Deep Multiview Stereo

3D Vision

University of Illinois

Derek Hoiem

Many slides adapted from Lana Lazebnik, Steve Seitz, Yasu Furukawa, Noah Snavely

This class: Deep Multiview Stereo

- Potential benefits of deep learning
- Deep learning background
- Deep network approaches to MVS
  - MVSNet
  - AttMVSNet

Benefits of deep learning

- Learn image and feature representations that are well-tuned for problems of interest
- Can optimize for many different kinds of losses or combinations of losses
- Can learn representations on large datasets and finetune them on smaller datasets
- Often, efficient inference on GPUs

Potential for deep networks in MVS

1. Learn better photometric scoring functions, e.g. more robust to smooth or reflective surfaces or boundaries, via better features or adaptive neighborhoods

2. Learn better prediction of visibility

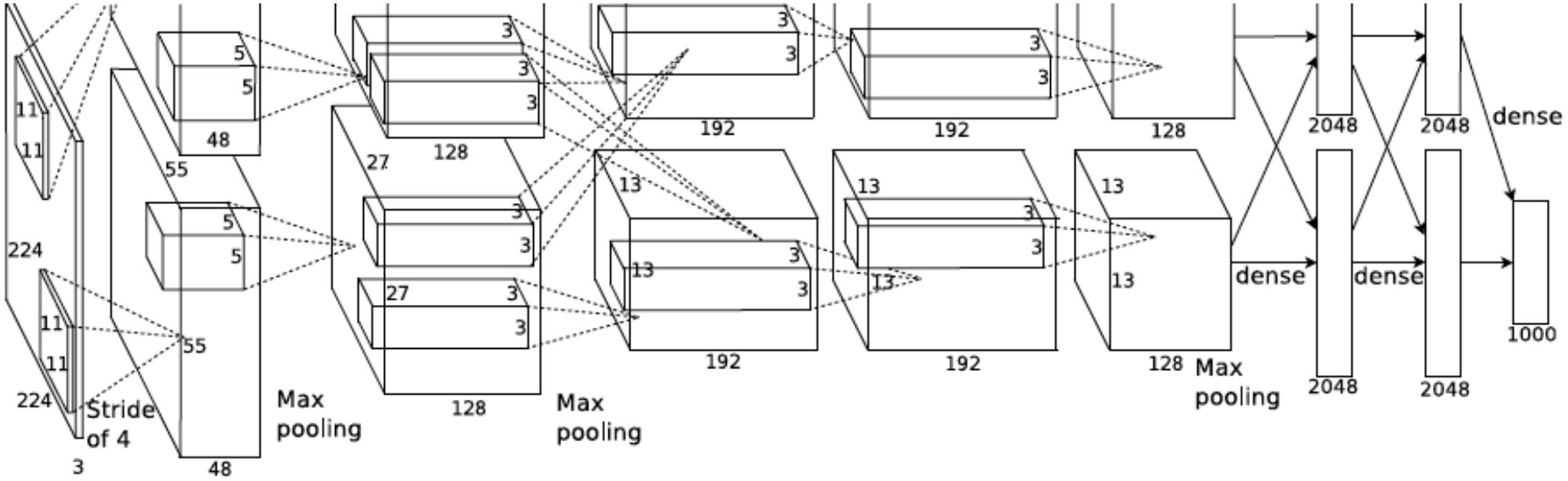
3. Learn better combination of cues, such as photometric score, geometric consistency, and surrounding predictions

4. Integrate multiview depth estimations and recognition

Deep networks: basic structures

- Image classification network (AlexNet shown)

Input image



Label confidences

Convolutional layers: aggregate/organize local information

Pooling: positional invariance

Fully connected (FC) layers : Integrate/organize all information

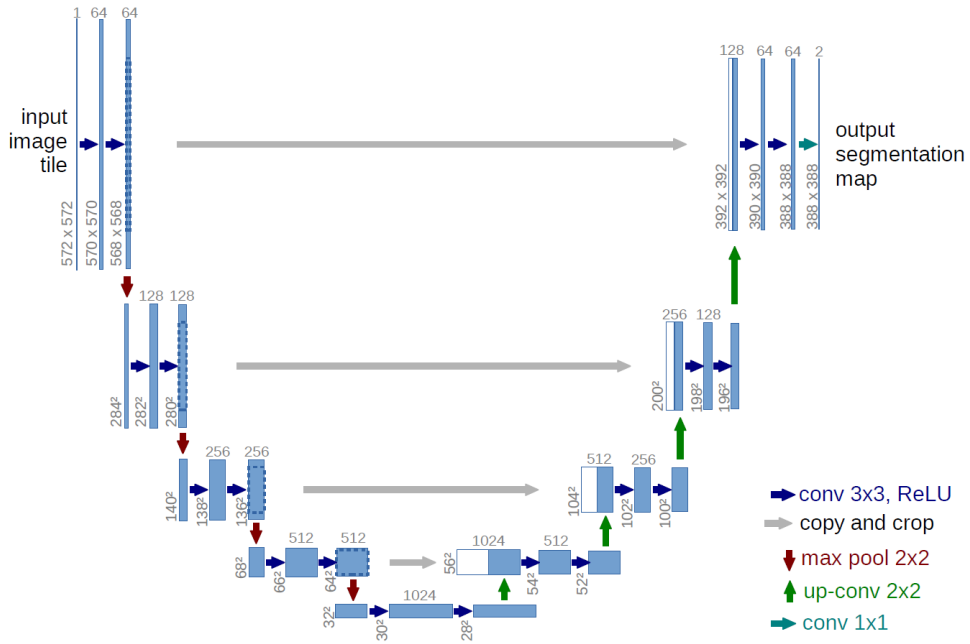
Prediction layer : Map final features to logistic scores

A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

Deep networks: basic structures

- Pixel labeling network (U-Net shown)

Input image



Segmentation map

Encoder:
convolve/pool to create feature image or vector that incorporates context

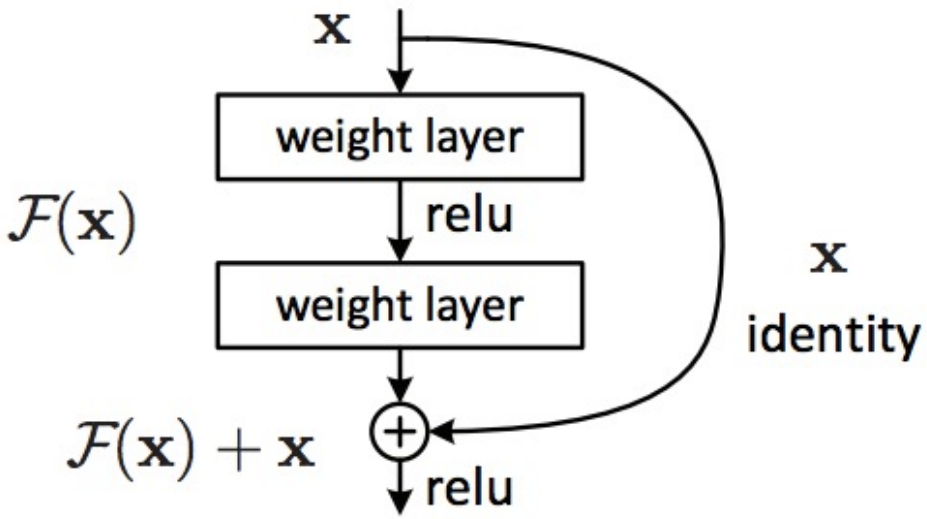
Decoder:
Upsample while combining with detail encoder layers to create features for local prediction

Prediction layer :
1x1 convolution to predict label at each position

O. Ronneberger, P. Fischer, and T. Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation, 2015

ResNet: the residual module

- Introduce *skip* or *shortcut* connections (existing before in various forms in literature)
- Make it easy for network layers to represent the identity mapping
- Sparser, more direct updates in training



Important but non-intuitive idea: to combine information, no need to concatenate – just add

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)

Transformers

- Can process an unordered set of tokens/vectors
  - Positional encoding adds spatial information
- Self-Attention is like clustering and replacing vectors with centers, except
  - Multiple, complex, learnable similarity functions
  - No need to preset number of clusters
- Vectors are iteratively aggregated and transformed
- Vectors can start as purely local information (e.g. 16x16 patch)

