LiStereo: Generate Dense Depth Maps from LIDAR and Stereo Imagery

Junming Zhang[1], Manikandasriram Srinivasan Ramanagopal[2], Ram Vasudevan[3] and Matthew Johnson-Roberson[4]

I. Architecture

A. LiStereo

The structural details of our proposed model, LiStereo, is shown in Table I. The pipeline in the model consists of following parts. Inputs: rectified stereo images and corresponding left sparse depth maps. Feature extraction: high-level features are extracted from stereo images and sparse depth maps. ResNet50 [1] structure is used. There are two branches, the color images branch and the LIDAR branch. The correlation layer computes correlation from one camera view to the other. Features from left color image are processed by transform layer to prepare for later sensor fusion. The PSP module [2] is used to extract more contextual information. Fusion: we fuse information by concatenation. Estimation: fused information is processed to do depth estimation. Output: both dense disparity maps and depth maps are generated.

In Table I, "convBlock" denotes the convolution block, where a convolution layer is followed by batch normalization and leaky ReLU activation. "ResBlock" denotes the residual block introduced by [1]. "upConvBlock" denotes the upsampling block, where a bilinear interpolation upsampling layer is followed by a convolution layer, batch normalization and leaky ReLU activation. The column of "Attributes" denotes key parameters or a short description. The column of "Channels I/O" denotes number of channels of inputs and output. The first term is for color images branch and the second term is for LIDAR branch. The column of "Scaling" denotes the scale of current layers relative to original input images. "corr_layer" is introduced by [3] and "PSP module" is introduced by [2].

B. LiMono

We introduce LiMono as a baseline model, which is created by removing right color image branch and some layers in the LiStereo. The structural details of LiMono is shown in Table II. There is no correlation layer and estimated disparity maps in LiMono.

II. More qualitative results

More qualitative results are shown.

[1]J. Zhang is with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA junming@umich.edu

[2]M. Srinivasan Ramanagopal is with the Robotics Program, University of Michigan, Ann Arbor, MI 48109 USA srmani@umich.edu

[3]R. Vasudevan is with the Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI 48109 USA ramv@umich.edu

[4]M. Johnson-Roberson is with the Department of Naval Architecture and Marine Engineering, University of Michigan, Ann Arbor, MI 48109 USA mattjr@umich.edu

