DEPTH ESTIMATION FROM A MONOCULAR IMAGE SEQUENCE

July 25, 2019

Swadhin Agrawal
Robert Bosch Center for Cyber Physical Systems
Guided by
Prof.Raghu Krishnapuram

Contents

0.1 Goal to achieve

We wanted a program which can take **Monocular images** from the camera of a robot and can tell us the depth of each object in it.

0.2 Basic Mathematics behind depth estimation from images

0.2.1 Pinhole Camera Model

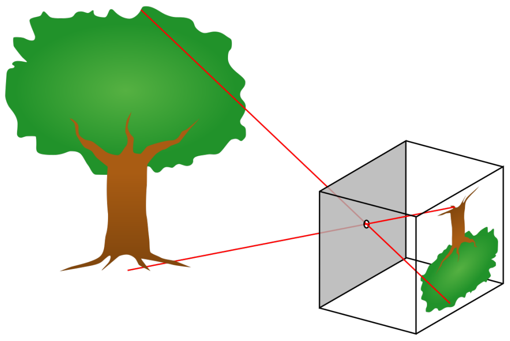


Figure 1: Taken from Wikipedia

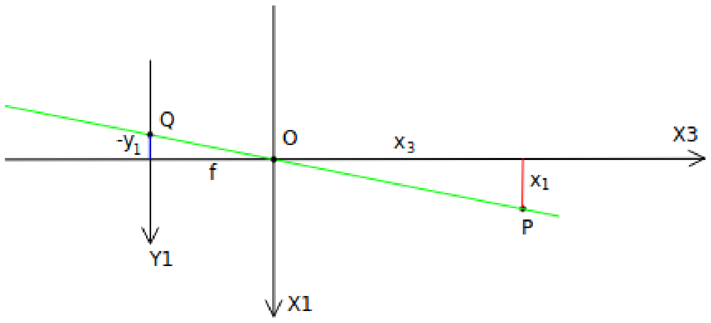


Figure 2: Taken from Wikipedia

$$\frac{-y_1}{f} = \frac{x_1}{x_3} \tag{1}$$

$$\frac{-y_2}{f} = \frac{x_2}{x_3} \tag{2}$$

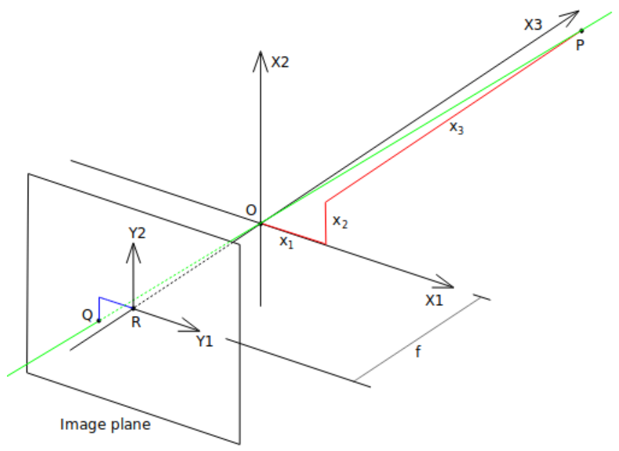


Figure 3: Taken from Wikipedia

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = -\frac{f}{x_3} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{3}$$

0.2.2 Homogeneous Coordinates

Advantage: The coordinates of points, including points at infinity, can be represented using finite coordinates.
Definition:Given a point (x, y) on the Euclidean plane, for any non-zero real number Z, the triple (xZ, yZ, Z) is called a set of homogeneous coordinates for the point.

0.2.3 Camera Matrix

Camera matrix or (camera) projection matrix is a 3 ÃŮ 4 matrix which describes the mapping of a pinhole camera from 3D points in the world to 2D points in an image.
Let x be a representation of a 3D point in homogeneous coordinates (a 4-dimensional vector), and let y be a representation of the image of this point in the pinhole camera (a 3-dimensional vector). Then the following relation holds

$$y \sim Cx \tag{4}$$

where C is the camera matrix and the ∼ sign which implies that the left and right hand sides are equal up to a non-zero scalar multiplication.
This equation (4) can be explained from basic linear algebra of coordinate transformations.

