Limited Implementation of Monodepth2 Monocular Depth Estimation

[redacted]
Dept. of Electrical and Computer Engineering
University of Victoria
Victoria, Canada

[redacted]
Dept. of Electrical and Computer Engineering
University of Victoria
Victoria, Canada

***Abstract* — Dense depth information is desirable in a plethora of different problem spaces. However, current approaches suffer from a variety of limitations. Instead of relying on custom hardware, Monodepth2 proposes a novel use of loss functions regarding photometric reconstruction and dense depth smoothness losses at multiple scales and multiple temporal samples to constrain a self-supervised monocular depth estimation optimization problem. This paper attempts to re-create this algorithm on a new domain of the DrivingStereo dataset, and analyze assumptions and performance related to this novel loss formulation.**

Keywords — depth estimation, disparity map, computer vision, machine learning, monocular, self-supervised

I. Introduction

Many of the today's problem spaces require dense depth data. Applications include autonomous cars, remote mapping, and robotics. The common approaches used today are *lidar*, *radar*, *sonar*, *stereoscopic*, and *structured light*. *Lidar* rapidly fires laser pulses onto the environment and measures the time of return for each pulse, calculating distance using time-of-flight. The drawback is that it requires expensive hardware, has moving parts within the sensor apparatus, and cannot work in harsh environments such as inclement weather where the light beams may be degraded or blocked. *Radar* also measures time-of-flight but uses radio waves instead of light. Because radio wavelengths are almost 3 orders of magnitude greater than lidar pulses, radar fails to resolve smaller objects and thus suffers from lower resolution. It is also difficult to detect static objects, especially in vehicle environments, where ground clutter may introduce false detections. *Sonar* and *ultrasonic* systems bounce low- and high-frequency sound waves, respectively, instead of electromagnetic waves off objects. However, sonar only works underwater, and ultrasonic lacks the range necessary for most applications. *Stereoscopic* approaches use two camera positions to calculate depth but require careful calibration of sensitive relative camera positions. Finally, *structured light* cameras project a known pattern onto a surface and calculate depth from the deviations in the pattern [1]. However, it is difficult to project patterns of sufficient brightness to illuminate objects far away, or in bright environments.

Another approach based on images can be used to augment or entirely replace other methods, leading to increased robustness and accuracy. These methods can be categorized into two forms. One approach, called *monocular*, uses a single camera perspective as input data. While it may seem difficult to estimate depth based on a single image and without triangulation or actively probing the environment, there are multiple elements within the image that can help estimate distance. For example, humans are able to estimate depth with one eye by using references within an image, such as perspective distortion, relative sizes of objects, occlusion of objects, sizes of known objects, and other situational cues. These prior knowledge and situational hints enable the estimation of depth given a new scene. Monocular vision has an additional benefit of requiring only a single camera, rather than a secondary camera that may require precise alignment and mounting for accurate triangulation.

The other image-based method uses a pair of stereo images as input. This consists of two images captured with a known distance between the sources. By correlating objects between these two frames, triangulation can be used to estimate depth. However, this approach suffers from the main problems of occlusion of objects, which can lead to discrepancies between the two images, potentially leading to unmatched pixels and thus discontinuities in the smoothness of the depth map.

Both methods can be improved with an additional source of information from the movement of the camera itself. If frames are captured temporally, this time delta can be used to enforce consistency between successive depth estimates. This can be achieved by projecting a depth map onto past and future viewpoints and comparing them.

Increased investments mainly from the mobile phone industry has greatly improved camera performance and reduced their cost. Cameras are much cheaper than other depth sensing hardware and capture high resolution and high dynamic range images. Thus, it is desirable to build a depth estimation system on this technology. Many present state-of-the-art techniques rely on machine learning to solve this complex problem. These deep neural networks require an expansive and varied training dataset to produce better results. To train on these datasets, accurate *ground truth* information is required, which may be cumbersome and expensive. For example, collecting ground truth data for depth estimation may require expensive lidar hardware to achieve the necessary resolution.

The ability to formulate an understanding of a scene's shape, independent of variations in the image, is important especially for autonomous cars, where dense depth data can augment or reduce the number of lidar units required. This also has applications in monocular or stereoscopic synthetic depth of field in computational photography, synthetic object insertion in computer graphics, and many more [6].

