Learned Visual Features For Depth Estimation

Lukas Willinger and Marcus Hott

Technical University of Munich
Seminar - Visual feature learning for autonomous driving
Munich, 12th of July 2019
Outline

1. Depth information in autonomous driving
2. Monocular and stereo vision
3. Traditional approach
4. Deep Learning approach
5. Comparison in the context of autonomous driving
Our stereo DNN computes depth from RGB input (shown colorcoded)
Why depth information is needed?

• Interact with a constantly changing environment
• Generate mapping of surrounding (e.g. SLAM method)
• Classify objects and localize objects around the car
• Generate safe path planning
• ToF, Lidar... sensors provide sparse and sensitive data
Incomplete depth information

Example from KITTI 2015 stereo dataset
Monocular and Stereo Vision

• Mono Vision:
  - Ill-posed problem to find depth from single image

• Stereo Vision:
  - Focus of research, since it yields better results
  - Defined by epipolar geometry
  - Goal: Approximate difference between the views

Approach: Calculate disparity of environment and deduce depth information
Monocular and Stereo Vision: Epipolar geometry

- Epipolar lines connect the optical center and a point on the image and meet behind the image plane in X.

- Disparity is the pixel shift between two identical points in both images.

\[
disp. = x - x' = \frac{Bf}{z}
\]

\[
depth = z = \frac{Bf}{\Delta x}
\]

Known:
- O: Optical center of camera
- B: Distance between cameras
- f: Focal length of camera

Unknown:
- x-x': distance between points in image plane

https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_calib3d/py_depthmap/py_depthmap.html
Scharsteiner and Szeliski (2002):

- Defined metrics to compare accuracy on the Middlebury stereo dataset
- Described stereo algorithm by four stages
  1. Matching cost computation
  2. Cost aggregation
  3. Disparity selection
  4. Disparity refinement
Traditional Computer Vision

• Global vs. local:
  Local: window based approach
  Global: minimize global energy function for all pixels
  with terms penalizing inconsistencies and enforcing smoothness

Calibration reduces search to 1D horizontal matching.

• 4 Stages:
  1. Matching cost computation: e.g. Absolute Difference, Squared difference,
     Normalized Cross Correlation; rank transform, census transform
  2. Cost aggregation: e.g. Fixed Window, Adaptive Window, Adaptive Support
     Weights
  3. Disparity selection: lowest matching cost
  4. Disparity refinement: Gaussian Filter, Median Filter, Bilateral Filter
State of the art algorithms

Algorithms:
*includes stage 1-3: cost computation, cost aggregation, disparity selection*

- Stereo Block Matching (local: FW, SAD, horizontal block matching)
- Stereo Belief Propagation (global: bayesian approach)

Filtering:
*handles stage 4: disparity refinement*

- Gaussian Filtering
- Stereo joint bilateral Filtering
  - combines range and domain filtering: evaluates locality and similarity
  - preserves edges
  - removes texture -> lower color map
  - cartoonish style for multiple iterations
We tried it out with OpenCV

Algorithm:

```python
cv2.StereoBM(...)
- numDisparities: 32
- blockSize: 15
```

Filter: (multiple iterations)

```python
cv2.BilateralFilter(...)
- d (neighborhood): 5
- sigmaColor: 50
```

Left image (Middlebury dataset)

Right image (Middlebury dataset)
We tried it out with OpenCV

Algorithm:

```python
cv2.StereoBM(...)
  - numDisparities: 32
  - blockSize: 15
```

Filter: (multiple iterations)

```python
cv2.BilateralFilter(...)
  - d (neighborhood): 5
  - sigmaColor: 50
```

Left image (Middlebury dataset)
We tried it out with OpenCV

Algorithm:

```python
cv2.StereoBM(...)
    - numDisparities: 32
    - blockSize: 15
```

Filter: (multiple iterations)

```python
cv2.BilateralFilter(...)
    - d (neighborhood): 5
    - sigmaColor: 50
```

Right image (Middlebury dataset)
We tried it out with OpenCV

Algorithm:

```python
cv2.StereoBM(...)
  - numDisparities: 32
  - blockSize: 15
```

Filter: (multiple iterations)

```python
cv2.BilateralFilter(...)
  - d (neighborhood): 5
  - sigmaColor: 50
```

