Deep Structure From Motion

Abhishek Mangla, Sheng-Yu Wang, and Aarash Heydari

Abstract— **We modified an existing unsupervised learning framework for the task of monocular depth and camera motion estimation from unstructured video sequences. The baseline model jointly trains on estimating a cameras motion relative to a rigid scene and Depth Estimation using the spatial structure of a target scene. The model synthesizes a novel view of the target scene which seen from a different camera pose and is used a supervisory signal. Our modified approach attaches a Mask R-CNN to segment images into different objects. We incorporate an additional loss function promoting depth gradients around object segments to sharpen our depth maps around object boundaries. Empirical evaluation on the KITTI dataset indicates that our approach performs slightly better than the baseline.**

I. INTRODUCTION

Two of the core problems in computer vision are depth estimation, defined as learning the 3D spatial structure of a scene from images, and egomotion, defined as estimating the 3D motion of a camera within a rigid environment [3]. Being able to estimate depth and egomotion is a given capability for humans and a good example of exercising it is driving: people have a good sense of how far away things are from their car while constantly changing their visions degrees of freedom (such as by moving ones head left or right). Another motivating example is how important visual odometry is for the Mars Exploration Rovers, which must determine their position and orientation in respect to a stream of images in real-time [2]. With the advent of autonomous cars and small robots zooming on college campuses, the ability to perceive depth from images is useful for preventing collisions while enabling smoother modes of operation.

Before the 2000s, approaches to depth estimation and egomotion focused on non-neural network techniques such as carefully designing optimal image processing algorithms. A comparison of the six most competitive algorithms at the time revealed the major differences in approaches: some computed translational velocity first while others computed rotational velocity first, some used numerical optimization methods, some compensated for bias, and others were based on motion parallax versus epipolar constraint [1].

A. Background

In the last two decades however, convolutional neural networks have been getting more interest and better results for particular problems in depth estimation and egomotion [4]. Computer vision algorithms have found success in feeding labeled image sequences into varying neural network architectures that output inferences for camera motion [3]. This is typically done using feature detection to construct an optical flow from two image frames in a sequence generated from either single cameras (monocular visual odometry) or two cameras (stereo VO). Stereo VO provides the system with more information because the image pairs can be used to reduce error and provide additional depth and scale information [5] [6]. For our project, we focused on monocular VO.

A recent paper by Google researchers and Tinghui Zhou at Berkeley illustrates using a fully convolutional architecture akin to DispNet[8] which simultaneously trains two networks - one that infers depth from one single image, and another one that infers camera pose and explainability [7] given multiple views of the scene. Training occurs in an unsupervised fashion, i.e. training does not use ground truth depth maps nor ground truth pose transformations. Instead, the baseline model reconstructs a target view and computes the reconstruction loss as a means of training the network. More specifically, it uses the depth map of a target image, nearby views to the target, and relative pose transformations between nearby views and the target in order to synthesize a reconstruction of the target image. The photometric reconstruction loss is used as the supervisory signal which trains the depth and pose networks.

By training the task of view reconstruction, the network is forced to improve at the intermediate tasks of depth and pose estimation. Even though this is an unsupervised approach, the results are on par with similar supervised approaches as illustrated from example video sequences from the KITTI dataset [13].

B. Problem Statement

The depth maps of the baseline model are often blurry and diffuse, and other methods are known to perform better. For the practical purpose of enabling robots to see and avoid obstacles, we wanted to specifically strengthen the depth maps around objects in the image. Therefore, the goal of our project is to improve the results in Unsupervised Learning of Depth and Ego-Motion from Video to produce sharper depth maps that are more attentive to objects in the image without sacrificing quality. Thus, we introduced an object segmentation network into the model and added a new loss signal to promote depth gradients at object boundaries. We tested different neural net hyperparameters. In some experiments, our new loss term regulated the first order gradient of the depth map, and in others, we regulated the second order gradient. As evaluation metrics of our depth maps, we used four metrics of ”error” relative to ground truth (Absolute Relative Error, Squared Relative Error, Root Mean Squared Error, and Log Root Mean Squared Error) and three metrics of ”accuracy”, defined as percentage of classifications which were epsilon-close to the ground truth for three different values of epsilon.

C. Data Sources

KITTI [13] is a rich benchmark dataset used for many computer vision tasks. It contains short timestamped image sequences of views from a driving car, captured and synchronized at 10 Hz. We used their "synced + rectified" processed dataset, where imaged have been undistorted and where the data frame numbers correspond across sensor streams. There are around 45K image frames in total. All images we are using are in color, come from a 0.5 megapixel camera, and are saved in png format.