This paper proposes a monocular self-supervised training approach, which only requires stereo pairs from videos as input. This data is much easier to collect. The proposed model attempts to predict the disparity map from a single image. It uses this disparity map to project this image onto the corresponding image of the stereo pair. The model attempts to minimize any errors from reconstructing the image.

II. Literary Review

Monocular depth estimation is an active field of research, with continuing innovations in reducing prediction artifacts and increasing resolution. We focus on analyzing work in

monocular and stereo self-supervised training without ground truth dense depth data.

One implementation is the Deep3D network [2], which is a fully automatic 2D-to-3D conversion algorithm. Its input is 2D images or videos, and its output is a 3D stereo image pair. The system aims to minimize the right image's pixel-wise reconstruction error, given the left view. The intermediate representation of the network is a probabilistic disparity map which is fed into a differentiable depth-based rendering layer that calculates the right view. Since the network generates all possible disparities for each pixel, and a corresponding probability distribution, increasing the number of candidate disparity values causes a corresponding increase in memory consumption. This limits the scalability of their algorithm to handle higher resolutions.

Another system, Monodepth [3], is a state-of-the-art monocular depth estimation network. Many attempts before Monodepth treated depth estimation as a supervised problem. However, obtaining adequate ground truth data is difficult. Monodepth proposes a novel method that replaces the need for ground truth dense depth data with a binocular stereo footage during training, greatly simplifying the data required to train the network. This enables the use of large image collections from driving environments where corresponding pixel depths are not available.

Monodepth uses a fully convolutional model that takes a rectified stereo image pair as input and predicts pixel-level disparity. The authors specifically outline the following contributions: a novel training loss and network architecture, enhanced network performance and evaluation metrics, and model generalization across three different datasets.

The goal of the network is to predict a per-pixel scene depth. The authors rely on image reconstruction to calculate the loss of the output, rather than comparisons to ground truth per-pixel depth data. If the network is given a calibrated stereo pair of images and it is successful in reconstructing one image from the other, then the system must have learned something about the 3D environment. The network predicts a disparity map that allows the other image to be reconstructed by modifying the input image. To improve results, the network predicts both disparities from a single image (i.e., left-to-right and right-to-left from a single left input). These disparities must be consistent with each other.

Monodepth uses three terms within its loss function: *appearance matching loss*, which prefers if the reconstructed image is visually similar to the corresponding image of the pair; *disparity smoothness loss*, which prefers smooth disparities, and *left-right disparity consistency loss*, which measures the consistency of the left-right disparities. In the authors' implementation, only the left image is used as input to the network.

The training dataset was the KITTI dataset [4] and the source code is available on GitHub [5]. The model is state-of-the-art and generalizes well to multiple datasets. However, Monodepth has difficulty at occlusion boundaries where pixels are not visible on both left and right images. Additionally, there are inconsistent depth predictions for transparent or specular surfaces.

In the following sections, we will outline the specifics of our source paper's approach, network architecture, dataset, and training loss. We describe any deviations of our implementation from the source. We then evaluate the results of our network. Finally, we conclude with our findings and recommendations.

III. Proposed Approach

This section first outlines the main innovations proposed by the Monodepth2 paper. We then explore in detail the network architecture, the theoretical basis behind the training process, and details on our dataset and training loss functions.

A. Innovations of Source Paper

Monodepth2 [6] builds on the work from Monodepth [2] by improving the handling of occlusions and camera motion assumptions. Specifically, it improves in handling occlusions, reducing visual artifacts, and ignoring pixels that do not conform to camera motion assumptions. The paper uses self-supervised training and proposes three architectural and loss function improvements that improve depth estimation.

The first is an appearance matching loss that improves the problem of occluded pixels. The authors propose a per-pixel minimum reprojection, which deals with occlusions and disocclusions that cause pixels to be inconsistent between frames. Instead of matching these occluded pixels, the paper's method only matches pixels to views in which they are visible. This improves the sharpness of the results.

The second is a 'auto-masking' method used to ignore pixels where no relative camera motion is observed. The rigid scene assumption from previous papers, where objects are mostly assumed to be stationary between frames, is inadequate. Between frames, objects may move, which require motion segmentation masks to decompose the world into rigid and non-rigid components.