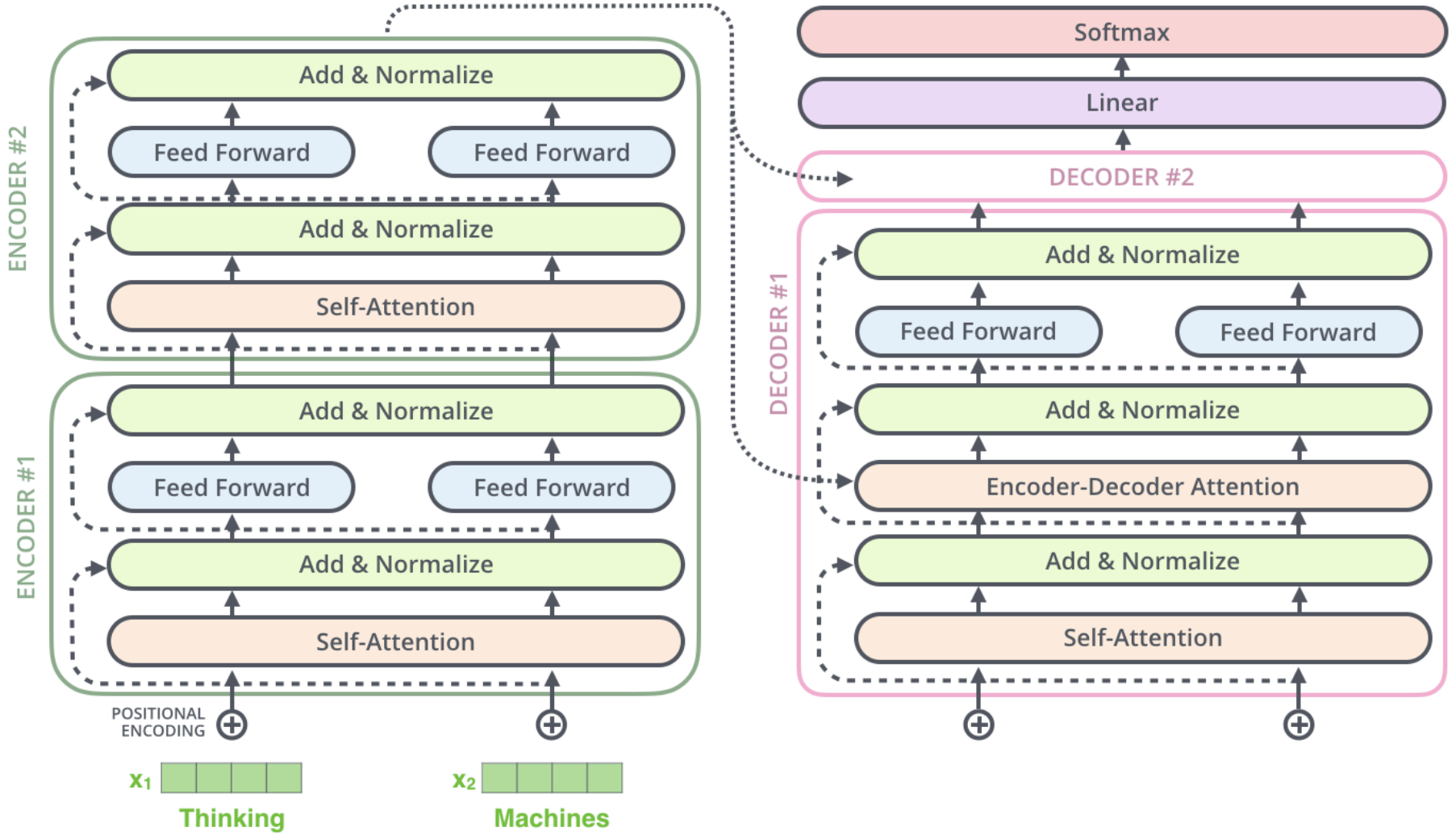
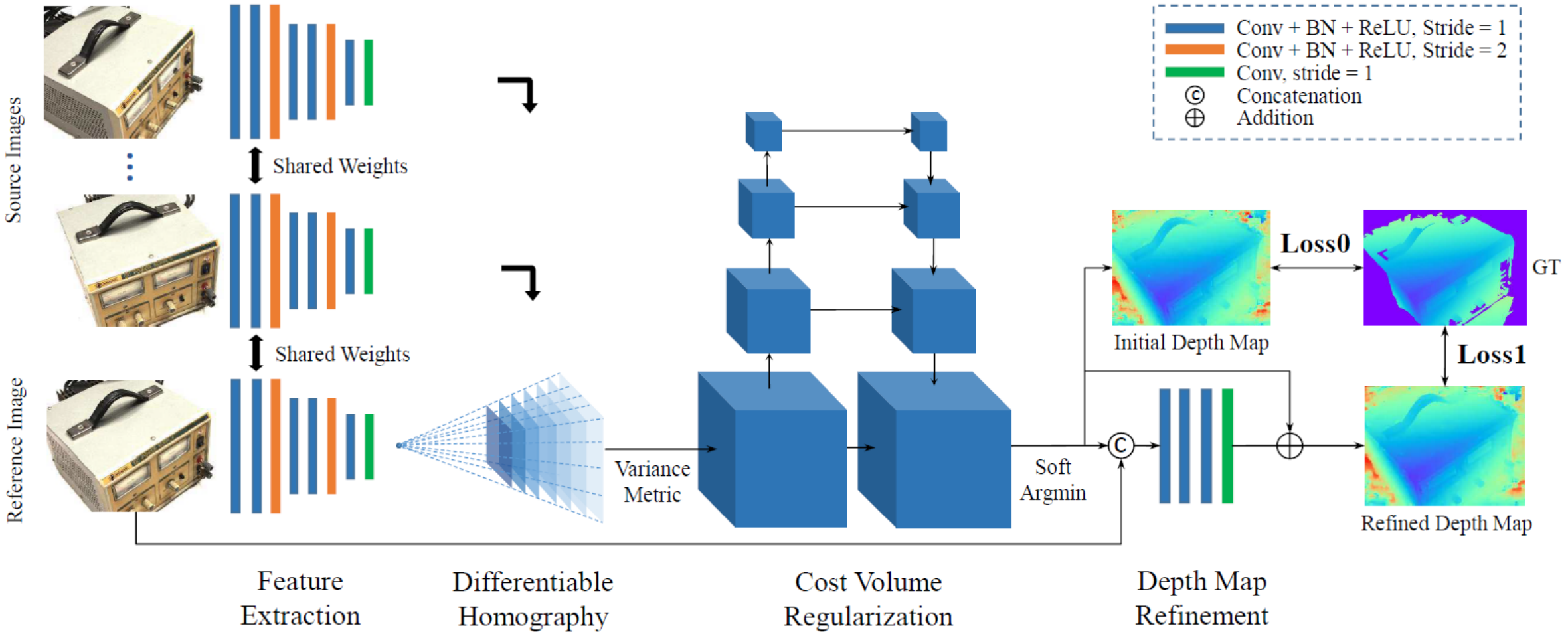
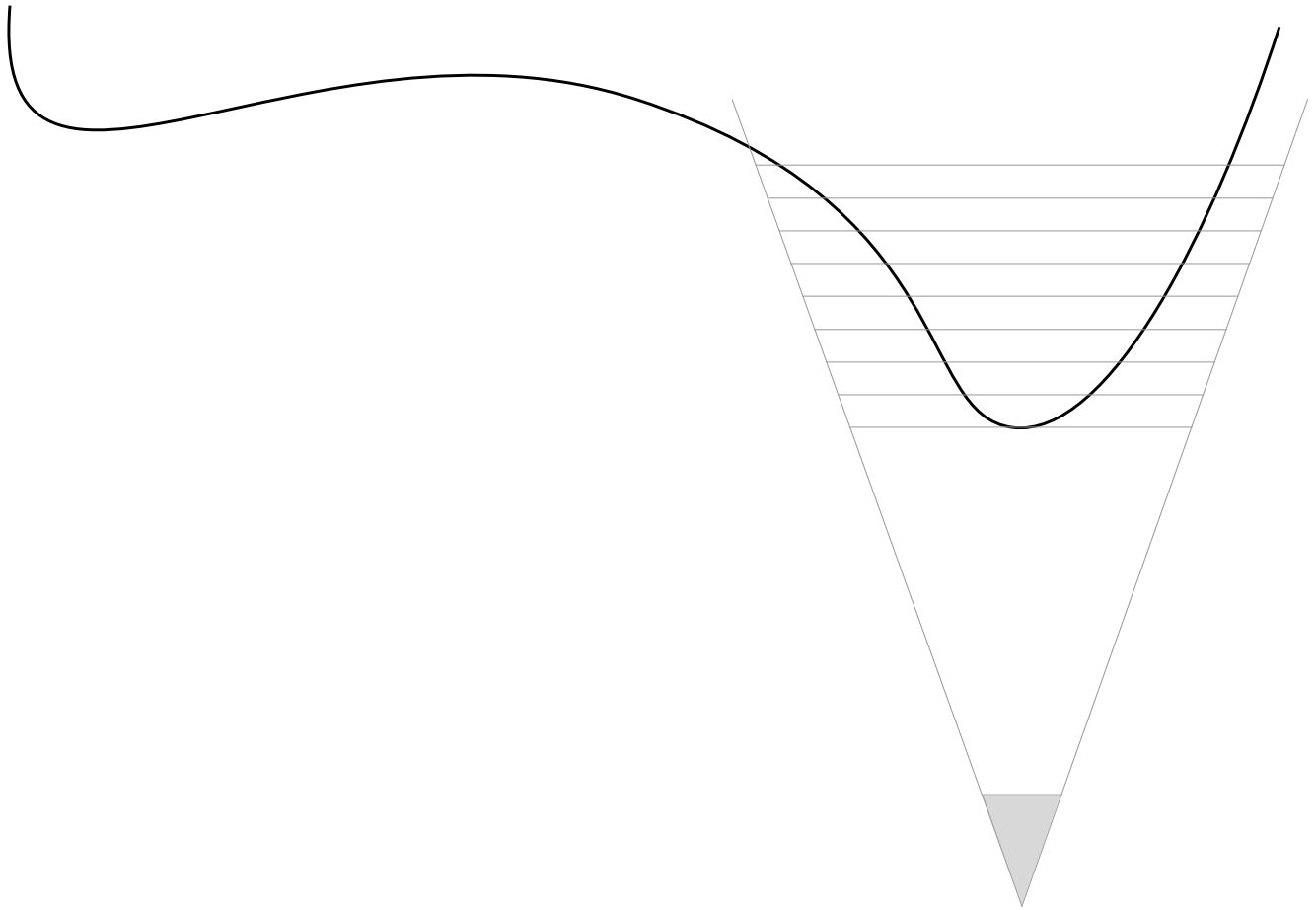