Derivation of Camera matrix:

Equation (3) can be re-written in homogeneous coordinates as:

$$\begin{pmatrix} y_1 \\ y_2 \\ 1 \end{pmatrix} = \frac{f}{x_3}\begin{pmatrix} x_1 \\ x_2 \\ \frac{x_3}{f} \end{pmatrix} \sim \begin{pmatrix} x_1 \\ x_2 \\ \frac{x_3}{f} \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{f} & 0 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix} \Rightarrow y \sim Cx \tag{5}$$

So, here

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{f} & 0 \end{pmatrix} \sim \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

It can further be normalized by taking f=1 and represented in terms of Rotation and translation matrices for generalization.

0.2.4 Depth and Disparity Relationship

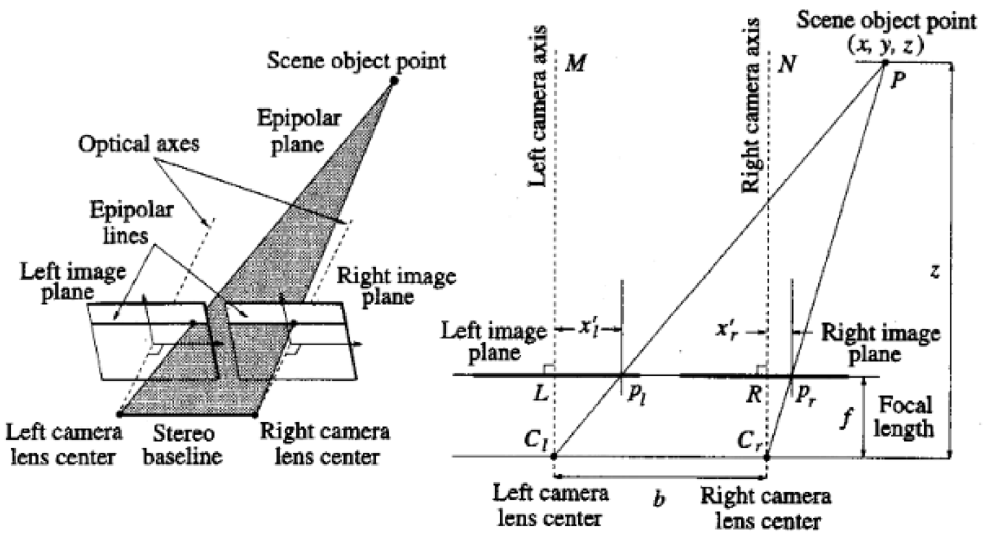


Figure 4: Taken from `http://www.cse.usf.edu/~r1k/MachineVisionBook/MachineVision.files/MachineVision_Chapter11.pdf`

Here in Similar triangles PMC_l and p_lLC_l

$$\frac{x}{z} = \frac{x_l^{'}}{f}$$

And, from similar triangles, PNC_r and p_rRC_r

$$\frac{x-b}{z} = \frac{x_r^{'}}{f}$$

where, f = Focal length, b = baseline(distance between the two camera centers) and z = depth of the object
Now, combining the above two equations we get,

$$z = \frac{bf}{x_l^{'} - x_r^{'}}$$

where,

$$x_l^{'} - x_r^{'} = Disparity$$

A **conjugate pair** is two points in different images that are the projections of the same point in the scene.
Disparity is the distance between points of a conjugate pair when the two images are superimposed.

0.3 Some Useful Codes, Papers and our Conclusions

0.3.1 DeepTAM

Problem: Their paper says that we don't need any depth maps while training but the available code in GitHub(`https://github.com/lmb-freiburg/deeptam`) requires depth maps while training.
This method works not by using geometry but by training deep networks. There are two networks they have used, Tracking (to track the camera motion using Depth maps generated from mapping) and mapping(which maps all the images and creates Depth maps using camera trajectory). And they need some good initialization since it is a cyclic process.

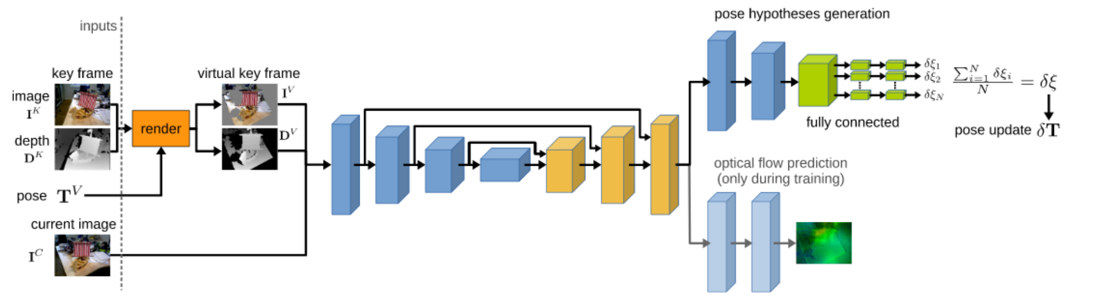


Figure 5: Taken from https://arxiv.org/pdf/1808.01900.pdf. This is then used in process shown below.

