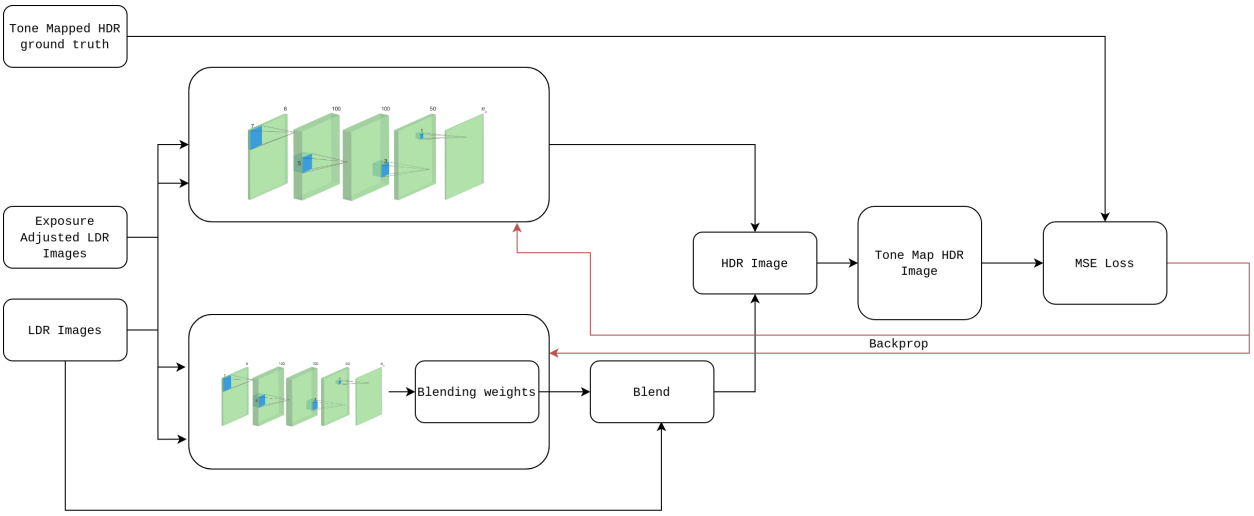

Figure 5. Merging pipeline

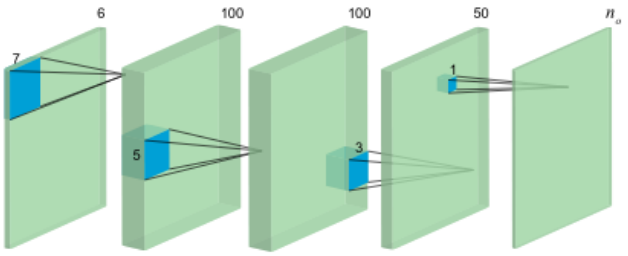

Figure 6. 4-layer Network model, taken from [6]

estimator (WE), I estimate the blending weights required for merging. Thus the network outputs a 9 channel image, with each channel corresponding to the blending weights used for each of the three color channels of the three input image.

Direct Method: The network outputted HDR image H is tone mapped to produce (see Figure 5) the final output T, which is given by:

$$T = \frac{log(1+\mu H)}{log(1+\mu)} \tag{3}$$

where $\mu = 5000$ and H is always in range $[0, 1]$. This ensures that we do not need to normalize or clip the output and introduce a non-differentiable operation. This tone mapping operation is similar to range compressing used in audio processing applications, and is also differentiable as opposed to Gamma encoding ($H^{\frac{1}{\gamma}}$ is not differentiable around 0)or any other complex tone mapping operation. The derivative of the tone mapped output T with respect to the network outputted image H is computed as shown in Equation 4.

$$\frac{\partial T}{\partial H} = \frac{\mu}{log(1+\mu)} \frac{1}{1+\mu H} \tag{4}$$

The derivative of the network output with its weights can be calculated through back propagation. Thus the network can be trained to learn a complex process, however over-fitting has to be avoided with limited data as input and hence the simple network architecture is good enough.