Layer	Attributes	Channels I/O	Scaling	Inputs
Feature Extraction				
left_convBlock1, right_convBlock1, lidar_convBlock1	kernel size = 7 stride = 2	3/32, 1 /16	1/2	inputs stereo images sparse depth maps
left_convBlock2, right_convBlock2, lidar_convBlock2	kernel size = 5 stride = 1	32/32, 16/16	1/2	left_convBlock1, right_convBlock1, lidar_convBlock1
left_resBlock1_1, right_resBlock1_1, lidar_resBlock1_1	kernel size = 3 stride = 2	32/64, 16/32	1/4	left_convBlock2, right_convBlock2, lidar_convBlock2
left_resBlock1_2, right_resBlock1_2, lidar_resBlock1_2	kernel size = 3 stride = 1	64/64, 32/32	1/4	left_resBlock1_1, right_resBlock1_1, lidar_resBlock1_1
left_resBlock1_3, right_resBlock1_3, lidar_resBlock1_3	kernel size = 3 stride = 1	64/64, 32/32	1/4	left_resBlock1_2, right_resBlock1_2, lidar_resBlock1_2
left_resBlock2_1, right_resBlock2_1, lidar_resBlock2_1	kernel size = 3 stride = 2	64/128, 32/64	1/8	left_resBlock1_3, right_resBlock1_3, lidar_resBlock1_3
left_resBlock2_2, right_resBlock2_2, lidar_resBlock2_2	kernel size = 3 stride = 1	128/128, 64/64	1/8	left_resBlock2_1, right_resBlock2_1, lidar_resBlock2_1
left_resBlock2_3, right_resBlock2_3, lidar_resBlock2_3	kernel size = 3 stride = 1	128/128, 64/64,	1/8	left_resBlock2_2, right_resBlock2_2, lidar_resBlock2_2
left_convBlock_pre, right_convBlock_pre, lidar_convBlock_pre	kernel size = 3 stride = 1	128/128, 64/64	1/8	left_resBlock2_3, right_resBlock2_3, lidar_resBlock2_3
corr_layer	max displacement = 24	128 / 25	1/8	left_convBlock_pre, right_convBlock_pre
context_layer	PSP module	128 / 128	1/8	left_convBlock_pre
trans_layer	kernel size = 3 stride = 1	128 / 128	1/8	left_convBlock_pre
Fusion				
concat_fusion	concatenation	128+128+64+25 / 345	1/8	corr_layer, context_layer, trans_layer, lidar_convBlock_pre
Estimation: encoder				
resBlock3_1	kernel size = 3 stride = 1	345 / 384	1/8	concat_fusion
resBlock3_2	kernel size = 3 stride = 1	384 / 384	1/8	resBlock3_1
resBlock3_3	kernel size = 3 stride = 1	384 / 384	1/8	resBlock3_2
resBlock4_1	kernel size = 3 stride = 2	384 / 512	1/16	resBlock3_3
resBlock4_2	kernel size = 3 stride = 1	512 / 512	1/16	resBlock4_1
resBlock4_3	kernel size = 3 stride = 1	512 / 512	1/16	resBlock4_2
Estimation: decoder				
upConvBlock1	kernel size = 3 stride = 2	512 / 256	1/8	resBlock4_3
skip_concat1	concatenation	256 + 384 / 640	1/8	upConvBlock1, resBlock3_3
convBlock3	kernel size = 3 stride = 1	640 / 256	1/8	skip_concat1
upConvBlock2	kernel size = 3 stride = 2	256 / 128	1/4	convBlock3
skip_concat2	concatenation	128 + 64 / 192	1/4	upConvBlock2, left_resBlock1_3
convBlock4	kernel size = 3 stride = 1	192 / 128	1/4	skip_concat2
upConvBlock3	kernel size = 3 stride = 2	128 / 64	1/2	convBlock_4
skip_concat3	concatenation	64 + 32 / 96	1/2	upConvBlock3, left_convBlock2
convBlock5	kernel size = 3 stride = 1	96 / 64	1/2	skip_concat3
upConvBlock4	kernel size = 3 stride = 2	64 / 64	1	convBlock5
disp_convBlock	kernel size = 3 stride = 1, no BN or lrelu	64 / 193	1	upConvBlock4
Output				
disparity	soft argmax	193 / 1	1	disp_convBlock
depth	baseline, focal length	1 / 1	1	disparity

