Generating disparity maps using Convolutional Neural Networks

Martin Asenov
The University of California, Irvine

masenov@uci.edu

Dimitar Dimitrov
The University of California, Irvine

ddimitro@uci.edu

Abstract

Extracting 3D information from a scene is both a fundamental problem in Computer Vision and one of the field's most valuable applications. A lot of solutions exists using specialized hardware like 3d cameras, different radar systems, Kinect, etc. However, all of them has trade-offs in terms of their accuracy, price and limitations that are usually unsatisfactory for commercial applications. In this paper we present a method for extracting 3D information from cameras that aims to partially mitigate the problems that prevent them to be the cheap and reliable solution for the stereo vision problem. Our solution is based on Convolutional Neural Networks which recently gained popularity for the problem of visual recognition. We modify their architecture to be better suited for depth map extraction and test them on both single and multi-view camera images. Furthermore, we make use of computer graphics generated data and experiment with transferring our results from animated pictures to the real ones.

1. Introduction

The Computer Vision community have proposed a lot of methods for extracting 3D information using image processing. Different methods include geometry models based on key points matching, volumetric and photometric stereo. However, all those methods suffer from the same weakness - they take into account only one visual cue at a time instead of combining and weighting the information provided by all of them. That is crucial not only because we miss on slew of information but because, as many psychological studies suggest humans do exactly that when they wrestle with the same problem. On top of that, there is whole another factor the previously mentioned methods totally miss out on - the image statistics. As human beings, we make all kinds of assumptions about the world we live in that serve us well. Those are based upon the repeating patterns we face in our everyday lives and are extremely useful part of our visual pipeline. Due to the complexity of the matter however, that remains something that hasn't been touched upon in the context of 3D reconstruction.

Our motivation to use CNNs for 3D reconstruction is based upon the fact that CNNs are a good way to catch those statistics, as the training process naturally picks the factors that really matter and weight them to determine how much they do according to the particular image we observe. We argue that although strictly mathematically speaking at least 2 pictures from different angles are needed to get accurate depth perception, combinations of the previously mentioned factors can give a really good estimate of the 3D scene, even from a single picture. That reasoning is supported by our own biology. We can get decent depth perception, even if we close one of our eyes. Our biology also dictates that although motion is very important input for our visual system, analyzing static images is more than enough for depth perception most of the time. Therefore, we decided to leave motion for future development. Last but not least, we want to make the argument that although sometimes we can only obtain approximations and relative distances from the factors mentioned, they are more than enough for the most tasks where 3D reconstruction is used.

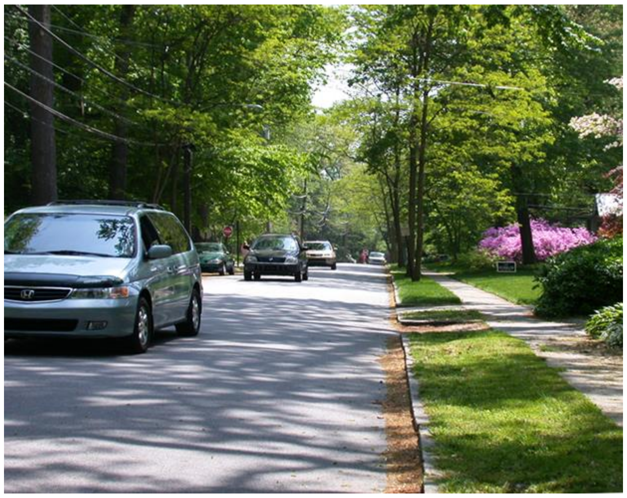

Figure 1. Perspective, relative size, occlusion and texture gradients all contribute to the three-dimensional appearance of this photo.

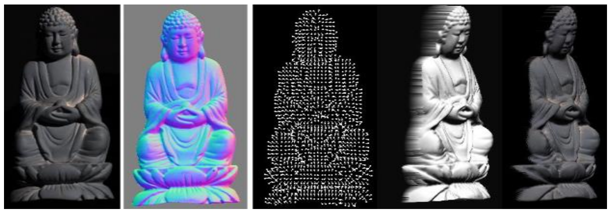
Figure 2. Reconstruction of the 3D shape of object from lighting.

1.1. Introduction continued

We regard to our experiment as way of exploring a radically different application of Convolutional Neural Networks. Looking at papers from the past 2-3 years one observes unpresented raise in successful application of convolutional neural networks. However, one can also argue that big part of problems people solve with CNNs these days are more or less connected to classification. Research is conducted on a more and more data, obtaining ever improving results mostly thanks to deeper models, trained on GPUs and super computers which experience exponential growth in computational power. Although, those advancements emerge from the research on neural networks for image classification they open the door for the use of CNNs for different purposes, as well.