The third is a multi-scale appearance matching loss that works on the original image resolution, which reduces the amount of depth artifacts. This full-resolution multi-scale module upsamples depth predictions and calculates losses at the original input resolution. This reduces the number of artifacts in the depth map.

Monodepth2 is trained in both monocular and stereo self-supervised depth estimation using the KITTI [4] and Make3D [7] dataset. It is implemented in PyTorch [8] and is available open source on GitHub [9]. It is state-of-the-art compared to similar leading algorithms in all test evaluation metrics.

We propose two improvements in implementation and training sets: (1) a stereo training method implemented solely in TensorFlow 1.13.1 [10], which improves accessibility, and (2) a change to the DrivingStereo dataset [11], up to 25 times larger than previously used datasets, to improve generalization and depth prediction smoothness. A larger dataset has previously been shown to improve results of models compared to those trained on smaller datasets [11].

B. Network Architecture

The architecture of the *depth* network is a standard, fully convolutional U-Net [12] which uses a colour image as input and generates a disparity map as output. This is an encoder-decoder network with skip connections, which allow both deep abstract features and local information to be retained. The paper starts training from weights pretrained on ImageNet, which improved their accuracy when compared to training from scratch.

Their implementation covers both monocular and stereo training modes. This is a self-supervised depth estimation training method which requires no ground truth depth data to perform estimation. The network consists of either a ResNet50 or ResNet18 backbone [13]. To limit the scope of our project and reduce training time, we only investigated the ResNet18 backbone. Their network takes a single input colour image and outputs a same size disparity map. We also limited our investigations to the stereo training mode, with no supervision.

Assuming similar resource requirements to the original implementation in PyTorch [8] we expect a required GPU memory of 6 GB for training. To further limit the scope of this project, we will also only train the 640x192 pixel scale input Monodepth2 algorithm with pre-training using ImageNet. This input image size limit also allows for easier use of our chosen dataset for training and evaluation. We trained for 20 epochs at a learning rate of 1e-3 for the first 15, then 1e-4 for the final 5. Training on our Nvidia GeForce GTX 1080 Ti GPU [14] required 9 hours per epoch with a batch size of 12.

A secondary *pose* network predicts the camera's position and orientation changes from temporally adjacent frames. This is necessary for monocular video, as the egomotion between temporal image pairs is used to enforce consistency of the disparity prediction. We only investigate the depth network.

Both these networks are illustrated below in Figure 1.

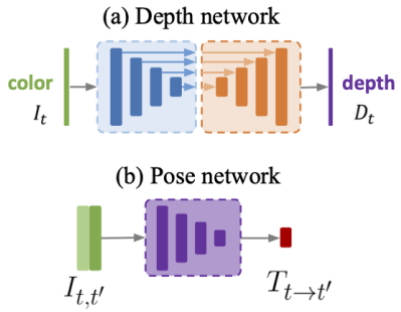

Figure 1: Network architecture [6]

C. Self-Supervised Training

The basis for the learning process is a novel view-synthesis problem, where a network attempts to predict the view of a target image by using a *different* image's vantage point. If this network is constrained to synthesize this transformation using an intermediate disparity variable, this can be used to extract depth. This disparity is the amount of shift in a pixel to produce the corresponding pixel from the other camera's perspective. This problem is ill-posed because there are a large number of incorrect depths assigned to each pixel that can still maintain consistency in the reconstructed image. This can be reduced by enforcing depth map smoothness and calculating photo-consistency.

Given a left (I^l) and right (I^r) rectified stereo image pair, the network predicts the disparity d^r or d^l such that applying this to the left image would reconstruct the right image. That is, the reconstructed image is represented by $\widetilde{I^r} = I^l(d^r)$ [3]. Using a similar process, we can instead use $\widetilde{I^l} = I^r(d^l)$. Only one of the two perspectives is fed into the network. This disparity can easily be used to calculate depth $\hat{d}$ using the equation $\hat{d} = bf/d$, where b is the baseline distance between cameras, and f is the camera focal length.

The disparity direction of movement is intrinsic to the chosen perspective. For example, the disparity from the left image to the right image has the implicit direction of *right*.

D. Dataset

We plan to implement the Stereo training method solely in TensorFlow 1.13.1 [10]. Mono and Mono + Stereo training modalities will not be implemented to limit the scope of the project. We also plan to use the DrivingStereo [11] dataset to train our model. The desired effect of the larger dataset is to improve evaluation results on the originally trained and evaluated on the evaluated KITTI eigen split [4].