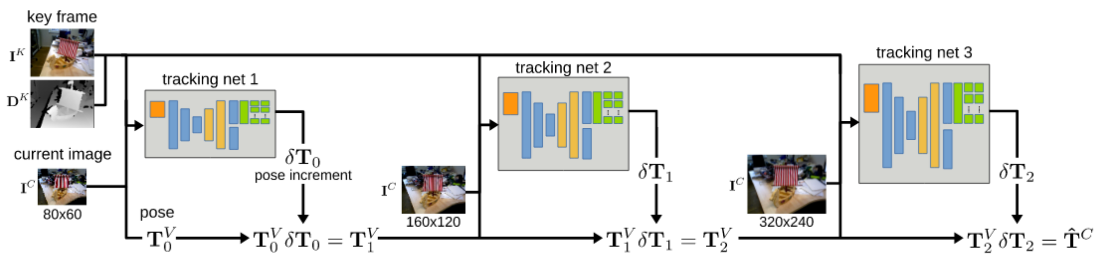


Figure 6: Taken from https://arxiv.org/pdf/1808.01900.pdf.

0.3.2 Demon

It(`https://github.com/lmb-freiburg/demon`) is also supervised learning method based on a CNN network. Their crucial component of the approach is a training loss based on spatial relative differences. The network also estimates surface normal's, optical flow between the images and confidence of the matching. Results were not very good on images take by us, and is slow to be used in real time. Also there is scaling factor issue.

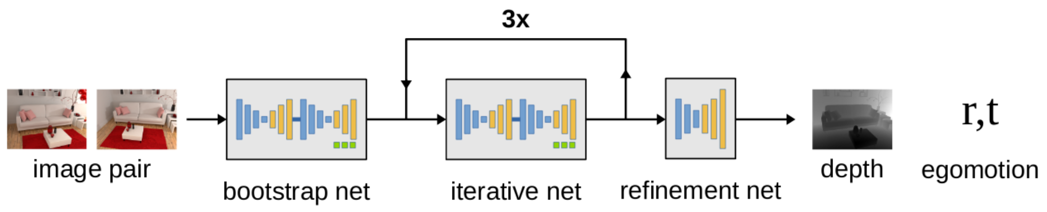


Figure 7: Overview of the architecture. DeMoN takes an image pair as input and predicts the depth map of the first image and the relative pose of the second camera. The refinement network increases the resolution of the final depth map. Taken from https://arxiv.org/pdf/1612.02401.pdf

0.3.3 SfMLearner

It(`https://github.com/tinghuiz/SfMLearner`) is an unsupervised learning method. This method uses single-view depth and multi-view pose networks, with a loss based on warping near by views to the target using the computed depth and pose. The networks are thus coupled by the loss during training, but can be applied independently at test time. This method works on predicting geometry. So, imperfect geometry and/or pose estimation can cheat with reasonable synthesized views for certain types of scenes (e.g., textureless), the same model would fail miserably when presented with another set of scenes with more diverse layout and appearance structures.
Here, they have used single-view depth CNN and a camera pose estimation CNN, which has to be jointly trained, but can be used independently during test-time inference. Also, additionally used explainability prediction network (jointly and simultaneously with the depth and pose networks) to overcome non-modeled factors effect, that outputs a per-pixel soft mask for each target-source pair, indicating the networkâĂŹs belief in where direct view synthesis will be success-fully modeled for each target pixel.. **Assumption**: the scenes we are interested in are mostly rigid, i.e., the scene appearance change across different frames is dominated by the camera motion. The scene is static without moving objects; there is no occlusion/disocclusion be-tween the target view and the source views; the surface is Lam-bertian so that the photo-consistency error is meaningful.

