Multiview Stereo with Cascaded Epipolar RAFT

Zeyu Ma
Princeton University
zeyum@princeton.edu

Zachary Teed
Princeton University
zteed@princeton.edu

Jia Deng
Princeton University
jiadeng@princeton.edu

Abstract

We address multiview stereo (MVS), an important 3D vision task that reconstructs a 3D model such as a dense point cloud from multiple calibrated images. We propose CER-MVS (Cascaded Epipolar RAFT Multiview Stereo), a new approach based on the RAFT (Recurrent All-Pairs Field Transforms) architecture developed for optical flow. CER-MVS introduces five new changes to RAFT: epipolar cost volumes, cost volume cascading, multiview fusion of cost volumes, dynamic supervision, and multiresolution fusion of depth maps. CER-MVS is significantly different from prior work in multiview stereo. Unlike prior work, which operates by updating a 3D cost volume, CER-MVS operates by updating a disparity field. Furthermore, we propose an adaptive thresholding method to balance the completeness and accuracy of the reconstructed point clouds. Experiments show that our approach achieves competitive performance on DTU (the second best among published results) and state-of-the-art performance on the Tanks-and-Temples benchmark (both the intermediate and advanced set). Code is available at https://github.com/princeton-vl/CER-MVS

1. Introduction

Multiview stereo (MVS) is an important task in 3D computer vision. It seeks to reconstruct a full 3D model, typically in the form of a dense 3D point cloud, from multiple RGB images with known camera intrinsics and poses. It is a difficult task that remains unsolved; the main challenge is producing a 3D model that is not only accurate but also complete, that is, no parts should be missing and all fine details should be recovered.

Many of the latest results of multiview stereo are achieved by deep networks. In particular, many recent leading methods [32, 42] are variants of MVSNet [34], a deep architecture that consists of two main steps: (1) constructing a 3D cost volume in the frustum of a reference view, by warping features from other views, and (2) using 3D convolutional layers to transform, or "regularize", the cost volume before using it to predict a depth map. The resulting depth maps, one from each reference view, are then combined to form a single 3D point cloud through a heuristic procedure.

However, a drawback of MVSNet is that regularizing the 3D plane-sweeping cost volume using 3D convolutions can be costly in terms of computation and memory, potentially limiting the quality of reconstruction under finite resources. Subsequent variants [35] of MVSNet have attempted to address this issue by replacing 3D convolutions with recurrent sequential processing of 2D slices. Despite significant empirical improvements, however, such sequential processing can be suboptimal because the 3D cost volume does not have a natural sequential structure.

In this work, we propose CER-MVS, a new deep-learning multiview stereo approach that is significantly different from existing methods. Like prior deep-learning work on multiview stereo, CER-MVS predicts individual depth maps and then fuses them, but differs significantly in how it predicts each depth map. Given a reference view and multiple neighbor views, CER-MVS constructs a 3D cost volume for each neighbor view by computing the similarity between each pixel in the reference view and pixels along the epipolar line, indexed by increments of inverse depth (i.e. disparity) in the reference view. Then, the cost volumes from all neighbor views are aggregated into a single cost volume. CER-MVS uses a GRU to iteratively update a disparity field—the field that represents pixel correspondence. Each update is generated by the GRU by sampling from the aggregated cost volume using the current disparity field.

The key difference of CER-MVS from MVSNet and its variants lies in how depth is predicted from the 3D cost volume. MVSNet updates (i.e. regularizes) the 3D cost volume and predicts depth through a soft argmax on the updated cost volume. In contrast, CER-MVS does not update the cost volume at all; instead it iteratively updates a disparity field, which is used to retrieve values from the cost volume. The final depth prediction is simply the inverted disparity field. Updating a disparity field, which is less expensive than updating the cost volume, can allow more effective use of finite computing resources.

CER-MVS builds upon RAFT [25], an architecture that

estimates optical flow between two video frames. Compared to RAFT, which cannot be directly applied to multiview stereo, CER-MVS introduces four novel changes:

- *Epipolar cost volume:* RAFT constructs a 4D cost volume that compares all pairs of pixels from two views, whereas we construct a 3D cost volume comparing each pixel in the reference view with pixels which are on the epipolar line in a neighbor view and spaced by uniform increments of disparity.

- *Cost volume cascading:* Unlike RAFT, the size of our epipolar cost volumes depends not only on the image resolution but also the number of disparity increments. To reconstruct fine details, a large number of disparity increments is necessary, but can blow up GPU memory. To address this issue, we introduce cascaded epipolar cost volumes, a novel design in the context of RAFT. In particular, after a fixed number of RAFT iterations, we construct additional finer-grained epipolar cost volumes centered around current disparity predictions with finer increments of disparity, allowing reconstruction of fine details with less memory.

- *Multiview fusion of cost volumes:* RAFT constructs a single cost volume from two views, whereas CER-MVS constructs multiple cost volumes, one for each neighbor of a reference view. The cost volumes are then aggregated into a single volume through a simple averaging operator.

- *Dynamic supervision:* RAFT uses exponentially decaying weights to add up flow errors in each iteration. We also use such weights, but supervise a dynamic combination of depth errors and disparity errors.

- *Multiresolution fusion of depth maps:* RAFT operates on a single resolution of the input images, whereas CER-MVS applies the same network to predict depth maps on multiple resolutions, and aggregate the depth maps into a single high-resolution depth map through a simple but novel heuristic.

When stitching the depth maps into point clouds, a filtering algorithm is often used, e.g., Dynamic Consistency Checking proposed in D2HC-RMVSNet [32]. However, a good balance of accuracy and completeness is required for high scores on the evaluation metric, which is ignored by these algorithms. Therefore, we propose an adaptive thresholding method built on top of [32].

We evaluate CER-MVS on two challenging benchmarks, DTU [2] and Tanks-and-Temples [15]. On DTU, CER-MVS achieves performance competitive to the current state of the art (the second best among published results). On Tanks-and-Temples, CER-MVS significantly advances the state of the art of the intermediate set from a mean F1 score of 61.68 to 64.82, and the advanced set from 37.44 to 40.19.

