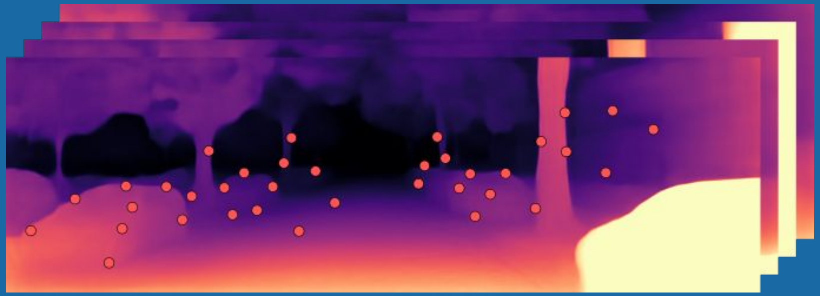
JUNE 18-22, 2023
CVPR
VANCOUVER, CANADA



SfM-TTR: Structure from Motion for Test-Time Refinement of Single-View Depth Networks

Sergio Izquierdo, Javier Civera

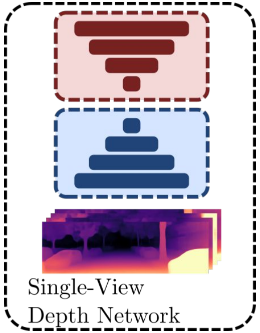
University of Zaragoza

THU-PM-082

Scene Reconstruction

Multi-view
traditional methods

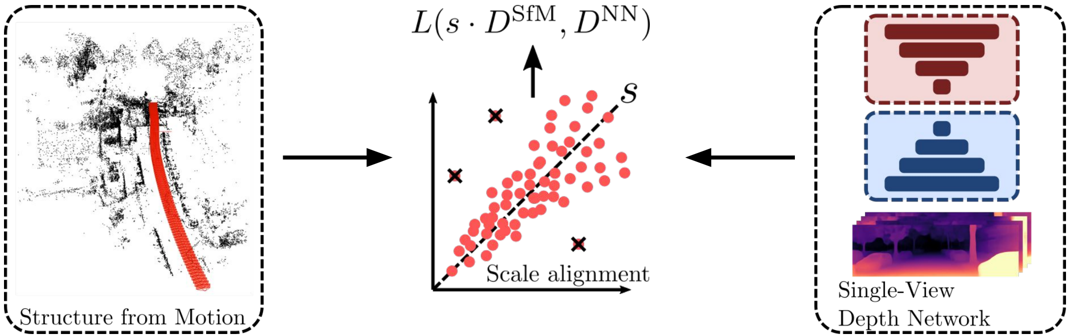
Single-view
deep learning



Method Overview

Multi-view
traditional methods

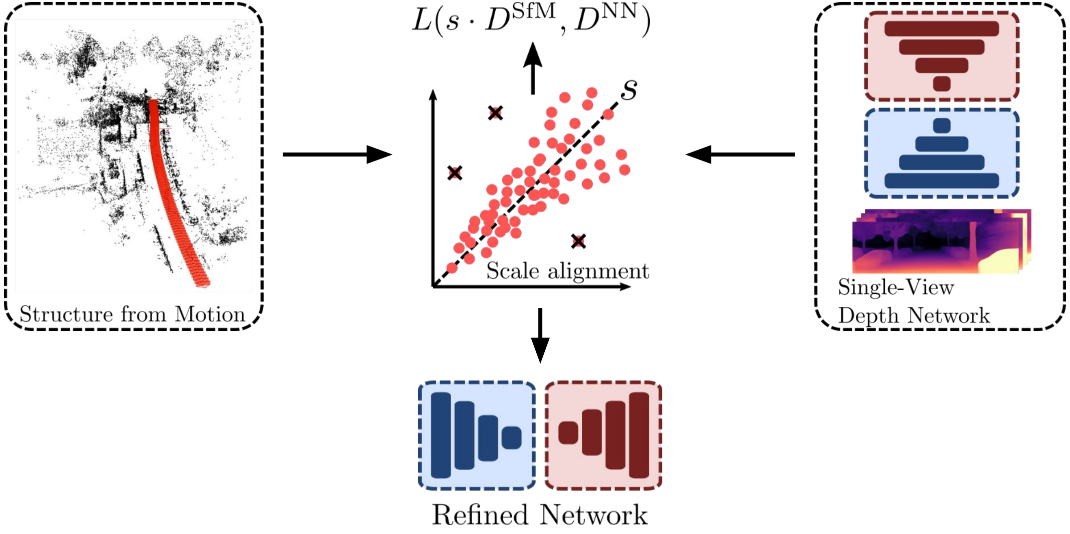
Single-view