2. Dataset

For our training we used a recently released computer generated dataset called MPI Sintel Flow Dataset [1]. It consists of 1064 frames, with resolution 1024x436. The dataset consist of different types of training data such as optical flow, depth and camera motion, segmentation and disparity maps. The data comes in two flavors - motioned blurred and pre-cleaned. In our experiments we trained and tested with the harder set of images in which the motion blur was added. For out final CNN we used only the disparity maps part of the dataset but we want to use some of the other types of information in our future experiments.

3. CNN architecture

For our experiments we used Matlab library called “MatConvNet” [3]. We decided to use late fusion architecture as described in [2]. Initially, we decided to feed the left and right camera images, convolve them with all the pre-trained filters learned on the problem of object classification on Imagenet and combine the results by the means of two fully connected layers. For the course of training we fixed the Imagenet filter and we used them as they are. At first we tried using rectify linear activation functions. No matter what picture we were feeding into the network, it was only outputting either random noise or a disparity map of all zeros. Thinking about the problem, we realized that a simpler linear activation function suits our problem better as it increases monotonous, which represents depth better than the sudden flatness of rectified linear. Due to the high resolution of the pictures and the relative small dataset, we decided that it is unfeasible to train on the whole images. Instead we trained our neural network on random 224x224 patches from the frames, adding different perturbations like flipping the images. In this way we were able to get a lot more training data and also reduce the complexity of our model. Even so, we didn’t accumulate that big of a dataset. Given its diversity however, we still obtained good results. In our experiments we used batched Stochastic Gradient Descent with momentum [6]. We ran experiments on different hyper parameters for four epochs, and then we trained the most successful one for another fifty (momentum of 0.95 and learning rate of 0.0001).

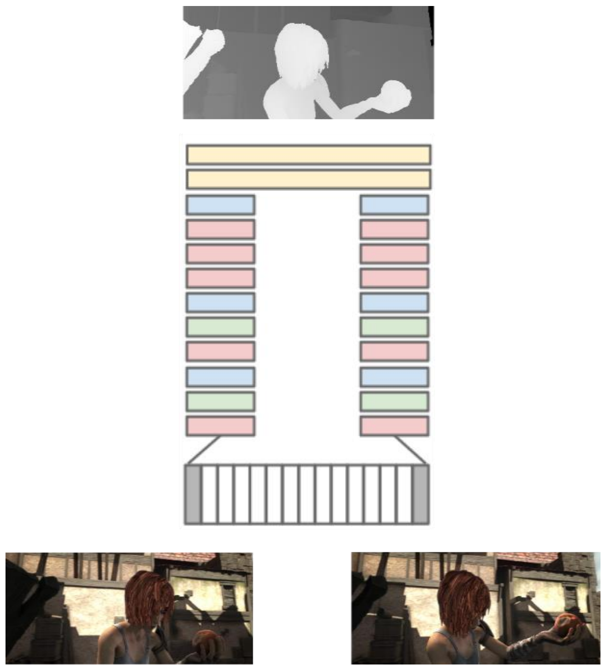
Figure 3. In our architecture we fed left and right camera image through pre-trained filters layers. We fed the output to a fully connected layers with linear activation functions and a final layer a disparity map.

4. Results

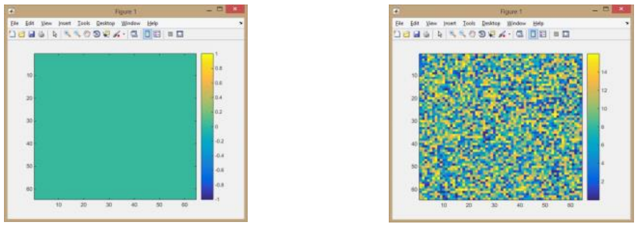
Figure 4. Initial results

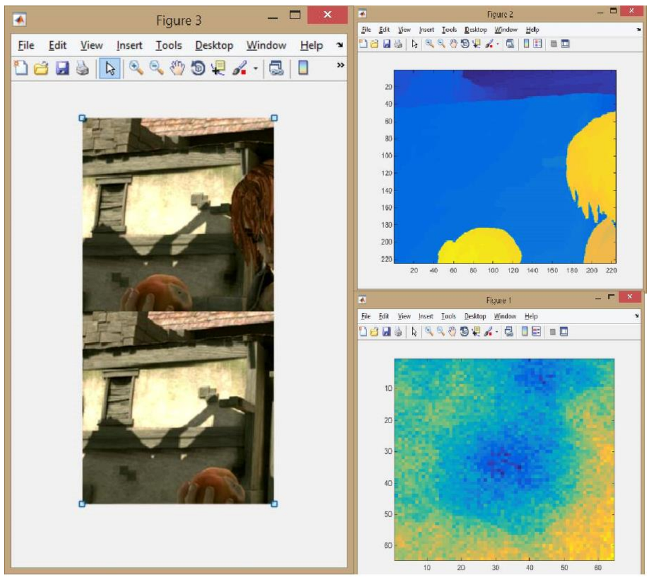

Figure 5. Input to our CNN (left), Ground truth disparity (top right), Output of our CNN (bottom right)