The data generator for the images consisted of a 50% chance of each of the following set of operations that are then applied to the input image: flip horizontal, brightness change by up to 0.2, change saturation by up to 0.2, change hue by up to 0.2. To limit our scope, we did not implement the horizontal flips.

E. Training Loss

The authors use a combination of two losses to train the model: photometric reprojection loss, and edge-aware smoothness.

1) Photometric Reprojection Loss

Photometric reprojection loss measures the error from using the disparity information to reconstruct an image compared to the image itself. It is implemented by using the pixel-wise disparity map to calculate the amount of predicted travel for the corresponding pixel in the other image. This forms a matrix of indices pointing to the pixels in the right image, rounding the indices as necessary. We then collect all of the referenced points in the right image and assign a weight based on how close the disparity value referenced each of the right image values.

The relative pose of the source view is denoted $I_{t'}$. This is with respect to the target pose, I_t. We then have the following equation for photometric reprojection error L_p:

$$L_p = \sum_{t'} pe(I_t, I_{t'\to t})$$

$$\text{and } I_{t'\to t} = I_{t'}\langle proj(D_t, T_{t\to t'}, K)\rangle$$

where pe represents the photometric reconstruction error measured as the L1 pixel-wise difference, $proj()$ generates 2D coordinates of the projected depths D_t in $I_{t'}$, $\langle\,\rangle$ is the sampling operator, $T_{t\to t'}$ refers to the relative pose of the source view with respect to the target image, and K represents pre-computed camera intrinsics that are assumed to be constant between the two views.

We use bilinear sampling on the reconstructed images and compare them using both L1 pixelwise difference and also SSIM [15]. This gives our final photometric error function pe:

$$pe(I_a, I_b) = \frac{\alpha}{2}\big(1 - SSIM(I_a, I_b)\big) + (1-\alpha)\|I_a - I_b\|_1$$

where $\alpha = 0.85$ emphasizes SSIM metric in the loss calculation.

The authors propose three enhancements to the photometric reprojection loss: per-pixel minimum reprojection loss, auto-masking stationary pixels, and multi-scale estimation.

The first, a *per-pixel minimum reprojection loss*, attempts to counter the detrimental effects of occluded pixels. In these instances, even if the depth prediction is correct, it is very likely that the source and target pixel colour will not match, causing a high photometric error penalty. Instead of the existing methods which average the reprojection error into each available source image, the minimum photometric error is taken across all source images. Thus, the final per-pixel photometric loss is:

$$L_p = \min_{t'} pe(I_t, I_{t'\to t})$$

This has the effect of reducing artifacts near image borders and improving the sharpness near occlusion boundaries. This is shown below in Figure 2.

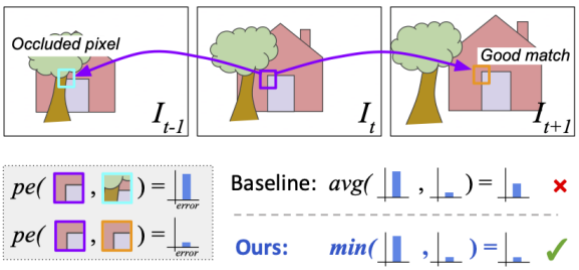

Figure 2: Per-pixel minimum reprojection [6]

The second, an *auto-masking stationary pixel* method, improves performance of the network when the assumption of a moving camera and static scene breaks down. These situations can include instances where the camera is stationary, or when object motion is present in the scene. This can affect the disparity by introducing discontinuities of infinite depth when objects are moving during training. Thus, the authors propose a simple auto-masking function which eliminates invariant pixels between frames. Thus, objects moving at the same velocity and frames where the camera is stationary are ignored. Specifically, the binary masking function $\mu \in \{0,1\}$ is calculated by the following equation:

$$\mu = \left[\min_{t'} pe(I_t, I_{t'\to t}) < \min_{t'} pe(I_t, I_{t'})\right]$$

where [ ] is the Iverson bracket. Thus, the only pixels included in the loss are those where the reprojection error of the warped image $I_{t'\to t}$ results in a lower score than that of the unmodified original source image $I_{t'}$. This is easily implemented by comparing per-pixel reprojection loss of the warped image with the per-pixel reprojection loss of the unwarped image, i.e., a direct comparison of the left-right image pair.