2. Related Work

Classical MVS Classical methods [4, 8, 9, 12, 22, 26] essentially formulate multiview stereo as an optimization problem, which seeks to find a 3D model that is most compatible with the observed images. The compatibility is typically based on some hand-designed notion of photo-consistency, assuming that pixels that are projections of the same 3D point should have similar appearance. Often photo-consistency alone does not sufficiently constrain the solution space, and the optimization objective can also include shape priors, which make additional assumptions about what shapes are likely. To solve the optimization problem, a concrete classical algorithm usually consists of a particular 3D representation (e.g. polygon meshes, voxels, or depth maps) and a optimization procedure to compute the best model under that representation. The different combinations of photo-consistency measures, shape priors, 3D representations, and optimization procedures give rise to a large variety of algorithms. For more details, we refer the reader to excellent surveys of these algorithms by Seitz et al. [23] and by Furukawa and Hernández [7].

One family of classical MVS methods [9, 21, 22, 28, 30, 43] is based on the PatchMatch [3] algorithm, which enables efficient dense matching of pixels across views. PatchMatch methods have proved very effective and have demonstrated highly competitive performance. In particular, Xu and Tao [30] introduced the ACMP algorithm, which, among other enhancements, incorporates planar priors and has achieved competitive results on Tanks-and-Temples.

Learning-based MVS Unlike classical algorithms, our approach is learning-based. Existing learning-based MVS methods either use learning to improve parts of a classical pipeline such as PatchMatch [11, 39–41], or develop end-to-end architectures [5, 6, 10, 13, 14, 18, 29, 31–35, 37, 38, 42]. A common step in existing end-to-end architectures is the construction of a 3D cost volume (or feature grid) through some differentiable geometric operations. Then, this 3D cost volume undergoes further updates, often through 3D convolutions, before being transformed into the final 3D model in some particular representation such as voxels [13, 14], depth maps [6, 10, 18, 20, 27, 29, 31–35, 37, 38, 42], or point clouds [5].

The main difference between our approach and existing works is that although we also construct a 3D cost volume, we do not update it. Instead, we update an inverse-depth field that is used to iteratively index from the 3D cost volume to produce 2D feature maps. Our approach thus avoids the costly operations of updating a 3D volume and focuses

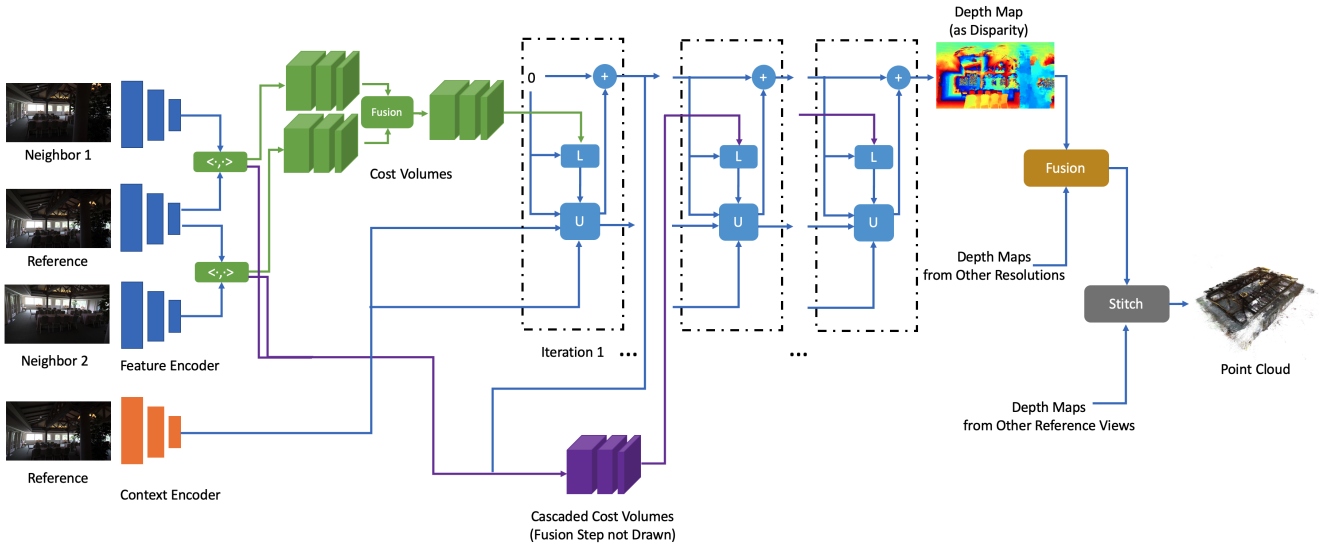

Figure 1. Overview of CER-MVS, which includes an architecture that constructs cascaded epipolar cost volumes and performs recurrent iterative updates of disparity (inverse depth) maps, with fusion of cost volumes from multiple views as well as fusion of disparity maps of multiple resolutions.

limited computing resources on refining the depth maps directly.

3. Approach

This section describes the detailed architecture and pipeline of CER-MVS, as shown in Fig. 1. Given a reference view and a set of neighbor views, we first extract features using a set of convolutional networks. Features are then used to build a collection of cost volumes. We then predict a depth map through recurrent iterative updates, followed by the fusion of multiresolution depths. Finally, depth maps from all references views are fused and stitched to produce a final point cloud.

3.1. Cost Volume Construction

Image Features We need to extract image features from both reference views and neighbor views before using them to construct the cost volumes. In addition, the iterative update unit, to be introduced later, needs context features from reference views. We extract these image features using convolutional encoders following RAFT: $\mathbb{R}^{H\times W\times 3} \rightarrow \mathbb{R}^{H/2^k\times W/2^k\times D_\mathrm{f}}$, where k and D_f are hyperparameters that control the feature resolution and dimension (See Sec. 4.1 and Appendix A for more details).

