Light Field Reconstruction from Single Coded Image using CNN

A Project Report

submitted by

SAIKIRAN CHOLLETI

in partial fulfilment of requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS

MAY 2017

THESIS CERTIFICATE

This is to certify that the thesis titled **Light Field Reconstruction from Single Coded Image using CNN**, submitted by **Saikiran Cholleti**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology**, is a bonafide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof.Kaushik Mitra
Research Guide
Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Place: Chennai

Date: 25th May 2017

ACKNOWLEDGEMENTS

This work would not have been possible without the guidance and the help of several people. I take this opportunity to extend my sincere gratitude to all those who made this thesis possible.

First, I would like to thank all my teachers who bestowed me with good academic knowledge. I am indebted to my advisor Prof. Kaushik Mitra whose expertise, generous guidance and support made it possible for me to work on a topic that was of great interest to me.

I would also like to thank my lab mate and dear friend Anil Kumar Vadathya for sharing his valuable ideas and helping me whenever I am stuck with some problem.

I would like to thank my family for giving support and guidance all through my life. I would also like to thank all my friends and well-wishers for helping me in difficult times and being a good source of support and guidance.

ABSTRACT

Light field photography captures rich structural information that may facilitate a number of traditional image processing and computer vision tasks. A crucial ingredient in such endeavors is accurate depth recovery. Light field photography has gained a significant research interest in the last two decades.However, there is an inherent trade-off between the angular and spatial resolution, and thus, our method tries to increase the spatial resolution by reconstructing entire 5x5 light field using coded image and centerview.We use deep learning along with compressive light field capture to get the novel views.We first warp the centerview using the disparity we obtain as a outpur from our network and interpolate the holes resulted present in warped image to get the novel view.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Light Field

Light fields provide a rich representation of real-world scenes, enabling exciting applications such as refocusing and viewpoint change. Generally, they are obtained by capturing a set of 2D images from different views Levoy and Hanrahan [1996],Wilburn *et al.* [2005] or using a microlens array Adelson and Wang [1992],Ng *et al.* [2005],Georgiev and Intwala [2006]. The early light field cameras required custom-made camera setups

(a) taken form Marwah *et al.* [2013]

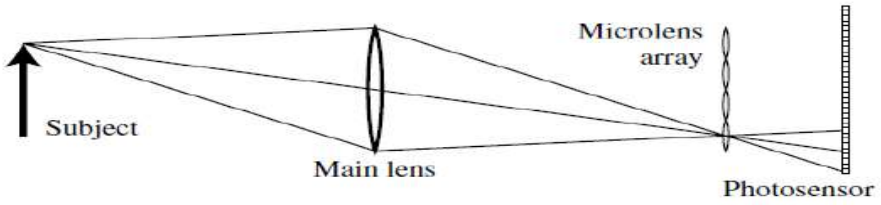
(b) taken from Levoy and Hanrahan [1996]

Figure 1.1: a)Different perspective shifts of image of pens in pot.
b)Refocussing of the flowers after the images have been taken from Lytro Illium camera

which were bulky and expensive, and thus, not available to the general public. Recently, there has been renewed interest in light field imaging with the introduction of commercial light field cameras such as Lytro [2016] and RayTrix [2016].However, because of

the limited resolution of the sensors, there is an inherent trade-off between angular and spatial resolution, which means the light field cameras sample sparsely in either the angular or spatial domain.

(a) taken frm Levoy and Hanrahan [1996]



(b) taken from Ng *et al.* [2005]

Figure 1.2: a)Stanford array of cameras used to take images at different view points, which is bulky. b)New technique which uses microlens array to get images with different perspective shifts

1.1.1 Compressive Light Field Capture

An image $i(x)$ captured by a camera sensor is the projection of an incident spatio-angular light field $l(x, \nu)$ along its angular dimension ν over the aperture area $\mathcal{V}$:

$$i(x) = \int_{\nu} l(x, \nu) \partial \nu \tag{1.1}$$

We adopt a two-plane parameterization Levoy and Hanrahan [1996], Gortler *et al.* [1996] for the light field where x is the 2D spatial dimension on the sensor plane and ν denotes the 2D position on the aperture plane at distance d_a (see Fig. 1.3, left). For brevity of notation, the light field in Equation 1 absorbs vignetting and other angle-dependent factors Ng *et al.* [2005]. We propose to insert a coded attenuation mask $f(\xi)$ at a distance d_l from the sensor, which optically modulates the light field prior to