Weight Estimator Method: Similar to existing methods that compute a set of blending weights for each channel of the input, the network model used for this architecture outputs a 9 channel image α as opposed to 3. This also helps in better constraining the problem compared to the previous approach. The final HDR image can be computed as shown below:

$$H(p) = \frac{\sum \alpha_j(p) \overline{\mathrm{H_j(p)}}}{\sum \alpha_j(p)} \tag{5}$$

where j corresponds to each of the aligned input images index and p corresponds to every pixel of the image. Estimating α using a neural network helps get rid of alignment artifacts better than conventional merging techniques. Differentiability of the network is ensured as the derivative of the output against the α is:

$$\frac{\partial H}{\partial \alpha_i} = \frac{\overline{\mathrm{H_j(p)}} - H(p)}{\sum \alpha_j(p)} \tag{6}$$

This output is then tone mapped for optimization. Similar to the direct method, the network can be back propagated to estimate the derivative of the network output with respect to the network weights. Thus this also results in a fully

differentiable architecture that can trained as an end to end network.

Loss Function The system is trained by minimizing the L^2 distance between the predicted output and the ground truth after tone mapping. The loss function is defined as:

$$E = \sum (T_k - T'_k)^2 \tag{7}$$

where $k = 1, 2, 3$ corresponding to each color channel of the output, T_k corresponds to the ground truth tone mapped HDR image based on range compressor function, and T'_k corresponds to the predicted output of the network. The error was not averaged but summed for all pixels instead since otherwise the loss converged to 0 very quickly due to numerical approximations and the network did not train afterwards.

4. Experimental Setup

Thankfully, the authors of [6] made the dataset they used for training public. It comprised of 3 RAW Tif files for merging at various exposures, the log_2 of which was also available in a txt file. A HDR image was also available in a format readable using skimage itself. There were about 100 scenes including test and train set. I also added a few captures of 3 different dynamic scenes to the dataset for training. I also augmented the dataset by flipping and rotating images after alignment to add more data and avoid over fitting. For optical flow, default parameters were used as in python implementation of [2]. For training the deep learning network, I used an Adam optimizer with a learning rate of $1e-5$ and default $\beta_1 = 0.9, \beta_2 = 0.999$. I trained for over 100 epochs on NVIDIA Tesla v100 for 3 hours per attempt at training from scratch. Code for this project can be found here.

5. Results

To evaluate the quality of the detected output, a common metric used in signal processing, Peak Signal to Noise Ratio (PSNR) (Refer Equation 8, N corresponds to the size of the image) was used. It estimates the quality of the produced HDR image since it is inversely proportional to mean square error. The loss obtained while training and testing the network and the corresponding PSNR values at those instances are graphed in Figure 8. As mentioned earlier, since the error was summed rather than averaged to achieve reasonable convergence, the loss values are scaled higher. The input used for training was always resized images of size $1500 \times 1000, N = 15e5$ and hence the average loss scales down to reasonable value. Convergence values at the end of training and testing (after averaging) is given in Tables 1 and 2 respectively. In comparison between the two different architectures, Direct method performs better than the weight estimation method. This makes sense, since the

Figure 7. Sample input (top row) and aligned images (bottom row)

Approach	Train MSE	PSNR
Direct	0.1986	18.30
WE	0.2113	16.92

Table 1. Results on train set

Approach	Test MSE	PSNR
Direct	0.2533	18.35
WE	0.2800	16.7

Table 2. Results on test set

WE method is estimating 9 output channels while the former is directly estimating the HDR image. It can also be seen that since the test loss is also decreasing with values along the same range, the network is simply not memorizing but learning the mapping between the LDR and HDR image. The PSNR value for test set is slightly higher for Direct, however the difference is quite small.