TABLE I: Structural details in LiStereo

Layer	Attributes	Channels I/O	Scaling	Inputs
Feature Extraction				
left_convBlock1, lidar_convBlock1	kernel size = 7 stride = 2	3/32, 1 /16	1/2	left color images sparse depth maps
left_convBlock2, lidar_convBlock2	kernel size = 5 stride = 1	32/32, 16/16	1/2	left_convBlock1, right_convBlock1, lidar_convBlock1
left_resBlock1_1, lidar_resBlock1_1	kernel size = 3 stride = 2	32/64, 16/32	1/4	left_convBlock2, lidar_convBlock2
left_resBlock1_2, lidar_resBlock1_2	kernel size = 3 stride = 1	64/64, 32/32	1/4	left_resBlock1_1, lidar_resBlock1_1
left_resBlock1_3, lidar_resBlock1_3	kernel size = 3 stride = 1	64/64, 32/32	1/4	left_resBlock1_2, lidar_resBlock1_2
left_resBlock2_1, lidar_resBlock2_1	kernel size = 3 stride = 2	64/128, 32/64	1/8	left_resBlock1_3, lidar_resBlock1_3
left_resBlock2_2, lidar_resBlock2_2	kernel size = 3 stride = 1	128/128, 64/64	1/8	left_resBlock2_1, lidar_resBlock2_1
left_resBlock2_3, lidar_resBlock2_3	kernel size = 3 stride = 1	128/128, 64/64,	1/8	left_resBlock2_2, lidar_resBlock2_2
left_convBlock_pre, lidar_convBlock_pre	kernel size = 3 stride = 1	128/128, 64/64	1/8	left_resBlock2_3, lidar_resBlock2_3
context_layer	PSP module	128 / 128	1/8	left_convBlock_pre
trans_layer	kernel size = 3 stride = 1	128 / 128	1/8	left_convBlock_pre
Fusion				
concat_fusion	concatenation	128+128+64 / 320	1/8	context_layer, trans_layer, lidar_convBlock_pre
Estimation: encoder				
resBlock3_1	kernel size = 3 stride = 1	320 / 384	1/8	concat_fusion
resBlock3_2	kernel size = 3 stride = 1	384 / 384	1/8	resBlock3_1
resBlock3_3	kernel size = 3 stride = 1	384 / 384	1/8	resBlock3_2
resBlock4_1	kernel size = 3 stride = 2	384 / 512	1/16	resBlock3_3
resBlock4_2	kernel size = 3 stride = 1	512 / 512	1/16	resBlock4_1
resBlock4_3	kernel size = 3 stride = 1	512 / 512	1/16	resBlock4_2
Estimation: decoder				
upConvBlock1	kernel size = 3 stride = 2	512 / 256	1/8	resBlock4_3
skip_concat1	concatenation	256 + 384 / 640	1/8	upConvBlock1, resBlock3_3
convBlock3	kernel size = 3 stride = 1	640 / 256	1/8	skip_concat1
upConvBlock2	kernel size = 3 stride = 2	256 / 128	1/4	convBlock3
skip_concat2	concatenation	128 + 64 / 192	1/4	upConvBlock2, left_resBlock1_3
convBlock4	kernel size = 3 stride = 1	192 / 128	1/4	skip_concat2
upConvBlock3	kernel size = 3 stride = 2	128 / 64	1/2	convBlock_4
skip_concat3	concatenation	64 + 32 / 96	1/2	upConvBlock3, left_convBlock2
convBlock5	kernel size = 3 stride = 1	96 / 64	1/2	skip_concat3
upConvBlock4	kernel size = 3 stride = 2	64 / 64	1	convBlock5
Output				
depth_convBlock	kernel size = 3 stride = 1,no BN or lrelu	64 / 1	1	upConvBlock4