Epipolar Cost Volume After extracting feature maps $\{\mathbf{f}_i, i = 0, ..., N+1\}$, where $\mathbf{f}_0$ is the reference view and others are neighbor views, each with resolution $(D_\mathrm{f}, H_\mathrm{f}, W_\mathrm{f}) = (D_\mathrm{f}, H/2^k, W/2^k)$, we construct a 3D cost volume by computing the correlation of each pixel in the reference view with pixels along its epipolar line in a neighbor view. Specifically, for a pixel in the reference view, we backproject it to D 3D points with disparity (inverse depth) uniformly spaced in the range from 0 to d_max (after proper scaling as described in Sec. 4.1), reproject the 3D points to the epipolar line in the neighbor view, and use differentiable bilinear sampling to retrieve the features from the neighbor view. This procedure outputs a volume $\mathbf{C} \in \mathbb{R}^{N\times H_\mathrm{f}\times W_\mathrm{f}\times D}$.

Like RAFT, we compute a stack of $\mathbf{C}_\mathrm{P}$ of multiscale cost volumes by repeated average-pooling, i.e., $\mathbf{C}_\mathrm{P} = \{\mathbf{C}_0, \mathbf{C}_1, ..., \mathbf{C}_{L-1}\}$ where $\mathbf{C}_l \in \mathbb{R}^{N\times H_\mathrm{f}\times W_\mathrm{f}\times D/2^l}$, for $l = 0, ..., L-1$.

Cost Volume Cascading Unlike RAFT, the size of an epipolar cost volume depends on not only the image resolution but also the number of disparity values sampled. A dense sampling of a large number of disparity values effectively increases the resolution of the cost volume along the depth dimension and can help reconstruct fine details. However, using a large number of disparity values can take too much GPU memory. To address this issue, we introduce a cascade design. The basic idea is to construct additional cost volumes that are finer-grained along the disparity dimension and centered around the current disparity predictions.

Concretely, after T_1 iterative updates, we create a new

stack of cost volumes $\mathbf{C}_{\mathrm{P}}^{\mathrm{f}} = \{\mathbf{C}_0^{\mathrm{f}}, \mathbf{C}_1^{\mathrm{f}}, ..., \mathbf{C}_{L-1}^{\mathrm{f}}\}$, $\mathbf{C}_l^{\mathrm{f}} \in \mathbb{R}^{N \times H_{\mathrm{f}} \times W_{\mathrm{f}} \times D^{\mathrm{f}}/2^l}, l = 0, ..., L-1$, where D^{f} is the number of disparity values uniformly sampled centered around the current prediction of disparities with smaller increments than those used in the initial stack of cost volumes. Specifically, the value of D^{f} is determined by $2^{L-1} * R$, where R is a hyperparameter that controls the size of the neighborhood described in Sec. 3.2. The factor 2^{L-1} is needed to allow repeated pooling. In this work we use up to 2 stages in our experiments, but the design can be trivially extended to more stages.

It is worth noting that cost volume cascading has been used in prior MVS work [10, 33], but it is a novel design in the context of a RAFT-like architecture, which differs significantly from prior MVS work in that the cost volumes are not updated and are only used as static lookup tables.

3.2. Iterative Updates

The iterative updates follow RAFT in overall structure. We iteratively update a disparity field $\mathbf{d} \in \mathbb{R}^{H_{\mathrm{f}} \times W_{\mathrm{f}}}$ initialized to zero. In each iteration, the input to the update operator includes a hidden state $\mathbf{h} \in \mathbb{R}^{H_{\mathrm{f}} \times W_{\mathrm{f}} \times D_{\mathrm{h}}}$, the current disparity field, the context features $\mathbf{i} \in \mathbb{R}^{H_{\mathrm{f}} \times W_{\mathrm{f}} \times D_{\mathrm{h}}}$ from the reference view, as well as per-pixel features retrieved from the cost volumes using the current disparity field. The output of the update operator includes a new hidden state and an increment to the disparity field.

Multiview Fusion of Cost Volumes Different from RAFT, in multiview stereo we need to consider multiple neighbor views. For each pixel in the reference view, we generate one correlation feature vector against each neighbor view. Given such feature vectors from multiple neighbor views, we take the element-wise mean as the final vector. The intuition behind this operator is that mean value is more robust as the number of neighbor views can vary in test time.

To generate the correlation feature vector for each pixel against a single neighbor view, we perform the same lookup procedure as RAFT. Given the current disparity estimate for the pixel and the stack of cost volumes $\mathbf{C}_{\mathrm{P}} = \{\mathbf{C}_0, \mathbf{C}_1, ..., \mathbf{C}_{L-1}\}$ against the neighbor views, we retrieve, from each cost volume, correlation values corresponding to a local 1D integer grid of length R centered around the current disparity. This is repeated for each level of the stack, and the values from all levels are concatenated to form a single feature vector.

Update Operator We use a GRU-based update operator to propose a sequence of incremental updates to the disparity field.

First, we extract features from the current disparity estimate $\mathbf{d}_t$. The feature vector is formed by subtracting the disparity of each pixel by its 7x7 neighborhood, then reshaping the result into a 49-dimensional vector. This operation has the effect of making the feature vector invariant to the disparity field up to a shift factor, since the retrieved vector only depends on relative disparity between neighboring pixels.

Second, because we have a cascade of cost volumes and our update operator accesses different cost volumes at different stages of the cascade, the operator, while still recurrent, should be given the flexibility to behave somewhat differently for different stages of the cascade. Thus, we modify the weight tying scheme of RAFT such that some weights are tied across all iterations while others are tied only within a single stage of the cascade. Specially, we tie all weights across iterations except the decoder layer that decodes a disparity update from the hidden state of the GRU. The weights of the decoder layer are tied only within each stage of the cascade.

Third, RAFT uses upsampling layers for final predictions of flow field, whereas we do not use any upsampling layer.