$$PSNR = 10\ log(\frac{N}{(T_k - T'_k)^2}) \tag{8}$$

Figure 7 shows a sample test case used to demonstrate results obtained. The boxes in the bottom row highlights the artifacts introduced due to misalignment. The output HDR image from the network using direct and WE methods are shown in Figures 9 and 10. It can be seen that both methods result in similar visual output. However when observed closely, it can be seen that the edges of the WE method are interpolated and additional artifacts are introduced as a result, this is not observed in direct method. However, if we look near the shoulder of the person, direct method introduces a high exposure value at that point whereas WE does not. I personally prefer the Direct method since it also has a better PSNR value and a lower loss during convergence.

The output produced by using the conventional weights used for tents is shown in Figure 11. In comparison with the learning based method, it can be seen that the ghostly artifacts still remain. Infact, the resulting output has combined

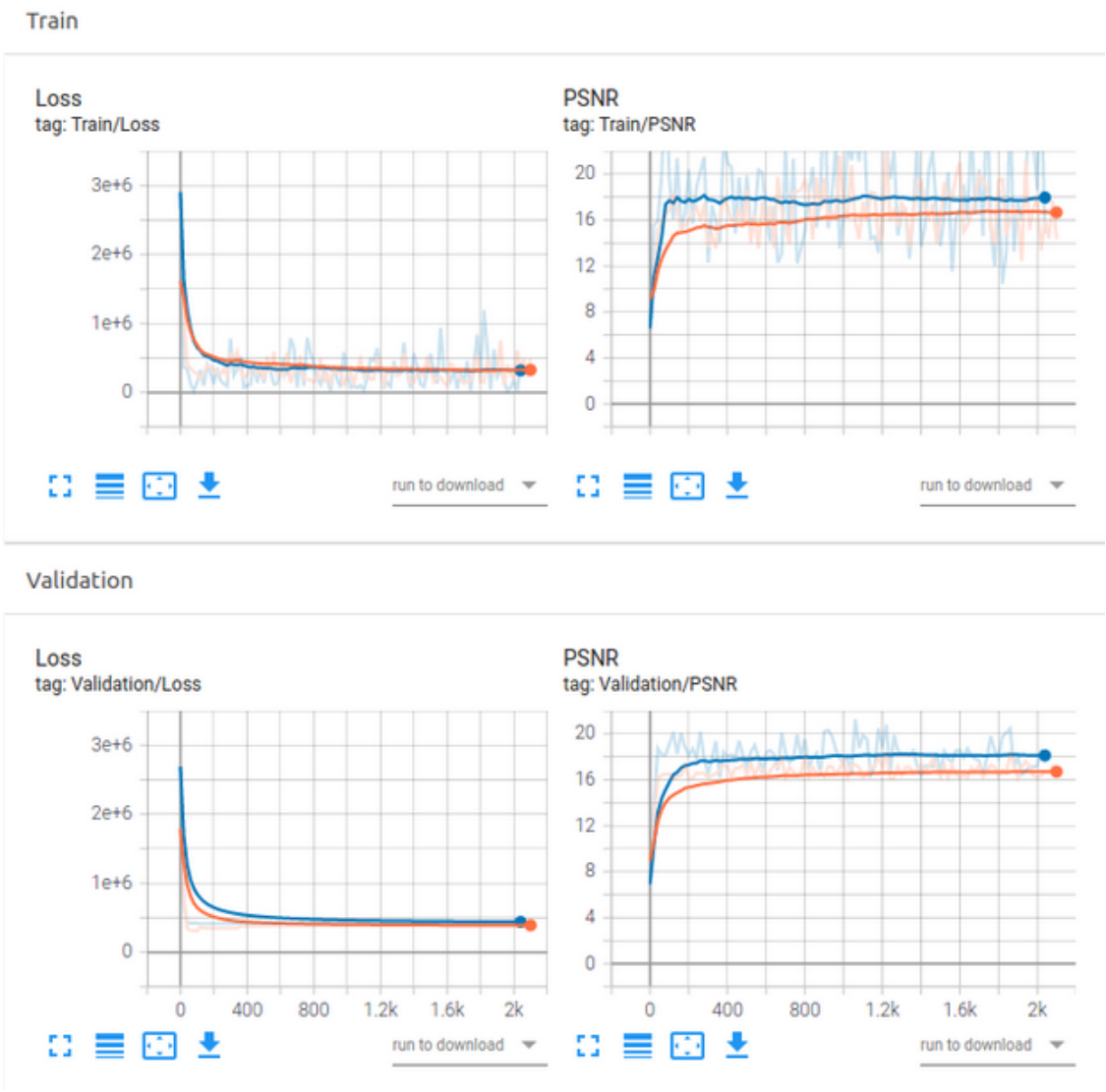

Figure 8. Training and testing loss and PSNR for WE, Direct

Figure 9. WE method

Figure 10. Direct method

Figure 11. Traditional merging using Tent weights

Metric	Tent	DL direct
Mean Square Error	0.0594	0.0037
PSNR	12.27	24.31
Run Time	9.58s	5.88s

Table 3. Metrics for tent vs DL

artifacts from the low and high exposure images. Thus this proves the effectiveness of the deep learning method that implicitly is able to reject dynamic pixels and also correctly extrapolate to produce reasonable output. Error metrics and run time for merging are compared for the traditional and direct methods in Table 3. It can be seen that in comparison with the ground truth, an improvement of $10\times$ is observed when using learning based method in mean square error, and approximately $2\times$ improvement in PSNR. The learning based method is also faster. It must be noted that the metric for time provided in Table 3 is only averaged over 2 runs and not profiled. However it can be generalized and said that inference time for this simple 4-layer network is lesser than conventional merging methods.