Figure from Ming Li. See full deck for details/illustrations.
https://cs.uwaterloo.ca/~mli/Lecture2.pptx

MVSNet (Yao et al. ECCV 2018)



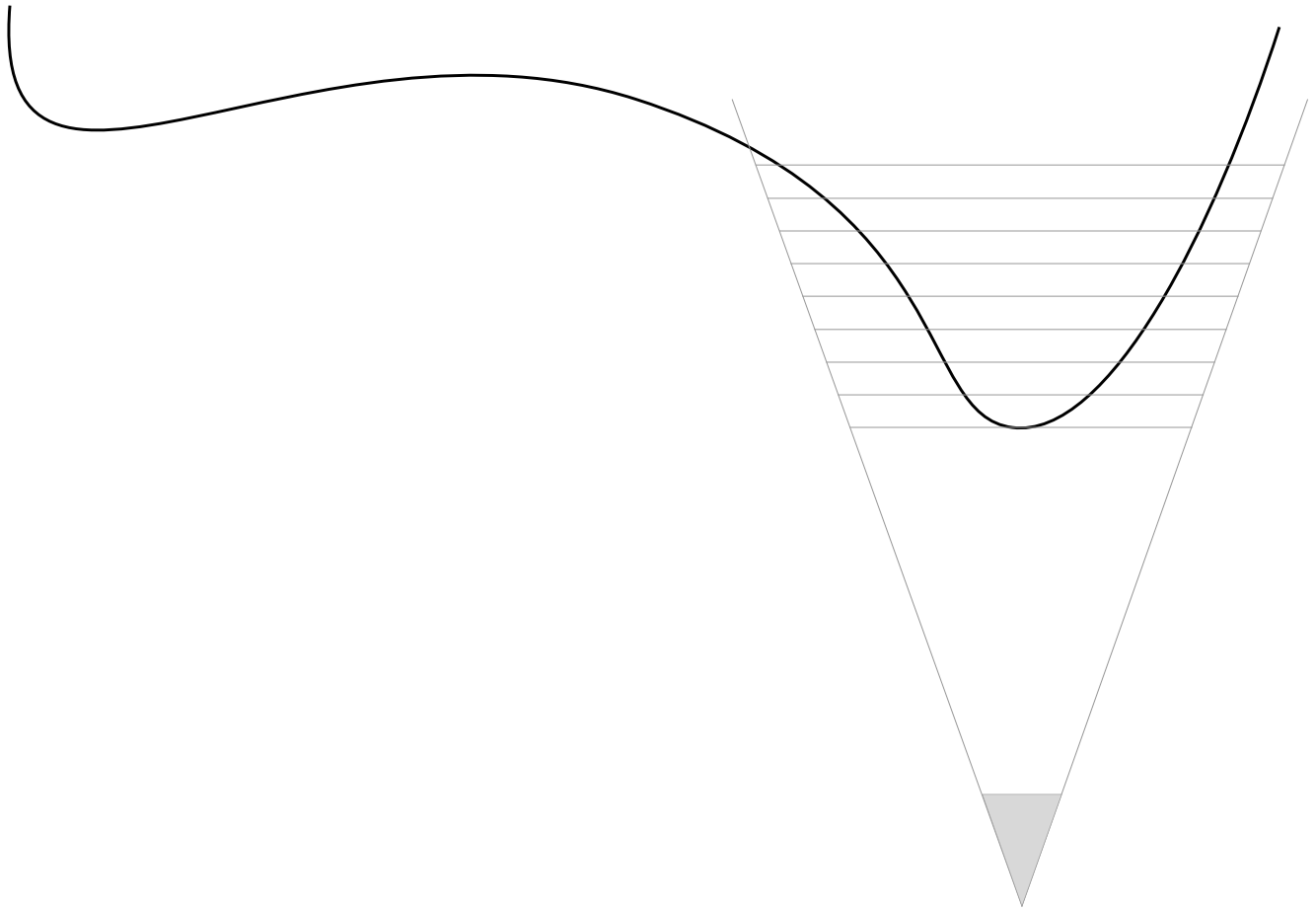
Plane Sweep Stereo



Plane Sweep Stereo

Slide credit: Jae Yong Lee

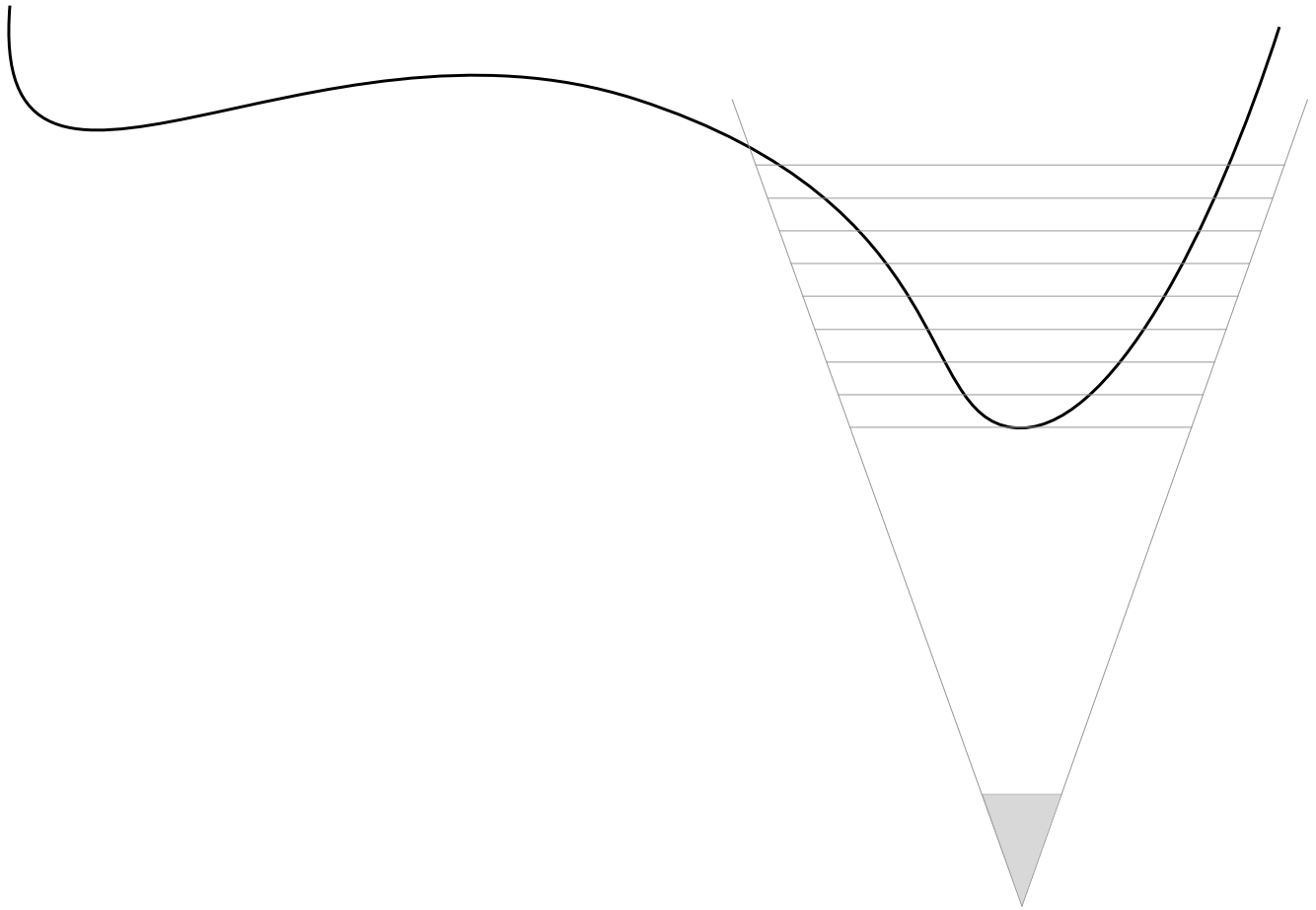
Plane Sweep Stereo



Plane Sweep Stereo

Slide credit: Jae Yong Lee

Plane Sweep Stereo



Plane Sweep Stereo

Slide credit: Jae Yong Lee

Plane Sweep Stereo

Slide credit: Jae Yong Lee

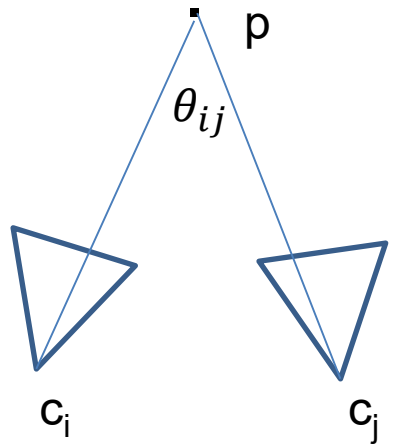
View Selection

- Score is weighted count of common sparse points, favoring those with triangulation angle close to 5 degrees

$$\theta_{ij}(\mathbf{p}) = (180/\pi)\arccos((\mathbf{c}_i - \mathbf{p}) \cdot (\mathbf{c}_j - \mathbf{p}))$$