The update equations are as follows, with a 2-stage cascade with T_1 iterations for stage 1.

$$\mathbf{x}_t = [\mathrm{Encoder}_d(\mathbf{d}_t), \mathrm{Encoder}_c(\mathbf{c}), \mathbf{i}] \tag{1}$$

$$\mathbf{z}_t = \sigma\left(\mathrm{Conv}_{3\times3}\left([\mathbf{h}_{t-1}, \mathbf{x}_t], W_z\right)\right) \tag{2}$$

$$\mathbf{r}_t = \sigma\left(\mathrm{Conv}_{3\times3}\left([\mathbf{h}_{t-1}, \mathbf{x}_t], W_r\right)\right) \tag{3}$$

$$\tilde{\mathbf{h}}_t = \tanh\left(\mathrm{Conv}_{3\times3}\left([\mathbf{r}_t \odot \mathbf{h}_{t-1}, \mathbf{x}_t], W_h\right)\right) \tag{4}$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t \tag{5}$$

$$\Delta\mathbf{d}_t = \begin{cases} \mathrm{Decoder}_1(\mathbf{h}_t), & t \le T_1 \\ \mathrm{Decoder}_2(\mathbf{h}_t), & t > T_1 \end{cases} \tag{6}$$

Here $\mathbf{i}$ is the context features, and $\mathrm{Encoder}_c$ is an encoder the transforms the correlation features using two convolution layers (see Appendix A for details).

3.3. Multiresolution Depth Fusion

To construct fine details, it generally helps to operate at high resolution, but the available GPU memory limits the highest resolution the network can access, especially during training with large mini-batches. One approach to get around this limit is to apply the network to a higher resolution during inference, which is the common approach adopted in prior works.

However, we find that while using a higher resolution during inference can help, an even better approach is to apply the same network on two input resolutions, the "low" resolution $W \times H$ used to train the network and the higher resolution $2W \times 2H$, and combine the two disparity maps LR and HR to form a fused disparity map MR with a control parameter t:

Table 1. Implementation hyperparameters

Training dataset	DTU	BlendedMVS
Native resolution (H, W)	(1200, 1600)	(1536, 2048)
# neighbor views	10	8
# training epochs	15	16
Feature map downsize ratio	4	
Feature map dimension	64	
Cost volume stack size L	3	
Retrieved neighborhood size R	11	
Cascaded stages	2	
Max disparity $d_{\max}$	0.0025	
Disparity increment in stage 1	$d_{\max}$ / 64	
Disparity increment in stage 2	$d_{\max}$ / 320	
# GRU iterations in each stage	8	
Batch size	2	
Loss parameter	$\lambda = 2.8 \times 10^{-6}, \kappa = 100, \gamma = 0.9$	

Test dataset	DTU	Tanks-and-Temples
Native resolution (H, W)	(1200, 1600)	(1080, 1920) or (1080, 2048)
# Neighbor views for native resolution input	10	15
# Neighbor views for 2 × native resolution input	10	25
Multires fusion threshold t	0.02	0.02
Resolution for point cloud stitching	native resolution	1/2 native resolution
Adaptive thresholding parameter p	0.25	0.25

$$\mathbf{d}_{\mathrm{MR}} = \begin{cases} \mathbf{d}_{\mathrm{HR}}, & \text{if} |\mathbf{d}_{\mathrm{LR}}^{-1} - \mathbf{d}_{\mathrm{HR}}^{-1}| < t * \mathbf{d}_{\mathrm{LR}}^{-1} \\ \mathbf{d}_{\mathrm{LR}}, & \text{otherwise} \end{cases} \tag{7}$$

That is, if the low resolution prediction and high resolution prediction are similar at a pixel, we use the high resolution prediction; otherwise we use the low resolution prediction. This is motivated by the observation that low resolution predictions are more reliable in term of texture-less large structures such as planes, whereas high resolution predictions are more reliable in terms of fine details, which do not tend to deviate drastically from low resolution predictions. Note that as the control parameter t varies from 0 to infinity, $\mathbf{d}_{\mathrm{MR}}$ varies from $\mathbf{d}_{\mathrm{LR}}$ to $\mathbf{d}_{\mathrm{HR}}$.

3.4. Adaptive Point Cloud Stitching

As a last step, the depth maps from the reference views are stitched together to form a single point cloud. We use an adaptive thresholding approach based on Dynamic Consistency Checking (DCC) proposed in D2HC-RMVSNet [32]. DCC hard-codes two thresholds t_1 and t_2 for reprojection errors, however, we use the thresholds kt_1 and kt_2 where k is different for each scene to ensure a fixed percentage, $p\%$ of all pixels pass through consistency test. And p is optimized through the validation set.

Table 2. Results on *DTU* test set

	DTU mean distance (mm)		
	Acc.	Comp.	Overall
COLMAP [22]	0.400	0.664	0.532
MVSNet [34]	0.396	0.527	0.462
D2HC-MVSNet [32]	0.395	0.378	0.386
Point-MVSNet [5]	0.342	0.411	0.376
Vis-MVSNet [42]	0.369	0.361	0.365
AA-RMVSNet [27]	0.376	0.339	0.357
CasMVSNet [10]	0.325	0.385	0.355
EPP-MVSNet [20]	0.413	**0.296**	0.355
CVP-MVSNet [33]	**0.296**	0.406	0.351
UCSNet [6]	0.338	0.349	0.344
IB-MVS [24]	0.334	0.309	**0.321**
Ours	0.359	0.305	0.332

3.5. Supervision

We supervise our network with a loss consisting of two parts. The first part measures the L1 error of the predicted disparity against the ground truth at each iteration, with exponentially increasing weights for later iterations. This part enables faster training of all disparity ranges regardless of outliers at the beginning. The second part of the loss is similar to the first part except that (1) it measures the error of depth (i.e. inverted disparity) so as to be more aligned with point cloud evaluation, and that (2) the error is capped at a constant κ so as to prevent outliers from dominating the loss.