projection as

$$i(x) = \int_{\nu} f(x + s(\nu - x))l(x, \nu)\partial\nu, \tag{1.2}$$

where $s = d_l = d_a$ is the shear of the mask pattern with respect to the light field (see Fig. 1.3, center). In discretized form, coded light field projection can be expressed as a matrix-vector multiplication:

$$i = \Phi l, \Phi = [\Phi_1 \Phi_2 ... \Phi_{p_\nu^2}] \tag{1.3}$$

where $i \in R^m$ and $l \in R^n$ are the vectorized sensor image and light field, respectively. All p_νxp_ν angular light field views $l_j (j = 1...p_\nu^2)$ are stacked in l. Note that each submatrix $\Phi_j \in R^{m \times m}$ is a sparse matrix containing the sheared mask code on its diagonal (see Fig. 1.3, right). For multiple recorded sensor images, the individual photographs and corresponding measurement matrices are stacked in i and Φ. The observed image $i = \sum_j \Phi_j l_j$ sums the light field views, each multiplied with the same mask code but sheared by different amounts. If the mask is mounted directly on the sensor, the shear vanishes $(s = 0)$ and the views are averaged. If the mask is located in the aperture $(s = 1)$, the diagonals of each submatrix Φ_j become constants which results in a weighted average of all light field views. In this case, however, the angular weights do not change over the sensor area. Intuitively, the most random, or similarly incoherent, sampling of different angular samples happens when the mask is located between sensor and aperture.

Equations 1–3 model a captured sensor image as the angular projection of the incident light field. These equations can be interpreted to either describe the entire sensor image or small neighborhoods of sensor pixels—2D patches—as the projection of the corresponding 4D light field patch. The sparsity priors discussed in the following sections exclusively operate on such small two-dimensional and four-dimensional patches.

1.2 Related Work

Inspired by the recent success of usage deep learning in a light field and depth reconstruction applications, such as Kalantari *et al.* [2016], Flynn *et al.* [2016] and Garg

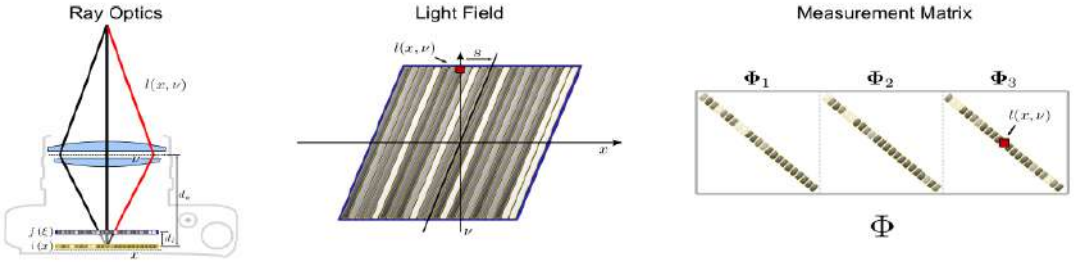


Figure 1.3: The proposed optical setup comprises a conventional camera with a coded attenuation mask mounted at a slight offset in front of the sensor (left). This mask optically modulates the light field (center) before it is projected onto the sensor. The coded projection operator is expressed as a sparse matrix Φ, here illustrated for a 2D light field with three views projected onto a 1D sensor (right). It is taken from Marwah *et al.* [2013]

et al. [2016], we propose to use convolutional neural networks (CNN) along with compressive light field photography to predict novel views using the coded image and the position of the novel view in the light field. However, the major challenge is that training a single end-to-end CNN for this task is difficult, producing novel views that are quite blurry. Existing view synthesis approaches Chaurasia *et al.* [2013], Wanner and Goldluecke [2014] and light field view typically first estimate the depth at the input views and use it to warp the input images to the novel view. They then combine these images in a specific way (e.g., by weighting each warped image Chaurasia *et al.* [2013]) to obtain the final novel view image). To make the learning more tractable, we build upon these methods and break down the task into disparity and in-painting components. The main contribution of our work is to use machine learning to model these two components and train both models by directly minimizing the error between the synthesized and ground truth images. Our system could potentially be used to decrease the required angular resolution of current cameras, which allows their spatial resolution to increase. The output of our first network is disparity and typically we would need ground truth disparities to train this network. However, we show how to train both networks simultaneously by directly minimizing the error between the synthesized and ground truth images. We propose a computational light field camera architecture that allows for high resolution light fields to be reconstructed from a single coded image and centerview. This is facilitated by exploring the co-design of camera optics and compressive computational processing.