deep learning



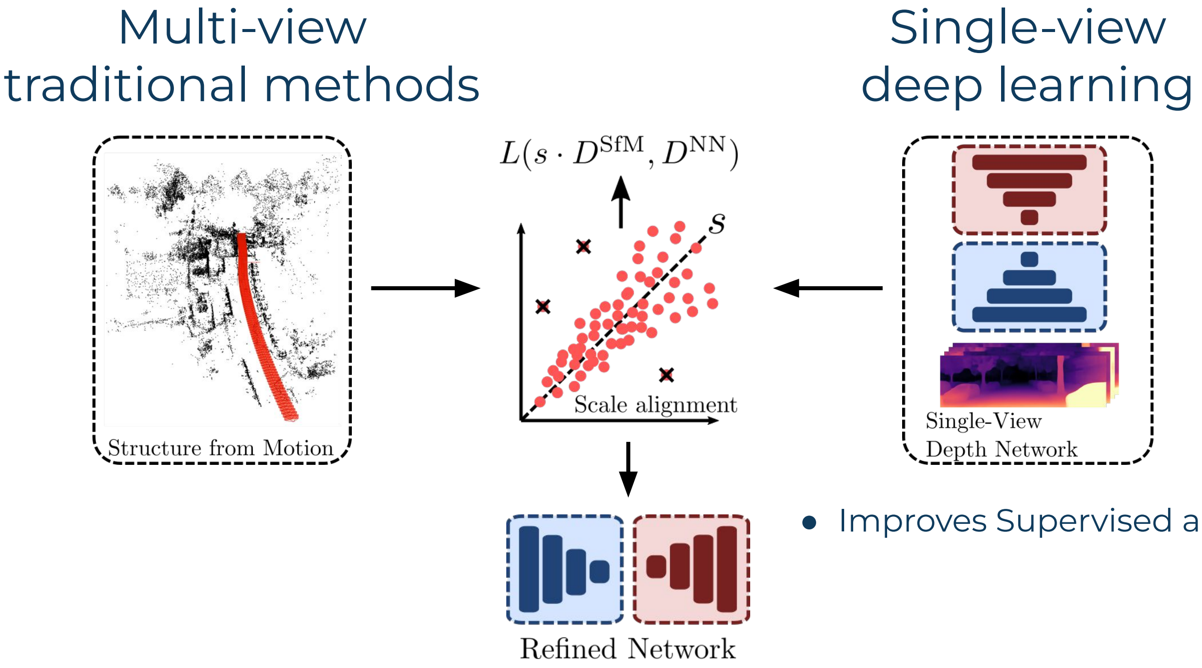
Method Overview

Multi-view
traditional methods

Single-view
deep learning



Method Overview

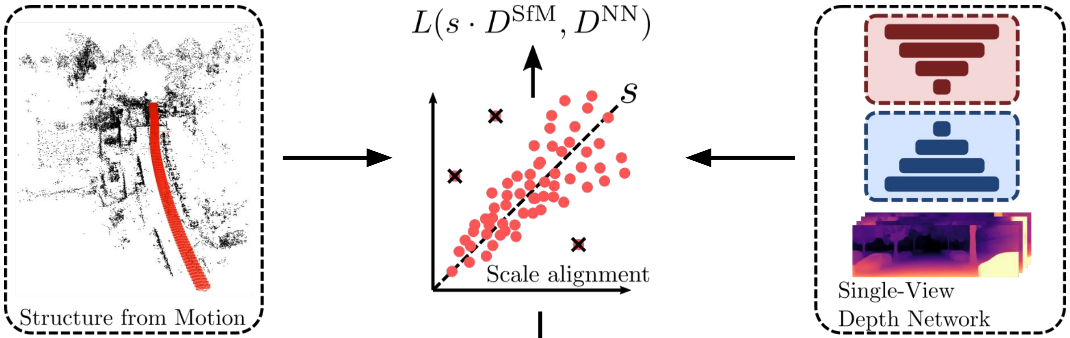


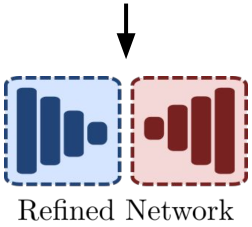
- Improves Supervised and Self-supervised

Method Overview

Multi-view
traditional methods

Single-view
deep learning



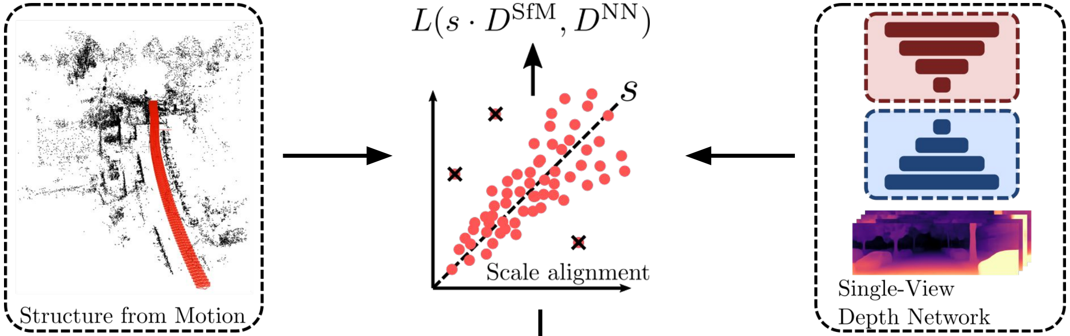


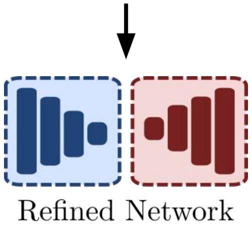
- Improves Supervised and Self-supervised
- 27% RMSE reduction

