GeoNet: Unsupervised Learning of Monocular Depth

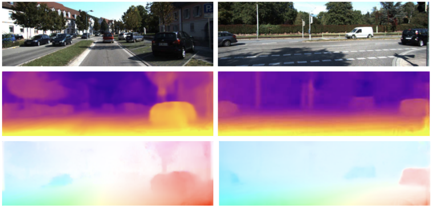
Zheyuan Wu

Washington University in St. Louis

September 9, 2025

Table of Contents

Introduction



The problem we are trying to solve is given below:
Let $\{I_1, I_2, \cdots, I_N\}$ be a sequence of images, we want to estimate the depth map D_t for each image.
This can be used to:

- Reconstruction of 3D scenes
- Generate Optical Flow
- Estimate Camera Pose
- Generate Scene Flow

Problems with Monocular Depth Estimation

- Camera Pose Estimation (translation and rotation)
- Illumination changes
- Rigid and non-rigid motions
- Occlusion
- Disocclusion
- Non-lambertian surfaces

Solution for Monocular Depth Estimation

Naive approach: Do with Convolutional Neural Networks (CNNs) and supervised learning.

- PoseNet: Estimate the camera pose (translation and rotation)
- DepthNet: Estimate the depth map (depth of each pixel)

Drawbacks: Need to manually annotate the depth map and camera pose.

Questions

How to obtain the depth map and camera pose or train with unlabeled data (unsupervised learning)?

GeoNet Architecture overview

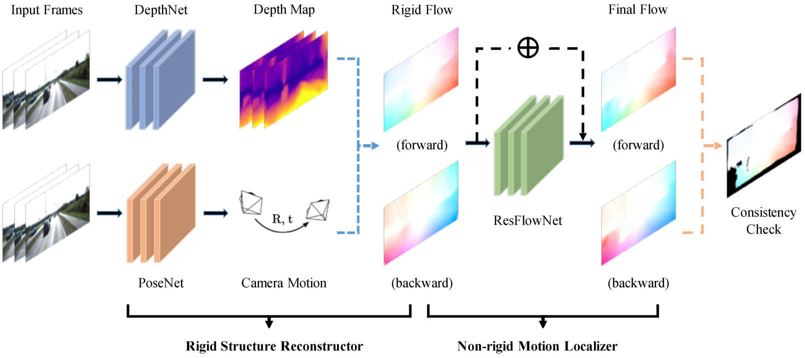


Figure 1: Image source: [Yin and Shi, 2018]

GeoNet Part I: (Rigid structure constructor)

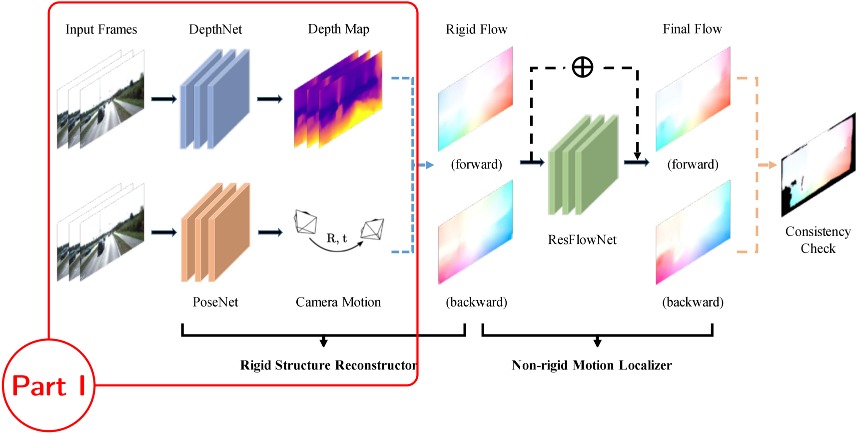


Figure 2: Image source: [Yin and Shi, 2018]

PoseNet and DepthNet

Unsupervised Learning of Depth and Ego-Motion From Video (CVPR 2017)[Zhou et al., 2017]

This is a method that estimates both depth and camera pose motion from a single video using CNN.
Jointly training a single-view depth CNN and a camera pose estimation CNN form unlabelled monocular video sequences.
Assumptions:

- The scene is static and the only motion is the camera motion.
- There is no occlusion/disocclusion between the target view and the source view.
- The surface is Lambertian.