II. APPROACH

A. Baseline model

The baseline model consists of two disjoint CNN architecture, with one predicting the depth of an image and the other predicting the relative pose of the camera between image views. Given a target image and other source images from the same scene with slightly different camera pose, the Depth CNN takes the target image as input and performs a single-frame depth estimation. Architecture-wise, the Depth CNN adopts DispNet architecture, which outputs a disparity map at multiple scales given a single-frame input. Note that a disparity map is the inverse of a depth map.

On the other hand, the Pose CNN takes in both target image and its nearby views and estimates the relative poses between the target frames and its source frames. With camera intrinsics given in the data and estimations from both models, one can then calculate projections and synthesize novel views of a target image from the original source image sequence. A graphical representation can be seen in Fig 1, and the more detailed architecture can be seen in Fig 2.

Mathematically, let I_t be the target image, $< I_1, \ldots, I_n > \in S$ denote the source image sequence, and $< \hat{I}_1, \ldots, \hat{I}_n >$ be the view syntheses of the target image with the corresponding source images. The view synthesis loss can be constructed as follows:

$$\mathcal{L}_{vs} = \sum_{<I_1,\ldots,I_n> \in S} \sum_p \hat{E}_s(p) |I_t(p) - \hat{I}_s(p)| \qquad (1)$$

The loss is calculating the ℓ-1 loss over pixels between the target image and its novel view syntheses. Note that $\hat{E}_s$ serves as an explanability mask which is learned to ignore areas that violate the model assumptions. This is important because the naive view synthesis loss is constructed based on three large assumptions. 1) The scenes are static, meaning variation from image to image is due entirely to the motion of the camera. 2) All surfaces are lambertian, meaning the luminous intensity of the surface appears the same regardless of the observer's angle of view. 3) There is no occlusion/disocclusion between the target view and source views. These assumptions are unrealistic and so they present significant limitations. In order to model the limitation of the model, the explainability mask $\hat{E}_s$ is used as a per-pixel soft mask that provides slack to areas that

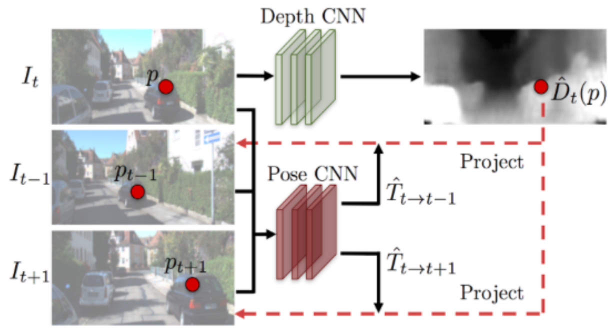


Fig. 1. Sequence of images feeds into two neural nets that help create a depth map

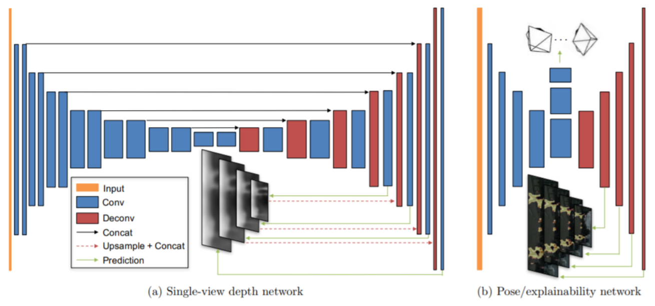

Input
Conv
Deconv
Concat
Upsample + Concat
Prediction
(a) Single-view depth network
(b) Pose/explainability network

Fig. 2. Network architecture for the baseline depth/pose/explainability prediction modules. The width and height of each rectangular block indicates the output channels and the spatial dimension of the feature map at the corresponding layer respectively, and each reduction/increase in size indicates a change by the factor of 2. (a) For single-view depth, the model adopt the DispNet [8] architecture with multi-scale side predictions. The kernel size is 3 for all the layers except for the first 4 conv layers with 7, 7, 5, 5, respectively. The number of output channels for the first conv layer is 32. (b) The pose and explainabilty networks share the first few conv layers, and then branch out to predict 6-DoF relative pose and multi-scale explainability masks, respectively. The number of output channels for the first conv layer is 16, and the kernel size is 3 for all the layers except for the first two conv and the last two deconv/prediction layers where the model use 7, 5, 5, 7, respectively.