Given the predicted disparity in each iteration be $\mathbf{d}_t, t = 1, ..., T_1 + T_2$ and ground truth disparity $\mathbf{d}_{\mathrm{gt}}$, the combined loss is defined as follows:

$$\mathcal{L}_1 = \sum_{t=1}^{T_1+T_2} \gamma^{T_1+T_2-t} \left\| \mathbf{d}_{\mathrm{gt}} - \mathbf{d}_t \right\|_1 \tag{8}$$

$$\mathcal{L}_2 = \sum_{t=1}^{T_1+T_2} \gamma^{T_1+T_2-t} \min(\left\| \mathbf{d}_{\mathrm{gt}}^{-1} - \mathbf{d}_t^{-1} \right\|_1, \kappa) \tag{9}$$

$$\mathcal{L} = (1 - w) \cdot \mathcal{L}_1 + w \cdot \mathcal{L}_2 \cdot \lambda \tag{10}$$

where γ controls the weights across iterations and λ makes the two parts have roughly the same range. The parameter w balances the two parts and changes from 0 to 1 linearly as training progresses to focus more on the depth error, e.g. for a total number of 16 training epochs, w would be 0.5 when 8 epochs are finished.

4. Experiments

4.1. Implementation Details

We evaluate our models on two datasets, DTU and Tanks-and-Temples. On DTU, we train on its training split

Table 3. Results on Tanks-and-Temples

	intermediate									advanced						
Method	mean	Fam.	Franc.	Horse	Light.	M60	Pan.	Play.	Train	mean	Audi.	Ballr.	Courtr.	Museum	Palace	Temple
COLMAP [22]	42.14	50.41	22.25	25.63	56.43	44.83	46.97	48.53	42.04	27.24	16.02	25.23	34.7	41.51	18.05	27.94
MVSNet [34]	43.48	55.99	28.55	25.07	50.79	53.96	50.86	47.90	34.69	-	-	-	-	-	-	-
Point-MVSNet [5]	48.27	61.79	41.15	34.20	50.79	51.97	50.85	52.38	43.06	-	-	-	-	-	-	-
CVP-MVSNet [33]	54.03	76.50	47.74	36.34	55.12	57.28	54.28	57.43	47.54	-	-	-	-	-	-	-
UCSNet [6]	54.83	76.09	53.16	43.03	54.00	55.60	51.49	57.38	47.89	-	-	-	-	-	-	-
Altizure-SFM, PCF-MVS [16]	55.88	70.99	49.60	40.34	63.44	57.79	58.91	56.59	49.40	35.69	28.33	38.64	35.95	48.36	26.17	36.69
IB-MVS [24]	56.02	75.76	57.65	41.61	55.90	58.09	51.89	59.48	47.73	31.96	22.41	37.00	31.60	41.01	28.20	31.54
CasMVSNet [10]	56.84	76.37	58.45	46.26	55.81	56.11	54.06	58.18	49.51	31.12	19.81	38.46	29.10	43.87	27.36	28.11
ACMM [28]	57.27	69.24	51.45	46.97	63.20	55.07	57.64	60.08	54.48	34.02	23.41	32.91	41.17	48.13	23.87	34.60
ACMP [30]	58.41	70.30	54.06	**54.11**	61.65	54.16	57.60	58.12	57.25	37.44	**30.12**	34.68	**44.58**	50.64	27.20	37.43
Altizure-HKUST-2019 [1]	59.03	77.19	61.52	42.09	63.50	59.36	58.20	57.05	53.3	37.34	24.04	44.52	36.64	49.51	30.23	39.09
DeepC-MVS [17]	59.79	71.91	54.08	42.29	66.54	55.77	**67.47**	60.47	**59.83**	34.54	26.30	34.66	43.50	45.66	23.09	34.00
Vis-MVSNet [42]	60.03	77.40	60.23	47.07	63.44	62.21	57.28	60.54	52.07	33.78	20.79	38.77	32.45	44.20	28.73	37.70
AttMVS [19]	60.05	73.90	62.58	44.08	64.88	56.08	59.39	63.42	56.06	31.93	15.96	27.71	37.99	**52.01**	29.07	28.84
D2HC-MVSNet [32]	60.13	77.36	57.74	45.74	63.39	63.30	57.82	60.71	54.99	-	-	-	-	-	-	-
AA-RMVSNet [27]	61.51	77.77	59.53	51.53	64.02	**64.05**	59.47	60.85	54.90	-	-	-	-	-	-	-
EPP-MVSNet [20]	61.68	77.86	60.54	52.96	62.33	61.69	60.34	62.44	55.30	35.72	21.28	39.74	35.34	49.21	30.00	38.75
Ours	**64.82**	**81.16**	**64.21**	50.43	**70.73**	63.85	63.99	**65.90**	58.25	**40.19**	25.95	**45.75**	39.65	51.75	**35.08**	**42.97**

(a) scan1 (b) scan4 (c) scan9

(d) scan10 (e) scan11 (f) scan12

Figure 2. Visualization of results on DTU (test set).

of DTU and evaluate on its test split, which was suggested by Yao et al. [34] and followed by most authors. On Tanks-and-Temples, we train on the BlendedMVS dataset [36], following the practice of prior work [20, 32, 34]. For all datasets, during training we use the native image resolutions after some random cropping and scaling as input to the network and other details on the hyperparameters are given in Table 1.

To pair neighbor views with reference views, we use the same method as MVSNet [34]. In BlendedMVS, which is used for training only, the scenes have large variations in the range of depth values, we scale each reference view, along with its neighbor views, so that its ground-truth depth has a median value 600 mm. When we evaluate on Tanks-and-Temples, due to lack of ground-truth and noisy background, we scale each reference view, along with its neighbor views, so that its minimum depth of a set of reliable feature points (computed by COLMAP [22] as in MVSNet [34]) is 400 mm. To stitch the predicted depth maps from multiple reference views, we simply scale back each depth map to its original scale.

4.2. Main Results

DTU The results on the DTU benchmark are presented in Table 2. Our method achieves the second best overall score, which is an average completeness and accuracy [2]. Visualizations of sample reconstructions on DTU are shown in Fig. 2.