Method Overview

Multi-view
traditional methods

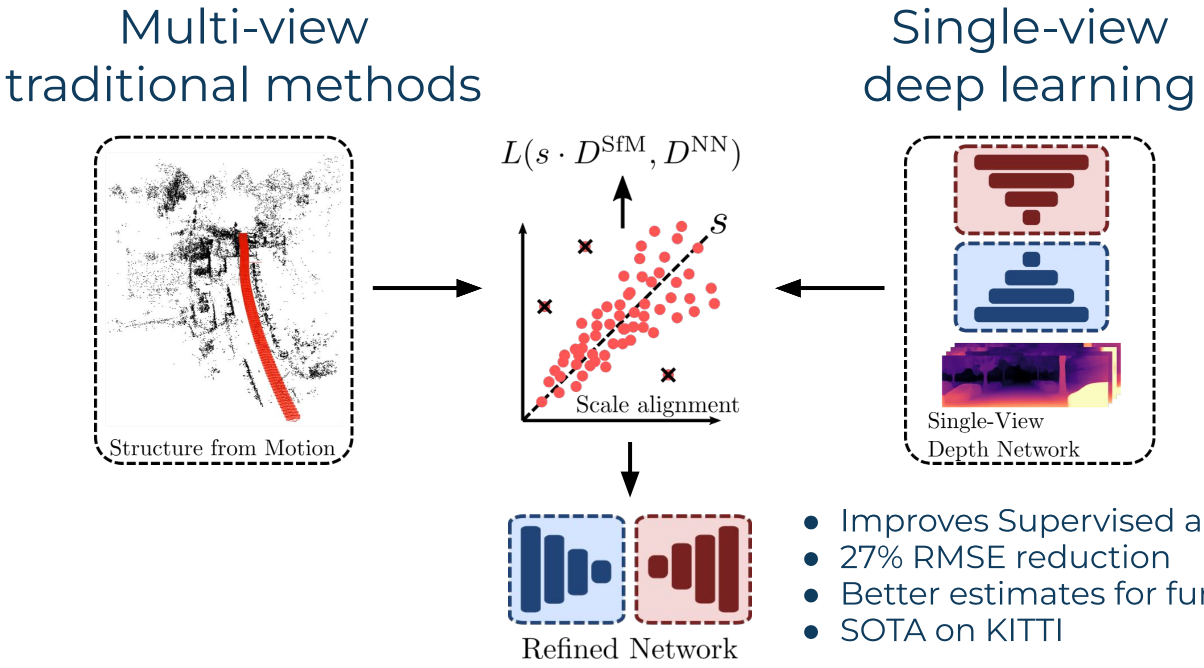
Single-view
deep learning





- Improves Supervised and Self-supervised
- 27% RMSE reduction
- Better estimates for further areas

Method Overview



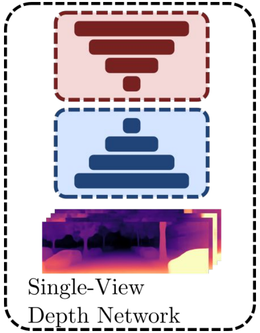
- Improves Supervised and Self-supervised
- 27% RMSE reduction
- Better estimates for further areas
- SOTA on KITTI

Motivation

- Multi-view traditional methods
  - ✔ 3D geometry based
  - ✔ Accurate estimations
  - ✗ Sparse depth map
  - ✗ Not learned priors

how to combine both?

- Single-view networks
  - ✔ Learned priors
  - ✔ Dense estimations
  - ✗ Vast collections of images
  - ✗ No geometry based