I also compared deep learning method's performance against traditional methods for static scenes. Figure 12 shows the input fed to the network. As can be seen from Figure 13, direct and conventional methods produce visually similar results, and have a mean square error of 0.0012 (before tone mapping, it is not fair to compare after tone mapping quantitatively since they use different tone mapping functions) between them. However the WE method introduces additional artifacts along the region near the wall in the bottom right. This is interesting since WE failed to interpolate along the edges of the image in previous cases, but it can be attributed to the fact that the network is predicting the set of blending weights and has higher constraints that prevents it from differentiating with static and dynamic pixels in background and foreground. Another reason could also be that the network was trained for input with dynamic objects primarily, and hence did not perform well for images with no subject or main object.

6. Conclusion and Future Work

In this project, I trained a 4-layer convolutional neural network to predict the HDR image from a set of 3 LDR images captured at various exposures while accounting for any misalignments due to camera motion and/or objects in the scene using optical flow. The results show the potential for learning based methods over conventional methods used. In the future, I would like to develop a unified framework for estimating flow and the HDR framework by replacing the optical flow part of the pipeline with a CNN that is also simultaneously trained along with the merging framework. I would also like to experiment with additional input modalities such as flow vectors and depth maps to implicitly guide the network in predicting the HDR image. My intuition is that flow vectors and depth maps essentially provide geometric information for the network to better extrapolate foreground and background pixels that are in motion. Another constraint with this approach is that it takes 3 images as input, while it can be used with higher number of inputs after fine tuning, reducing the number of images will require additional work to produce decent input, thus I would also like to experiment with certain encoder decoder architecture with latent space encoding and decoding in HDR domain.