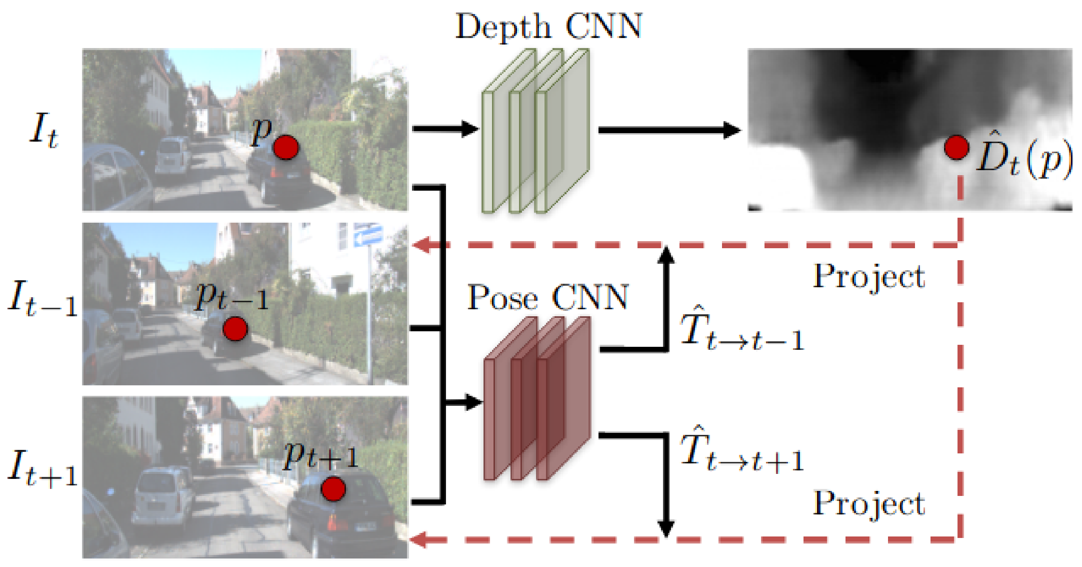
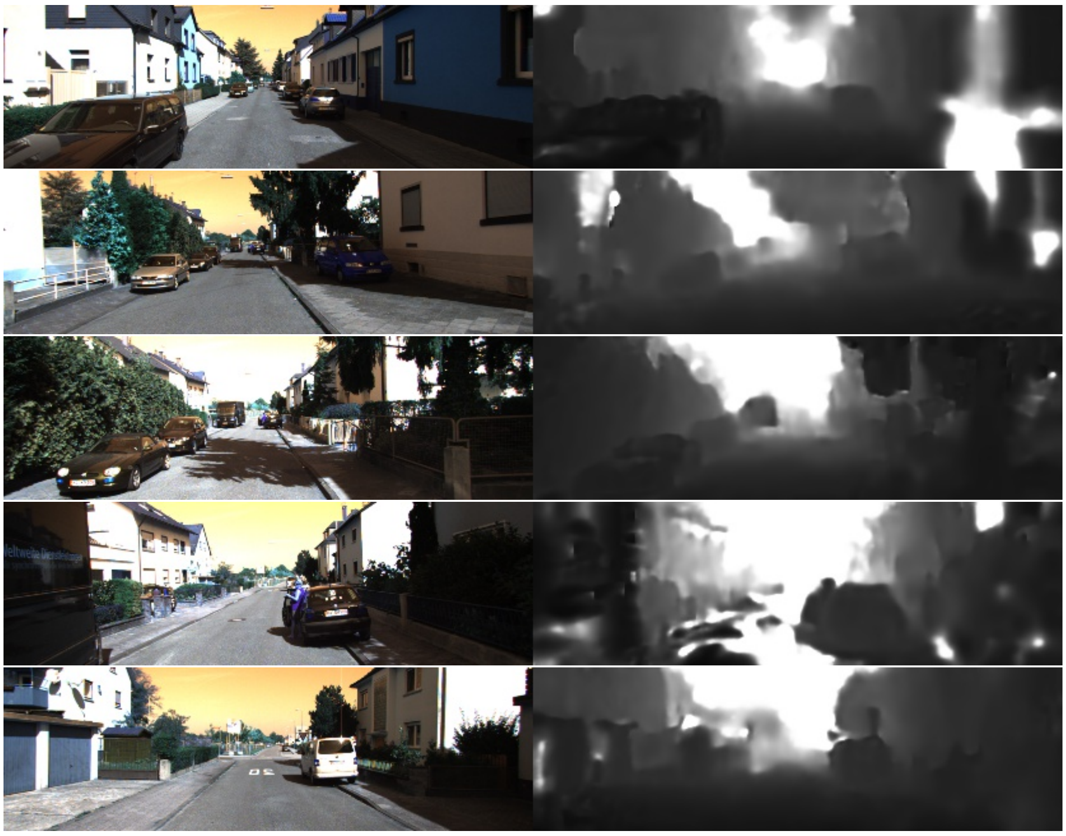