the network believes view synthesis will not be successful. To avoid a trivial solution where the explanability mask is zero everywhere, the baseline model applies a regularization term $\mathcal{L}_{reg}$ for the mask, which is a cross-entropy loss with constant label 1 at each pixel location. This regularization enforces non-zero predictions of the explanability mask. In addition, a regularization on the second order gradient of the disparity map $\mathcal{L}_{smooth}$ is applied to encourage a smooth depth estimation. Combining everything and calculating loss on every scale of the disparity map, the total loss for the baseline model will be as follows:

$$\mathcal{L}_{tot} = \sum_l \mathcal{L}^l_{vs} + \frac{\lambda_s}{2^l} \mathcal{L}^l_{smooth} + \lambda_m \sum_S \mathcal{L}_{reg}(\hat{E}^l_s) \qquad (2)$$

λ_s and λ_m are smoothness weight and explanability weight respectively, and both of them are hyperparameters to be tuned.

B. Final model

To encourage sharpness of the depth map on object boundaries, our model adds an additional regularization term on the

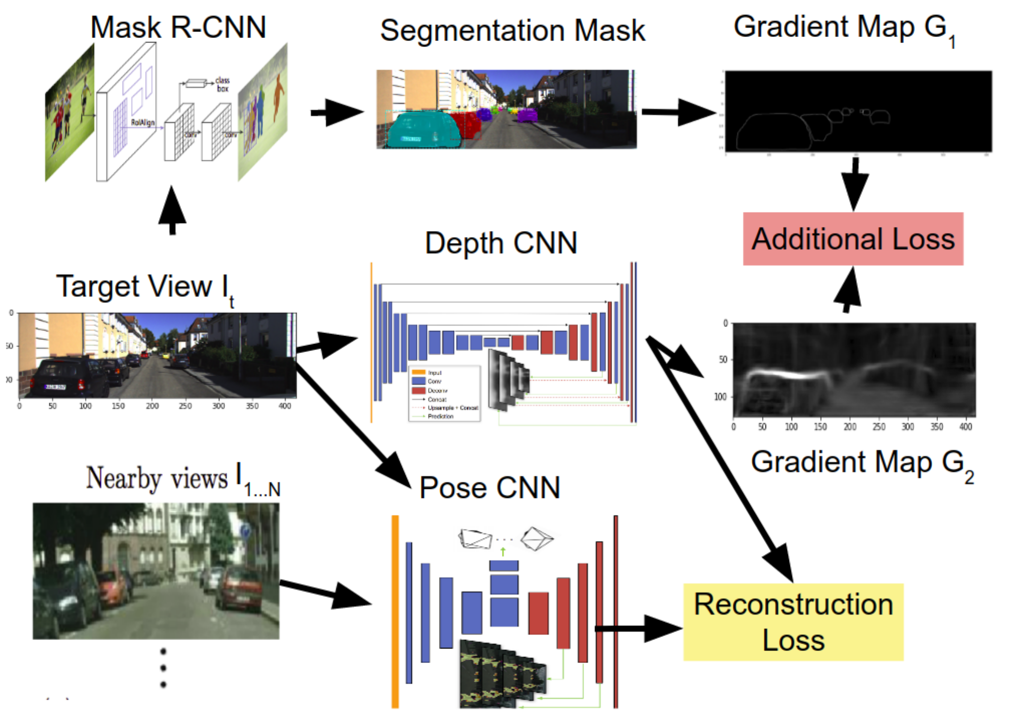


Fig. 3. The final model architecture. The main difference from the baseline is that the input view feeds into the Mask R-CNN which outputs a segmentation mask. This is used to produce a gradient map outlining object boundaries. These boundaries are used to evaluate our additional loss which is combined with the baseline model loss.

baseline model to encourage the gradients of the disparity map on object boundaries. The reason for calculating the gradients of the disparity map is that depths have an inverse relation with respect to the pixel space for planar geometry. Therefore, the gradients of the depth map will diverge to infinity at the vanishing point of the scene, which is definitely not an ideal behavior for our object boundary matching regularization. On the other hand, disparity maps, which are the inverse of the depth maps, have a nice linearity with respect to pixels. The linearity makes the 1st order gradients of the disparity map constant on planes and further makes the 2nd order gradients to be 0 for planes. In a nutshell, taking gradients on disparity maps instead of depth maps prevents huge gradients close to the vanishing horizon while still preserves large gradients on the object boundaries, so the gradients of the disparity map will be relatively larger on the object boundaries given that the depth estimation is correct.

To obtain the object boundary, the target image is passed into a trained Mask R-CNN model to get segmentation mask predictions. The boundary is the gradient intensity of the segmentation masks, which will be further discussed in the data preparation section.