CHAPTER 2

Neural Networks and Deep learning

Neural networks are a class of machine learning models that are used to approximate real valued, discrete valued and vector-valued functions. Motivated by the biological neural networks which are a set of interconnected neurons that enable information processing in organisms, artificial neural networks are presented as a set of interconnected nodes (also called neurons) which exchange messages with each other. The connections have numeric weights that can be tuned based on experience, making neural nets adaptive to inputs and capable of learning.

A deep neural network (DNN) is an artificial neural network with multiple hidden layers of units between input and output layers. DNNs attempt to model highlevel abstraction in data by using multiple processing layers, with complex structures composed of multiple non-linear transformations. Deep learning is a part of a family of machine learning methods based on learning representations of data. An observation (e.g., an image) can be represented in many ways such as a vector of intensity values per pixel, or in a more abstract way as a set of edges, regions of particular shape, etc. Some representations are better than the others at simplifying the learning task from examples. One of the promises of deep learning methods is replacing hand-crafted features with better feature representations that are automatically learnt for a given task.

Deep learning methods have seen a recent surge in success and popularity due to advances in Graphics Processing Units (GPU) hardware as well as new and improved methods for training them. A type of feed-forward neural network called convolutional neural network (CNN) is extensively used in computer vision and has shown state-of-the-art results in many problems such as object detection, image classification, image denoising, etc. In this work, we explore the use of CNNs in non-bind deblurring of images.

2.1 Modeling one neuron

One of the popular models for neurons in an Artificial Neural Network is the sigmoid model. The perceptrons (neurons) with out this sigmoid function at the end are essentially linear classifiers. The perceptron training rule will converge if the data is linearly separable. However, for overlapping data, convergence is not assured. Here we require our neural network to learn non-linear functions. A new model of neuron (the sigmoid

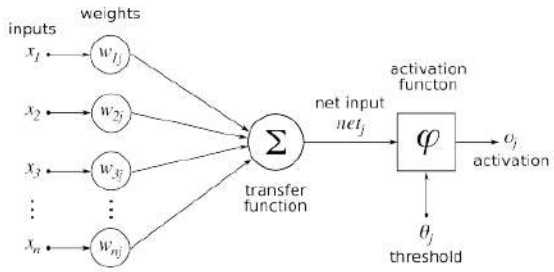


Figure 2.1: Artificial Neuron

model) was introduced to circumvent the problem. The sigmoid model is illustrated in Fig.2.1. The sigmoid unit first computes a linear combination of its inputs, then applies a threshold to the result. In the case of sigmoid unit, however, the threshold output is a continuous function of its input. More precisely, the sig- moid unit computes its output O as

$$o(\vec{x}) = \sigma(\vec{w}, \vec{x})$$
$$\sigma(y) = \frac{1}{1+e^{-y}}$$

The sigmoid function has a very useful property that its derivative is easily expressed in terms of its output.

$$\frac{\partial\sigma(y)}{\partial y} = \sigma(y)(1\text{-}\sigma(y))$$

Other activation functions which are generally used include tanh function.

2.1.1 Training sigmoid neurons

For the sigmoid neurons to predict output given an input vector, it becomes necessary to estimate the weights $\{w_i\}_0^n$ based on a set of training examples D. Let us denote t_d to be

the target output for the training example d. The set D is essentially the set of ordered pairs (d, t_d). In order to optimize w_i's, we need to specify a measure of training error relative to the training examples. One common measure of error between the predicted and the true output is the mean square error between them.

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} (t_d \text{ - } o_d)^2$$

Here $\vec{w} = (\vec{w}_0, \vec{w}_1 ... \vec{w}_n)$ is the weight vector to be learnt. We can optimize this error using the Gradient Descent Algorithm. In gradient descent, the error function is minimized by starting with an arbitrary weight vector, then iteratively updating the weight vector by moving in the direction of steepest descent along the error surface. The direction of the steepest descent is essentially the vector derivative of E with respect to $\vec{w}$.

$$\Delta E(\vec{w}) = (\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, .., \frac{\partial E}{\partial w_n})$$
$$\frac{\partial E}{\partial w_i} = \sum_{d \in D} (t_d - o_d)^2 = \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d \in D} (t_d - \sigma(\vec{w}, \vec{x}_d))^2$$
$$\frac{\partial E}{\partial w_i} = \sum_{d \in D} (t_d - \sigma(\vec{w}, \vec{x}_d))(o_d(1 - o_d)(-x_d))$$

