Robot Vision: Multi-view Stereo

Prof. Friedrich Fraundorfer

SS 2023
Outline

- Multi-view stereo principle
- Feature extraction networks
- Cost volume generation
- Cost volume regularization
- Depth inference
- Data sets
- Results
Multi-View Stereo

- Input: set of images + camera poses (from SFM)
- Output: 3D model (as dense point cloud)
Multi-View Stereo Pipeline
Plane-sweep multi-view stereo

- Classical plane sweeping stereo [8]
- Sweep family of planes at different depths with respect to reference camera
- With CNNs: Warp deep features instead of raw pixel values

\[ H_i(d) = K_i \cdot R_i \cdot \left( I - \frac{(t_{ref} - t_i) \cdot n_{ref}^T}{d} \right) \cdot R_{ref}^T \cdot K_{ref}^T \]
Deep Learning for Multi-View Stereo (MVS)

- Advantages:
  - fast
  - usually works better in terms of completeness
  - can work on non-lambertian surfaces

- Disadvantages:
  - often huge (GPU) memory requirements
  - needs large amount of data to train on
  - might fail in a completely new environment
Deep Learning for MVS: Features

- **Hand-crafted Features:**
  - Designed by human experts to extract a given set of chosen characteristics
  - Trade-off between accuracy and computational efficiency
  - e.g.: Census

- **Learned Features:**
  - Extracted via Convolutional Neural Network (CNN)
  - Learned from data
Deep Learning for MVS: Regularization

- Needed to filter incorrect correspondences (e.g. from occlusions, noise)

- Traditional Regularization:
  - Find local correspondences
  - Apply regularization methods
    - Semi global matching
    - Belief propagation
    - Graph cut
    - Smoothness priors
  - Apply filters

- Learned Regularization:
  - Network learns to regularize raw feature output
  - Often 3D convolutions
Pre-process Images

- Crop/scale to fit network requirements
  - Due to convolutions, width/height usually need to be a multiple of $2^n$ (e.g. 32 or 64)
  - Adjust camera parameters accordingly!

- Images usually need to be stacked in network -> need same sizes!

- Augment data for training: Change brightness, contrast, etc
Deep Feature Extraction

- Acquired from RGB image via CNN
- Encode image information in a way that it can be compared to other images
- Can have many layers
  - Usually a combination of 2D convolutions, Normalization and ReLU
- Original neighboring information can be encoded to smaller resolution
  - Save memory for next step
Deep Feature Extraction: 2D Convolution

- Example: Kernel=3x3, Stride = 1

\[
I \ast K = \begin{pmatrix}
0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0
\end{pmatrix} \ast \begin{pmatrix}
1 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 1
\end{pmatrix} = \begin{pmatrix}
1 & 4 & 3 & 4 \\
1 & 2 & 4 & 3 \\
1 & 2 & 3 & 4 \\
1 & 3 & 3 & 1
\end{pmatrix}
\]
Deep Feature Extraction: 2D Convolution

- Example: Kernel=3x3, Stride = 3

\[
\begin{pmatrix}
0 & 1 & 1 \\
0 & 0 & 1 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 1 & 1 \\
0 & 1 & 0 \\
0 & 1 & 0 \\
\end{pmatrix}
\begin{pmatrix}
1 & 1 & 1 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
1 & 1 & 0 \\
\end{pmatrix}
\ast
\begin{pmatrix}
1 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 1 \\
\end{pmatrix}
=
\begin{pmatrix}
1 & 4 \\
1 & 1 \\
\end{pmatrix}
\]
Deep Feature Extraction: 2D Convolution

- Input and output channels can be arbitrary (modelled through more kernel weights)
Ex. Deep Feature Extraction: Simple Feature Net

- **Simple Feature Net**
  - **Layer 1**
    - Kernel: 3x3, Stride: 1
    - BN, ReLU
  - **Layer 2**
    - Kernel: 5x5, Stride: 2
    - BN, ReLU
  - **Layer 3**
    - Kernel: 3x3, Stride: 1

- **Convolutional Layers**:
  - **Conv1**: Input 8x8, Output 16
  - **Conv2**: Input 16x16, Output 32
  - **Conv3**: Input 32x32, Output 32
Ex. Deep Feature Extraction: Unet

- kernel=3x3, stride=1, BN, ReLU
- kernel=3x3, stride=2, BN, ReLU
- upsampling, scale=2
Cost Volume

- Aggregate N feature volumes to one cost volume C via homography warping (plane sweep)
- Variance cost metric using the average feature volume:

\[
C = \frac{\sum_{i=1}^{N} (F_i - \bar{F}_i)^2}{N}
\]

- Each point in the cost volume can be seen as a similarity measure
Cost Volume Regularization

- Raw cost volume
  - could be noise-contaminated
  - has no smoothness constraint
- Use CNNs to regularize the obtained cost volume variance
- Usually 3D convolutions
Cost Volume Regularization

- Last 3D convolution layer maps output to single channel
- Search for lowest cost / highest probability
### Depth Inference