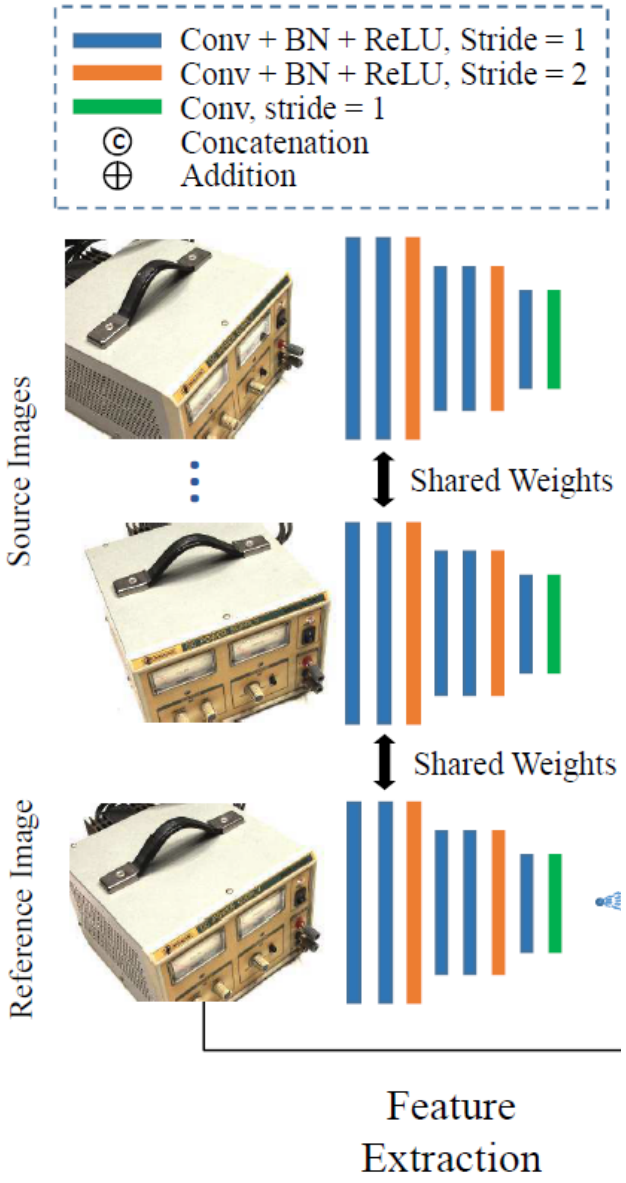
$$\mathcal{G}(\theta) = \begin{cases} \exp(-\frac{(\theta-\theta_0)^2}{2\sigma_1^2}), \theta \leq \theta_0 \\ \exp(-\frac{(\theta-\theta_0)^2}{2\sigma_2^2}), \theta > \theta_0 \end{cases} \qquad \begin{matrix} \theta_0 = 5 \\ \sigma_1 = 1 \\ \sigma_2 = 10 \end{matrix}$$

$$\text{score } s(i,j) = \sum_{\mathbf{p}} \mathcal{G}(\theta_{ij}(\mathbf{p}))$$



Photometric costs: features

- ConvNet: W/4 x H/4 x F features
  - Same # of values as W x H when F = 32
  - Same model extracts feature for each image
- BatchNorm and ReLU

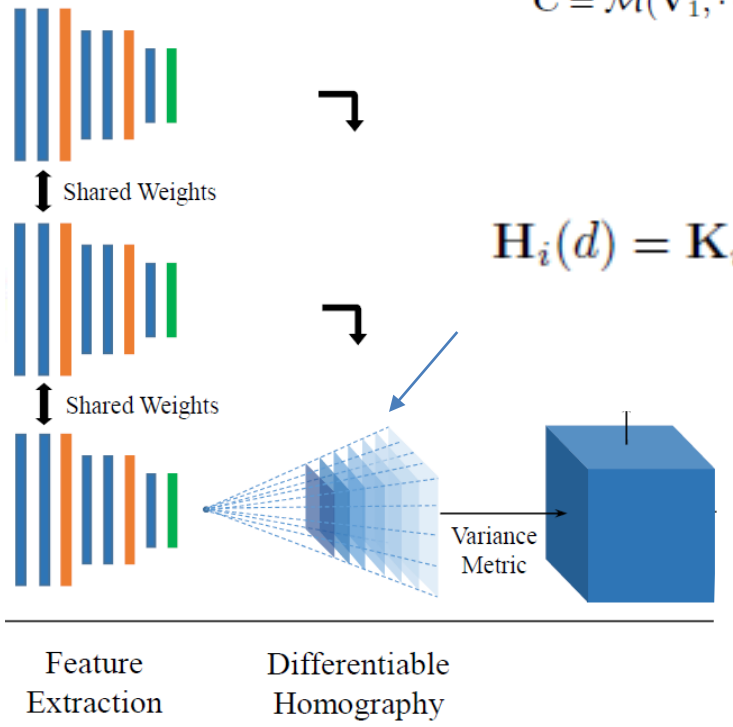


Photometric cost volume computation

- Homography maps from each reference depth plane to each view to obtain interpolated feature values (D = # depths = 256, N = # views = 5)
- Stack features into N volumes of size W/4 x H/4 x D x F
- Compute **variance** over volumes to get W/4 x H/4 x D x F photometric score volume

$$\mathbf{C} = \mathcal{M}(\mathbf{V}_1, \cdots, \mathbf{V}_N) = \frac{\sum_{i=1}^{N} (\mathbf{V}_i - \overline{\mathbf{V}_i})^2}{N}$$

$$\mathbf{H}_i(d) = \mathbf{K}_i \cdot \mathbf{R}_i \cdot \left(\mathbf{I} - \frac{(\mathbf{t}_1 - \mathbf{t}_i) \cdot \mathbf{n}_1^T}{d}\right) \cdot \mathbf{R}_1^T \cdot \mathbf{K}_1^T$$

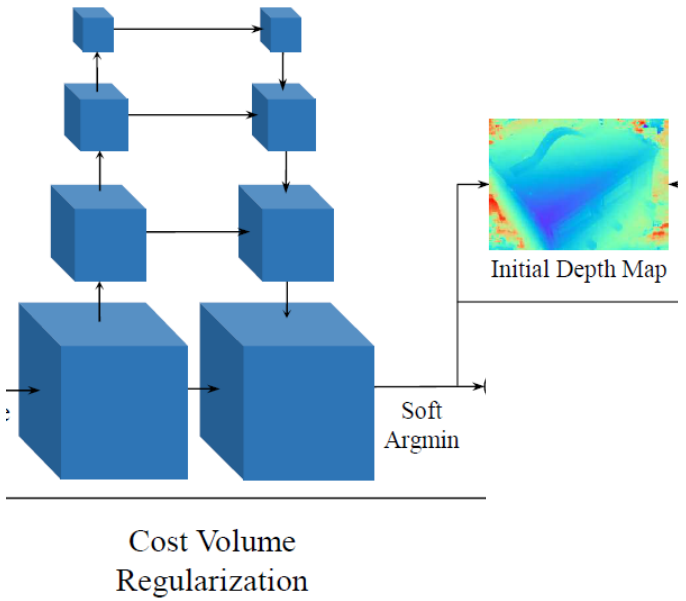


What do you think of feature variance vs. NCC with reference image?

Cost volume regularization and initial depth estimation

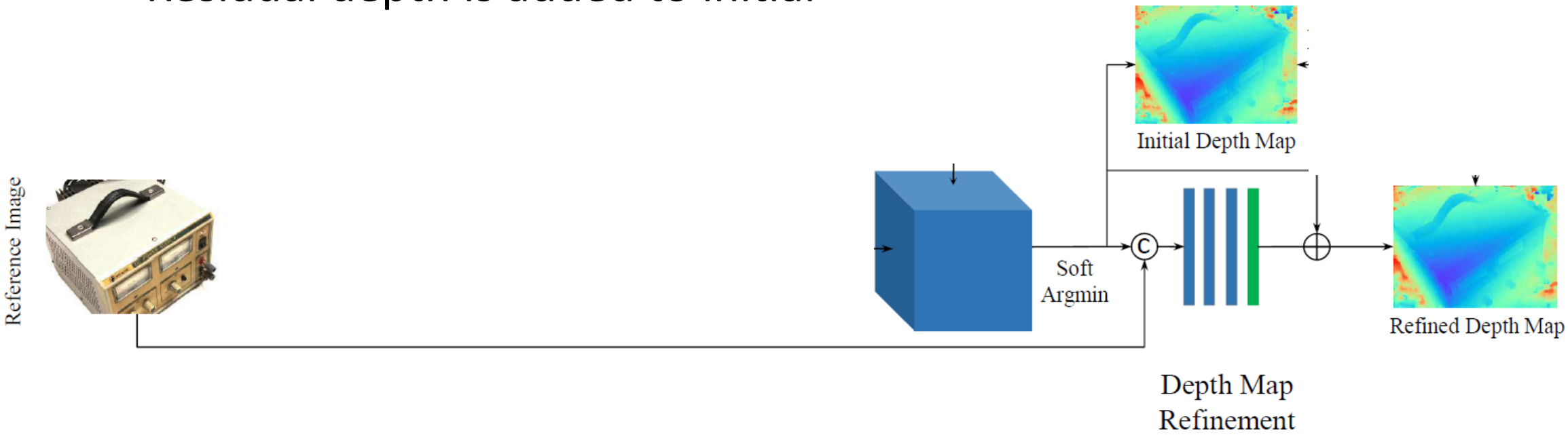
- 3D UNet: compress while accumulating spatial context and then uncompress with skip connections
- Softmax in depth direction results in W/4 x H/4 x D probability volume P
  - P(x, y, d) is probability that depth at coordinate (x,y) had depth d
- Compute expectation of depth to get initial estimate

$$\mathbf{D} = \sum_{d=d_{min}}^{d_{max}} d \times \mathbf{P}(d)$$



Depth refinement

- Predict a residual depth from reference image and initial depth map
  - Attempts to refine around boundaries
  - Residual depth is added to initial