Here x_{id} is the ith component of the vector $\vec{x}_d$. Now the iterative training rule can be written as

$$w_i \leftarrow w_i + \eta \sum_{d \in D} (t_d - \sigma(\vec{w}, \vec{x}_d))(o_d(1 - o_d)(-x_d))$$

2.2 Multilayer network

In the previous section, we have discussed a prominent model for neuron which is the fundamental unit in an ANN. By combining multiple units in a cascade, we obtain a model that will learn complex functions. Fig 2.2 shows one such network composed of 4 layers - one input layer, followed by 2 hidden layers which is then connected to an output layer. For the rest of this chapter, we shall consider only sigmoid neurons.

2.2.1 Backpropagation Algorithm

In 2.1.1 we have discussed gradient descent algorithm for training single sigmoid unit. To train multilayered network, we follow a similiar approach, but apply gradient updates in a layered fashion - for layer j, gradient output from layer j+1 is taken as input, gradient is then computed with respect to this output and then it is passed in to layer j-1. Since gradient propagates backwards, this algorithm is called "Backpropagation Algorithm". Let us now derive the update rules for the backpropagation algorithm.

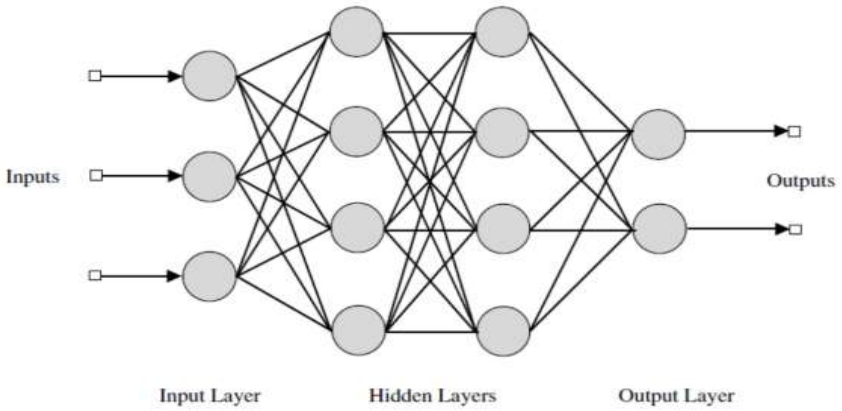


Figure 2.2: A neural network

The notation adopted for this section is given below.

- x_{ij} = i^{th} input to unit j
- w_{ij} = weight associated with i^{th} input to unit j
- net_j = the weighted sum of inputs for unit $j(\sum_i w_{ji}x_{ji})$
- o_j = output computed by unit j
- t_j = target output for unit j

As before, we use mean squared error between the target and output units as our error function. But now, since the output layer consist of multiple units, we need to compute the average error over all the units.

$$\frac{\partial E}{\partial w_i} = \frac{1}{2}\sum_{d\in D}\sum_{k\in outputs}(t_d - o_d)^2$$

Let us derive the update rule for a single training example- the overall gradient is the summation of gradients corresponding to all training examples.

$$E_d = \frac{1}{2} \sum_{k \in outputs} (t_d - o_d)^2$$
$$\frac{\partial E_d}{\partial w_{ij}} = \frac{\partial E_d}{\partial net_j} \frac{\partial net_j}{\partial w_{ij}} = \frac{\partial E_d}{\partial net_j} x_{ij}$$

Case 1: Training rule for output weights

In case of output weights, w_{ij} can influence the network only through net_j and net_j can influence the network only through o_j. Hence,

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j} \frac{\partial o_j}{\partial net_j}$$
$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \frac{1}{2} \sum_{k \in outputs} (t_d - o_d)^2$$
$$= -(t_j - o_j)$$
$$\frac{\partial o_j}{\partial net_j} = \frac{\partial \sigma(net_j)}{\partial net_j} = o_j(1 - o_j)$$

Summing everything, we get

$$\frac{\partial E_d}{\partial net_j} = -(t_j - o_j)(o_j(1 - o_j))$$

Case 2: Training rule for hidden unit weights

In case of hidden units, the derivative of the training rule for w_{ji} must take into account the indirect ways in which w_{ji} can influence the network outputs and hence Ed. For this reason, we will find it useful to refer to the set of all units immediately downstream of unit j in the network. We can then write

$$\frac{\partial E_d}{\partial net_j} = \sum_{k \in Downstreams_i} \frac{\partial E_d}{\partial net_k} \frac{\partial net_k}{\partial net_j}$$