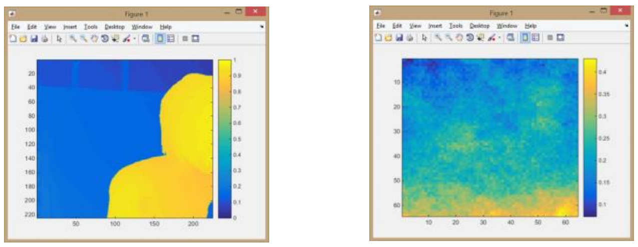

Figure 6. Training after 4 epochs, 0.95 momentum, 0.0001 learning rate. Ground truth disparity (left), our result (left)

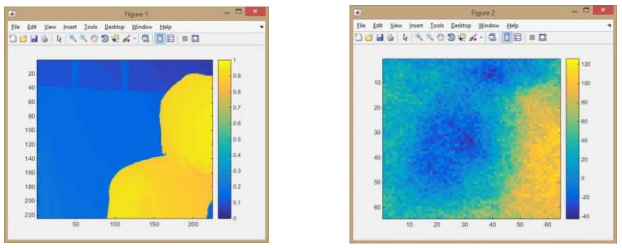

Figure 7. Training after 20 epochs, 0.95 momentum, 0.0001 learning rate. Ground truth disparity (left), our result (left)

5. Challenges

We came across numerous challenges throughout the project. First of all, we had to extend MatConvNet so we can feed two pictures as input, instead of one. Second, we created custom final layer error function that does SSD over the disparities maps. Moreover, we had to experiment with many different architectures and parameters. Last but not least, we came across a lot of GPU and CPU memory issues. The authors of the library suggest using at least of 8GB of video memory. Because we had only two available in our GPU we had to make some compromises between efficiency and memory usage, but managed to squish our neural network in the place available. Even after that we continued to run out of memory, because of improper freeing up of the memory and temporary variables. At the end we came up with quite 'hacky' solution of using batch script to train the network for only limited number of epochs, saving the model, restarting Matlab, loading the model back in memory.

6. Further improvements

Numerous papers [4] have concluded that first few filter layers are pretty generic and are not required to be trained for every newly introduced classification problem. However, because of the different nature of the problem we are trying to solve, it will be actually interesting to see if there is going to be a significant increase of the performance for our if we train our own filters. We also want to expand our model so we can feed a sequence of more than two frames, thus introducing motion cues into the mix. Moreover, the project can be expanded in the direction of generating different types of information from the images like optical flows or camera motions, both available in the dataset. Last but not least, it would be nice to work with a bigger dataset. Although our data augmentation helped training our model quite successfully, we expect to produce better results with more data.

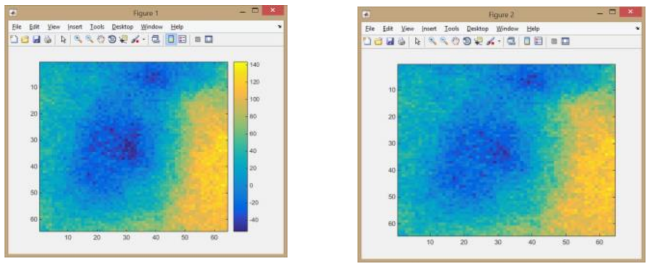

Figure 8. We experiment with different input as well. Instead of feeding left and right camera images, we fed the same 2D picture to our neural network. As you can see the results are almost identical. This is another proof towards our suggestion, that in order to generate 3D information, a lot of times just a single picture is enough.