With the object boundary G_1 and the gradient map G_2 of the disparity map, the object boundary matching loss $\mathcal{L}_{seg}$ is formulated as follows:

$$\mathcal{L}_{seg} = \sum_{(x,y)} G_1^2(x,y) \cdot max(0, G_1(x,y) - G_2(x,y)) \quad (3)$$

The loss penalizes weak gradients at object boundaries, since $G_2(x,y)$ gives contribution to the loss when it's smaller than $G_1(x,y)$. On the other hand, the $G_1^2(x,y)$ serves as a mask to prevent loss contribution to pixels where there's no object boundary response. The term is squared for empirical reasons. We experimented both the 1st and 2nd order gradients of disparity map for G_2 to see which one works better.

The new total loss for our final model will be the sum of Eq (2) and Eq (3) weighted by the segmentation weight λ_{seg}.

C. Data Preparation

Given a target image of size 1226x370, we scale it down to 416x128 and then compress it with JPG. We then horizontally stack the target image with all its associated source images as well as the object boundary image generated by the Mask R-CNN. This stacked image is one training sample. Each training sample also comes with camera intrinsics in a text file. This data includes focal points, angles, principle points, and other camera specifics.

To obtain object boundaries, we first pass in the downsized target image into Mask R-CNN to get the object segmentation masks. We then calculate the gradient intensity by

Fig. 4. Compression of a target image.

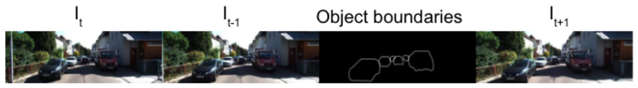


Fig. 5. Single training sample with stacked images along with object boundary image.

applying both horizontal and vertical sobel filter and taking ℓ-2 norm of the two gradient responses. The gradient intensity map will be the object boundaries for the target image.

D. Training Details

We did not train our final model from scratch. Rather, we took the weights of the depth and pose networks trained by the baseline model after approximately 190,000 iterations of training. These weights were made downloadable by Zhou et. al. We using the processed KITTI dataset, which consists of around 40,000 training samples, to fine-tune the model. We fine-tuned those networks for 35,000 more iterations with batch size 4, trying different hyperparameters for the weight of our additional segmentation loss, the weight of the smoothness regularizer, and the learning rate of ADAM. We experimented with restarting ADAM's momentum when we began fine-tuning with our additional loss or letting it keep its momentum from the training of the baseline. In most of our experiments, we used the object segmentation to regulate the first order disparity gradient, but in some we tried regulating the second order disparity gradient instead.

Another detail about our training is that the base model was trained using the explainability mask as described above, but in our fine-tuning we disabled the explainability mask for the sake of simplicity and to lower the computational complexity of each iteration.

It would be interesting for future work to see how results would compare if our final model was trained from scratch using the explainability mask and with a more deeply rigorous hyperparameter search.

III. RESULTS

A. Depth Prediction Performance

We evaluate the depth prediction performances on the 697 images from the test split of the KITTI dataset. Our results are summarized in Fig. 8 and the visual difference between our model and the baseline model and ground truth are displayed in Fig. 7. Only one of our models had better empirical results on the test set than the baseline. It was trained for 35,000 iterations by restarting the ADAM optimizer's momentum, slightly increasing the smoothness regularization weight from 0.5 (baseline) up to 0.75, and using 2.5 as the weight of the new segmentation loss term.

In terms of squared relative error, this model achieved 1.8053 versus the 1.8363 of baseline. The absolute and root mean squared errors were both less than baseline as well. Accuracy on the smallest value of epsilon also improved from 0.717 to 0.743. Visually, our model is similar to baseline yet noticeably better. For example, in the first row of images in Fig. 8, the boundary of the car's depth into the road is better captured by our model than the baseline which was the goal of including object segmentation into the loss.

B. Overfit Model

One surprising but interesting result is our overfit model which used the hyperparameters $\lambda_{seg} = 100$ and $\lambda_s = 0.5$ and used segmentation to regulate the 1st order gradient. It exhibited an extremely jagged and bar-like depth pattern around objects, as seen in Fig 6. The jagged and bar-like depth patterns correspond to large gradient intensities, which indicates that the Depth CNN had learned to maximize the gradients around object boundaries in order to minimize the boundary matching loss. However, the jagged patterns contributes to higher smooth regularization loss and view synthesis loss. In this case, the segmentation weight was too large so that the model overfitted at the cost of violating the smooth depth map constraint, resulting in an extremely unnatural looking depth map. One thing to notice is that the segmentation loss and the smoothness regularization pull the model in different directions, as the segmentation loss favors sudden sharp gradients on objects boundaries, which incurs high 2nd order gradients for both gradient settings, while the smooth loss penalizes high second order gradients over the entire disparity map. This realization motivated us to slightly increase the smoothness loss during our hyperparameter search and to be weary of setting the segmentation weight too large.