Method (View synthesis)

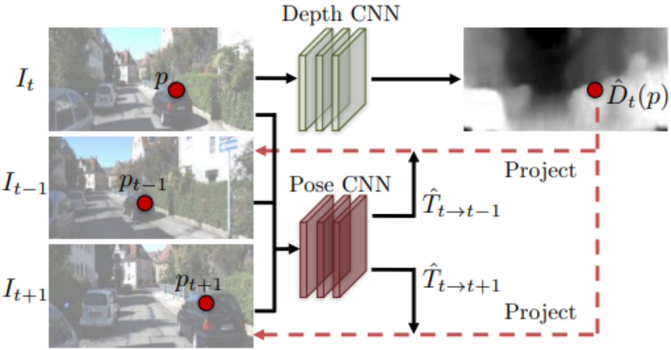


Figure 3: Image source: [Zhou et al., 2017]

Let I_{t-1}, I_t, I_{t+1} be three consecutive frames in the video.
Take the current frame as target view, we synthesize the image $\hat{I}_s$ from previous view I_{t-1} and successor view I_{t+1} based on predicted depth and camera pose motion.

Differentiable depth image-based rendering

Assume that the transition and rotation between the frames are smooth and differentiable.

Let

- p_t denote the pixel coordinates of I_t at time t.
- K denote the camera intrinsic matrix.
- p_s denote the pixel coordinates of I_s at time s.
- $\hat{T}_{t\to s}$ denote the predicted camera pose motion between I_t and I_s.
- $\hat{D}_t$ denote the predicted depth map of I_t.

We can always obtain p_t's projected coordinates to our source view p_s by the formula:

$$p_s \sim K\hat{T}_{t\to s}\hat{D}_t(p_t)K^{-1}p_t$$

Differentiable bilinear interpolation

Then we use differentiable bilinear interpolation to sample continuous pixel coordinates.

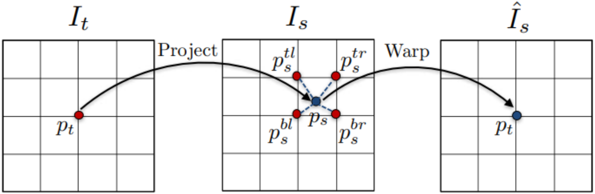


Figure 4: Image source: [Zhou et al., 2017]

$$\hat{I}_s(p_t) = I_s(p_s) = \sum_{i \in \{t,b\}, j \in \{l,r\}} w^{ij} I_s(p_s^{ij})$$

Where w^{ij} is the weight of the pixel p_s^{ij} in the bilinear interpolation. And $\sum_{i \in \{t,b\}, j \in \{l,r\}} w^{ij} = 1$.

Loss function from view synthesis

Let $\mathcal{I} = \{I_1, I_2, I_3, \cdots, I_n\}$ be the video sequence. Note that $I_t(p)$ is the pixel value of I_t at point p.

The loss function generated by view synthesis is:

$$\mathcal{L}_{vs} = \sum_{I_s \in \mathcal{I}} \sum_{p \in I_s} \left| I_t(p) - \hat{I}_s(p) \right|$$