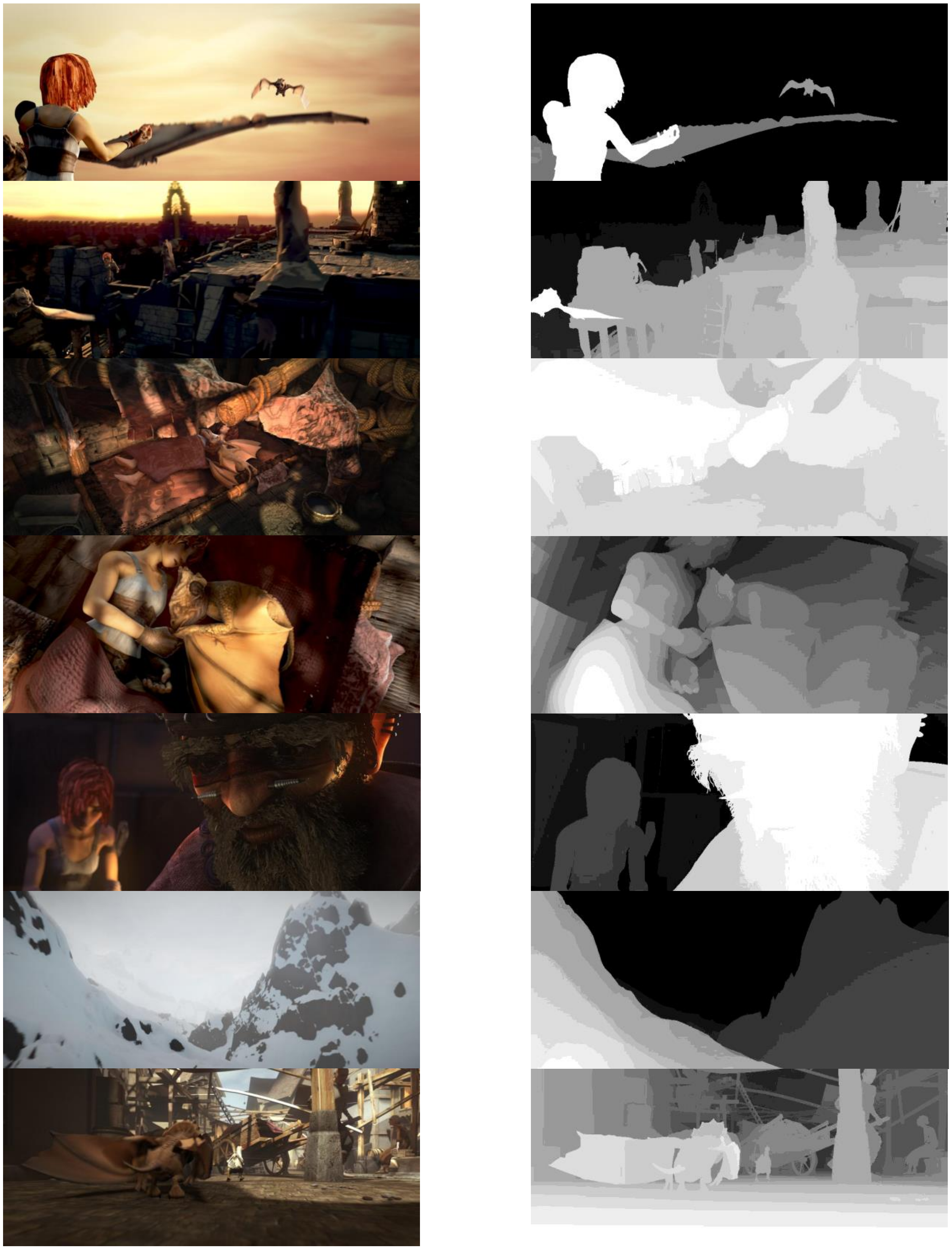

Figure 9.Sample images and their corresponding disparity maps from the MPI Sintel Dataset. Although they are really diverse and well suited for our purpose, having only 1064 really limits what can be learned by a neural network. However even with such a small dataset, through taking random smaller patches, we were able to produce promising results. In the future we are looking forward to train a similar model with a bigger dataset, and more powerful hardware to be able to fully extract all the features necessary for accurate 3D predictions.

References

[1] Butler, D. J. and Wulff, J. and Stanley, G. B. and Black, M. J., A naturalistic open source movie for optical flow evaluation, European Conf. on Computer Vision (ECCV), A. Fitzgibbon et al. (Eds.), Springer-Verlag, Part IV, LNCS 7577, oct, 611--625, 2012

[2] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.:Large-scale video classication with convolutional neural networks. In: Proc. CVPR(2014)

[3] A. Vedaldi and K. Lenc, MatConvNet - Convolutional Neural Networks for MATLAB", arXiv:1412.4564, 2014

[4] M. Oquab, L. Bottou, I. Laptev and J. Sivic "Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks" CVPR 2014

[5] Zhu, X., Vondrick, C., Ramanan, D. Fowlkes, C.: Do we need more training data or better models for object detection? BMVC 2012

[6] Orr, Genevieve, CS 449 Neural Networks, Momentum, http://www.willamette.edu/~gorr/classes/cs449/momrate.html