IV. TOOLS

We used Tensorflow to modify and implement our additional loss function. Tensorflow was the original framework used by Zhou in the original paper, so it was a natural extension to continue using it. We also used Numpy to do our custom data processing which involved compression and horizontal stacking of 3 images with an object segmented image.

For the image segmentation neural network model, we used Python 3.4, TensorFlow 1.3, and Keras 2.0.8. Keras, which is a high level wrapper over Tensorflow, helped us run and evaluate the Mask R-CNN models on provided weights to validate original source results. We did not have to write any code for this.

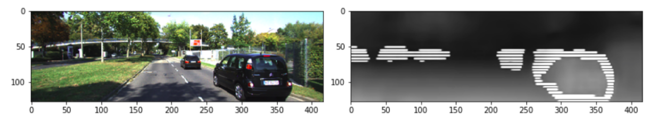


Fig. 6. Results from the overfit model with $\lambda_{seg} = 100$, $\lambda_s = 0.5$

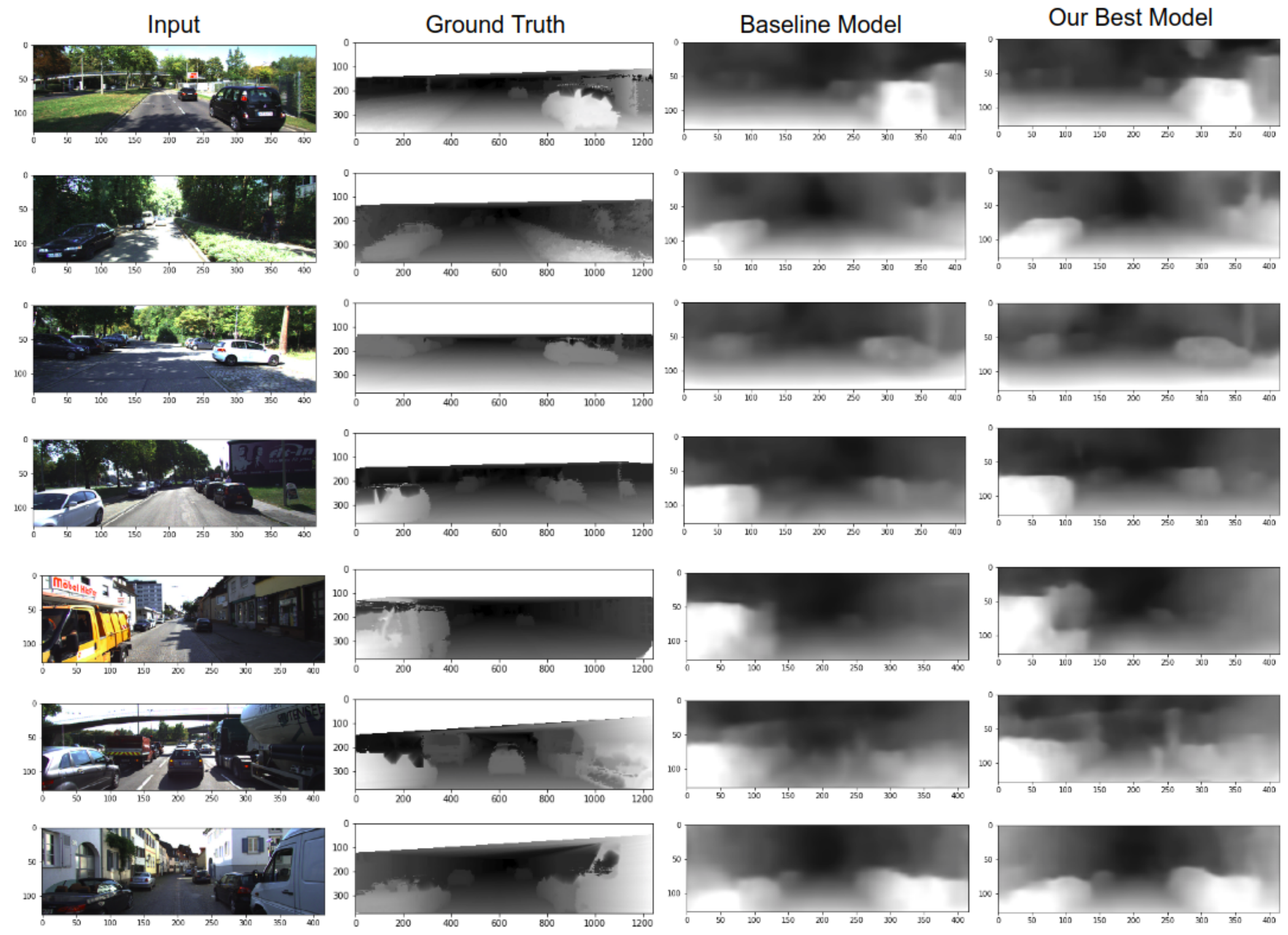


Fig. 7. Comparison of single-view depth estimation between Zhou et al. [7], and ours. The ground-truth depth map is interpolated from sparse measurements for visualization purpose.

V. LESSONS LEARNED

The first lesson we learned was to know and download our data well before beginning the task itself. We underestimated the size of our dataset and the time to download and preprocess took nearly 3 days which was expensive to say the least.