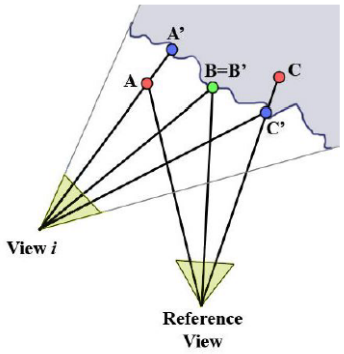


Filtering and Fusion

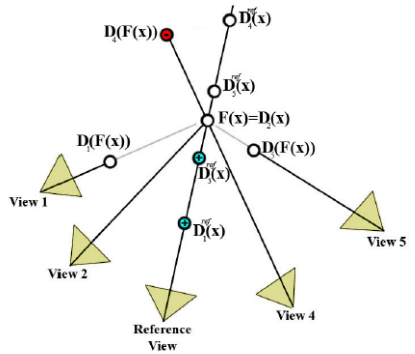
- Keep points that have:
  - probability of at least 0.8
  - low reprojection error and low reprojection depth difference with at least two other images

$$|p_{reproj} - p_1| < 1 \qquad |d_{reproj} - d_1|/d_1 < 0.01$$ (Note: depth ratio)

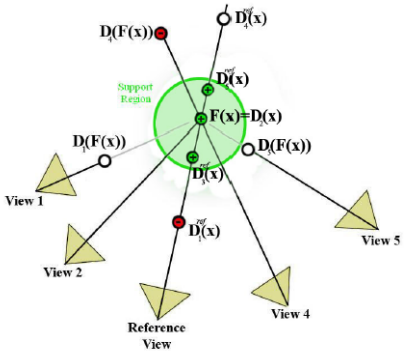
- Visibility-based fusion (Merrell et al. ICCV 2007)
  - Efficient GPU implementation



(a) Visibility relations between points



(b) Stability calculation

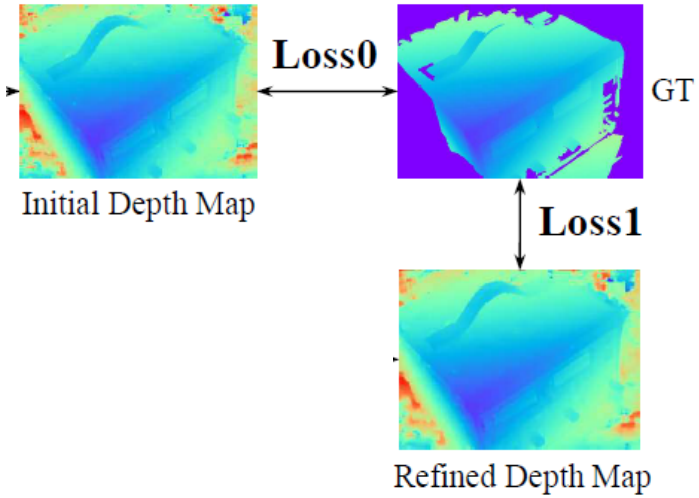


(c) Support estimation

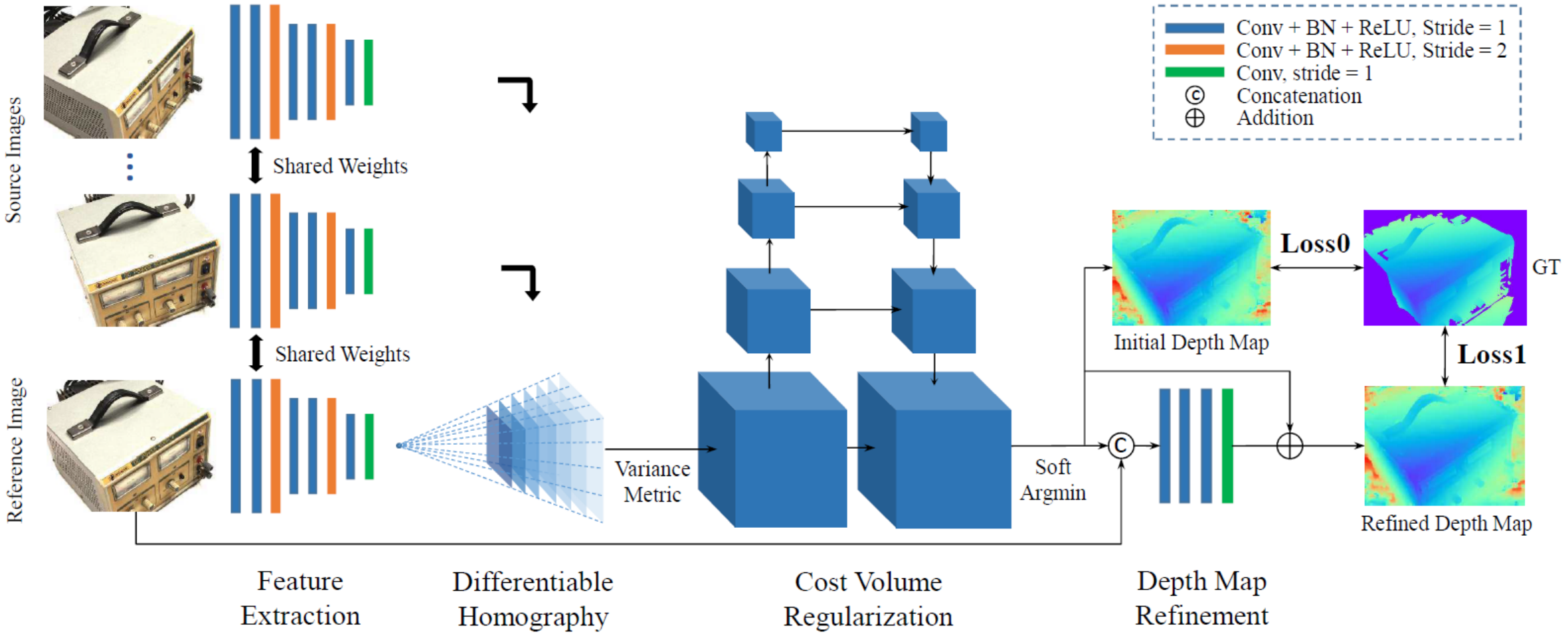
Training

- Loss on initial and refined maps
- N = 3, W=640, H=512 for training
- Ground truth from DTU depth-rendered meshes

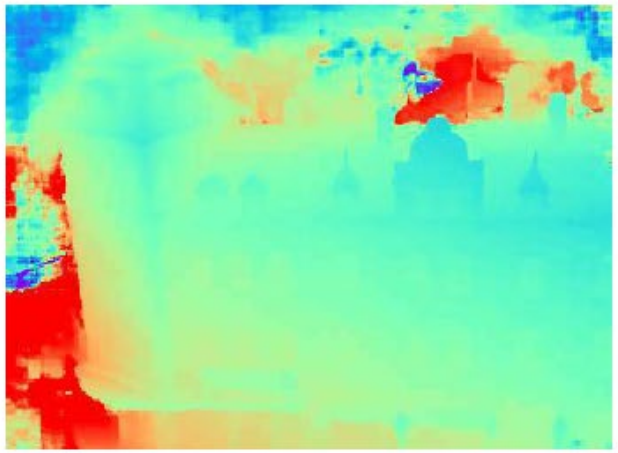
$$Loss = \sum_{p \in \mathbf{p}_{valid}} \underbrace{\|d(p) - \hat{d}_i(p)\|_1}_{Loss0} + \lambda \cdot \underbrace{\|d(p) - \hat{d}_r(p)\|_1}_{Loss1}$$



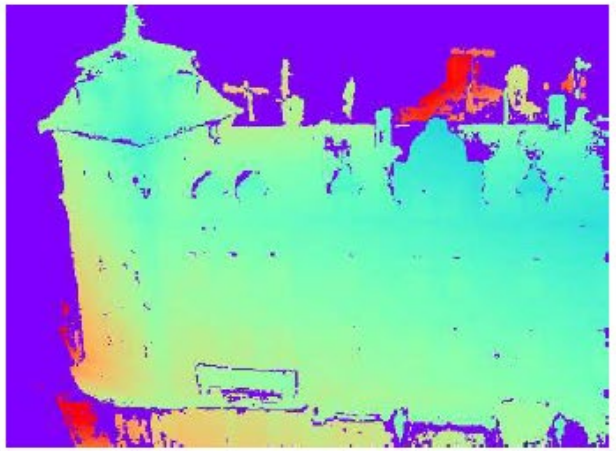
Recap


Source Images
Reference Image
Shared Weights
Shared Weights
Conv + BN + ReLU, Stride = 1
Conv + BN + ReLU, Stride = 2
Conv, stride = 1
Concatenation
Addition
Variance Metric
Soft Argmin
Initial Depth Map
Loss0
GT
Loss1
Refined Depth Map
Feature Extraction
Differentiable Homography
Cost Volume Regularization
Depth Map Refinement