Architecture of PoseNet and DepthNet

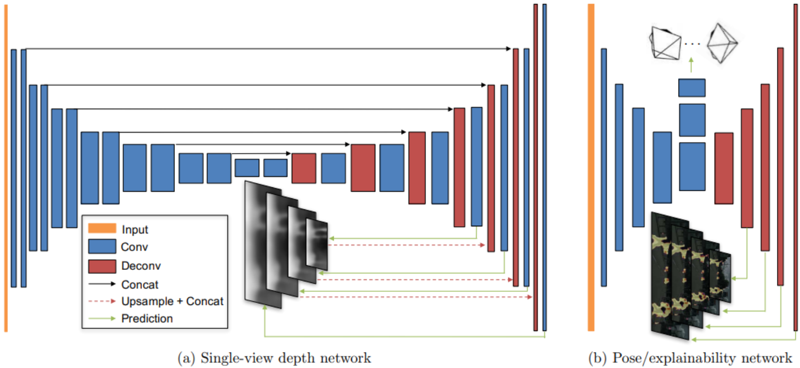


Figure 5: Since it's inevitable that there are some moving objects or occlusions between the target view and the source view, in PoseNet, we add a few additional layers to estimate our confidence (explainability mask) of the camera motion. Image source: [Zhou et al., 2017]

Rigid structure constructor [GeoNet Part I]

Combines the DepthNet and PoseNet to estimate the depth and camera pose motion from Unsupervised Learning of Depth and Ego-Motion From Video.[Zhou et al., 2017]
We denote the output of the Rigid structure constructor from frame t to s as $f^{rig}_{t\to s}$. The function output is a 2D vector showing the shift of the pixel coordinates.
Recall from previous paper,

$$\begin{aligned} p_t + f^{rig}_{t\to s}(p_t) &= KT_{t\to s}D_t(p_t)K^{-1}p_t \\ f^{rig}_{t\to s}(p_t) &= K(T_{t\to s}D_t(p_t) + I)K^{-1}p_t - p_t \end{aligned}$$

$f^{rig}_{t\to s}$ is a function that outputs a 2D vector showing the shift of the pixel coordinates that you are interested in.

Rigid structure constructor Implementation (PoseNet)

```python
def pose_net(opt, posenet_inputs):
    is_training = opt.mode == 'train_rigid'
    batch_norm_params = {'is_training': is_training}
    with tf.variable_scope('pose_net') as sc:
        with slim.arg_scope([slim.conv2d],
                            normalizer_fn=slim.batch_norm,
                            normalizer_params=batch_norm_params,
                            weights_regularizer=slim.l2_regularizer(0.0001),
                            activation_fn=tf.nn.relu):
            conv1  = slim.conv2d(posenet_inputs, 16,  7, 2)
            conv2  = slim.conv2d(conv1, 32,  5, 2)
            conv3  = slim.conv2d(conv2, 64,  3, 2)
            conv4  = slim.conv2d(conv3, 128, 3, 2)
            conv5  = slim.conv2d(conv4, 256, 3, 2)
            conv6  = slim.conv2d(conv5, 256, 3, 2)
            conv7  = slim.conv2d(conv6, 256, 3, 2)
            pose_pred = slim.conv2d(conv7, 6*opt.num_source, 1, 1,
                                    normalizer_fn=None, activation_fn=None)
            pose_avg = tf.reduce_mean(pose_pred, [1, 2])
            pose_final = 0.01 * tf.reshape(pose_avg, [-1, opt.num_source, 6])
            return pose_final
```

Figure 6: CNN that outputs the camera pose motion. (6-DoF translation and rotation (3 euler angles)) Image source: GeoNet Github

Rigid structure constructor Implementation (DepthNet)

```python
def build_dispnet(self):
    opt = self.opt

    # build dispnet_inputs
    if opt.mode == 'test_depth':
        # for test_depth mode we only predict the depth of the target image
        self.dispnet_inputs = self.tgt_image
    else:
        # multiple depth predictions; tgt: disp[:bs,:,:,:] src.i: disp[bs*(i+1):bs*(i+2),:,:,:]
        self.dispnet_inputs = self.tgt_image
        for i in range(opt.num_source):
            self.dispnet_inputs = tf.concat([self.dispnet_inputs, self.src_image_stack[:,:,:,3*i:3*(i+1)]], axis=0)

    # build dispnet
    self.pred_disp = disp_net(opt, self.dispnet_inputs)

    if opt.scale_normalize:
        # As proposed in https://arxiv.org/abs/1712.00175, this can
        # bring improvement in depth estimation, but not included in our paper.
        self.pred_disp = [self.spatial_normalize(disp) for disp in self.pred_disp]

    self.pred_depth = [1./d for d in self.pred_disp]
```