7. Unsuccessful Experiments

In this approach, the network takes as input the exposure adjusted HDR images and the the LDR images as input. I instead tried giving as input LDR images and the flow vector as input. Thus instead of a 18 channel input the network was trained with 11 channel input. It was based on the intuition that the flow vector can implicitly guide the network. However, it only made it worse, and I am guessing that is because flow vectors estimated are not ideal and the exposure adjusted HDR images (that I ignored) give it additional information for extrapolation. The input images and the flow vectors are shown in Figures 14 and 15 respectively. The output obtained, compared against the ground truth is shown in 16. The output has no significant contributions from the blue and the green channels. This is interesting to note since the network has 11 input channels and did not understand the significance of each channel, which was easier to learn when there are 3 channels for each of the 2 sets of 3 images. I also tried experimenting with a 20 channel

Figure 12. Static input at low, medium and high exposures (from left to right)

Figure 13. Predicted HDR image using Diect method (left), using tent weighting (middle), using WE method (right)

Figure 14. Sample test input: unaligned images (top row) and aligned images (bottom row)

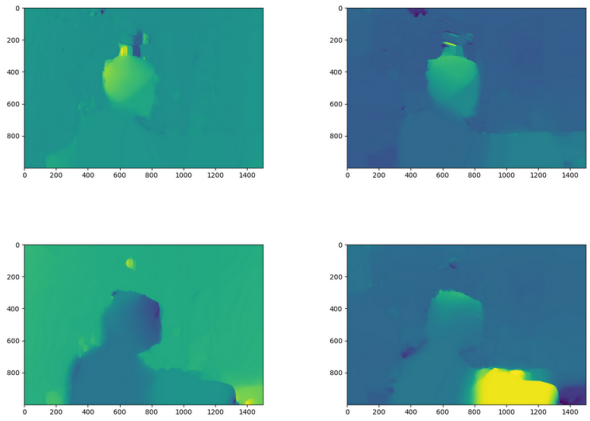

Figure 15. Additional input modality used: flow vectors along x (left) and y (right) directions for low (top row) and high exposure images (bottom row)

input, in addition to the 18 already used by the approach, I appended the flow vectors along the x and y direction as well. As expected it took a lot more time to train and I had to limit the batch size to 1 even when trained on AWS, and the network started overfitting. The validation loss started increasing while the training loss decreased, indicating that the network merely tried to memorize data it had already seen. Thus I feel that adding additional input modalities such as depth, and reducing either one set of 3 images used in current approach can help network train better.

References

[1] https://arxiv.org/abs/1504.06852. 2

[2] https://people.csail.mit.edu/celiu/opticalflow/. 2, 3, 5

[3] Gabriel Eilertsen, Joel Kronander, Gyorgy Denes, Rafał K Mantiuk, and Jonas Unger. Hdr image reconstruction from a single exposure using deep cnns. *ACM transactions on graphics (TOG)*, 36(6):1–15, 2017. 2

Figure 16. Output obtained from using 11 input channel (left) method vs direct method for merging (right)

[4] Michaël Gharbi, Jiawen Chen, Jonathan T Barron, Samuel W Hasinoff, and Frédo Durand. Deep bilateral learning for real-time image enhancement. *ACM Transactions on Graphics (TOG)*, 36(4):118, 2017. 2

[5] Berthold KP Horn and Brian G Schunck. Determining optical flow. In *Techniques and Applications of Image Understanding*, volume 281, pages 319–331. International Society for Optics and Photonics, 1981. 2

[6] Nima Khademi Kalantari and Ravi Ramamoorthi. Deep high dynamic range imaging of dynamic scenes. *ACM Trans. Graph.*, 36(4):144–1, 2017. 1, 2, 3, 4, 5

[7] Kristian Kirk and Hans Jørgen Andersen. Noise characterization of weighting schemes for combination of multiple exposures. In *BMVC*, volume 3, pages 1129–1138. Citeseer, 2006. 2

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 2

[9] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. *arXiv preprint arXiv:2003.12039*, 2020. 2

[10] Shangzhe Wu, Jiarui Xu, Yu-Wing Tai, and Chi-Keung Tang. Deep high dynamic range imaging with large foreground motions. In *The European Conference on Computer Vision (ECCV)*, 2018. 2