1. Why Depth?
2. Mono vs. Stereo
3. Traditional Approach
4. Deep Learning Approach
5. Conclusions
Semi global matching by Hirschmüller (2008) based on belief propagation (bayesian approach with coupled Markov Random Fields)

- Matching Cost from Mutual Information $MI$ (uses the notion of entropy $h$ and information gain)
  
  $$C_{MI}(p, d) \approx h_{I_1, I_2}$$

- Disparity found through minimization of:
  
  $$E(D) = \sum_p C(p, D_p) + \sum_{q \in N_p} P_1 f_1(D_p - D_q) + \sum_{q \in N_p} P_2 f_1(D_p - D_q)$$

  with $P_1 < P_2$ and $P_2 = \nabla(I)$

- 1D approximation (8 or 16 directions) to avoid NP-completeness in 2D case
Performance in the context of autonomous driving

- Up until 2016, traditional methods rank at the top on the Kitti dataset
- Global methods are nowhere near real-time implementations

Kitti Stereo 2015 Leaderboard: (2017) from Janai et al.
D1 error: <3px or <5%

Güney & Geiger (2015): SGM with context information to avoid ambiguities caused by e.g. reflections

---

### Table: Kitti Stereo 2015 Leaderboard

<table>
<thead>
<tr>
<th>Method</th>
<th>D1-bg</th>
<th>D1-fg</th>
<th>D1-all</th>
<th>Density</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Displets v2 – Güney &amp; Geiger (2015)</td>
<td>3.00 %</td>
<td>5.56 %</td>
<td>3.43 %</td>
<td>100.00 %</td>
<td>265 s / 8 cores</td>
</tr>
<tr>
<td>PBCP – Seki &amp; Pollefeys (2016)</td>
<td>2.58 %</td>
<td>8.74 %</td>
<td>3.61 %</td>
<td>100.00 %</td>
<td>68 s / GPU</td>
</tr>
<tr>
<td>MC-CNN-acrt – Žebontar &amp; LeCun (2016)</td>
<td>2.89 %</td>
<td>8.88 %</td>
<td>3.89 %</td>
<td>100.00 %</td>
<td>67 s / GPU</td>
</tr>
<tr>
<td>PRSM – Vogel et al. (2015)</td>
<td>3.02 %</td>
<td>10.52 %</td>
<td>4.27 %</td>
<td>99.99 %</td>
<td>300 s / 1 core</td>
</tr>
<tr>
<td>DispNetC – Mayer et al. (2016)</td>
<td>4.32 %</td>
<td>4.41 %</td>
<td>4.34 %</td>
<td>100.00 %</td>
<td>0.06 s / GPU</td>
</tr>
<tr>
<td>Content-CNN – Luo et al. (2016)</td>
<td>3.73 %</td>
<td>8.58 %</td>
<td>4.54 %</td>
<td>100.00 %</td>
<td>1 s / GPU</td>
</tr>
<tr>
<td>SPS-St – Yamaguchi et al. (2014)</td>
<td>3.84 %</td>
<td>12.67 %</td>
<td>5.31 %</td>
<td>100.00 %</td>
<td>2 s / 1 core</td>
</tr>
<tr>
<td>MDP – Li et al. (2016a)</td>
<td>4.19 %</td>
<td>11.25 %</td>
<td>5.36 %</td>
<td>100.00 %</td>
<td>11.4 s / 4 cores</td>
</tr>
<tr>
<td>OSF – Menze &amp; Geiger (2015)</td>
<td>4.54 %</td>
<td>12.03 %</td>
<td>5.79 %</td>
<td>100.00 %</td>
<td>50 min / 1 core</td>
</tr>
<tr>
<td>CSF – Lv et al. (2016)</td>
<td>4.57 %</td>
<td>13.04 %</td>
<td>5.98 %</td>
<td>99.99 %</td>
<td>80 s / 1 core</td>
</tr>
<tr>
<td>MBM – Einecke &amp; Eggert (2014)</td>
<td>4.69 %</td>
<td>13.05 %</td>
<td>6.08 %</td>
<td>100.00 %</td>
<td>0.13 s / 1 core</td>
</tr>
<tr>
<td>AABM – Einecke &amp; Eggert (2013)</td>
<td>4.88 %</td>
<td>16.07 %</td>
<td>6.74 %</td>
<td>100.00 %</td>
<td>0.08 s / 1 core</td>
</tr>
<tr>
<td>SGM – Hirschmüller (2008)</td>
<td>5.15 %</td>
<td>15.29 %</td>
<td>6.84 %</td>
<td>100.00 %</td>
<td>4.5 min / 1 core</td>
</tr>
</tbody>
</table>
Deep Learning Approach
Deep Learning Approach: Stereo timeline