Let δ_k denote $-\frac{\partial E_d}{\partial net_j}$ then

$$\delta_k = - \sum_{k \in Downstreams_i} \frac{\partial E_d}{\partial net_k} \frac{\partial net_k}{\partial net_j}$$
$$= - \sum_{k \in Downstreams_i} \delta_k \frac{\partial net_k}{\partial o_j} \frac{\partial o_j}{\partial net_j}$$
$$= o_j(1 - o_j) \sum_{k \in Downstreams_i} \delta_k w_{kj}$$

We can combine all these derived update rules to obtain the Backpropagation algorithm.

CHAPTER 3

Approach

The work by Marwah *et al.* [2013] states that coded image has enough information to recover the novel view. We propose a deep learning approach instead of dictionary learning used by Marwah *et al.* [2013]. The proposed method is shown below:

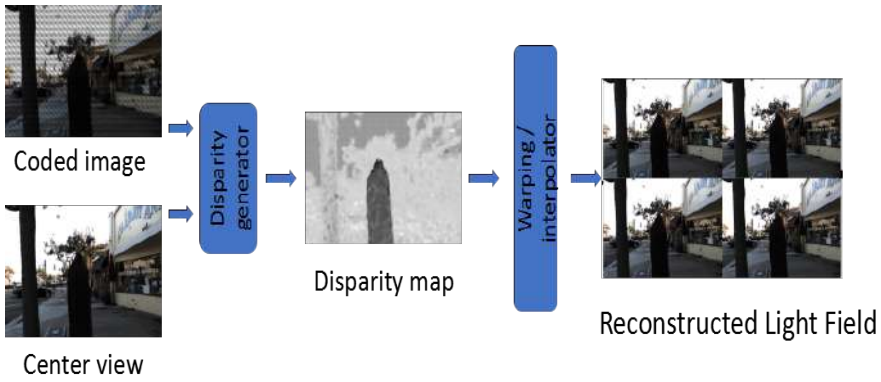


Figure 3.1: Proposed method

3.1 Network Architectures

Getting to optimal network architecture is a iterative process.We have tried different network architectures and tried to improve the results.The main Network architectures are shown below:

3.1.1 Network 1

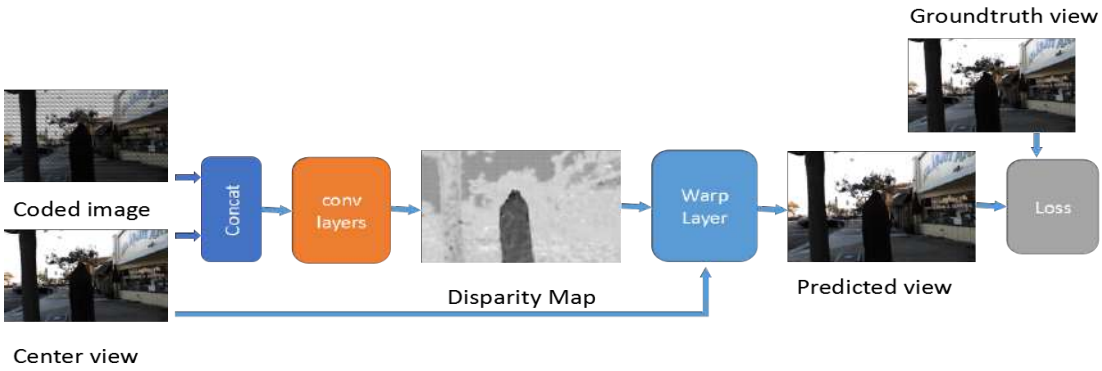


Figure 3.2: Network 1

3.1.2 Network 2

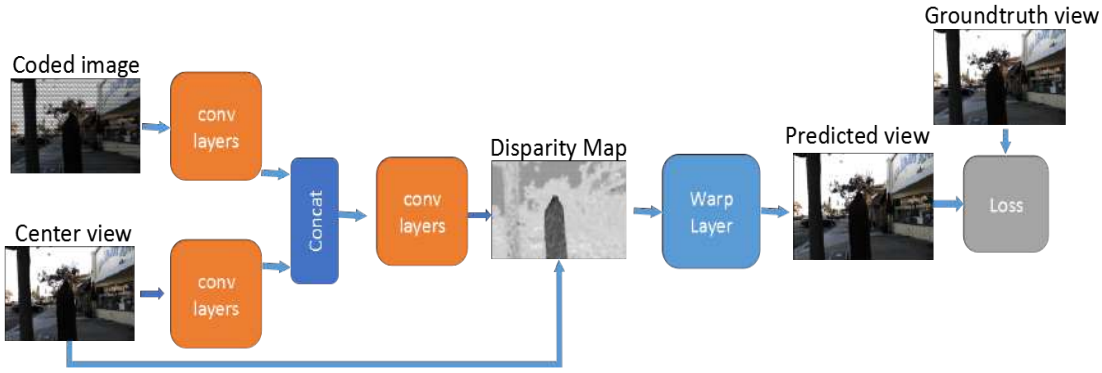


Figure 3.3: Network 2

3.1.3 Network 3

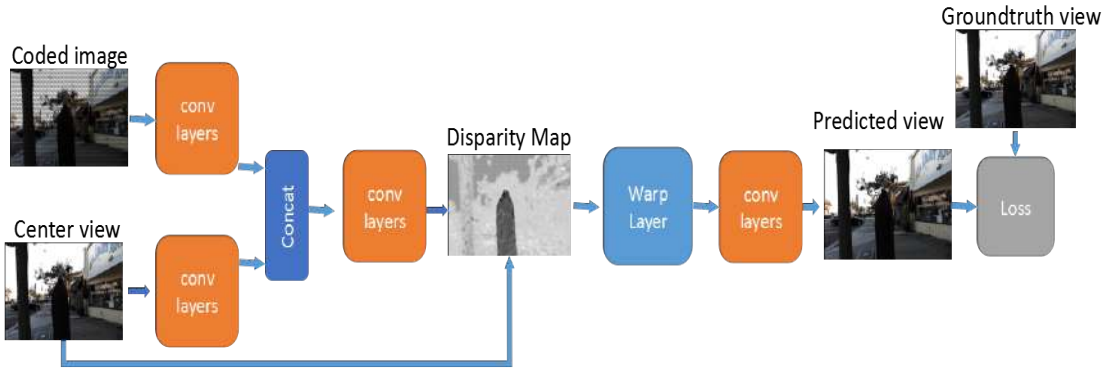


Figure 3.4: Network 3

3.1.4 Network 4

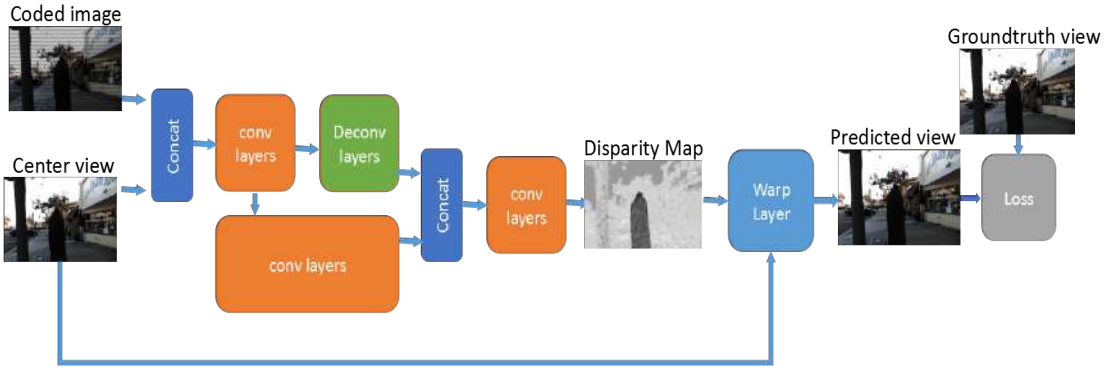


Figure 3.5: Network 4

3.1.5 Network 5 (final architecture)

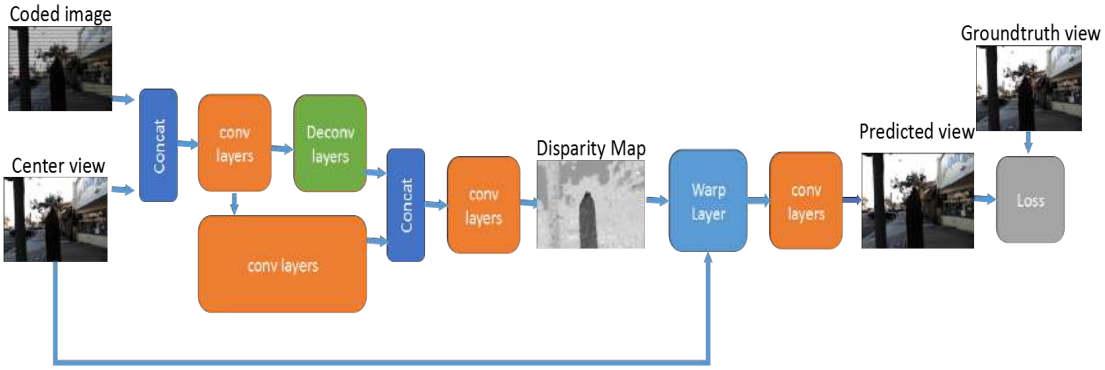


Figure 3.6: Network 5 (final architecture)

3.2 Performance of the networks

Table 3.1: Table showing performance of different network architectures

Newtork Architec-ture	Train error (avg)	Validation error (avg)	Test error (avg)
1	4	8	7
2	3	8	6
3	1	4	3.1
4	2	4	3.2
5	1.5	3.5	2.6

Though the errors of network 3 are less but the qualitative results are not good compared to network 4 and network 5 which implies that contextual knowledge is required to get good disparity maps in turn good lf images. Direct CNN network like network 1 gave blurry results as it is unable to learn the disparity.

3.3 Data Generation and Training