Finally, the third, a *multi-scale estimation* method, improves performance of the model by reducing the effect of the gradient locality of the bilinear sampler. The loss is typically evaluated at each scale of the decoder. However, previous works have shown that this creates 'holes' in large low-texture regions from the downscaled depth maps [6]. Since holes in depth at low resolutions result in ambiguity as to which pixel in the full-resolution reprojected image is incorrect, this results in the network not being penalized enough when it predicts incorrect depths in those layers. Instead, the authors upsample the lower resolution depth maps first, before calculating the photometric error. This forces the depth maps from all resolution scales to work together to improve the reconstruction of the high-resolution input target image. This is shown below in Figure 3.

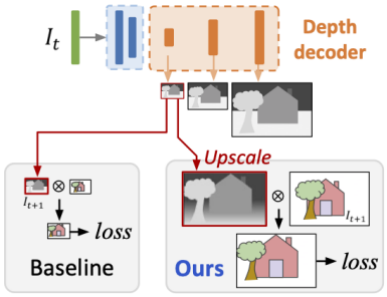

Figure 3: Multi-scale estimation [6]

The authors calculate these losses using frames temporally adjacent to I_t as well. Thus, the image $I_{t'} \in \{I_{t-1}, I_t, I_{t+1}\}$. This improves the probability that occluded regions may be revealed with a different camera spatial location. The minimum reprojection loss across these images is taken. In our implementation, we only use the image immediately before the current image, i.e. $I_{t'} \in \{I_{t-1}, I_t\}$ because of dataset limitations. We also only perform this calculation with L1 loss for masking.

Instead of weighting SSIM loss greater, we use the mean of the pixelwise losses, to simplify the implementation. This is taken as the loss for each scale level. We then take the mean of each scale level. In summary, we perform the following calculations:

1. Calculate the L1 loss pixelwise
2. Calculate SSIM only for current left and right images
3. Keep the minimum of the I_{t-1} and I_t images
4. Mask using L2 distance between the direct comparison of the left and right images without warping
5. Calculate the mean of the L1 and single SSIM loss value, and scale by their corresponding weights
6. Repeat steps 1 to 5 for all scales, and sum into one final photometric loss

In step 5, there is ambiguity as to whether they use the full image SSIM loss or neighbourhood difference. Thus, our loss function implementation may perform worse than theirs, as we chose to implement the global image SSIM difference.

2) Edge-Aware Smoothness Loss

The edge-aware smoothness loss encourages disparities to be locally smooth. Since discontinuities are easily seen in image gradients, we calculate this loss using the following equation:

$$L_s = |\partial_x d_t^*| e^{-|\partial_x I_t|} + |\partial_y d_t^*| e^{-|\partial_y I_t|}$$

where $d_t^* = d_t / \overline{d_t}$ represents the mean-normalized inverse depth, which penalizes the network from successively estimating smaller and smaller depths. In our implementation, this term was overlooked, and likely contributed to the results.

To implement this function, the left image I_t is successively convolved with a Gaussian kernel to the match

the scale level. Next, the gradients of the disparity are calculated. Finally, the per-pixel edge-aware weight is calculated using the left image gradient.

3) Final Training Loss Calculation

The final training loss is calculated as

$$L = \mu L_p + \lambda L_s$$

where L_p is the masked photometric loss and L_s is the per-pixel smoothness loss. L is averaged over each pixel, scale, and batch.

IV. Evaluation

This implementation was trained and evaluated using the DrivingStereo dataset [11]. This dataset is grouped into days in which data was captured. These sets are unbalanced in size, but to create a strong dataset we split the data along the days where data was captured. This decision was also supported by our need for sequential frames to be grouped. Further splitting the data is this manner makes each set more independent, giving our testing results stronger predictability of actual generalization for the trained model. The dataset was split into 10,030 test, 28,231 validation and 136,176 train image pairs.

Due to our constraint on the data being temporally local, the data loader rejected frames which did not have a prior and post frame neighbor within 700ms based on the timestamps provided on the images. This brought the yield of images down to 80% over the entire dataset. In general, the forced 1.4 Hz sample rate of our dataset was worse than the 10 Hz sample rate of the source paper's KITTI dataset [4]. Since minimum reprojection loss assumes that predictions of disparity on occluded objects – which result in the failure to recreate that portion of the image – may be compensated over temporally adjacent frames, it is important that the consecutive samples are not too far apart in time. Otherwise, the entire image changes too much between frames. This was expected to lead to poorer performance due to this time scale.