Figure 7: CNN that outputs the disparity map. (rigid motion assumed) Here disp_net is just choice of VGG or Resnet50. Image source: GeoNet Github

GeoNet Part II: (Non-rigid motion localizer)

Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose (CVPR 2018)[Yin and Shi, 2018]

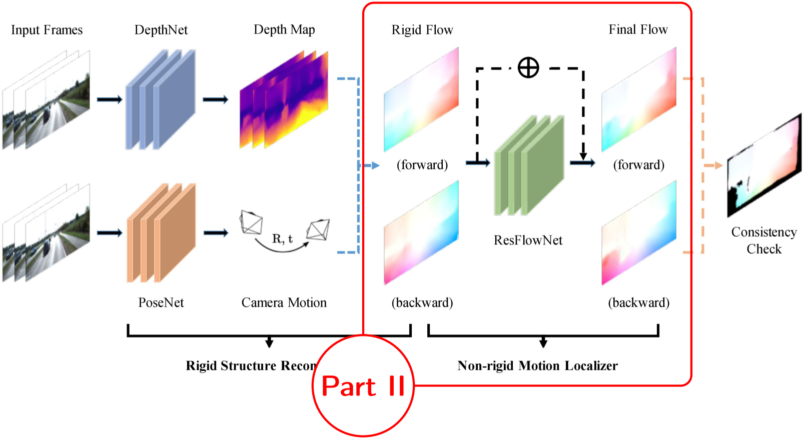


Figure 8: Image source: [Yin and Shi, 2018]

Depth estimation with Left-Right Consistency

Unsupervised Monocular Depth Estimation with Left-Right Consistency(CVPR 2016)[Godard et al., 2016]

This is a method that use pair of images as Left and Right eye to estimate depth. Increased consistency by flipping the right-left relation.
Intuition: Given a calibarated pair of binocular cameras, if we can learn a function that is able to reconstruct one image from the other (disparity map), then we have learned something about the depth of the scene.
Assumptions:

- Lambertian surface.
- No occlusion/disocclusion between the left and right image (for computing scene flow).

Architecture of Left-right consistency Network

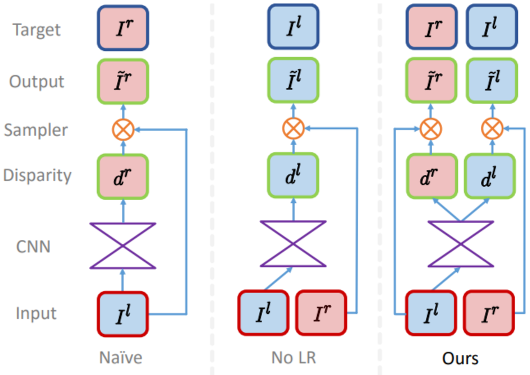


Figure 9: Native: Use source image as left image, produce disparity map for right image. No LR: Use source image as left image, produce disparity map for left image. Ours: Produce disparity map for both left and right image and do cross-validation. Image source: [Godard et al., 2016]

Appearance matching loss

Let I^l denote the left image, I^r denote the right image, $\hat{L}^l$ denote the left image reconstructed from the right image.
$\hat{I}^l$ denote the left image reconstructed from the right image with the predicted disparity map d^l.
The appearance matching loss for left image is:

$$\mathcal{L}_{ap}^l = \frac{1}{N} \sum_{p \in I^l} \alpha \frac{1 - \mathrm{SSIM}(I^l(p), \hat{I}^l(p))}{2} + (1 - \alpha) \left\| I^l(p) - \hat{I}^l(p) \right\|_1$$

Here SSIM is the structural similarity index.[Wang et al., 2004].
α is a hyperparameter that balances the importance of the structural similarity and the pixel-wise difference. In this paper, $\alpha = 0.85$.

