Depth from Stereo

Dominic Cheng
February 7, 2018
Agenda

1. Introduction to stereo
2. Efficient Deep Learning for Stereo Matching
   (W. Luo, A. Schwing, and R. Urtasun. In CVPR 2016.)
3. Cascade Residual Learning: A Two-stage Convolutional Neural Network for Stereo Matching
   (J. Pang, W. Sun, J. SJ. Ren, C. Yang, and Q. Yan. In CVPR 2017.)
Introduction to Stereo
What is stereo?

Depth from images is a very intuitive ability

- Given two images of a scene from (slightly) different viewpoints, we are able to infer depth

Can we do the same using computers?

- Yes (kind of?)
- First, we need to appreciate the geometry of the situation

Source: https://s.hswstatic.com/gif/pc-3-d-brain.png
Geometry in stereo (a visual overview)

- Think of images as projections of 3D points (in the real world) onto a 2D surface (image plane)
- $X_L$ is the projection of $X, X_1, X_2, X_3, \ldots$ onto the left image
- $X, X_1, X_2, X_3$ will also project onto the right image

Source: https://upload.wikimedia.org/wikipedia/commons/thumb/1/14/Epipolar_geometry.svg/640px-Epipolar_geometry.svg.png?1517775941158
Geometry in stereo (a visual overview)

- What do you notice?
- Projections of $X_1, X_2, X_3$ on right image all lie on a line
- This line is known as an epipolar line
  - Points $e_L, e_R$ are known as epipoles
  - Projections of cameras’ optical centers $O_L, O_R$ onto the images
  - All epipolar lines will intersect at epipoles
  - Left image has corresponding epipolar line
- Geometry of stereo vision also known as epipolar geometry

Source: https://upload.wikimedia.org/wikipedia/commons/thumb/1/14/Epipolar_geometry.svg/640px-Epipolar_geometry.svg.png?1517775941158
Geometry in stereo (a visual overview)

- What does this give us?
- All 3D points that could have resulted in $X_L$ must have a projection on the right image, and must be on the epipolar line $e_R - x_R$
- Given just the left/right images and $X_L$, you can search on the corresponding epipolar line in the right image. If you can find the corresponding match $X_R$, you can uniquely determine the 3D position of $X$. 

Source: https://upload.wikimedia.org/wikipedia/commons/thumb/1/14/Epipolar_geometry.svg/640px-Epipolar_geometry.svg.png?1517775941158
Geometry in stereo (a visual overview)

- In practice ...
- Epipolar lines can be made parallel through a process called **rectification**
- Simplifies the process of finding a match and calculating the 3D point

Credit: S. Savarese
Source: http://web.stanford.edu/class/cs231a/lectures/lecture6_stereo_systems.pdf
Geometry in stereo (a visual overview)

- How do you actually get depth?
- If you find correspondences $x$ and $x'$, the quantity $x - x'$ is known as the **disparity**
- By similar triangles, you can convince yourself that disparity is inversely proportional to depth
- Problem statement, reformulated: **Find the disparity** for every pixel in the left (or right) image by finding matches in the right (or left) image

Credit: L. Shapiro
Source: https://courses.cs.washington.edu/courses/cse455/16wi/notes/11_Stereo.pdf
Practical example: KITTI
Efficient Deep Learning for Stereo Matching [1]

Features for stereo correspondence

- Finding a good match is hard
- What is a good feature?
- Can we learn the features instead?

Source: https://upload.wikimedia.org/wikipedia/en/3/3b/Stereo_empire.jpg
Key idea

- Construct a neural network that takes input images (left/right) and produces representative features that can be used to find stereo correspondences efficiently.
Network architecture

- Siamese network
  - Shared weights enforce similar features are learned on both left/right images
- Several convolution layers
  - Paper implements a fairly vanilla network
  - Several variants are tested; the key behind the choices of kernel size / stride is the effective receptive field
Training

- Pose this as a multi-class classification problem
  - Differentiating from earlier work which poses as binary classification [3]
- Left image patch is equal to the receptive field
  - Final feature volume after passing through the network is $1 \times 1 \times 64$ ($H \times W \times C$)
- Right patch is larger to accommodate more context across range of possible disparities
  - Final feature volume is $1 \times S \times 64$ ($S$ is total number of search locations)
- **Inner product** of left feature with every spatial location of right feature $\Rightarrow S$ scores
Training

- Multi-class cross entropy loss over these $S$ scores
- Each class is an actual spatial bin
- Probability mass is diffused across ground truth bin +/- 2 bins, to allow for some ambiguity

$$\min_{w} \sum_{i,y_i} p_{gt}(y_i) \log p_i(y_i, w)$$

$$p_{gt}(y_i) = \begin{cases} 
\lambda_1 & \text{if } y_i = y_i^{GT} \\
\lambda_2 & \text{if } |y_i - y_i^{GT}| = 1 \\
\lambda_3 & \text{if } |y_i - y_i^{GT}| = 2 \\
0 & \text{otherwise}
\end{cases}$$
Testing