Moreover, depth for planar objects (e.g. ground, walls) in the image is not linear with respect to pixels, and the farther parts of the scene generate larger gradients in the depth map. That is, as parts of the scene are closer to the horizon, the difference in depth across one pixel increases dramatically, which causes issues for calculating the object boundaries using gradients. In order to fix this, we needed to first calculate the disparity map (1 / depth) to linearize the depths, and then calculate the gradients.

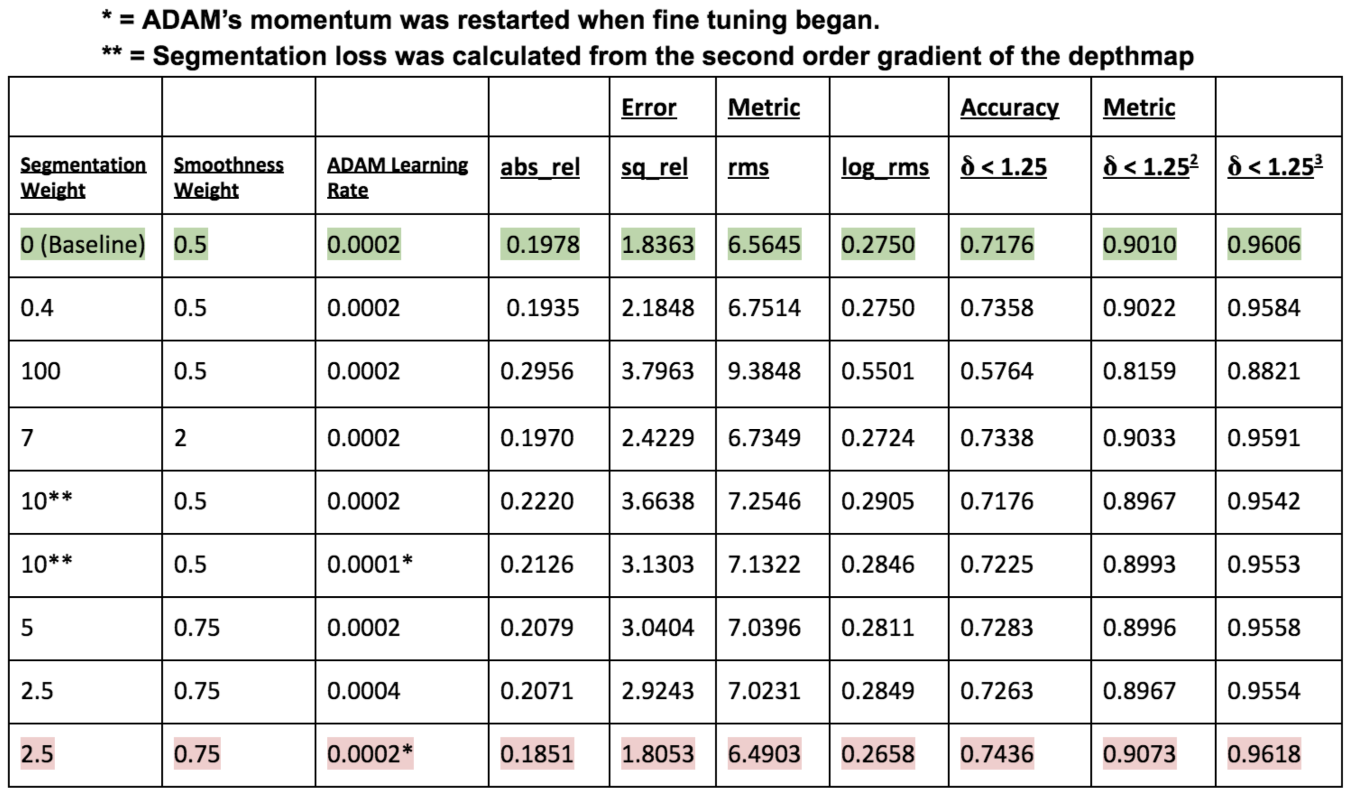
We noticed that the second order gradient smoothness regularization from the baseline model and our additional segmentation loss were pulling the model in different directions. Our segmentation loss wanted the model to learn sudden, sharp changes in depth at object boundaries, but the smoothness regularizer penalized large second order gradients everywhere in the image. Our model which overfit the segmentation (Fig. 6) illustrates why depth maps should generally be smooth, or else they look completely unnatural. The exception is at boundaries of objects, where occlusion causes sudden change in depth. We had this in mind while doing our hyperparameter search.