Disparity Smoothness Loss

Let ∂d denote the disparity gradient. $\partial_x d_p^l$ and $\partial_y d_p^l$ are the disparity gradient in the x and y directions respectively on the left image of pixel p.

The disparity smoothness loss is:

$$\mathcal{L}_{ds}^l = \frac{1}{N} \sum_{p \in I^l} \left|\partial_x d_p^l\right| e^{-\left|\partial_x d_p^l\right|} + \left|\partial_y d_p^l\right| e^{-\left|\partial_y d_p^l\right|}$$

Left-right disparity consistency loss

Our network produces two disparity maps, d^l and d^r. We can use the left-right consistency loss to enforce the consistency between the two disparity maps.

$$\mathcal{L}_{lr}^{l} = \frac{1}{N} \sum_{p \in I^l} \left| d_p^l - d_{p+d_p^l}^r \right|$$

Total loss for Left-right consistency Network

Loss functions for Left-right consistency Network

$$\mathcal{L} = \alpha_{ap}(\mathcal{L}_{ap}^{l} + \mathcal{L}_{ap}^{r}) + \alpha_{ds}(\mathcal{L}_{ds}^{l} + \mathcal{L}_{ds}^{r}) + \alpha_{lr}(\mathcal{L}_{lr}^{l} + \mathcal{L}_{lr}^{r})$$

where $\alpha_{ap}, \alpha_{ds}, \alpha_{lr}$ are hyperparameters that balance the importance of the appearance matching loss, the disparity smoothness loss, and the left-right disparity consistency loss.

Non-rigid motion localizer [GeoNet Part II]

Use Left-right consistency[Godard et al., 2016] to estimate the non-rigid motion by training the ResFlowNet.

Let $\hat{I}_s^{rig}$ denote the inverse wrapped image from frame s to t. Note that $\hat{I}_s^{rig}$ is the prediction of I_t from I_s, using the rigid structure constructor defined in GeoNet Part I.

Recall from previous paper, we rename the $\mathcal{L}_{ap}^{l}$ (appearance matching loss for left image) to $\mathcal{L}_{rw}$ (rigid warping loss).

$$\mathcal{L}_{rw} = \frac{1}{N} \sum_{p \in I_t} \alpha \frac{1 - \mathrm{SSIM}(I_t(p), \hat{I}_s^{rig}(p))}{2} + (1 - \alpha) \left\| I_t(p) - \hat{I}_s^{rig}(p) \right\|_1$$

Non-rigid motion localizer (cont.)

Then we use $\mathcal{L}_{ds}$ to enforce the smoothness of the disparity map. (localize the non-rigid motion)

$$\mathcal{L}_{ds} = \sum_{p \in I^l} \left|\partial_x d_p^l\right| e^{-\left|\partial_x d_p^l\right|} + \left|\partial_y d_p^l\right| e^{-\left|\partial_y d_p^l\right|} = \sum_{p_t} |\nabla D(p_t)| \cdot \left(e^{-|\nabla I(p_t)|}\right)^T$$

We denote the output of the Non-rigid motion localizer from frame t to s as $f_{t \to s}^{res}$. So the final full flow prediction is the sum of the rigid and non-rigid motion predictions, $f_{t \to s}^{full} = f_{t \to s}^{res} + f_{t \to s}^{rig}$.

Replacing $\hat{I}_s^{rig}$ with $\hat{I}_s^{full}$, in $\mathcal{L}_{rw}$ and $\mathcal{L}_{ds}$, we get the $\mathcal{L}_{fw}$ and $\mathcal{L}_{fs}$ for the non-rigid motion localizer.