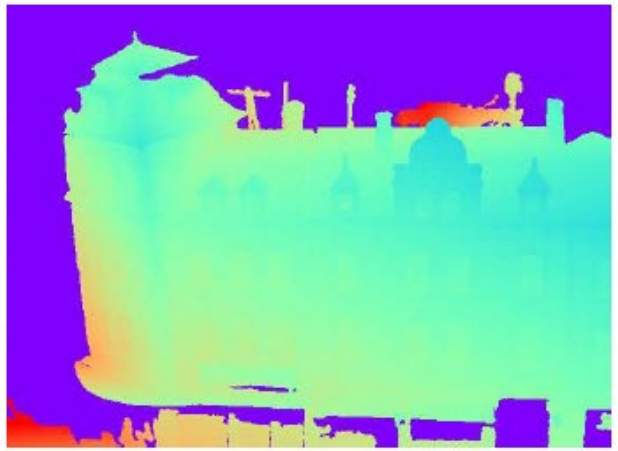
Results: example



(a) Inferred depth map



(b) Filtered depth map



(c) GT depth map

(d) Reference image

(e) Fused point cloud

(f) GT point cloud

Table 1: Quantitative results on the DTU's evaluation set [1]. We evaluate all methods using both the distance metric [1] (lower is better), and the percentage metric [18] (higher is better) with respectively $1mm$ and $2mm$ thresholds

	Mean Distance (mm)			Percentage ($<1mm$)			Percentage ($<2mm$)		
	Acc.	Comp.	*overall*	Acc.	Comp.	*f-score*	Acc.	Comp.	*f-score*
Camp [3]	0.835	0.554	0.695	71.75	64.94	66.31	84.83	67.82	73.02
Furu [7]	0.613	0.941	0.777	69.55	61.52	63.26	78.99	67.88	70.93
Tola [35]	0.342	1.190	0.766	90.49	57.83	68.07	93.94	63.88	73.61
Gipuma [8]	**0.283**	0.873	0.578	**94.65**	59.93	70.64	**96.42**	63.81	74.16
SurfaceNet[14]	0.450	1.04	0.745	83.8	63.38	69.95	87.15	67.99	74.4
MVSNet (Ours)	0.396	**0.527**	**0.462**	86.46	**71.13**	**75.69**	91.06	**75.31**	**80.25**

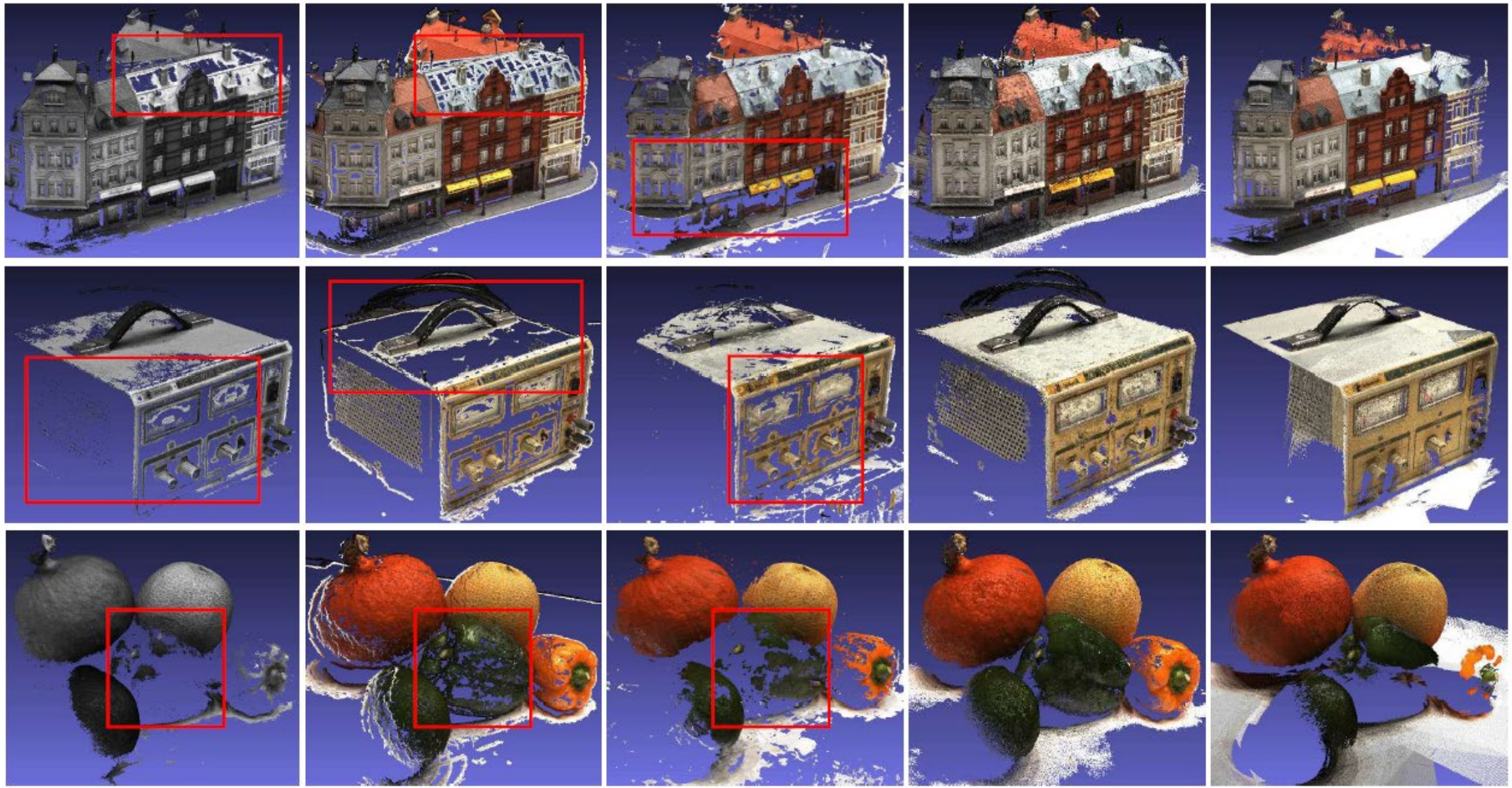

Scan 9
Scan 11
Scan 75
Gipuma
PMVS
SurfaceNet
MVSNet (Ours)
Gound Truth

Table 2: Quantitative results on *Tanks and Temples* benchmark [18]. MVSNet achieves best *f-score* result among all submissions without any fine-tuning

Method	Rank	Mean	Family	Francis	Horse	Lighthouse	M60	Panther	Playground	Train
MVSNet (Ours)	**3.00**	**43.48**	55.99	28.55	25.07	50.79	**53.96**	**50.86**	47.90	34.69
Pix4D [30]	3.12	43.24	**64.45**	31.91	**26.43**	54.41	50.58	35.37	47.78	34.96
COLMAP [32]	3.50	42.14	50.41	22.25	25.63	**56.43**	44.83	46.97	**48.53**	**42.04**
OpenMVG [27] + OpenMVS [29]	3.62	41.71	58.86	**32.59**	26.25	43.12	44.73	46.85	45.97	35.27
OpenMVG [27] + MVE [6]	6.00	38.00	49.91	28.19	20.75	43.35	44.51	44.76	36.58	35.95
OpenMVG [27] + SMVS [21]	10.38	30.67	31.93	19.92	15.02	39.38	36.51	41.61	35.89	25.12
OpenMVG-G [27] + OpenMVS [29]	10.88	22.86	56.50	29.63	21.69	6.55	39.54	28.48	0.00	0.53
MVE [6]	11.25	25.37	48.59	23.84	12.70	5.07	39.62	38.16	5.81	29.19
OpenMVG [27] + PMVS [7]	11.88	29.66	41.03	17.70	12.83	36.68	35.93	33.20	31.78	28.10

(a) *Family*

(b) *Panther*

(c) *Horse*

(d) *Playground*

(e) *Francis*

(f) *Train*

(g) *Lighthouse*

(h) *M60*

Runtime: 4.7s per view

- 100x faster than COLMAP, 5x faster than Gipuma

2 min break (and think)

- Compared to non-ML MVS algorithms, how is this one better?
- How is it worse?

Pros and Cons of MVSNet

- Pros
  - Fast and relatively simple
  - Learnable features
  - Good completeness
- Cons
  - Loses benefits of pixelwise view selection and normal estimates
  - Requires dense views (lack of pixelwise view selection) and small depth range (cost volume)
  - (Maybe) depth expectation can lead to inaccurate estimates

State-of-the-art works in MVS

Model	F1-Score
Vis-MVSNet (BMVC 2020)	60.03
AttMVSNet (CVPR 2020)	60.05
…	
ACMP (AAAI 2020)	58.41
ACMM (CVPR 2019)	57.27