In this report two major experiments were performed: training both a model with SSIM [15] and a model without SSIM, using only L1 distance. Both experiments resulted in nearly identical models, as a local minimum in the smoothness loss function was discovered by the model. With regards to the L1-only trained model, it was noted that the model only estimates a flat value, specifically zero for the full-scale model output and values on the order of 1e-6 otherwise. This can be seen in Figure 4. Then the edges of the images hold a constant disparity, either positive or negative, to compensate for the right image being offset from the left image, and abusing wrapping found in the flattening function used in the calculation of photometric reprojection error.

Then the edges of the images hold a constant disparity positive or negative to compensate for the right image being offset from the left image, and abusing wrapping found in photometric reprojection error.

These local minima of small estimated depths is likely a result of erroneously omitting the multiplication of the inverse of the mean normalized disparity as described in the source paper [6]. This leads to any flat small region to have near zero loss for smoothness, and any deviation from this will lead to the smoothness loss increasing significantly. This is shown in the training of the L1 model, where for a small period the photometric error loss decreases, but is associated with a large spike in smoothness loss, leading to the model finding the same minima again (see Figure 6). It can be noted that the bars for compensating for movement of the camera frame are larger for lower sampled outputs of the network. This is due to the Gaussian blur applied to the edge aware weights generated from the gradients of the left source image, where regions with lower gradients in the left image are weighted less. So, the blur makes the overall gradients at lower scales and therefore the weights lower, incentivizing the model to make the bars to improve the photometric reconstruction error. Blurring was increased at lower scales to compensate for the up-sampling done to the estimated disparity at these scales.

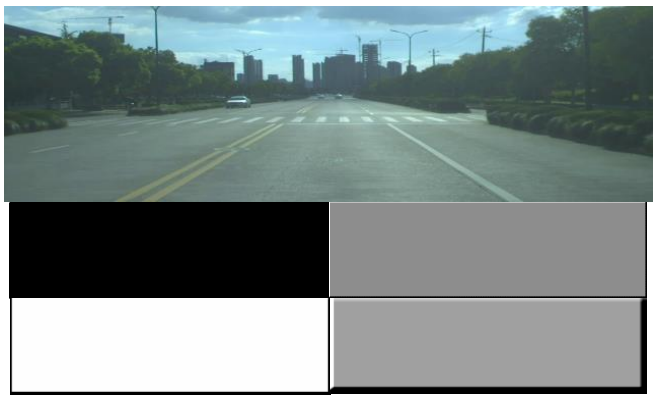

Figure 4: From top to bottom and left to right, source image, full scale estimation, ½ downscale estimation, ¼ downscale estimation, 1/8 downscale estimation, all for L1 only trained model. Darker regions imply lower disparity, larger depth, images are normalized for display

Figure 5: From top to bottom and left to right, source image, full scale estimation, ½ downscale estimation, ¼ downscale estimation, 1/8 downscale estimation, SSIM and L1 trained model. Darker regions imply lower disparity, larger depth, images are normalized for display.

Regarding the final trained model for the SSIM and L1 photometric loss, it was noted that the highest scale and lowest scale outputs were consistent with the L1 only model the intermediate layers did show some level of representation of actual image disparity, and are shown in Figure 5. These representations are partial and do not fully represent accurate depth but show that some features of the input image are not being lost. This shows that the SSIM loss appeared to be more robust to the zero-biased smoothness loss. This is qualitatively consistent with the findings of the original paper where SSIM modified training was more robust and gave better results.

For completeness we will now give the quantitative results of the trained models on the pre-allocated test set. We evaluated our models by taking their output on the test set and tracking the same loss functions used for training and also evaluating the highest scale predictions against the ground truth data. In Table 1 we show the smoothness loss is much lower than the reprojection for both the model with and without SSIM. With regards to the ARD (Absolute Relative Difference) and SQ Rel (Squared Relative Distance), the trained models perform significantly worse; this is due to the continuous estimation of near zero and the DrivingStereo dataset [11] providing a sparse disparity ground truth where all zeros are not to be used in evaluation. As we are using the top scale output of our models, the ARD is approximately 1 for both models and the SQ Rel gives the mean ground truth disparity. Both are the worst possible outputs of the network. This is due to the top output of the model learning to output all zeros as shown and discussed previously.