The coded images should ideally be obtained from compressive light field camera, but due to the unavailability of the hardware we have simulated the coded images by multiplying the light field views with the code which in this case is random normal gaussian code or mura code. centerviews are used groudtruth views. Centerviews can also be generated from coded images using series of convolutional layers but groudtruth centerviews are used as of now to get sharp disparity and light field views. other views of light field are used as the reference views.

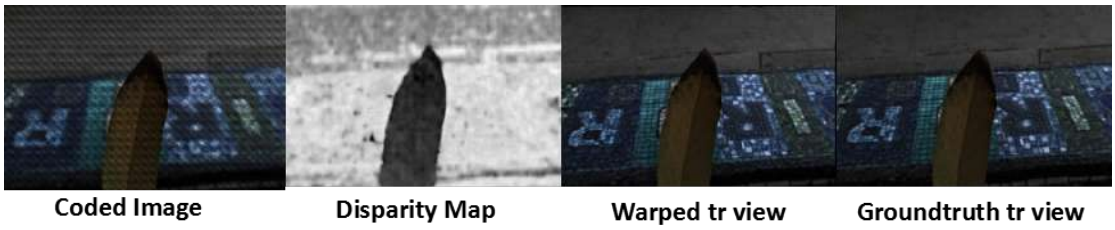
The dataset we used is same as the dataset used by Kalantari *et al.* [2016], the light field images are captured using lytro illium camera which produces 14x14 views but only middle 8x8 views are usable because of microlens drawback at corners. we use only 5x5 views of 8x8 views so there are 16 possible 5x5 views in which we take 10 random 5x5 views in each light field for training.The training is done patchwise with the patch size of 60x60 so this makes the training set bigger and makes easier for network to learn parameters.Coming to implementation details we used caffe framework to train and Adam solver is used.

CHAPTER 4

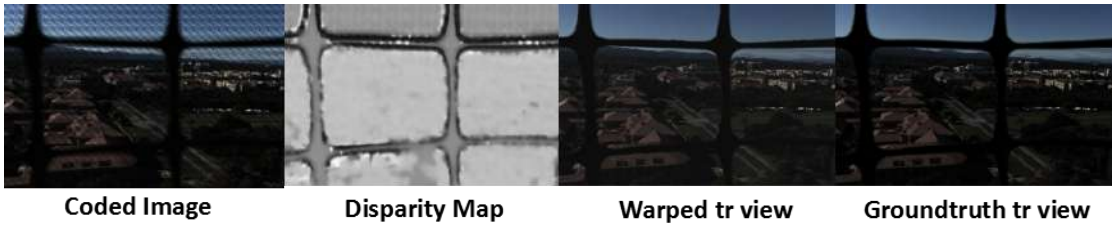
Results

The best results are obtained by network 5, the qualitative results are shown below.