Tanks and Temples
Intermediate Benchmark

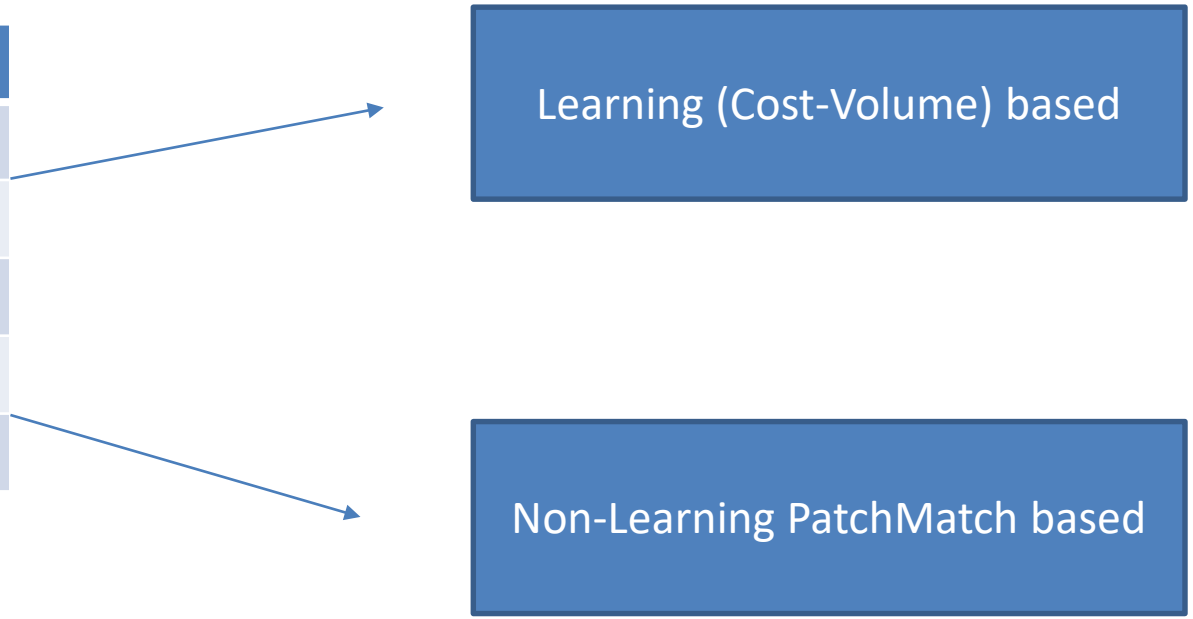
Dense views for object scenes

Slide credit: Jae Yong Lee

State-of-the-art works in MVS

Model	F1-Score
Vis-MVSNet (BMVC 2020)	60.03
AttMVSNet (CVPR 2020)	60.05
…	
ACMP (AAAI 2020)	58.41
ACMM (CVPR 2019)	57.27

Tanks and Temples
Intermediate Benchmark



Slide credit: Jae Yong Lee

In more Challenging Benchmarks…

Model	F1-Score
MARMVS (CVPR2020)	81.84
ACMP (AAAI 2020)	81.51
ACMM (CVPR 2019)	80.78
…	
Vis-MVSNet (BMVC 2020)	78.36
PVSNet	72.08

ETH3D High-Res Benchmark

With wide baselines with strict thresholds

Slide credit: Jae Yong Lee

In more Challenging Benchmarks…

Model	F1-Score
MARMVS (CVPR2020)	81.84
ACMP (AAAI 2020)	81.51
ACMM (CVPR 2019)	80.78
…	
Vis-MVSNet (BMVC 2020)	78.36
PVSNet	72.08

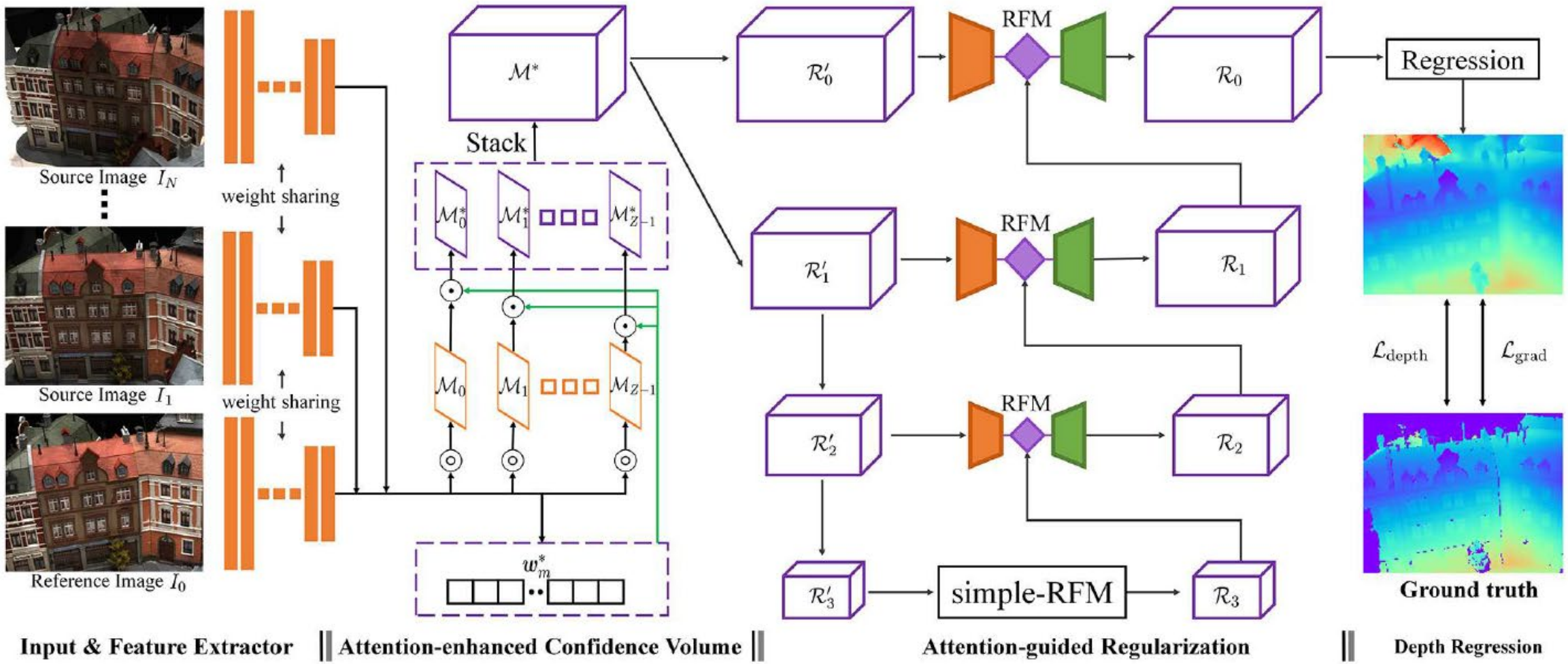
ETH3D High-Res Benchmark

Model	F1-Score
ACMP (AAAI 2020)	37.44
ACMM (CVPR 2019)	34.02
ACMH (CVPR 2019)	33.73
…	
AttMVSNet (CVPR 2020)	31.93
CasMVSNet (CVPR 2020)	31.12

Tanks and Temples Advanced Benchmark

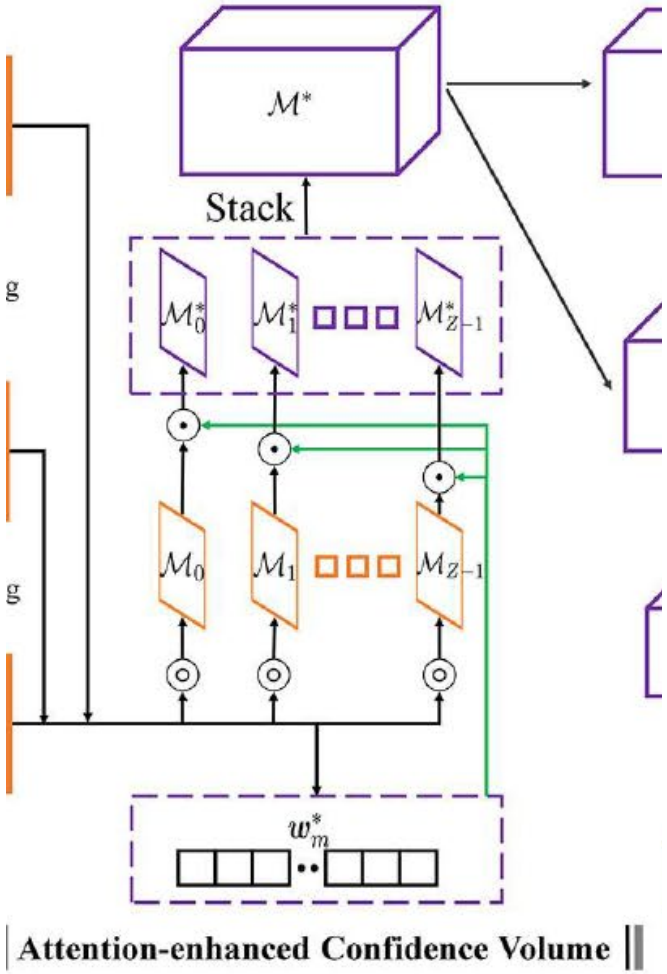
Slide credit: Jae Yong Lee

Attention-Aware MVS (Luo et al. CVPR 2020)