Table 1: Results of original paper on KITTI Eigen split and this implementation on the DrivingStereo test set.

Metric	SSIM + L1	L1 only	Original [3]
Total Loss	1.70706	1.70683	N/A
Reprojection Loss	1.70706	1.70683	N/A
Smoothness Loss	0.00132	0.00056	N/A
Reprojection Loss L1	0.05043	0.04861	N/A
Total Loss L1	0.05043	0.04862	N/A
ARD [11]	0.99999	0.99999	0.115
SQ Rel	23.3151	23.3151	0.903

V. Conclusion

Within the monocular stereo depth estimation domain, an unsupervised learning method which uses provided stereo image pairs over time is desirable as it allows for easy data collection and annotation. The Monodepth2 [6] method is effective in developing a novel loss function which constrains the training of the network using smoothness and photometric reconstruction losses across multiple scale outputs of the network and taking the minimum over stereo samples to reduce the effects of occlusion.

In our implementation, it was noted that no real training was able to be performed. In the smoothness loss function, a term that constrained the model from estimating successively smaller depths was accidentally omitted, which created a simple local minimum that the model exploited. As a result, the output of both networks at full scale was all zeros, causing the trained algorithm to perform poorly with an ARD of approximately 1 and a SQ Rel of 23.3151. Since the output was all zeros, this gave the mean of the disparities. Both our SSIM/L1 and L1 only models converged to this same local minimum, implying that this constraint of the loss function is independent of the other portions of the loss function in creating this minimum. Some support of claims from the original paper [6] showed partially true as the SSIM/L1 model produced some disparity features shown on the 1/2 and 1/4 scale outputs. Further investigations should be continued with the inverse of the mean normalized depths term added into the smoothness loss to allow for a more comprehensive evaluation of the entire loss function.

References

[1] Achuta Kadambi, Ayush Bhandari, and Ramesh Raskar. *Computer Vision and Machine Learning with RGB-D Sensors, 1st ed.* Switzerland: Springer International Publishing, 2014, pp. 3-4

[2] J. Xie, R. Girshick, and A. Farhadi, "Deep3D: Fully Automatic 2D-to-3D Video Conversion with Deep Convolutional Neural Networks," *Computer Vision – ECCV 2016 Lecture Notes in Computer Science*, pp. 842–857, Apr. 2016.

[3] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left- right consistency. In *CVPR*, 2017 Godard,

[4] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012.

[5] "monodepth," GitHub, [Online]. Available: https://github.com/mrharicot/monodepth. [Accessed: 30-Apr-2020].

[6] Clement and Mac Aodha, Oisin and Firman, Michael and Brostow, Gabriel J. Digging Into Self-Supervised Monocular Depth Estimation. In ICCV, 2019.

[7] A. Saxena, M. Sun, and A. Ng, "Make3D: Learning 3D Scene Structure from a Single Still Image," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 5, pp. 824–840, 2009.

[8] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In NeurIPS-W, 2017.

[9] "monodepth2," GitHub, [Online]. Available: https://github.com/nianticlabs/monodepth2. [Accessed: 30-Apr-2020].

[10] "TensorFlow White Papers: TensorFlow," TensorFlow. [Online]. Available: https://www.tensorflow.org/about/bib. [Accessed: 01-Feb-2020].

[11] G. Yang, X. Song, C. Huang, Z. Deng, J. Shi, and B. Zhou, "DrivingStereo: A Large-Scale Dataset for Stereo Matching in Autonomous Driving Scenarios," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. UNet: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Dec. 2015.

[14] "It's Here: The New GeForce GTX 1080Ti Graphics Card," *NVIDIA*. [Online]. Available: https://www.nvidia.com/en-sg/geforce/products/10series/geforce-gtx-1080-ti/. [Accessed: 30-Apr-2020].

[15] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

Figure 6: TensorBoard log for Full_data_no_mu_with_SSIM_on_left_right_only_full_loss_smoothness_0_3_disparity_scalling_res_18_no_SSI__2020_4_19_batchsize_12. Note the spike in smoothness loss and corresponding drop in reprojection loss.