Finally, the Mask R-CNN does not segment trees, only objects such as pedestrians and cars. We only were able to improve sharpness of depth maps on images that contains objects segmented by mask R-CNN so this means that our performance is limited by how well the R-CNN is trained. This dependency on 3 neural net architectures instead of 2 meant we had to train a lot more to get good results.

VI. TEAM CONTRIBUTIONS

Abhishek Mangla: I helped draft our poster's background and introduction sections, worked on diagrams for the poster and drafted the report. I also helped pinpoint places in code where we would have to make changes to incorporate our new loss function.

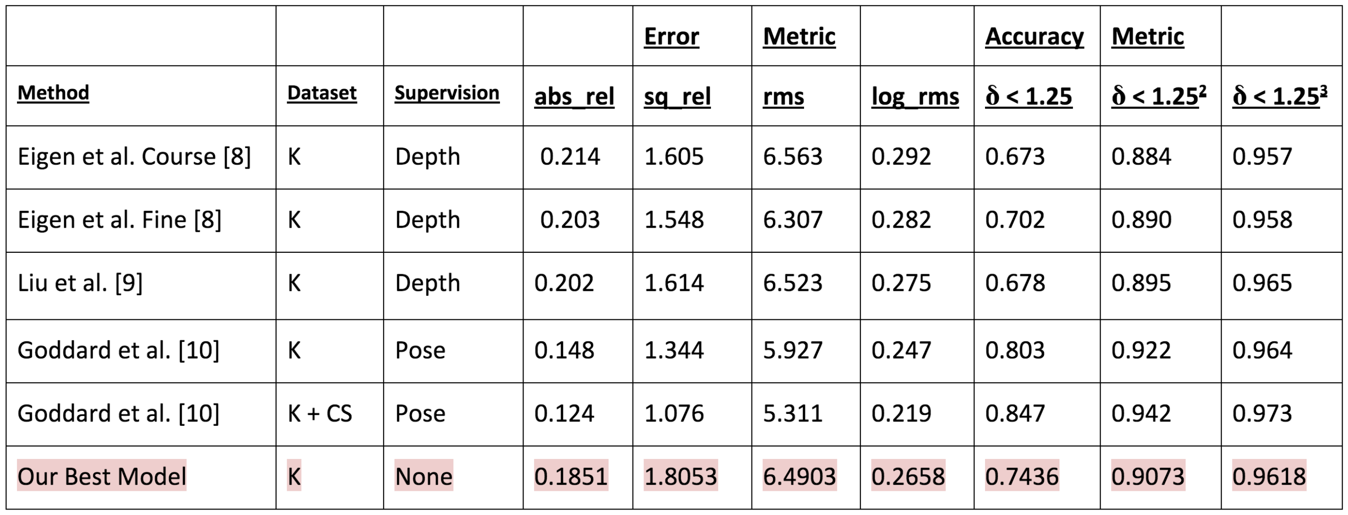
Sheng-Yu Wang: I worked on downloading the KITTI dataset and preprocessing the data. Also, I designed and experimented with different boundary matching loss function