Stereo Neural Networks have been evolving in complexity over time:

• Zbontar and LeCun (2016):
  CNN performs calculation of matching cost

• Mayer et al. (2016):
  Added conv. layers after cost computation to minimize distance between ground truth and network output

• Kendall et al. (2017):
  Includes classification information generated by additional conv. layer

• Chang et al. (2018):
  Adds spatial pooling to compute improved Cost Volume
Deep Learning Approach: Baseline Architecture

Architecture includes all 4 stages, s.t. end-to-end training is possible

Matching Cost Computation

Disparity estimation and refinement

3D Cost Volume dimensions:
- W: pixel width
- H: pixel height
- D: disparity

Global context information

Exemplary stereo baseline architecture, Kendall et al. (2017)
Deep Learning Approach: Current leader

Chang et al. (2018): 3D Convolutional Spatial Propagation Network (CSPN) is leading KITTI 2012 and 2015 stereo benchmark

• Extends PMNet (ranked 13)

• Uses Pyramid Scene Parsing Network to include global context information (ranked 1st on ImageNet challenge)
Deep Learning Approach: Leader’s architecture

Architecture overview from Chang et al. (2018):

- Emphasis on disparity refinement

1. Why Depth?
2. Mono vs. Stereo
3. Traditional Approach
4. Deep Learning Approach
5. Conclusions
Deep Learning Approach: Current leader

Chang et al. (2018): Learning Depth with Convolutional Spatial Propagation Network
Deep Learning - Depth from single image?
Deep Learning Approach: Mono Architecture

- MonoDepth does not use disparity ground truth
- Learns from global context through Encoder-Decoder architecture
- Includes a loss, that enforces:
  - Similar appearance
  - Disparity smoothness
  - Left-right consistency

\[ C_s = \alpha_{ap}(C_{ap}) + \alpha_{ds}(C_{ds}) + \alpha_{lr}(C_{lr}) \]

Godard et al. (2017): MonoDepth Architecture
Deep Learning Approach: Monodepth Network

1. Why Depth?
2. Mono vs. Stereo
3. Traditional Approach
4. Deep Learning Approach
5. Conclusions

*Godard et al. (2017): Unsupervised Monocular Depth Estimation*
But...

Neural Network seem to heavily overfit on street scenes
Deep Learning Approach: Depth Completion

Depth completion uses Lidar and RGB image to generate dense depth map

Chang et al. (2018) also rank among the top on the KITTI depth completion challenge
Comparison and Conclusion
Deep Learning: How good is it really?

1. Why Depth?
2. Mono vs. Stereo
3. Traditional Approach
4. Deep Learning Approach
5. Conclusions
Deep Learning: How good is it really?

Drouyer et al.: MC-CNN, Sparse stereo disparity densification using hierarchical image segmentation