Method

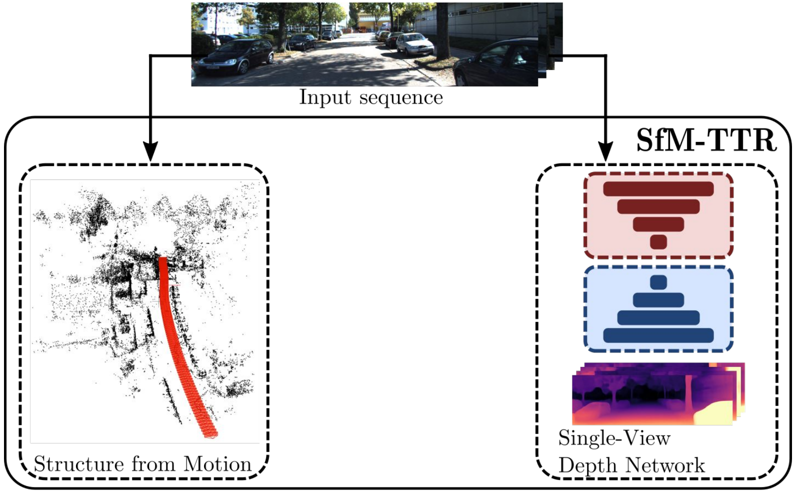
Input sequence

Method

Input sequence

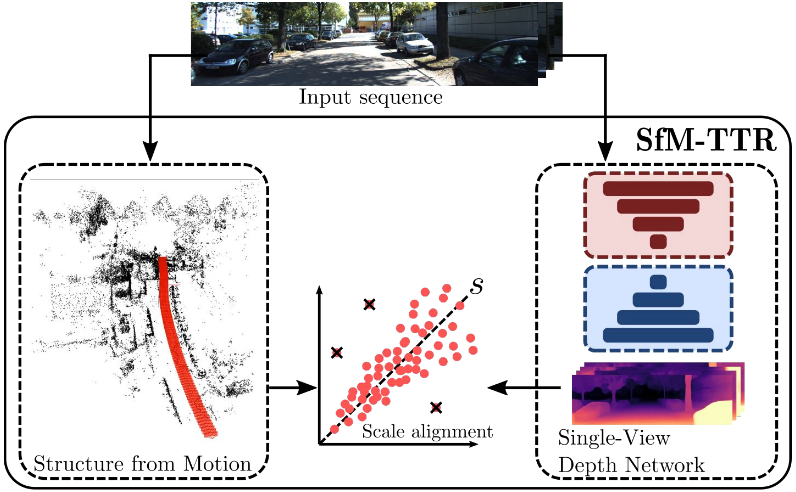
SfM-TTR

Method



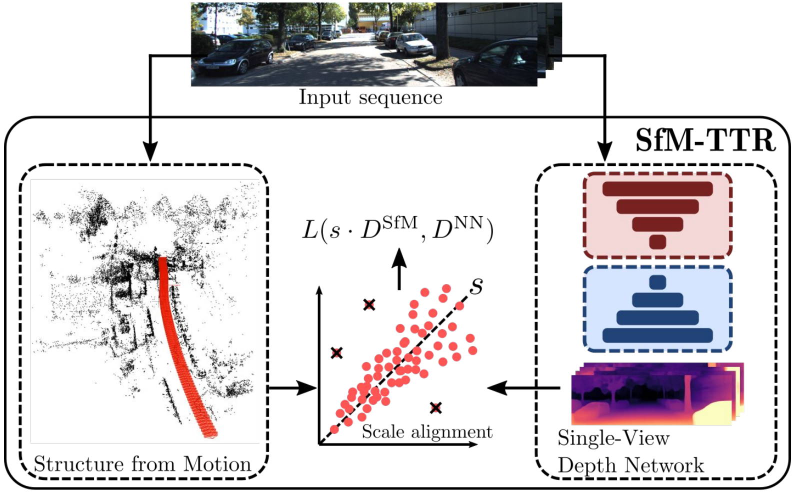
1. Compute SfM reconstruction
2. Compute Network predictions

Method



1. Compute SfM reconstruction
2. Compute Network predictions
3. Align both depth maps

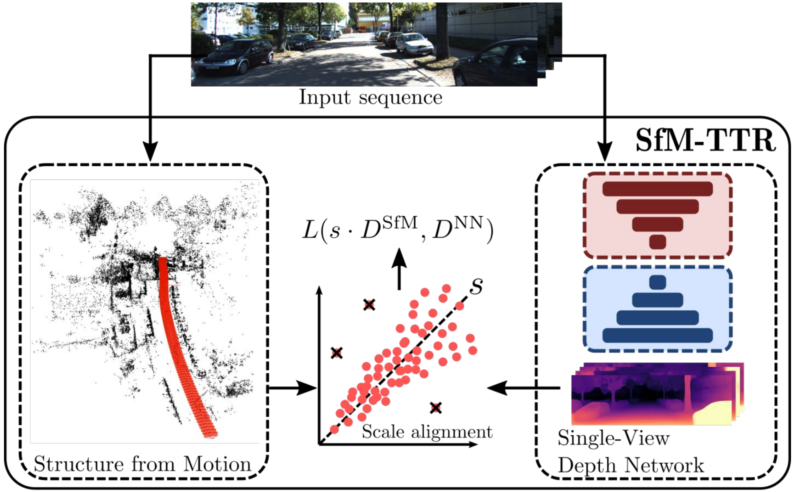
Method



1. Compute SfM reconstruction
2. Compute Network predictions
3. Align both depth maps
4. Use sparse depth as pseudo-gt

$$\mathcal{L} = \frac{1}{|\mathcal{D}_j^{\mathrm{SfM}}|} \sum_l w_{l,j}^{\boldsymbol{\theta}} \| \hat{s} \cdot D_{l,j}^{\mathrm{SfM}} - D_{l,j}^{\mathrm{NN}} \|_1$$