*** = ADAM's momentum was restarted when fine tuning began.**

**** = Segmentation loss was calculated from the second order gradient of the depthmap**

				Error	**Metric**		**Accuracy**	**Metric**	
Segmentation Weight	**Smoothness Weight**	**ADAM Learning Rate**	**abs_rel**	**sq_rel**	**rms**	**log_rms**	**$\delta < 1.25$**	**$\delta < 1.25^2$**	**$\delta < 1.25^3$**
0 (Baseline)	0.5	0.0002	0.1978	1.8363	6.5645	0.2750	0.7176	0.9010	0.9606
0.4	0.5	0.0002	0.1935	2.1848	6.7514	0.2750	0.7358	0.9022	0.9584
100	0.5	0.0002	0.2956	3.7963	9.3848	0.5501	0.5764	0.8159	0.8821
7	2	0.0002	0.1970	2.4229	6.7349	0.2724	0.7338	0.9033	0.9591
10**	0.5	0.0002	0.2220	3.6638	7.2546	0.2905	0.7176	0.8967	0.9542
10**	0.5	0.0001*	0.2126	3.1303	7.1322	0.2846	0.7225	0.8993	0.9553
5	0.75	0.0002	0.2079	3.0404	7.0396	0.2811	0.7283	0.8996	0.9558
2.5	0.75	0.0004	0.2071	2.9243	7.0231	0.2849	0.7263	0.8967	0.9554
2.5	0.75	0.0002*	0.1851	1.8053	6.4903	0.2658	0.7436	0.9073	0.9618

Fig. 8. Summary of the results of our hyperparameter search



				Error	**Metric**		**Accuracy**	**Metric**	
Method	**Dataset**	**Supervision**	**abs_rel**	**sq_rel**	**rms**	**log_rms**	**$\delta < 1.25$**	**$\delta < 1.25^2$**	**$\delta < 1.25^3$**
Eigen et al. Course [8]	K	Depth	0.214	1.605	6.563	0.292	0.673	0.884	0.957
Eigen et al. Fine [8]	K	Depth	0.203	1.548	6.307	0.282	0.702	0.890	0.958
Liu et al. [9]	K	Depth	0.202	1.614	6.523	0.275	0.678	0.895	0.965
Goddard et al. [10]	K	Pose	0.148	1.344	5.927	0.247	0.803	0.922	0.964
Goddard et al. [10]	K + CS	Pose	0.124	1.076	5.311	0.219	0.847	0.942	0.973
Our Best Model	K	None	0.1851	1.8053	6.4903	0.2658	0.7436	0.9073	0.9618

Fig. 9. Summary of results of other supervised approaches. Some approaches supervised the depth estimation, and others supervised the pose estimation. "K" in the dataset column indicates that the KITTI dataset was used, and "CS" refers to the Cityscape dataset [12]. Unsurprisingly, the strongest state-of-the-art supervised approaches outperform our best model, but our performance is still quite competitive.

and Mask R-CNN object boundary retrievals. I helped with model training and reports as well.

Aarash Heydari: I performed training and hyperparameter search for the final model. I substantially wrote and edited the presentations and final report.

REFERENCES

[1] Tian, T.; Tomasi, C.; Heeger, D. (1996). ”Comparison of Approaches to Egomotion Computation”. IEEE Computer Society Conference on Computer Vision and Pattern Recognition: 315.

[2] Maimone, M.; Cheng, Y.; Matthies, L. (2007). ”Two years of Visual Odometry on the Mars Exploration Rovers”. Journal of Field Robotics. 24 (3): 169186.

[3] Irani, M.; Rousso, B.; Peleg S. (June 1994). ”Recovery of Ego-Motion Using Image Stabilization” (PDF). IEEE Computer Society Conference on Computer Vision and Pattern Recognition: 2123. Retrieved 7 June 2010

[4] Vijayanarasimhan, Sudheendra, et al. ”Sfm-net: Learning of structure and motion from video.” arXiv preprint arXiv:1704.07804 (2017).

[5] Tulsiani, Shubham, et al. ”Multi-view supervision for single-view reconstruction via differentiable ray consistency.” CVPR. Vol. 1. No. 2. 2017.

[6] Milella, A.; Siegwart, R. (January 2006). ”Stereo-Based Ego-Motion Estimation Using Pixel Tracking and Iterative Closest Point” (PDF). IEEE International Conference on Computer Vision Systems: 21. Retrieved 7 June 2010.

[7] Zhou, Tinghui, et al. ”Unsupervised learning of depth and ego-motion from video.” CVPR. Vol. 2. No. 6. 2017.

[8] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4040 4048, 2016

[9] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In Advances in Neural Information Processing Systems, 2014.

[10] F. Liu, C. Shen, G. Lin, and I. Reid. Learning depth from single monocular images using deep convolutional neural fields. IEEE transactions on pattern analysis and machine intelligence, 38(10):20242039, 2016.

[11] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In Computer Vision and Pattern Recognition, 2017.

[12] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes dataset for semantic urban scene understanding. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 32133223, 2016.

[13] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pages 33543361. IEEE, 2012.