TABLE II: Structural details in LiMono

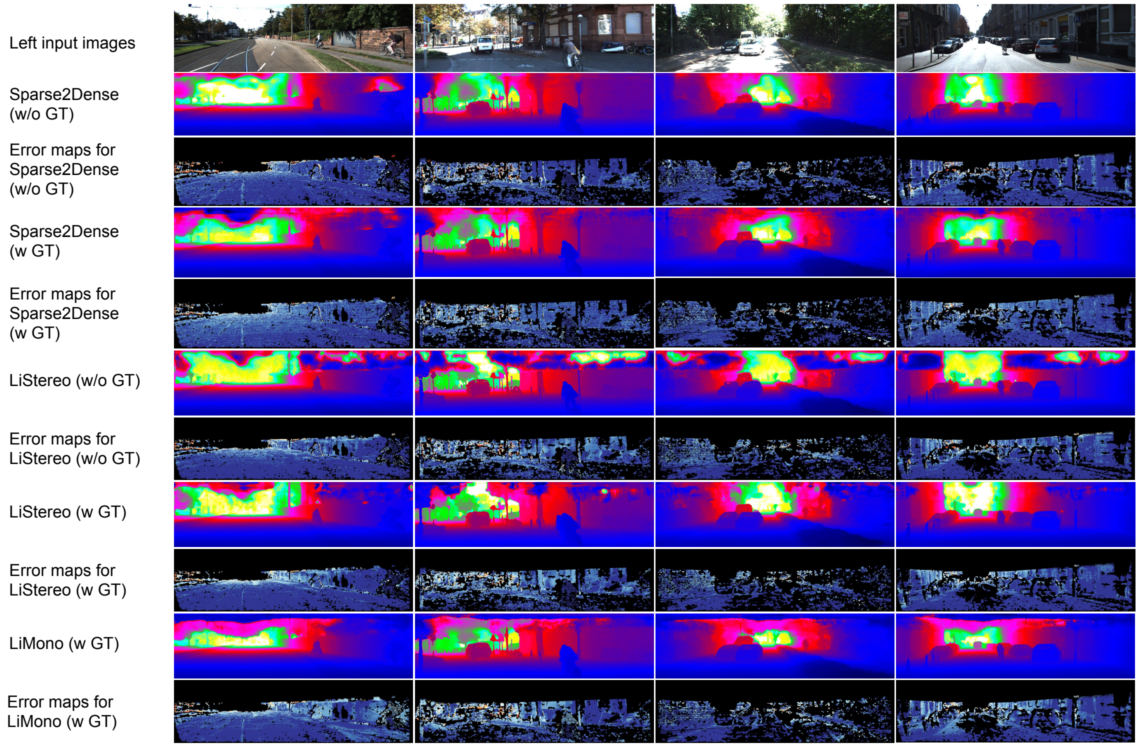

Fig. 1: Qualitative results on KITTI validation set. From top to bottom: left input image, estimated dense depth map and corresponding error maps of different methods. 'w/o GT' refers to training in a self-supervised manner. 'w GT' refers to training using ground-truth depth maps. The error maps use the log-color scale, depicting correct estimates in blue and wrong estimates in red. Pixels that have no ground-truth depth are colorized in black in error images.