Tanks-and-Temples On the Tanks-and-Temples dataset, we achieve state of the art performance, as shown in Table 3. Notably, the model is trained on the BlendedMVS dataset without finetuning on Tanks-and-Temples except for some test-time hyperparameter selection using the validation set,

(a) Family (Intermediate) (b) Francis (Intermediate) (c) Horse (Intermediate)

(d) Lighthouse (Intermediate) (e) M60 (Intermediate) (f) Auditorium (Advanced)

(g) Ballroom (Advanced) (h) Courtroom (Advanced) (i) Museum (Advanced)

Figure 3. Visualization of results on Tanks-and-Temples.

as described in Table 1. This indicates a good generalization ability of our approach. A visualization of some results is shown in Fig. 3, from which we can see that many reconstructed scenes look reasonably accurate, detailed, and complete, but there is still substantial room for improvement, especially on low-texture planar regions.

4.3. Ablations

We show our ablation experiments on Tanks-and-Temples official training set (used as validation set) in a restricted setting where we only train the model on BlendedMVS for 2 epochs but keep everything else the same as in Table 1.

Cost Volume Cascading We study the effect of cost volume cascading on memory consumption. In Fig. 4, we plot the GPU memory usage versus F_1-score on Tanks-and-Temples validation set for (1) a series of cascaded model (with different disparity increments in the first stage), (2) its non-cascaded counterpart, which matches the first-stage disparity resolution used in the cascaded model and has equal total GRU iterations. We train all models as described in Sec. 4.1 and finally chose the cascaded model (64, 320) for long-time training and benchmarking. It uses 44 disparity values with an increment of $d_{\max}/320$ in the second stage, and uses 64 values with a coarser increment $d_{\max}/64$ in the first stage to cover the entire disparity range from 0 to $d_{\max}$. For the non-cascaded model, because it needs to fill the entire disparity range from 0 to $d_{\max}$, it needs significantly more memory as the disparity resolution increases. We see from Fig. 4 that cascading produces significant savings of memory. Note the reported memory is the peak memory reported by the command "nvidia-smi".

Table 4. Ablation on supervision

Method	F_1-score
(1) Truncated L_1 depth loss	N/A
(2) L_1 disparity loss	66.79
(3) Average of (1) and (2)	67.32
(4) Proposed dynamic loss	**67.36**

Table 5. Ablation of neighbor view number

Mean F_1-score (%)		# Neighbor views in 2× native resolution		
		5	15	25
# Neighbor views in native resolution	5	62.62	66.42	67.27
	15	62.73	66.48	**67.36**
	25	62.66	66.37	67.27

Dynamic Supervision In Table 4, we show our model trained with different loss supervision. Among them, the truncated L_1 depth loss does not help the model to start up; and L_1 disparity loss has inferior performance; while the proposed dynamic loss is marginally better than the direct average of L_1 depth loss and L_1 disparity loss.

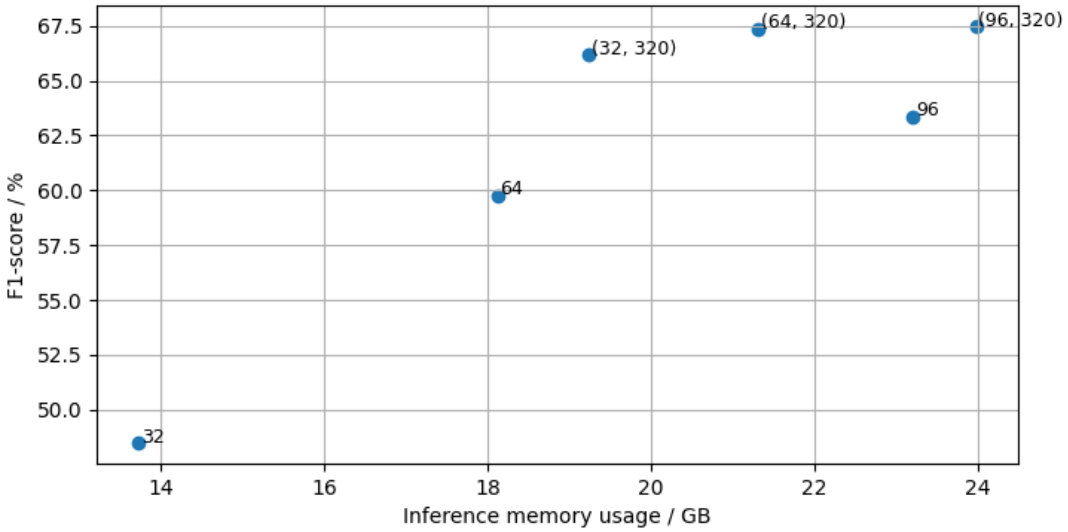

Figure 4. Memory usage of cascaded v.s. non-cascaded model. The label for cascaded models means coarse and fine disparity increments (larger number means smaller increments), and the label for non-cascaded models means the single disparity increment.

Number of Neighbor Views During inference, our network can use a different number of neighbor views than in training. In table 5, we study the effect of changing the number of neighbor views during inference. In particular, we study how this number can be chosen differently for the two resolutions we use to predict depth maps. As the results on the validation set show, the best combination is 15 views for native resolution prediction and 25 views for 2 $\times$ native resolution prediction. And these are the numbers we use on the test set.

Table 6. Ablation of aggregation options

Aggergation option	Mean F_1-score (%)
max	57.77
max + mean	65.37
std	59.90
std + mean	66.85
mean	**67.36**

Table 7. Ablation of adaptive thresholding

Controlled percentage p%	15%	20%	25%	30%	35%
Mean F_1-score (%)	66.83	67.31	**67.36**	67.11	66.60
Fixed threshold k	1	1.5	2	2.5	3
Mean F_1-score (%)	65.33	66.10	66.33	66.32	66.13

Aggregation of Cost Volumes Here in Table 6 we study the effect of aggregation options different from our simple averaging including both one-channel and two-channel ones. It shows that taking the mean is the best.