GeoNet Part III: (Geometric consistency enforcement)

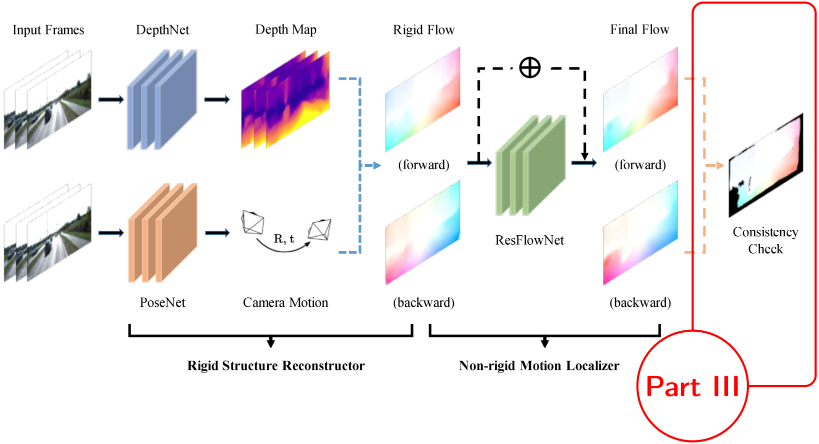


Figure 10: Image source: [Yin and Shi, 2018]

Geometric consistency enforcement

Finally, we use an additional geometric consistency enforcement to handle non-Lambertian surfaces (e.g., metal, plastic, etc.).
This is done by additional term in the loss function.
Let $\Delta f^{full}_{t\to s}(p_t) = f^{full}_{t\to s}(p_t) - f^{full}_{s\to t}(p_t)$.
Let $\delta(p_t)$ denote the function belows for arbitrary $\alpha, \beta > 0$:

$$\delta(p_t) = \begin{cases} 1 & \text{if } \|\Delta f^{full}_{t\to s}(p_t)\|_2 < \max\{\alpha, \beta \|f^{full}_{t\to s}(p_t)\|_1\} \\ 0 & \text{otherwise} \end{cases}$$

The geometric consistency enforcement loss is:

$$\mathcal{L}_{gc} = \sum_{p_t} \delta(p_t) \|\Delta f^{full}_{t\to s}(p_t)\|_2$$

Loss function for GeoNet

Let l be the set of pyramid image scales. $\langle t, s\rangle$ denote the set of all pairs of frames in the video and their inverse pairs, $t \neq s$.

$$\mathcal{L} = \sum_{l} \sum_{\langle t,s\rangle} \mathcal{L}_{rw} + \lambda_{ds}\mathcal{L}_{ds} + \lambda_{fw}\mathcal{L}_{fw} + \lambda_{fs}\mathcal{L}_{fs} + \lambda_{gc}\mathcal{L}_{gc}$$

$\lambda_{ds}, \lambda_{fw}, \lambda_{fs}, \lambda_{gc}$ are hyperparameters that balance the importance of the different losses.

Metrics (Monocular Depth Estimation)

KITTI dataset: Stereo image pairs

Cityscapes dataset: Monocular video sequences

- Absolute relative error ($\text{AbsRel}(I, \hat{I}) = \frac{1}{N}\sum_{p\in I}\frac{|I(p)-\hat{I}(p)|}{I(p)}$). Lower is better.
- Squared relative error ($\text{SqRel}(I, \hat{I}) = \frac{1}{N}\sum_{p\in I}\frac{(I(p)-\hat{I}(p))^2}{I(p)^2}$). Lower is better.
- Root mean squared error ($\text{RMSE}(I, \hat{I}) = \sqrt{\frac{1}{N}\sum_{p\in I}(I(p)-\hat{I}(p))^2}$). Lower is better.
- Root mean squared logarithmic error ($\text{RMSElog}(I, \hat{I}) = \sqrt{\frac{1}{N}\sum_{p\in I}(\log I(p)-\log \hat{I}(p))^2}$). Lower is better.
- Fraction accuracy ($\delta < 1.25$). Higher is better.