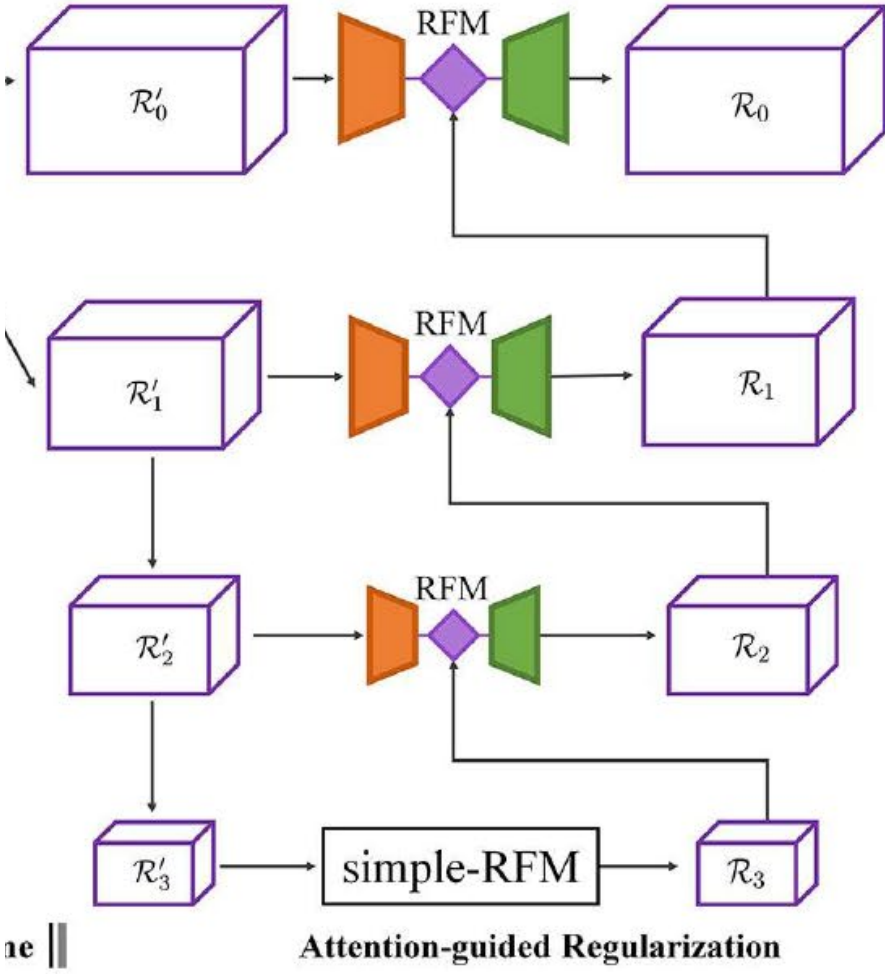
Attention-enhanced confidence volume

- Features: like MVSNet but 10 layers, 16 channels, LeakyReLU, InstanceNorm
- Predict “attention” weights based on variance of average feature channels across images
- Feature confidence channels based on mean squared diff of sources with ref along each channel $\mathcal{M}(d, \boldsymbol{p}, c) = \exp\left(-\frac{\sum_{j=1}^{N-1}(F_j(\boldsymbol{p}', c) - F_0(\boldsymbol{p}, c))^2}{N-1}\right)$
- Sum feature confidence channels weighted by attention weights
- Allows scene-specific feature importance



Attention-guided hierarchical regularization

- Create multiscale cost volume by stacking feature x depth channels and using stride=2 to downsample
- Confidence volumes are processed with 3D convolution and linear layers at multiple scales

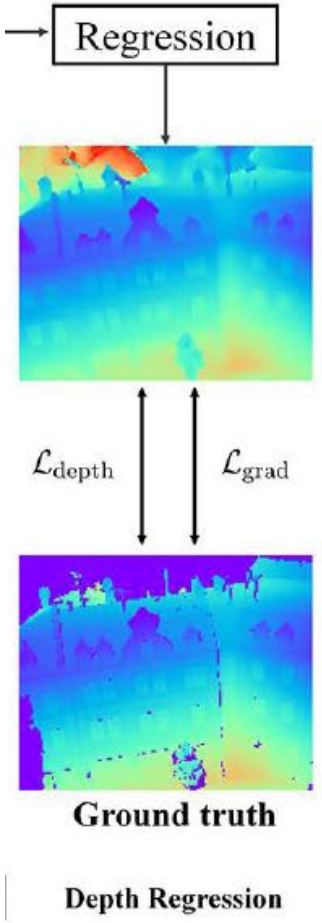


Depth regression

- Compute expected depth like MVSNet
- In training, optimize over relative depth and depth gradient (train on DTU w/ improved GT maps)

$$\mathcal{L}_{\text{depth}}(d^*, \hat{d}) = \frac{1}{\delta \mathcal{N}_d} \sum_{(i,j)} \left| d^*_{i,j} - \hat{d}_{i,j} \right|$$

$$\mathcal{L}_{\text{grad}}(d^*, \hat{d}) = \sum_{(i,j)} \left(\frac{1}{\mathcal{N}_x} \left| \varphi_x(d^*_{i,j}) - \varphi_x(\hat{d}_{i,j}) \right| + \frac{1}{\mathcal{N}_y} \left| \varphi_y(d^*_{i,j}) - \varphi_y(\hat{d}_{i,j}) \right| \right)$$



Ablation

Table 1: Comparison results of the proposed AttMVS with different model variants on the DTU validation set.

Models	Settings					MADE	Pred. prec. ($\tau = \delta$)	Pred. prec. ($\tau = 3\delta$)
	Mod. fea. extr.	Att MCV	Simple-RFM	RFMs	Joint loss			
Baseline						2.14	83.11	95.77
Model-A	√					1.96	84.57	96.25
Model-B	√	√				1.91	84.98	96.36
Model-C	√	√	√			1.89	85.64	96.45
Model-D	√	√	√	√		1.82	87.08	96.84
Full	√	√	√	√	√	**1.79**	**87.61**	**97.04**

Table 2: Comparisons on the recovered three-dimensional models for the DTU evaluation scenes by different methods. AttMVS* denotes inclusion of the refinement of the depth maps by (9).

Method	Mean accuracy	Mean completeness	Overall
Gipuma [11]	**0.274**	1.193	0.734
tola [34]	0.343	1.190	0.767
furu [10]	0.612	0.939	0.776
camp [5]	0.836	0.555	0.696
SurfaceNet [16]	0.450	1.043	0.746
MVSNet [41]	0.396	0.527	0.462
R-MVSNet [42]	0.385	0.459	0.422
Point-MVSNet [7]	0.342	0.411	0.376
P-MVSNet [25]	0.406	0.434	0.420
AttMVS ($Z = 256$)	0.412	0.394	0.403
AttMVS ($Z = 384$)	0.391	**0.345**	**0.368**
AttMVS* ($Z = 384$)	0.383	**0.329**	**0.356**

(a) Reference image

(b) MVSNet

(c) P-MVSNet

(d) AttMVS

Table 3: Performance comparisons of various reconstruction algorithms on the *intermediate sequences* of the Tanks & Temples benchmark. Our AttMVS ranks 1st among all of the submissions.

Method	Rank	Mean	Family	Francis	Horse	Lighthouse	M60	Panther	Playground	Train
AttMVS (**Ours**)	**2.38**	**60.05**	73.90	**62.58**	44.08	64.88	56.08	59.39	**63.42**	**56.06**
Altizure-HKUST-2019 [3]	4.00	59.03	**77.19**	61.52	42.09	63.50	59.36	58.20	57.05	53.30
3Dnovator [1]	4.62	58.37	73.43	52.51	37.08	64.55	**59.58**	**62.88**	62.88	51.40
ACMM [40]	6.12	57.27	69.24	51.45	**46.97**	63.20	55.07	57.64	60.08	54.48
Altizure-SFM, PCF-MVS [21]	7.38	55.88	70.99	49.60	40.34	63.44	57.79	58.91	56.59	49.40
OpenMVS [28]	7.75	55.11	71.69	51.12	42.76	58.98	54.72	56.17	59.77	45.69
P-MVSNet [25]	7.75	55.62	70.04	44.64	40.22	**65.20**	55.08	55.17	60.37	54.29
ACMH [39]	9.75	54.82	69.99	49.45	45.12	58.86	52.64	52.37	58.34	51.61
PLC_ [23]	10.62	54.56	70.09	50.30	41.94	59.04	49.19	55.53	56.41	54.13
Point-MVSNet [7]	18.25	48.27	61.79	41.15	34.20	50.79	51.97	50.85	52.38	43.06
Dense R-MVSNet [42]	18.38	50.55	73.01	54.46	43.42	43.88	46.80	46.69	50.87	45.25
R-MVSNet [42]	21.50	48.40	69.96	46.65	32.59	42.95	51.88	48.80	52.00	42.38
MVSNet [41]	27.88	43.48	55.99	28.55	25.07	50.79	53.96	50.86	47.90	34.69
COLMAP [31, 32]	30.12	42.14	50.41	22.25	25.63	56.43	44.83	46.97	48.53	42.04

Table 4: Performance comparisons of various reconstruction approaches on the *advanced sequences* of the Tanks & Temples benchmark.

Method	Rank	Mean	Auditorium	Ballroom	Courtroom	Museum	Palace	Temple
Altizure-HKUST-2019 [3]	**3.17**	**37.34**	24.04	**44.52**	36.64	49.51	**30.23**	**39.09**
Altizure-SFM, PCF-MVS [21]	4.33	35.69	**28.33**	38.64	35.95	48.36	26.17	36.69
OpenMVS [28]	5.50	34.43	24.49	37.39	38.21	47.48	27.25	31.79
3Dnovator [1]	5.67	34.51	18.61	40.77	37.17	50.30	27.60	32.61
PLC_ [23]	5.83	34.44	23.02	30.95	42.50	49.61	26.09	34.46
COLMAP-SFM, PCF-MVS [21]	6.17	34.59	26.87	31.53	**44.70**	47.39	24.05	32.97
ACMM [40]	6.33	34.02	23.41	32.91	41.17	48.13	23.87	34.60
AttMVS (**Ours**)	8.00	31.93	15.96	27.71	37.99	**52.01**	29.07	28.84
Dense R-MVSNet [42]	11.83	29.55	19.49	31.45	29.99	42.31	22.94	31.10
R-MVSNet [42]	15.67	24.91	12.55	29.09	25.06	38.68	19.14	24.96

Summary

- Deep MVS applies learned features and cost volume regularization
- Outperforms for dense views, moderate scene depth
- Underperforms non-ML methods for sparse views, large scene depths
- Deep patch-match based methods try to address this and are catching up to non-ML methods, but so far ACMMP