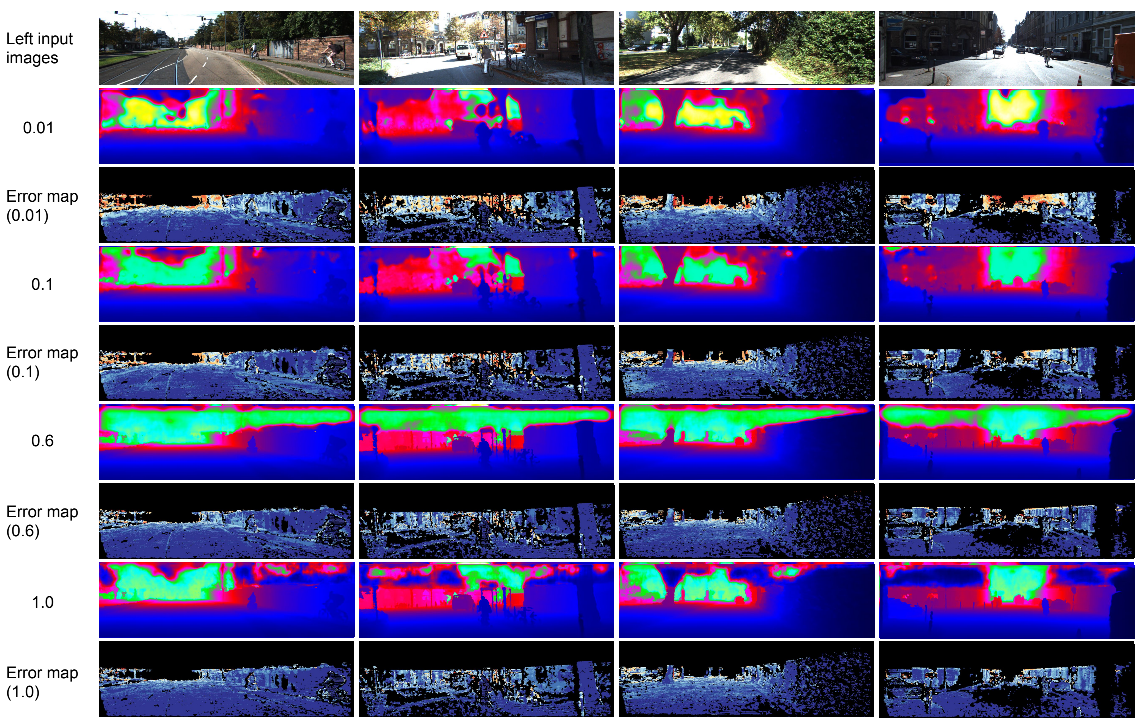

Fig. 2: Qualitative results on different levels of input sparsity for self-supervised model (LiStereo w/o GT) during training. The model is trained and evaluated using different levels of sparsity of input depth maps. From top to bottom: left input image, estimated dense depth maps and corresponding error maps of different input sparsity levels indicated on the left. The error maps use the log-color scale, depicting correct estimates in blue and wrong estimates in red. Pixels that have no ground-truth depth are colorized in black in error images.