- **Classification:**
  - Predicts label
  - Discrete output: Class with highest probability
  - Can be filtered through probability threshold
  - Example: Class 4 has highest prob -> Result: 4

- **Regression:**
  - Predicts quantity
  - Continuous output
  - Can be filtered through entropy threshold
  - Example: $0.1\times1 + 0.1\times2 + 0.1\times3 + 0.5\times4 + 0.2\times5 = 3.6$

<table>
<thead>
<tr>
<th>Network Output</th>
<th>Class Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>0.1</td>
<td>2</td>
</tr>
<tr>
<td>0.1</td>
<td>3</td>
</tr>
<tr>
<td>0.5</td>
<td>4</td>
</tr>
<tr>
<td>0.2</td>
<td>5</td>
</tr>
</tbody>
</table>
Training loss: Classification

- Multi-class classification problem with cross entropy loss:

\[
loss = \sum_{\mathbf{p}} \left( \sum_{i=1}^{D} -P(i, \mathbf{p}) \cdot \log Q(i, \mathbf{p}) \right)
\]

where:

- \(\mathbf{p}\) = spatial image coordinate
- \(D\) = maximum depth value
- \(P(i, \mathbf{p})\) = voxel in the probability volume \(P\)
- \(Q(i, \mathbf{p})\) = ground truth voxel
Training loss: Regression

- Regress depth outputs using the soft argmin [7] operation and L1 loss:

\[
\text{soft argmin} := \sum_{d=1}^{D_{max}} d \times \sigma(-c_d)
\]

where:

- \(D_{max}\) = maximum depth value
- \(c_d\) = predicted cost
- \(\sigma(\cdot)\) = softmax operation

\[
\text{loss} = \frac{1}{N} \sum_{n=1}^{N} \| d_{n,gt} - d_{n,pred} \|_1
\]
Post-Processing and Filtering

- Geometric verification
  - Project each pixel into different view and back
  - Check if reprojected image lies within some threshold

- Photometric verification
  - Measures matching quality for each pixel
  - Directly implemented in network: probability, standard deviation or entropy
Datasets

- Quality of dataset very important for training
- Benchmarks for evaluation

- Examples: DTU, Tanks and Temples, ETH3D, Blended MVS
DTU dataset

- http://roboimagedata.compute.dtu.dk
- Recorded using industrial robot arm with a structured light scanner
- Indoor, small scale, different light settings, 49 or 64 images per scene
- Ground-truth available as point clouds
- “Ground-truth” depth maps available from MVSNet
  - Screened Poisson surface reconstruction: point cloud -> mesh
  - Render mesh to each viewpoint
  - Not perfect: holes and wrong labelling in depth maps
    - Attention-Aware MVS [6]: improve ground-truth depth maps
DTU dataset
Tanks and Temples dataset

- [https://www.tanksandtemples.org/](https://www.tanksandtemples.org/)
- Ground-truth point cloud captured with industrial laser scanner
- Outdoor and indoor environments
- High-res video available for each scene
Tanks and Temples dataset
ETH3D dataset

- [https://www.eth3d.net/datasets](https://www.eth3d.net/datasets)
- Ground-truth point cloud from laser scan
- 13 training and 12 test scenes in high resolution
- 5 training and 5 test videos in low resolution
- Challenging
  - large image size
  - large viewpoint change
  - small amount of images
- Deep learning methods not (yet) competitive
ETH3D dataset
Evaluation

- **Overall Score**: mean of accuracy and completeness (DTU)
  - Measures the mean distance to the groundtruth point cloud
  - Lower is better

- **F-Score**: harmonic mean of precision and recall (TaT, ETH3D)
  - Measured at a certain distance threshold $d$
  - If either $P(d) \rightarrow 0$ or $R(d) \rightarrow 0$, then $F(d) \rightarrow 0$
  - Better summary measure than the arithmetic mean

\[
F(d) = \frac{2P(d)R(d)}{P(d) + R(d)}
\]
Examples

- MVSNet (ECCV 2018): CostRegNet after volume variance calculation
- R-MVSNet (CVPR 2019): regularizes 2D costmaps along depth direction via GRU to save memory
- MVSCRF (ICCV 2019): CRF after cost volume regularization
- CasMVSNet (CVPR 2020): Multiscale feature extraction, refine depth values in every step
- Cost Volume Pyramid (CVPR 2020)
- Attention-Aware MVS (CVPR 2020)
## HighRes-MVSNet: Evaluation DTU