Our OpenCV output
Comparison: Traditional vs. Deep Learning

**Kitti Stereo 2015 Leaderboard: (2017) from Janai et al.**

<table>
<thead>
<tr>
<th>Method</th>
<th>D1-bg</th>
<th>D1-fg</th>
<th>D1-all</th>
<th>Density</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Displets v2 – Güney &amp; Geiger (2015)</td>
<td>3.00 %</td>
<td>5.56 %</td>
<td>3.43 %</td>
<td>100.00 %</td>
<td>265 s / 8 cores</td>
</tr>
<tr>
<td>PBCP – Seki &amp; Pollefeys (2016)</td>
<td>2.58 %</td>
<td>8.74 %</td>
<td>3.61 %</td>
<td>100.00 %</td>
<td>68 s / GPU</td>
</tr>
<tr>
<td>MC-CNN-acrt – Žbontar &amp; LeCun (2016)</td>
<td>2.89 %</td>
<td>8.88 %</td>
<td>3.89 %</td>
<td>100.00 %</td>
<td>67 s / GPU</td>
</tr>
<tr>
<td>PRSM – Vogel et al. (2015)</td>
<td>3.02 %</td>
<td>10.52 %</td>
<td>4.27 %</td>
<td>99.99 %</td>
<td>300 s / 1 core</td>
</tr>
<tr>
<td>DispNetC – Mayer et al. (2016)</td>
<td>4.32 %</td>
<td>4.41 %</td>
<td>4.34 %</td>
<td>100.00 %</td>
<td>0.06 s / GPU</td>
</tr>
<tr>
<td>Content-CNN – Luo et al. (2016)</td>
<td>3.73 %</td>
<td>8.58 %</td>
<td>4.54 %</td>
<td>100.00 %</td>
<td>1 s / GPU</td>
</tr>
<tr>
<td>SPS-St – Yamaguchi et al. (2014)</td>
<td>3.84 %</td>
<td>12.67 %</td>
<td>5.31 %</td>
<td>100.00 %</td>
<td>2 s / 1 core</td>
</tr>
<tr>
<td>MDP – Li et al. (2016a)</td>
<td>4.19 %</td>
<td>11.25 %</td>
<td>5.36 %</td>
<td>100.00 %</td>
<td>11.4 s / 4 cores</td>
</tr>
<tr>
<td>OSF – Menze &amp; Geiger (2015)</td>
<td>4.54 %</td>
<td>12.03 %</td>
<td>5.79 %</td>
<td>100.00 %</td>
<td>50 min / 1 core</td>
</tr>
<tr>
<td>CSF – Lv et al. (2016)</td>
<td>4.57 %</td>
<td>13.04 %</td>
<td>5.98 %</td>
<td>99.99 %</td>
<td>80 s / 1 core</td>
</tr>
<tr>
<td>MBM – Einecke &amp; Eggert (2014)</td>
<td>4.69 %</td>
<td>13.05 %</td>
<td>6.08 %</td>
<td>100.00 %</td>
<td>0.13 s / 1 core</td>
</tr>
<tr>
<td>AABM – Einecke &amp; Eggert (2013)</td>
<td>4.88 %</td>
<td>16.07 %</td>
<td>6.74 %</td>
<td>100.00 %</td>
<td>0.08 s / 1 core</td>
</tr>
<tr>
<td>SGM – Hirschmüller (2008)</td>
<td>5.15 %</td>
<td>15.29 %</td>
<td>6.84 %</td>
<td>100.00 %</td>
<td>4.5 min / 1 core</td>
</tr>
</tbody>
</table>

**Kitti Stereo 2015 Leaderboard: (07/2019)**

<table>
<thead>
<tr>
<th>Method</th>
<th>D1-bg</th>
<th>D1-fg</th>
<th>D1-all</th>
<th>Density</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>M2S_CSPN – Cheng et al. (2018)</td>
<td>1.51 %</td>
<td>2.88 %</td>
<td>1.74 %</td>
<td>100.00 %</td>
<td>0.5 s / GPU @ 2.5 Ghz (C/C++)</td>
</tr>
<tr>
<td>GANet-deep – Zhang et al. (2019)</td>
<td>1.48 %</td>
<td>3.46 %</td>
<td>1.81 %</td>
<td>100.00 %</td>
<td>1.8 s / GPU @ 2.5 Ghz</td>
</tr>
<tr>
<td>AMNet – Du et al. (2019)</td>
<td>1.53 %</td>
<td>3.43 %</td>
<td>1.84 %</td>
<td>100.00 %</td>
<td>0.9 s / GPU @ 2.5 Ghz</td>
</tr>
</tbody>
</table>

1. Why Depth?
2. Mono vs. Stereo
3. Traditional Approach
4. Deep Learning Approach
5. Conclusions
Summary

• Traditional methods perform qualitatively well

• Global methods yield better results than local methods, but are computationally very expensive

• Semi-global combines advantages

• Deep Learning encapsulates all 4 stages of a stereo algorithm

• Deep Learning permits stereo and mono depth estimation, while stereo yields better results

• CNNs that follow encoder-decoder structure dominate
Conclusions

• Huge improvements over the past two years with Deep Learning
  - Better accuracy scores
  - Runtime is approaching real-time

• Deep Learning methods require high performance GPUs

• Image based depth estimation is still an open research topic

• Depth estimation from RGB data in combination with Lidar may be used to provide a dense depth map for autonomous driving applications
Primary sources

Thanks for your attention!