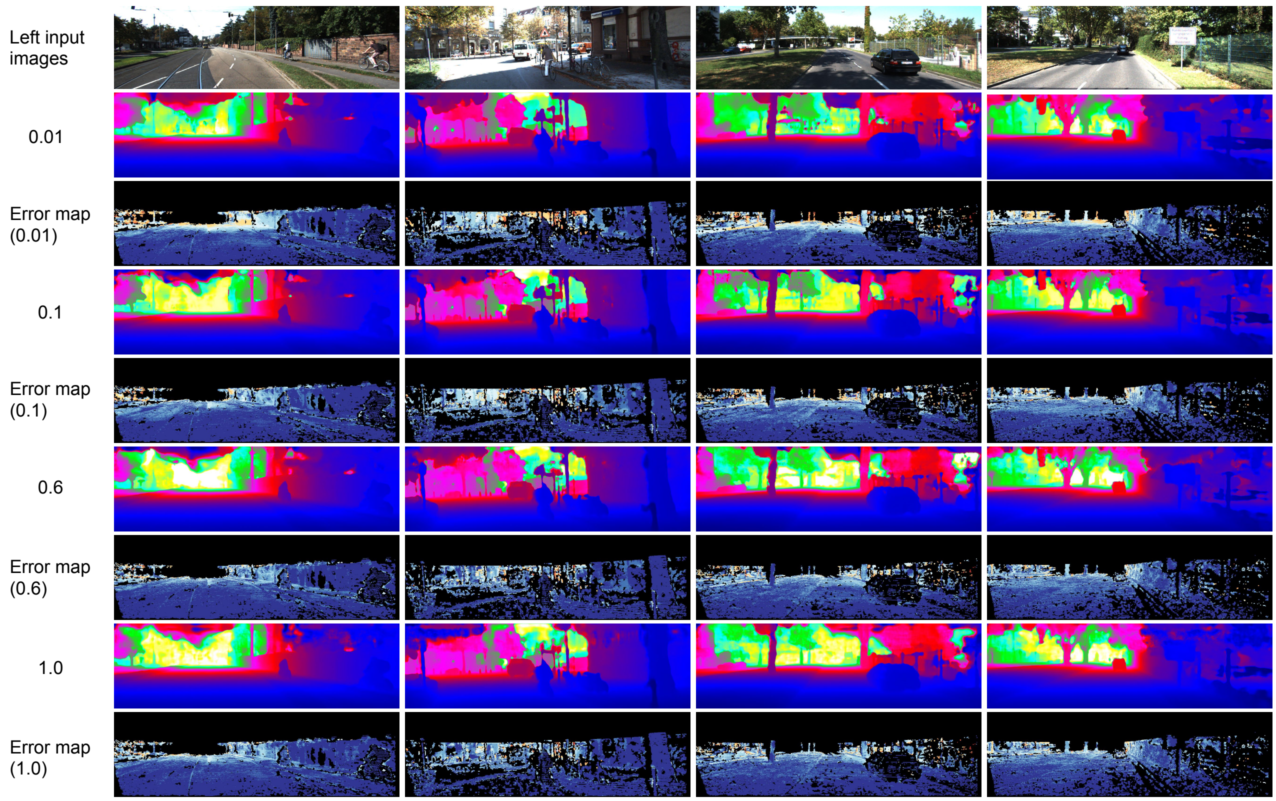

Fig. 3: Qualitative results on different levels of input sparsity for supervised model (LiStereo with GT) during training. The model is trained and evaluated using different levels of sparsity of input depth maps. From top to bottom: left input image, estimated dense depth maps and corresponding error maps of different input sparsity indicated on the left. The error maps use the log-color scale, depicting correct estimates in blue and wrong estimates in red. Pixels that have no ground-truth depth are colorized in black in error images.

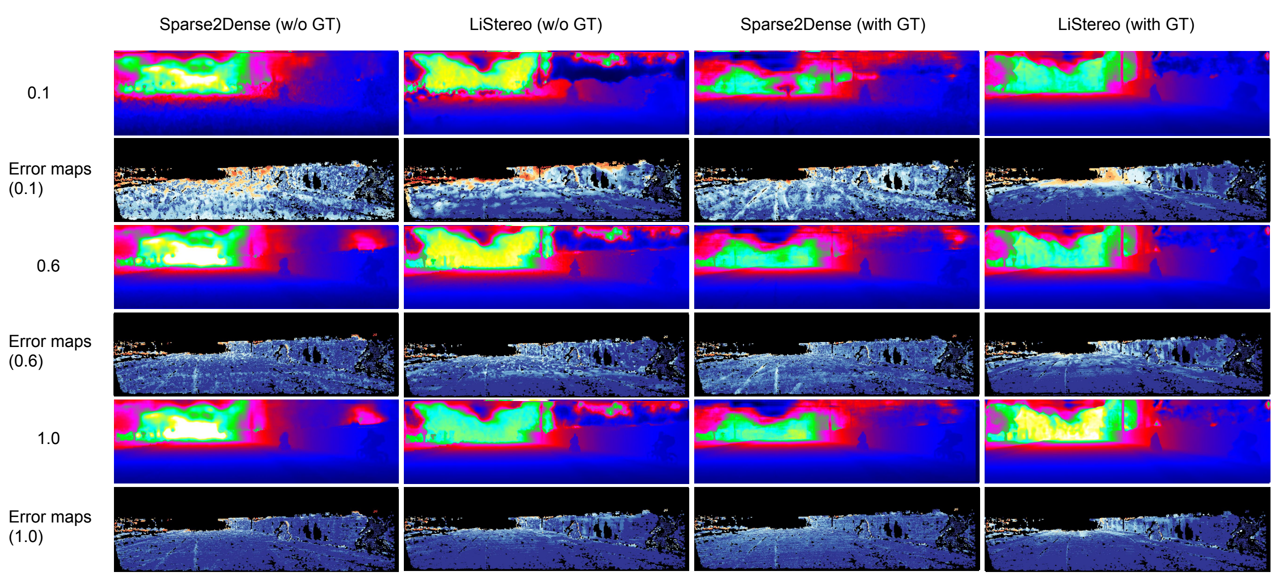

Fig. 4: Qualitative results on different levels of input sparsity during inference. Models are trained using original sparse depth maps but are provided different input sparsity levels during inference. From top to bottom: estimated dense depth maps and corresponding error maps of different different input sparsity indicated on the left. From left to right: results of Sparse2Dense (w/o GT), LiStereo (w/o GT), Sparse2Dense (with GT) and LiStereo (with GT). The error maps use the log-color scale, depicting correct estimates in blue and wrong estimates in red. Pixels that have no ground-truth depth are colorized in black in error images.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[2] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2881–2890.

[3] 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*, 2016, pp. 4040–4048.