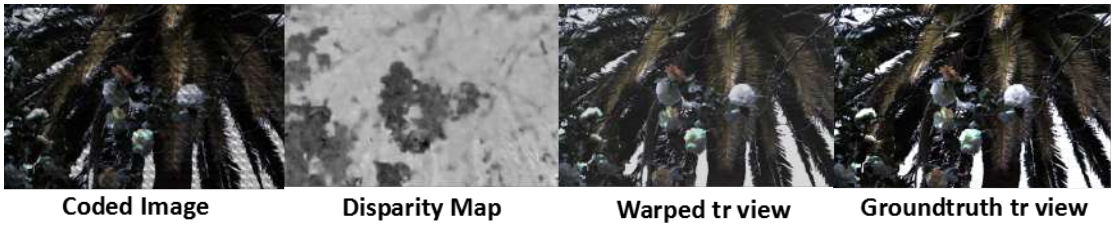
(a) psnr:31.4123 ssim:0.9516



(b) psnr:33.7650 ssim:0.9303



(c) psnr:26.9835 ssim:0.7196



(d) psnr:20.7460 ssim:0.7665

Figure 4.1: Topright (tr) views generated from coded image and cneter views, the edges of warped images matches with the ground truth views but the intepolation of the occlusion edges is inconsistent when there are high gradients in its neighbourhood which can be seen in 4.1c and 4.1d

The refocusing results obtained from the 5x5 light fields generated.

All Focus Image Focus in front Focus in the back

(a)

All Focus Image Focus in front Focus in the back

(b)

All Focus Image Focus in front Focus in the back

(c)

Figure 4.2: Refocusing the above images using the generated 5x5 light fields by warping the centerview with the generated disparity map

4.1 Conclusion and Future Work

The proposed method produces consistent disparity maps but decrease in psnr and ssim in 4.1c and 4.1d is due to contrast differences in predicted view and topright view. As of now we are using normal interpolation technique used in opencv but in future we will build a interpolation network which will do good job.

We are reconstructing light field views from both coded image and centerview, in our future work we will use only coded image and centerview will be reconstructed from coded image.

REFERENCES

1. **Adelson, E. H.** and **J. Y. Wang** (1992). Single lens stereo with a plenoptic camera. *IEEE transactions on pattern analysis and machine intelligence*, **14**(2), 99–106.

2. **Chaurasia, G.**, **S. Duchene**, **O. Sorkine-Hornung**, and **G. Drettakis** (2013). Depth synthesis and local warps for plausible image-based navigation. *ACM Transactions on Graphics (TOG)*, **32**(3), 30.

3. **Flynn, J.**, **I. Neulander**, **J. Philbin**, and **N. Snavely**, Deepstereo: Learning to predict new views from the world's imagery. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

4. **Garg, R.**, **G. Carneiro**, and **I. Reid**, Unsupervised cnn for single view depth estimation: Geometry to the rescue. *In European Conference on Computer Vision*. Springer, 2016.

5. **Georgiev, T.** and **C. Intwala** (2006). Light field camera design for integral view photography. *Adobe Technical Report*.

6. **Gortler, S. J.**, **R. Grzeszczuk**, **R. Szeliski**, and **M. F. Cohen**, The lumigraph. *In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 1996.

7. **Kalantari, N. K.**, **T.-C. Wang**, and **R. Ramamoorthi** (2016). Learning-based view synthesis for light field cameras. *ACM Transactions on Graphics (TOG)*, **35**(6), 193.

8. **Levoy, M.** and **P. Hanrahan**, Light field rendering. *In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 1996.

9. **Marwah, K.**, **G. Wetzstein**, **Y. Bando**, and **R. Raskar** (2013). Compressive light field photography using overcomplete dictionaries and optimized projections. *ACM Transactions on Graphics (TOG)*, **32**(4), 46.

10. **Ng, R.**, **M. Levoy**, **M. Brédif**, **G. Duval**, **M. Horowitz**, and **P. Hanrahan** (2005). Light field photography with a hand-held plenoptic camera. *Computer Science Technical Report CSTR*, **2**(11), 1–11.

11. **Wanner, S.** and **B. Goldluecke** (2014). Variational light field analysis for disparity estimation and super-resolution. *IEEE transactions on pattern analysis and machine intelligence*, **36**(3), 606–619.

12. **Wilburn, B.**, **N. Joshi**, **V. Vaish**, **E.-V. Talvala**, **E. Antunez**, **A. Barth**, **A. Adams**, **M. Horowitz**, and **M. Levoy**, High performance imaging using large camera arrays. *In ACM Transactions on Graphics (TOG)*, volume 24. ACM, 2005.