<table>
<thead>
<tr>
<th>Method</th>
<th>Acc.</th>
<th>Comp.</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Geometric</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Furu [6]</td>
<td>0.613</td>
<td>0.941</td>
<td>0.777</td>
</tr>
<tr>
<td>Tola [27]</td>
<td>0.342</td>
<td>1.190</td>
<td>0.766</td>
</tr>
<tr>
<td>Camp [2]</td>
<td>0.835</td>
<td>0.554</td>
<td>0.695</td>
</tr>
<tr>
<td>Gipuma [7]</td>
<td>0.283</td>
<td>0.873</td>
<td>0.578</td>
</tr>
<tr>
<td>COLMAP [25, 26]</td>
<td>0.400</td>
<td>0.664</td>
<td>0.532</td>
</tr>
<tr>
<td><strong>Learning</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MVSNet [32]</td>
<td>0.396</td>
<td>0.527</td>
<td>0.462</td>
</tr>
<tr>
<td>R-MVSNet [33]</td>
<td>0.383</td>
<td>0.452</td>
<td>0.417</td>
</tr>
<tr>
<td>SurfaceNet [14]</td>
<td>0.450</td>
<td>1.040</td>
<td>0.745</td>
</tr>
<tr>
<td>MVS-CRF [29]</td>
<td>0.371</td>
<td>0.426</td>
<td>0.398</td>
</tr>
<tr>
<td>Point-MVSNet [4]</td>
<td>0.342</td>
<td>0.411</td>
<td>0.376</td>
</tr>
<tr>
<td>CasMVSNet [9]</td>
<td>0.346</td>
<td>0.351</td>
<td>0.348</td>
</tr>
<tr>
<td>CVP-MVSNet [31]</td>
<td>0.296</td>
<td>0.406</td>
<td>0.351</td>
</tr>
<tr>
<td>AttMVS [20]</td>
<td>0.383</td>
<td>0.329</td>
<td>0.356</td>
</tr>
<tr>
<td>Fast-MVSNet [35]</td>
<td>0.336</td>
<td>0.403</td>
<td>0.370</td>
</tr>
<tr>
<td><strong>Ours</strong></td>
<td>0.354</td>
<td>0.393</td>
<td>0.373</td>
</tr>
<tr>
<td><strong>Ours (HR)</strong></td>
<td>0.346</td>
<td>0.345</td>
<td><strong>0.346</strong></td>
</tr>
</tbody>
</table>
### HighRes-MVSNet: Evaluation TaT

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean</th>
<th>Family</th>
<th>Francis</th>
<th>Horse</th>
<th>Lighthouse</th>
<th>M60</th>
<th>Panther</th>
<th>Playground</th>
<th>Train</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLMAP [25, 26]</td>
<td>42.41</td>
<td>50.41</td>
<td>22.25</td>
<td>25.63</td>
<td>56.43</td>
<td>44.83</td>
<td>46.97</td>
<td>48.53</td>
<td>42.04</td>
</tr>
<tr>
<td>MVSNet [32]</td>
<td>43.48</td>
<td>55.99</td>
<td>28.55</td>
<td>25.07</td>
<td>50.79</td>
<td>53.96</td>
<td>50.86</td>
<td>47.90</td>
<td>34.69</td>
</tr>
<tr>
<td>R-MVSNet [33]</td>
<td>48.40</td>
<td>69.96</td>
<td>46.65</td>
<td>32.59</td>
<td>42.95</td>
<td>51.88</td>
<td>48.80</td>
<td>52.00</td>
<td>42.38</td>
</tr>
<tr>
<td>Point-MVSNet [4]</td>
<td>48.27</td>
<td>61.79</td>
<td>41.15</td>
<td>34.20</td>
<td>50.79</td>
<td>51.97</td>
<td>50.85</td>
<td>52.38</td>
<td>43.06</td>
</tr>
<tr>
<td>AttMVS [20]</td>
<td>60.05</td>
<td>73.90</td>
<td>62.58</td>
<td>44.08</td>
<td>64.88</td>
<td>56.08</td>
<td>59.39</td>
<td>63.42</td>
<td>56.06</td>
</tr>
<tr>
<td>CasMVSNet [9]</td>
<td>56.42</td>
<td>76.36</td>
<td>58.45</td>
<td>46.20</td>
<td>55.53</td>
<td>56.11</td>
<td>54.02</td>
<td>58.17</td>
<td>46.56</td>
</tr>
<tr>
<td>CVP-MVSNet [31]</td>
<td>54.03</td>
<td>76.50</td>
<td>47.74</td>
<td>36.34</td>
<td>55.12</td>
<td>57.28</td>
<td>54.28</td>
<td>57.43</td>
<td>47.54</td>
</tr>
<tr>
<td>MVSCRF [29]</td>
<td>45.73</td>
<td>59.83</td>
<td>30.60</td>
<td>29.93</td>
<td>51.15</td>
<td>50.61</td>
<td>51.45</td>
<td>52.60</td>
<td>39.68</td>
</tr>
<tr>
<td>Fast-MVSNet [35]</td>
<td>47.39</td>
<td>65.18</td>
<td>39.59</td>
<td>34.98</td>
<td>47.81</td>
<td>49.16</td>
<td>46.20</td>
<td>53.27</td>
<td>42.91</td>
</tr>
<tr>
<td>Ours</td>
<td>49.81</td>
<td>66.62</td>
<td>44.17</td>
<td>30.84</td>
<td>55.13</td>
<td>53.20</td>
<td>50.32</td>
<td>55.45</td>
<td>42.73</td>
</tr>
</tbody>
</table>
References


References