Method



1. Compute SfM reconstruction
2. Compute Network predictions
3. Align both depth maps
4. Use sparse depth as pseudo-gt

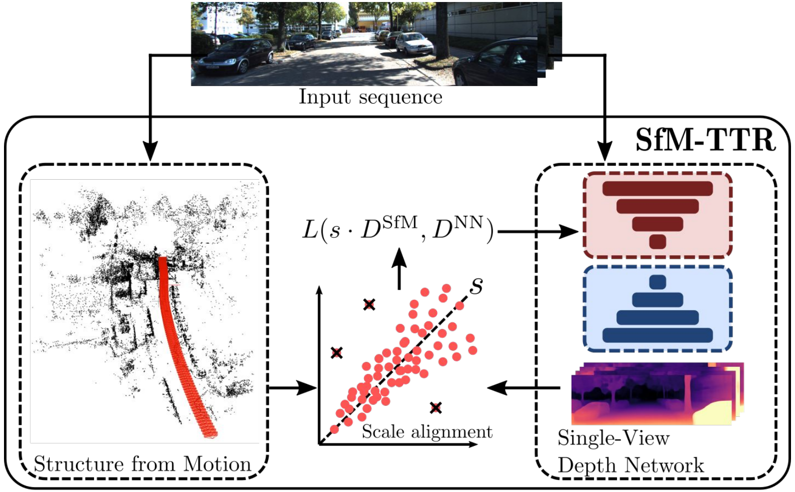
$$\mathcal{L} = \frac{1}{|\mathcal{D}_j^{\mathrm{SfM}}|} \sum_l w_{l,j}^{\boldsymbol{\theta}} \| \hat{s} \cdot D_{l,j}^{\mathrm{SfM}} - D_{l,j}^{\mathrm{NN}} \|_1$$

reprojection error

scale from alignment

$$w_{l,j}^{\boldsymbol{\theta}} = \exp(-\|\mathbf{r}_{l,j}\|_2^2)$$

Method

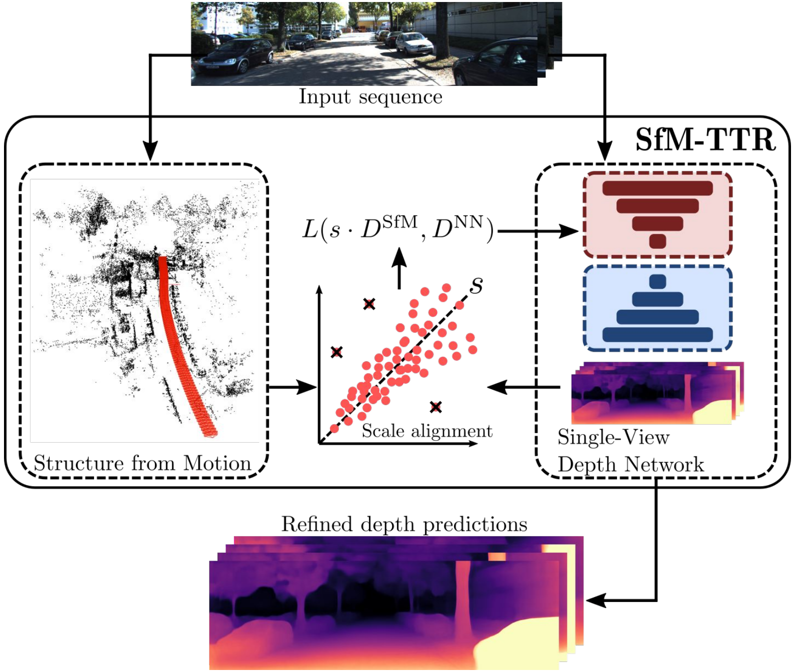


1. Compute SfM reconstruction
2. Compute Network predictions
3. Align both depth maps
4. Use sparse depth as pseudo-gt

$$\mathcal{L} = \frac{1}{|\mathcal{D}_j^{\mathrm{SfM}}|} \sum_l w_{l,j}^{\boldsymbol{\theta}} \|\hat{s} \cdot D_{l,j}^{\mathrm{SfM}} - D_{l,j}^{\mathrm{NN}}\|_1$$

5. Optimize the encoder

Method

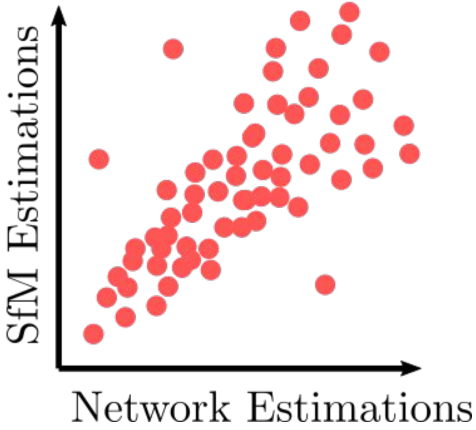


1. Compute SfM reconstruction
2. Compute Network predictions
3. Align both depth maps
4. Use sparse depth as pseudo-gt

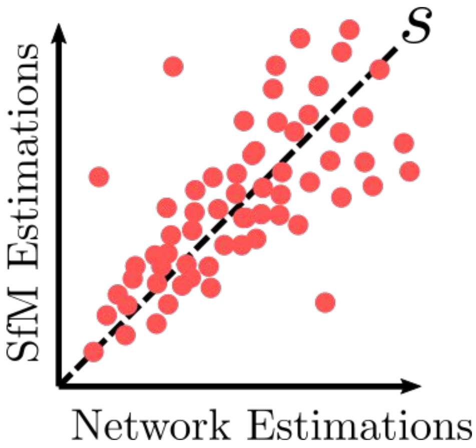
$$\mathcal{L} = \frac{1}{|\mathcal{D}_j^{\mathrm{SfM}}|} \sum_l w_{l,j}^{\boldsymbol{\theta}} \| \hat{s} \cdot D_{l,j}^{\mathrm{SfM}} - D_{l,j}^{\mathrm{NN}} \|_1$$