Figure 8: Taken from `https://people.eecs.berkeley.edu/~tinghuiz/projects/SfMLearner/cvpr17_sfm_final.pdf`

Given one input view of a scene, they synthesize a new image of the scene as seen from a different camera pose. They have handled the visibility, non-rigidity and other non-modeled factors with something called explainability mask.
Hence, they have a Loss function which has three terms, first, visual synthesis loss, second, depth smoothness loss and third, explainability along with its regularization loss. Also, they have a very interesting scaling factor that is = median(Ground truth depth)/ median(Predicted Depth).

Network Architecture
* For **single-view depth prediction**, they have adopted the DispNet architecture.
* The input to the **pose estimation** network is the target view(middle image) concatenated with all the source views(left and right image) (along the color channels),and the outputs are the relative poses between the target view and each of the source views. The network consists of 7 stride-2 convolutions followed by a 1x1 convolution with 6(N-1) output channels (corresponding to 3 Euler angles and 3-D translation for each source view).
* The **explainability prediction** network shares the first five feature encoding layers with the pose network,followed by 5 deconvolution layers with multi-scale side predictions. All conv/deconv layers are followed by ReLU except forthe prediction layers with no nonlinear activation. The number of output channels for each prediction layer is 2(N-1), with every two channels normalized by softmax to obtain the explainability prediction for the corresponding source-target pair (the second channel after normalization is $\hat{E}_s$ and used in computing the loss.
Things that I have tried on this:
`https://drive.google.com/open?id=1Mmo6SZBBfueI37h71z4Z3CwmNFdATh2q` Here are the codes available. I tried training it as it is explained, but For image size 416x128(in which the authors have trained), I got nothing but a sunrise kind of visual with whole KITTI data-set, but on over-fitting the model with some selected images, I got the very nice depth maps.
Next one might look into over-fitting the model with larger image size. And then one can trace and attack the problem of model not getting trained. And if works well, one can try to make it more accurate.
Here are some images from over-fitted models which I have trained:



0.3.4 Vid2Depth

This is the successor of SfMLearner. And precursor of Struct2Depth.

0.3.5 Struct2Depth

This is the successor of Vid2Depth. It's results are better than any other paper that I could find as per their claim. Also, they claim it to be working at the speed of 50 frames/sec, hence it is a very good candidate for our purpose(`https://github.com/tensorflow/models/tree/master/research/struct2depth`).
The main idea is to introduce geometric structure in the learning process,by modeling the scene and the individual objects; camera ego-motion and object motions are learned from monocular videos as input. Also, an online refinement method is introduced to adapt learning on the fly to unknown domains. They have modeled motions as SE3 transforms; done by fully differentiable operations aand trained with uncalibrated monocular videos.