Adaptive Thresholding To strike a balance between accuracy and completeness scores, we use adaptive thresholding method and search for the best parameter p. The results are in Table 7 in comparison with results from fixed thresholds. We see that our adaptive thresholding approach is significantly better than fixed thresholding.

Multiresolution Fusion of Depth Maps An important part of CER-MVS is the multiresolution fusion of depth maps. Different from previous components, its effect is most obvious on our final model trained for 16 epochs. We report the following results on the validation sets of Tanks-and-Temples: (1) Different control parameter t, and (2) simple weighted average of native input results and 2 $\times$ native input results with weight w . We see from Table 8 that our novel fusion approach is significantly better than all the other approaches.

4.4. Memory and Runtime

The computational cost of CER-MVS is compared with other methods in Table 9. When using similar resolution and numbers of views, the time and memory cost of our method is comparable to others.

5. Conclusion

We have proposed CER-MVS, a new approach based on the RAFT architecture developed for optical flow. CER-MVS introduces five new changes to RAFT: epipolar cost volumes, cost volume cascading, multiview fusion of cost

Table 8. Ablation of multiresolution fusion

Multi-resolution with control threshold t	0 =native input	0.01	0.02	0.04	∞ =2×native input
Mean F_1-score (%)	64.38	68.47	**68.49**	68.39	68.08
Weighted average with w	0 =native input	0.25	0.5	0.75	1 =2×native input
Mean F1-score (%)	64.38	65.30	66.55	67.51	68.08

Table 9. Comparison of running time and memory cost

Method	# Neighbor views	Input Resolution	Output Resolution	Times per view (ms)	Mem. (GB)
CasMVSNet	4	(1056, 1920)	(1056, 1920)	792.2	9.5
Vis-MVSNet			(528, 960)	864.2	4.5
PatchmatchNet			(1056, 1920)	317.7	3.2
EPP-MVSNet			(528, 960)	522.2	8.2
Ours			(264, 480)	664.4	3.0
Ours		(2112, 3840)	(528, 960)	1754.5	7.0
Ours	25			7611.3	22.6

volumes, dynamic supervision, and multiresolution fusion of depth maps, as well as adaptive thresholding to construct point clouds. Experiments show that our approach achieves competitive performance on DTU and state-of-the-art performance on Tanks-and-Temples.

Acknowledgments: This work is partially supported by the National Science Foundation under Award IIS-1942981.

References

[1] https://www.altizure.com. 6

[2] Henrik Aanæs, Rasmus Ramsbøl Jensen, George Vogiatzis, Engin Tola, and Anders Bjorholm Dahl. Large-scale data for multiple-view stereopsis. *International Journal of Computer Vision*, 120(2):153–168, 2016. 2, 6

[3] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24, 2009. 2

[4] Neill DF Campbell, George Vogiatzis, Carlos Hernández, and Roberto Cipolla. Using multiple hypotheses to improve depth-maps for multi-view stereo. In *European Conference on Computer Vision*, pages 766–779. Springer, 2008. 2

[5] Rui Chen, Songfang Han, Jing Xu, and Hao Su. Point-based multi-view stereo network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1538–1547, 2019. 2, 5, 6

[6] Shuo Cheng, Zexiang Xu, Shilin Zhu, Zhuwen Li, Li Erran Li, Ravi Ramamoorthi, and Hao Su. Deep stereo using adaptive thin volume representation with uncertainty awareness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2524–2534, 2020. 2, 5, 6

[7] Yasutaka Furukawa and Carlos Hernández. Multi-view stereo: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 9(1-2):1–148, 2015. 2

[8] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1362–1376, 2009. 2

[9] Silvano Galliani, Katrin Lasinger, and Konrad Schindler. Massively parallel multiview stereopsis by surface normal diffusion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 873–881, 2015. 2

[10] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2495–2504, 2020. 2, 4, 5, 6

[11] Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar, and Alexander C Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3279–3286, 2015. 2

[12] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2007. 2

[13] Mengqi Ji, Juergen Gall, Haitian Zheng, Yebin Liu, and Lu Fang. Surfacenet: An end-to-end 3d neural network for multiview stereopsis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2307–2315, 2017. 2

[14] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. *arXiv preprint arXiv:1708.05375*, 2017. 2

[15] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017. 2

[16] Andreas Kuhn, Shan Lin, and Oliver Erdler. Plane completion and filtering for multi-view stereo reconstruction. In *German Conference on Pattern Recognition*, pages 18–32. Springer, 2019. 6

[17] Andreas Kuhn, Christian Sormann, Mattia Rossi, Oliver Erdler, and Friedrich Fraundorfer. Deepc-mvs: Deep confidence prediction for multi-view stereo reconstruction. In *2020 International Conference on 3D Vision (3DV)*, pages 404–413. IEEE, 2020. 6

[18] Keyang Luo, Tao Guan, Lili Ju, Haipeng Huang, and Yawei Luo. P-mvsnet: Learning patch-wise matching confidence aggregation for multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10452–10461, 2019. 2

[19] Keyang Luo, Tao Guan, Lili Ju, Yuesong Wang, Zhuo Chen, and Yawei Luo. Attention-aware multi-view stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1590–1599, 2020. 6

[20] Xinjun Ma, Yue Gong, Qirui Wang, Jingwei Huang, Lei Chen, and Fan Yu. Epp-mvsnet: Epipolar-assembling based depth prediction for multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5732–5740, 2021. 2, 5, 6

[21] Andrea Romanoni and Matteo Matteucci. Tapa-mvs: Textureless-aware patchmatch multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10413–10422, 2019. 2

[22] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision*, pages 501–518. Springer, 2016. 2, 5, 6

[23] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 1, pages 519–528. IEEE, 2006. 2