5. Optimize the encoder
6. Get refined predictions

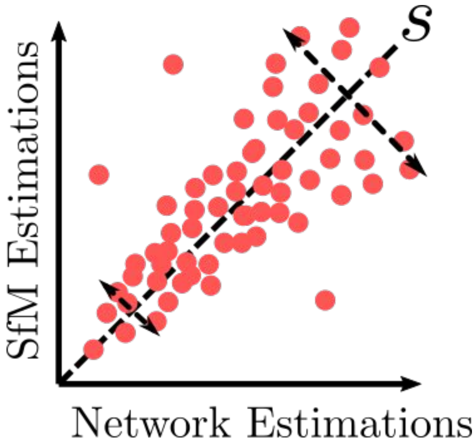
Alignment



Alignment



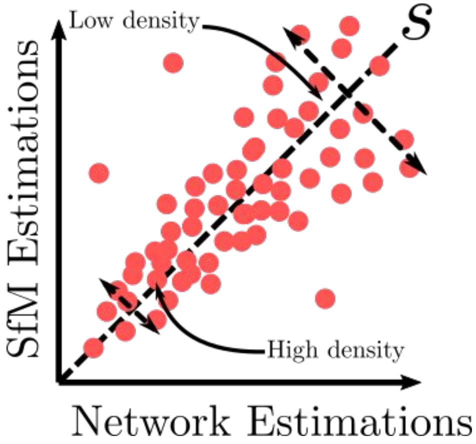
Alignment

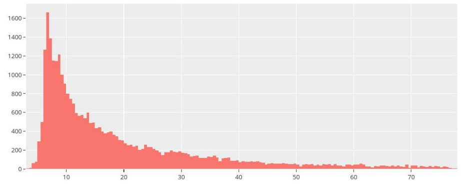


- Heteroscedasticity

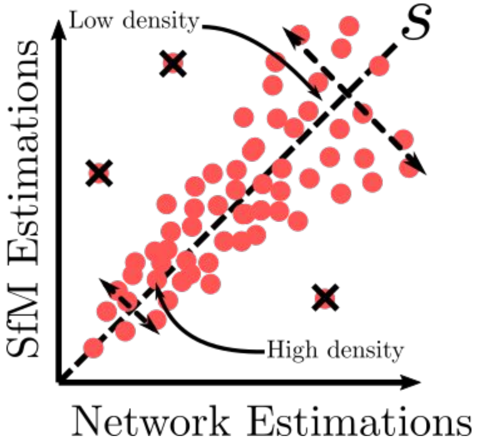
Alignment

- Heteroscedasticity
- Uneven distribution of points

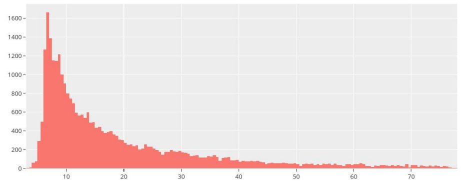




Alignment



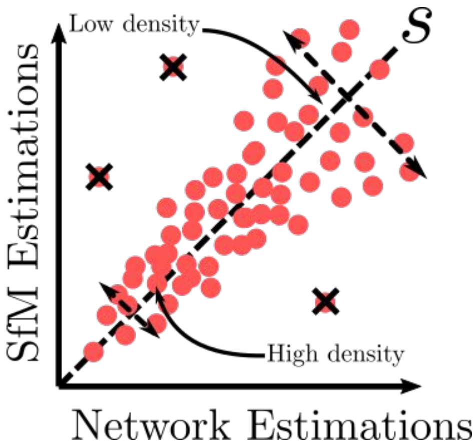
- Heteroscedasticity
- Uneven distribution of points



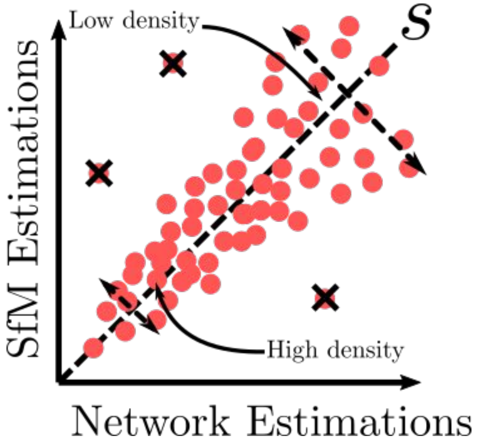
- Many outliers (both distributions)

Alignment

Two Steps: Strict & Relaxed Model



Alignment



Two Steps: Strict & Relaxed Model

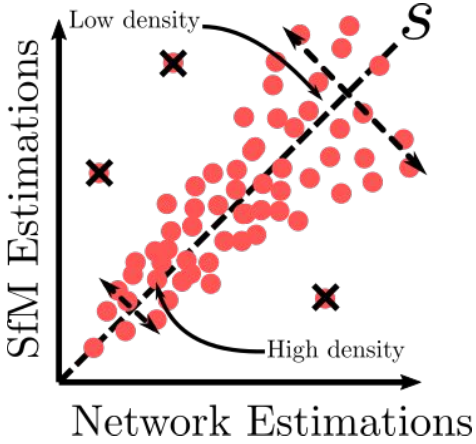
1. Strict model
   - RANSAC to remove outliers

$$\frac{\left(s_{l,j} \cdot D^{\mathrm{SfM}}_{l',j'} - D^{\mathrm{NN}}_{l',j'}\right)^2}{s_{l,j} \cdot D^{\mathrm{SfM}}_{l',j'}} \leq \tau$$