$$\delta < 1.25(I, \hat{I}) = \frac{1}{N}\sum_{p\in I}\begin{cases} 1 & \text{if } \max\left\{\frac{I(p)}{\hat{I}(p)}, \frac{\hat{I}(p)}{I(p)}\right\} < 1.25 \\ 0 & \text{otherwise}\end{cases}$$

Results

Method	Supervised	Dataset	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Eigen *et al.* [9] Coarse	Depth	K	0.214	1.605	6.563	0.292	0.673	0.884	0.957
Eigen *et al.* [9] Fine	Depth	K	0.203	1.548	6.307	0.282	0.702	0.890	0.958
Liu *et al.* [28]	Depth	K	0.202	1.614	6.523	0.275	0.678	0.895	0.965
Godard *et al.* [15]	Pose	K	**0.148**	1.344	5.927	0.247	**0.803**	0.922	0.964
Zhou *et al.* [56]	No	K	0.208	1.768	6.856	0.283	0.678	0.885	0.957
Zhou *et al.* [56] updated[2]	No	K	0.183	1.595	6.709	0.270	0.734	0.902	0.959
Ours VGG	No	K	0.164	1.303	6.090	0.247	0.765	0.919	0.968
Ours ResNet	No	K	0.155	**1.296**	**5.857**	**0.233**	0.793	**0.931**	**0.973**
Garg *et al.* [14] cap 50m	Pose	K	0.169	1.080	5.104	0.273	0.740	0.904	0.962
Ours VGG cap 50m	No	K	0.157	0.990	4.600	0.231	0.781	0.931	0.974
Ours ResNet cap 50m	No	K	**0.147**	**0.936**	**4.348**	**0.218**	**0.810**	**0.941**	**0.977**
Godard *et al.* [15]	Pose	CS + K	**0.124**	**1.076**	**5.311**	**0.219**	**0.847**	**0.942**	**0.973**
Zhou *et al.* [56]	No	CS + K	0.198	1.836	6.565	0.275	0.718	0.901	0.960
Ours ResNet	No	CS + K	0.153	1.328	5.737	0.232	0.802	0.934	0.972

Table 1. Monocular depth results on KITTI 2015 [31] by the split of Eigen *et al.* [9]. For training, K is the KITTI dataset [31] and CS is Cityscapes [7]. Errors for other methods are taken from [15, 56]. We show the best result trained only on KITTI in bold. The results of Garg *et al.* [14] are capped at 50m and we seperately list them for comparison.

Figure 11: Recall that [15] is the left-right consistency network, [56] is the Unsupervised Learning of Depth and Ego-Motion From Video (PoseNet and DepthNet). In section III, inferior results may due to distinctions between training data characteristics. (stereo image pairs: KITTI. and monocular video sequences: Cityscapes) Image source: [Yin and Shi, 2018]

References I

Godard, C., Aodha, O. M., and Brostow, G. J. (2016).
Unsupervised monocular depth estimation with left-right consistency.
CoRR, abs/1609.03677.

Mayer, N., Ilg, E., Häusser, P., Fischer, P., Cremers, D., Dosovitskiy, A., and Brox, T. (2015).
A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation.
CoRR, abs/1512.02134.

Wang, Z., Bovik, A., Sheikh, H., and Simoncelli, E. (2004).
Image quality assessment: from error visibility to structural similarity.
IEEE Transactions on Image Processing, 13(4):600–612.

References II

Yin, Z. and Shi, J. (2018).
Geonet: Unsupervised learning of dense depth, optical flow and camera pose.
CoRR, abs/1803.02276.

Zhou, T., Brown, M., Snavely, N., and Lowe, D. G. (2017).
Unsupervised learning of depth and ego-motion from video.
CoRR, abs/1704.07813.

Q&A