[24] Christian Sormann, Mattia Rossi, Andreas Kuhn, and Friedrich Fraundorfer. Ib-mvs: An iterative algorithm for deep multi-view stereo based on binary decisions. In *British Machine Vision Conference (BMVC) 2021*. The British Machine Vision Association, 2021. 5, 6

[25] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision*, pages 402–419. Springer, 2020. 1

[26] Engin Tola, Christoph Strecha, and Pascal Fua. Efficient large-scale multi-view stereo for ultra high-resolution image sets. *Machine Vision and Applications*, 23(5):903–920, 2012. 2

[27] Zizhuang Wei, Qingtian Zhu, Chen Min, Yisong Chen, and Guoping Wang. Aa-rmvsnet: Adaptive aggregation recurrent multi-view stereo network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6187–6196, 2021. 2, 5, 6

[28] Qingshan Xu and Wenbing Tao. Multi-scale geometric consistency guided multi-view stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5483–5492, 2019. 2, 6

[29] Qingshan Xu and Wenbing Tao. Learning inverse depth regression for multi-view stereo with correlation cost volume. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12508–12515, 2020. 2

[30] Qingshan Xu and Wenbing Tao. Planar prior assisted patchmatch multi-view stereo. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12516–12523, 2020. 2, 6

[31] Youze Xue, Jiansheng Chen, Weitao Wan, Yiqing Huang, Cheng Yu, Tianpeng Li, and Jiayu Bao. Mvscrf: Learning multi-view stereo with conditional random fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4312–4321, 2019. 2

[32] Jianfeng Yan, Zizhuang Wei, Hongwei Yi, Mingyu Ding, Runze Zhang, Yisong Chen, Guoping Wang, and Yu-Wing Tai. Dense hybrid recurrent multi-view stereo net with dynamic consistency checking. In *European Conference on Computer Vision*, pages 674–689. Springer, 2020. 1, 2, 5, 6

[33] Jiayu Yang, Wei Mao, Jose M Alvarez, and Miaomiao Liu. Cost volume pyramid based depth inference for multi-view stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4877–4886, 2020. 2, 4, 5, 6

[34] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 767–783, 2018. 1, 2, 5, 6

[35] Yao Yao, Zixin Luo, Shiwei Li, Tianwei Shen, Tian Fang, and Long Quan. Recurrent mvsnet for high-resolution multi-view stereo depth inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5525–5534, 2019. 1, 2

[36] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. Blendedmvs: A large-scale dataset for generalized multi-view stereo networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1790–1799, 2020. 6

[37] Hongwei Yi, Zizhuang Wei, Mingyu Ding, Runze Zhang, Yisong Chen, Guoping Wang, and Yu-Wing Tai. Pyramid multi-view stereo net with self-adaptive view aggregation. In *European Conference on Computer Vision*, pages 766–782. Springer, 2020. 2

[38] Zehao Yu and Shenghua Gao. Fast-mvsnet: Sparse-to-dense multi-view stereo with learned propagation and gauss-newton refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1949–1958, 2020. 2

[39] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4353–4361, 2015. 2

[40] Jure Zbontar and Yann LeCun. Computing the stereo matching cost with a convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1592–1599, 2015. 2

[41] Jure Zbontar and Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches. *J. Mach. Learn. Res.*, 17(1):2287–2318, 2016. 2

[42] Jingyang Zhang, Yao Yao, Shiwei Li, Zixin Luo, and Tian Fang. Visibility-aware multi-view stereo network. *arXiv preprint arXiv:2008.07928*, 2020. 1, 2, 5, 6

[43] Enliang Zheng, Enrique Dunn, Vladimir Jojic, and Jan-Michael Frahm. Patchmatch based joint view selection and depthmap estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1510–1517, 2014. 2

Appendix A. Network Architecture

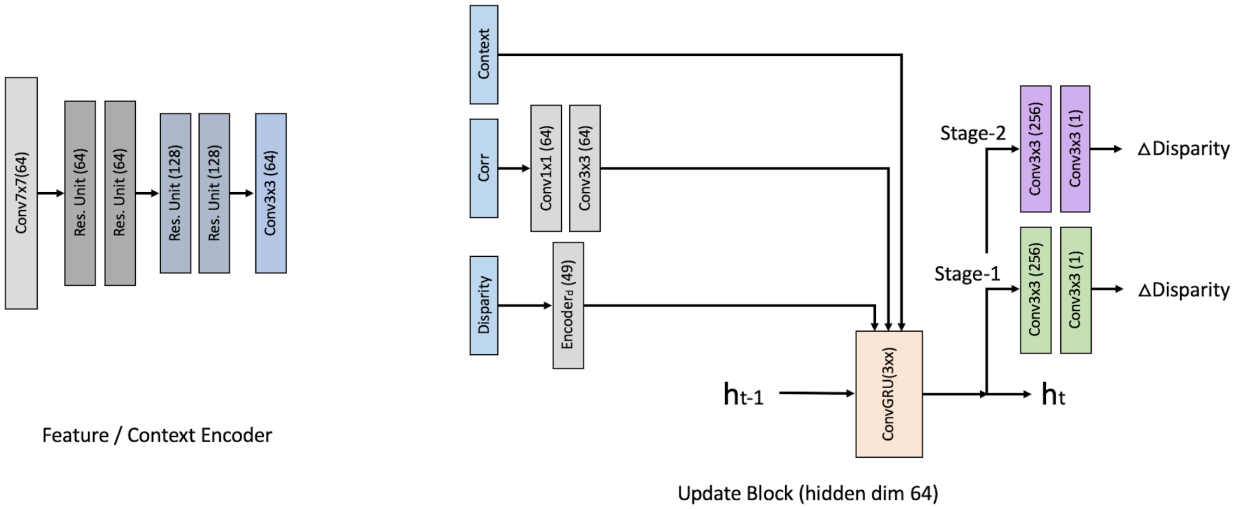

Figure 5. Network architecture details. The context and feature encoders have the same architecture, the only difference is that the feature encoder uses instance normalization while the context encoder uses batch normalization.