   - Weighted Least Squares

$$\hat{s} = \underset{s}{\operatorname{argmin}} \sum_j \sum_l w^s_{l,j} \left(s \cdot D^{\mathrm{SfM}\checkmark}_{l,j} - D^{\mathrm{NN}\checkmark}_{l,j}\right)^2$$

Alignment



Two Steps: Strict & Relaxed Model

1. Strict model
   - RANSAC to remove outliers

$$\frac{\left(s_{l,j} \cdot D^{\mathrm{SfM}}_{l',j'} - D^{\mathrm{NN}}_{l',j'}\right)^2}{s_{l,j} \cdot D^{\mathrm{SfM}}_{l',j'}} \leq \tau$$

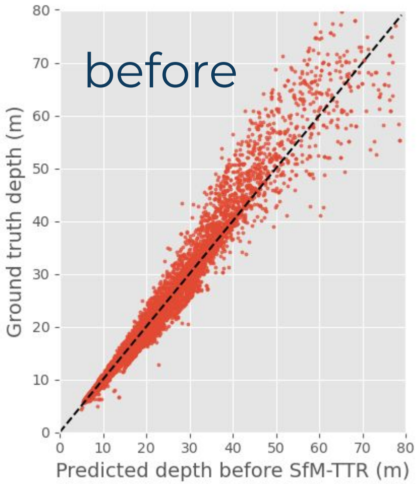
   - Weighted Least Squares

$$\hat{s} = \underset{s}{\operatorname{argmin}} \sum_j \sum_l w^s_{l,j} \left(s \cdot D^{\mathrm{SfM}\checkmark}_{l,j} - D^{\mathrm{NN}\checkmark}_{l,j}\right)^2$$