- Does not have to take the same form as training
- Efficiency comes from enforcing that similarity between features is measured by their inner product
- Can compute all these features at once on left/right images
- Produce a cost volume by computing similarity across multiple disparities
  - \( H \times W \times D \), where \( D \) is number of disparity candidates
Smoothing

- How to get final result?
- Could just take most likely assignments across this volume
- Drawback: These predictions tend to be rough (no smoothness prior)
- Can smooth in various ways through averaging, energy minimization (semi-global block matching), slanted-plane, and other post-processing techniques

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15.25</td>
<td>8.95</td>
<td>7.23</td>
<td>7.13</td>
<td>12.45</td>
<td>14.96</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11.43</td>
<td>8.00</td>
<td>6.60</td>
<td>6.58</td>
<td>7.78</td>
<td>-</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>5.18</td>
<td>4.74</td>
<td>4.62</td>
<td>4.73</td>
<td>3.48</td>
<td>5.05</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>4.41</td>
<td>4.23</td>
<td>4.31</td>
<td>4.38</td>
<td>3.10</td>
<td>4.74</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>4.25</td>
<td>4.20</td>
<td>4.14</td>
<td>4.19</td>
<td>3.11</td>
<td>4.79</td>
</tr>
</tbody>
</table>
Evaluation

- Train and test on KITTI only (training set has 200 image pairs)
- Very straightforward training procedure
- Competitive results (on D1 error reported by testbench) with significant speed-up
  - Highlighting similar approach of [3]

<table>
<thead>
<tr>
<th></th>
<th>All/All</th>
<th>All/Est</th>
<th>Noc/All</th>
<th>Noc/Est</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D1-bg</td>
<td>D1-fg</td>
<td>D1-all</td>
<td>D1-fg</td>
</tr>
<tr>
<td>MBM [9]</td>
<td>4.69</td>
<td>13.05</td>
<td>6.08</td>
<td>4.69</td>
</tr>
<tr>
<td>SPS-St [27]</td>
<td>3.84</td>
<td>12.67</td>
<td>5.31</td>
<td>3.84</td>
</tr>
<tr>
<td>MC-CNN [30]</td>
<td>2.89</td>
<td>8.88</td>
<td>3.89</td>
<td>2.89</td>
</tr>
<tr>
<td>Displets v2 [12]</td>
<td>3.00</td>
<td>5.56</td>
<td>3.43</td>
<td>3.00</td>
</tr>
<tr>
<td>Ours(37)</td>
<td>3.73</td>
<td>8.58</td>
<td>4.54</td>
<td>3.73</td>
</tr>
</tbody>
</table>
Sample output

- From submission to KITTI 2015 stereo benchmark
- Middle is prediction, bottom is error
- Even small differences in prediction can result in large disparity errors

Another approach

- This can be posed as a classification problem, why not regression?
  - Based on idea of DispNet presented in [4]
  - Feed two images in, get dense disparity prediction out

- Advantage:
  - Note that in previous approach, smoothing was still necessary for good results
  - We could try to make the entire prediction process end-to-end learnable

- Disadvantage:
  - Do not get to explicitly incorporate geometric priors
Architecture

- Two parts
  - DispFulNet: Predict initial disparity
  - DispResNet: Refine the prediction
Architecture

- **DispFulNet**
  - Based on DispNet [4]
  - Encoder/decoder architecture; take left/right images as input, share lower level features, combine, predict
  - Train with $L_1$ loss against ground truth disparity map
  - Make predictions at multiple scales during decode ($d_1^{(S)}$, ..., $d_1^{(0)}$)
  - Produce initial disparity map $d_1$
Architecture

- **DispResNet**
  - Idea from ResNet
  - Given initial prediction, have another network predict the residuals
  - Again, produce predictions at multiple scales to incorporate more supervision
  - Output is final disparity
Evaluation

- Train on a lot of data
  - FlyingThings3D: Synthetic dataset with 22k+/4k+ train/test examples
  - Finetuning on KITTI
- Test on FlyingThings, Middlebury, and KITTI
- Currently #8 on KITTI 2015 stereo leaderboard!
  - Keep in mind submitted March 2017
Evaluation

- Qualitative assessment of refinement
Sample output

- From submission to KITTI 2015 stereo benchmark
- Middle is prediction, bottom is error
- Generally smoother outputs with ability to define sharp boundaries for objects
Questions
References