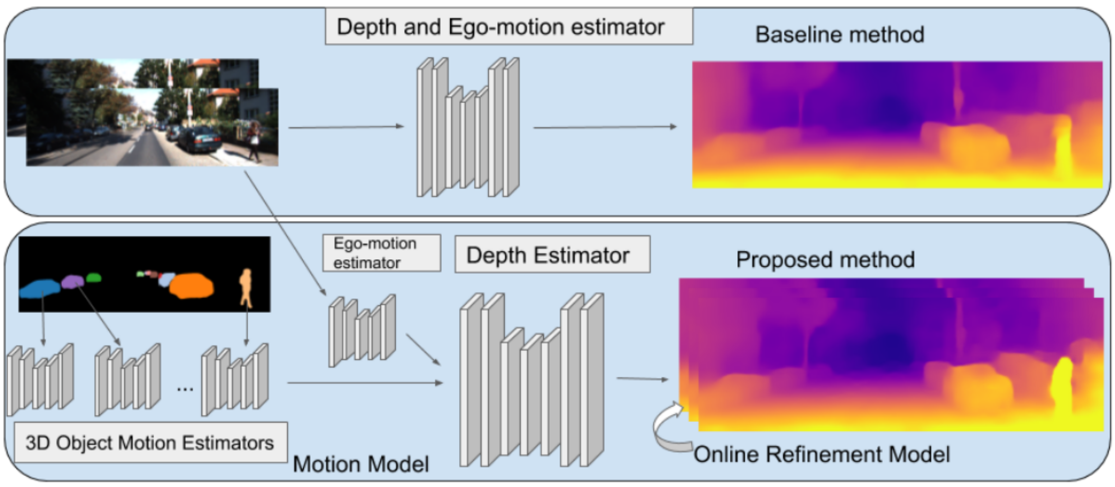
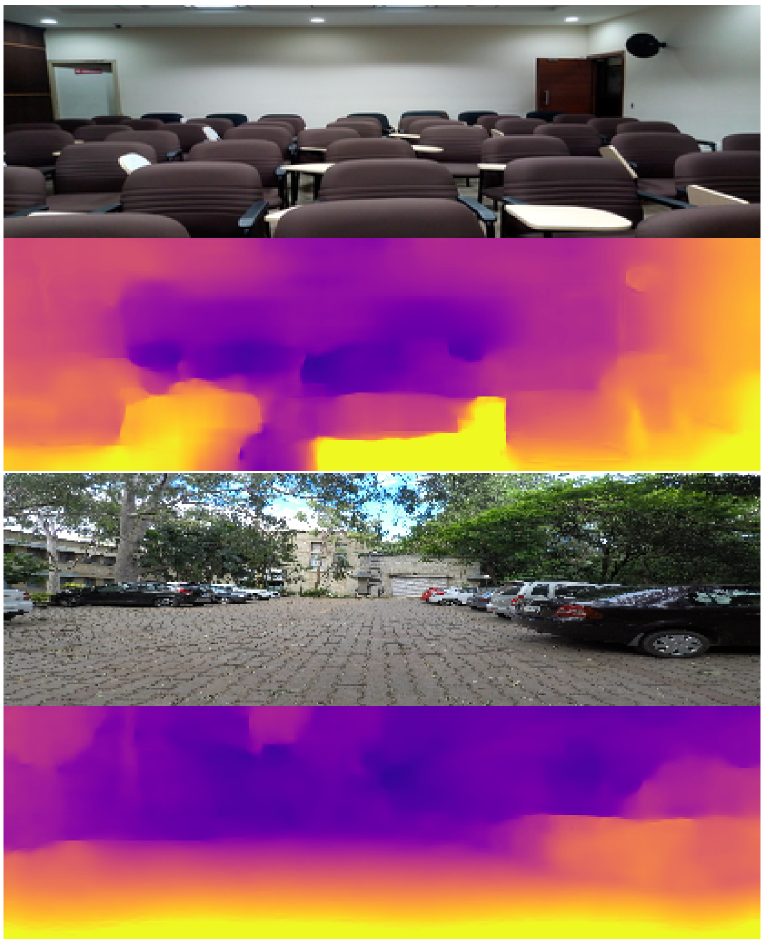


Figure 9: Taken from `https://arxiv.org/pdf/1811.06152.pdf`

Both modeling object motion and online refinement are tangential, but can be used either separately or jointly. And since here also they need to reconstruct image, they have defined a loss function that has a term of reconstruction loss, a depth smoothness loss, SSIM loss for the baseline. They also apply a depth normalization while training.The network learn all constraints (like object size, height and segmentation size in the segmented image) simultaneously without requiring additional inputs. And a loss is trained to learn those constraints too. They have also observed that this additional loss can successfully correct wrong depth estimates when applying it to already trained models, in which case it works by correcting depth for moving objects.
A single-frame depth estimator comes at a cost when running continuous depth estimation on image sequences as consecutive predictions are often misaligned or discontinuous. These are caused by two major issues:
1) Scaling inconsistencies between neighboring frames, since both our and related models have no sense of global scale.
2) Low temporal consistency of depth predictions.
Here are the results of pre-trained models that were available for KITTI dataset on two images taken by us at RBCCPS:

The Depth Estimates were close to the ground truth, but there were an error of more than 2m length.



0.4 Conclusion

Hence, on the basis of results obtained, I conclude that working on SfMLearner might be fruitful, but if that works, one should look into Stuct2Depth, and if that also works, then that can be made more accurate.
Also, I had an idea of using light aberrations to get different images of same object might help in determining depth much more accurately than current methods, because they don't evolve over time. Those pair of images will be of same time.

References

[1] Multiple View Geometry in Computer Vision By Richard Hartley, Andrew Zisserman
[2] On Epipolar Geometry: `https://en.wikipedia.org/wiki/Epipolar_geometry`.