2. Relaxed model
   - Use $\hat{s}$ to include outliers and correct them

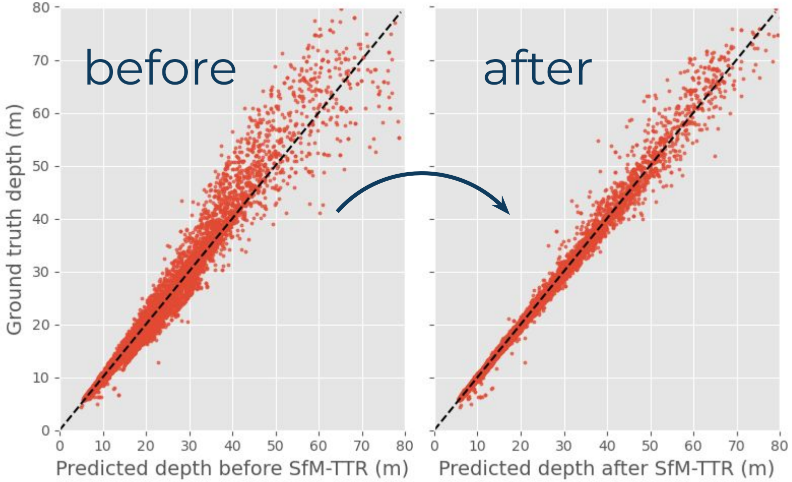
Experiments

- SfM-TTR improves depth estimations



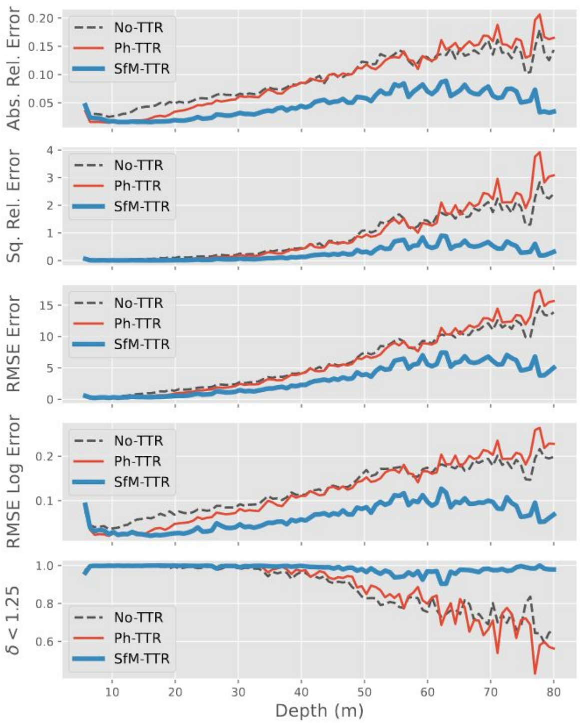
Experiments

- SfM-TTR improves depth estimations

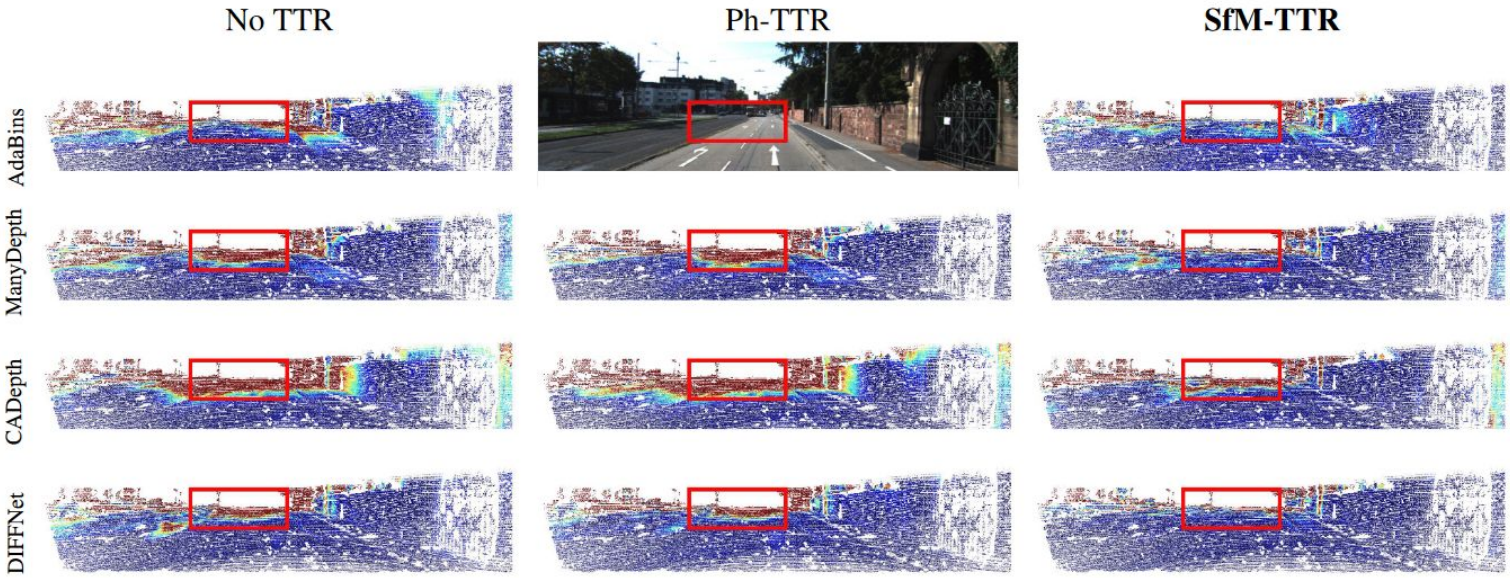


Experiments

- SfM-TTR improves depth estimations
- Bigger improvement than photometric refinement
- Specially for further areas



Experiments



Experiments

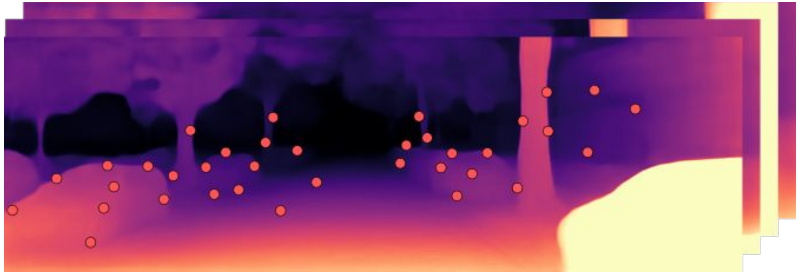
- Improvement in different network architectures
- Improvement in self and supervised models

TTR	Method	Abs Rel ↓	Sq Rel ↓	RMSE ↓	RMSE log ↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑
✗	AdaBins [6] ◇ †	0.058	0.190	2.360	0.088	0.964	0.995	**0.999**
✓	**AdaBins [6] + SfM-TTR †**	**0.054**	**0.138**	**1.885**	**0.078**	**0.978**	**0.996**	**0.999**
✗	ManyDepth [48] ∗	0.059	0.297	2.960	0.097	0.954	0.991	**0.998**
✓	ManyDepth [48] + Ph-TTR ∗	**0.053**	**0.252**	2.774	**0.089**	0.962	**0.993**	**0.998**
✓	**ManyDepth [48] + SfM-TTR**	0.054	**0.252**	**2.510**	**0.089**	**0.966**	0.992	**0.998**
✗	CADepth [51] ∗	0.073	0.359	3.287	0.112	0.941	0.990	**0.997**
✓	CADepth [51] + Ph-TTR ∗	0.082	0.426	3.565	0.124	0.923	0.986	**0.997**
✓	**CADepth [51] + SfM-TTR**	**0.060**	**0.263**	**2.620**	**0.096**	**0.962**	**0.992**	**0.997**
✗	DIFFNet [58] ∗	0.066	0.318	3.078	0.103	0.953	0.992	**0.998**
✓	DIFFNet [58] + Ph-TTR ∗	0.053	0.252	2.778	0.090	0.965	0.993	**0.998**
✓	**DIFFNet [58] + SfM-TTR**	**0.052**	**0.229**	**2.444**	**0.085**	**0.973**	**0.994**	**0.998**

KITTI

Conclusions

- SfM serves as good pseudo ground truth
- Significant improvement of the estimations
  - Specially in distant points
- Better results than other refinements



Paper+code