3D Vision Topics

Jaesik Park
jaesik.park@postech.ac.kr
Assistant Professor
Computer Science & Engineering
In This Lecture

• Addressing two practical problems
  • Camera tracking / pose estimation
  • Geometric Feature from Point Clouds
In This Lecture

• Addressing two practical problems
  • Camera tracking / pose estimation
  • Geometric Feature from Point Clouds
Motion Estimation

• Essential parts for intelligent systems
  • Self-driving cars, drones, robots
  • Virtual reality and augmented reality

Image courtesy of Waymo

Image courtesy of Forbes
Motion Estimation

• Sensory based approaches
  • An inertial measurement unit (IMU)
    • using a combination of accelerometers, gyroscopes, and sometimes magnetometers
  • Global positioning system (GPS)
    • satellite-based, outdoor only

• Camera based approaches
  • Fisheye Camera
  • Taking sequential inputs
  • Feature extraction
  • Pose estimation

Image courtesy of nasa.gov
Image courtesy of wikipedia.org
Camera tracking – VR/AR applications

Headset tracker

Oculus Rift
March 2016

Fish Eye cameras

Oculus Quest
May 2019
Camera based Motion Estimation

\[ \begin{align*}
\mathbf{c}_1 & \quad K_1: \text{known} \\
\mathbf{x}_1 & \\
\mathbf{c}_2 & \quad K_2: \text{known} \\
\mathbf{x}_2 & 
\end{align*} \]
Camera based Motion Estimation

\[ \begin{align*} 
\mathbf{c}_1 & \rightarrow x_1 \\
\mathbf{K}_1 : \text{known} & \\
\mathbf{c}_2 & \rightarrow x_2 \\
\mathbf{K}_2 : \text{known} & \\
\mathbf{R}, \mathbf{T} & \\
\end{align*} \]
Literature survey:
Direct Sparse Odometry

Jacob Engel et al., DSO: Direct Sparse Odometry, PAMI 2016
DSO - Pipeline

• Preprocessing
  • Prepare global Shutter camera with Fisheye lens
  • Camera calibration on: radiometric response, lens distortion, camera intrinsic

• Main tracking loop
  • Undistort images
  • Recover radiometric responses
  • Feature extraction
    • Intensity based, simple rule-based
  • Feature matching
  • Local pose optimization
    • With a few keyframes
Direct Sparse Odometry

Jakob Engel\textsuperscript{1,2}, Vladlen Koltun\textsuperscript{2}, Daniel Cremers\textsuperscript{1}

July 2016

\textsuperscript{1}Computer Vision Group
Technical University Munich

\textsuperscript{2}Intel Labs
Image Features

• Uses intensity directly
  • Requires careful radiometric calibration

• Simple features
  • Generate candidate points
    • Region adaptive gradient thresholds
  • Candidate points (green) are spread uniformly
    • Treating 32x32 blocks. Select if the largest gradient surpasses the threshold
    • Repeat this procedure three times.

Requirements

Simple image feature
Simple feature matching
Large field-of-view

• Detected points for tracking
• From the second pass
• From the third pass
Frame management

• Keyframe creation
  • Always keep a window of up to 7 active keyframes
  • Based on the field of view changes (detected by optical flow)
  • If camera exposure time changes significantly

• Discrete search along the Epipolar line
  • Inspired by LSD-SLAM
  • Photometric error, relatively distinguishable match

• Compute relative pose between
  • New frame
  • Key frame

• Blocked optimization

Requirements

- Simple image feature
- Simple feature matching
- Large field-of-view
Large field of view

• Narrow focal length
• Extremely large field-of-view (about 175 deg.), or called Fisheye
• Wider FoV = more clues for the motion estimation
  • Good for image based tracking

Requirements

- Simple image feature
- Simple feature matching
- Large field-of-view

https://vision.in.tum.de/data/datasets/visual-inertial-dataset
Lens Distortions

• Linear camera model
  • Image point and optical center are collinear.
  • Not realistic in the real (non-pinhole) lenses

• Radial distortion
  • Non-linear error
  • More significant as the focal length of the lens decrease

barrel distortion  pincushion distortion
Lens Distortions

• Radial distortion
  • Note: Lens distortion takes place during the initial projection of the world onto the image plane

\[ (\tilde{x}_{2d}, \tilde{y}_{2d}, 1)^T = x_{2d} = K[I|0]\tilde{x}_{3d} \]

\[ \begin{align*}
(x_{2d}) &= L(\hat{r}) (\tilde{x}_{2d}) \\
y_{2d} &= L(\hat{r}) (\tilde{y}_{2d})
\end{align*} \]

• Correction: using (low-order) polynomials

\[ \hat{x} = x_c + L(r)(x - x_c) \]
\[ \hat{y} = y_c + L(r)(y - y_c) \]
\[ L(r) = 1 + \kappa_1 r + \kappa_2 r^2 + \kappa_3 r^3 + \cdots \]
\[ r = (x - x_c)^2 + (y - y_c)^2 \]
Global shutter camera

**Rolling Shutter**
- Pixels are exposed roll by roll
- Usually has higher pixel count
- Good for stationary or slow-moving objects
- May distort image for moving objects

**Global Shutter**
- All pixels are exposed simultaneously
- Good for moving objects
- No image distortion
- Expensive

Images by Danielle Osman
Take-home message – Camera Tracking

• Practical issue
  • Use wide field of view camera
  • Use global shutter camera

• Simple approach runs real-time on CPU
  • Intensity based local feature for matching
    • It requires radiometric calibration and compensation
  • Blocked optimization with keyframes
    • Not frame by frame to minimize drift error

• Many rule-based approaches
  • Requires hand-craft designs
Literature survey:
DeepTAM: Deep Tracking and Mapping

Huizhong Zhou et al., DeepTAM: Deep Tracking and Mapping, ECCV 2018 Oral
Camera Tracking – Implicit Approach

• Using **deep learning** for the pose estimation
  • Deep network to extract features + match features + estimate poses
  • A encoder-decoder network architecture
  • Ensemble camera pose model for better stability
  • Adaptive cost matching scheme

• Learn *incremental movement* of camera motion
  • Direct regression of 6DoF camera pose estimation is challenging
DeepTAM: Deep Tracking and Mapping

Huizhong Zhou*  Benjamin Ummenhofer*  Thomas Brox

University of Freiburg

*equal contribution
DeepTAM: network architecture

• Tracking network

• Mapping network
• A map is a single depth map – can be rendered with the arbitrary view point
• Encoder-decoder type architecture for large receptive field
• The last part of the pose generation
  • Consists of $N = 64$ branches of stacked, fully connected layers
  • More stable and accurate than a single branch
A coarse-to-fine approach to efficiently estimate the current camera pose.

- We train three tracking networks:
  - Each specialized for a distinct resolution level
  - (80 × 60), (160 × 120) and (320 × 240)
Stereo Matching with Cost Volume

Image courtesy: https://github.com/PRiME-project/PRiMEStereoMatch/
• For cost-volume-based methods, accuracy is limited by the number of depth labels.
• **An adaptive narrow band strategy** to increase the sampling density while keeping the number of labels constant.
• Effects of the narrow band refinement.
• Without the refinement, the module lacks the knowledge of the band shape. This can help in capturing more details, but also causes strong artifacts.
### Comparison

<table>
<thead>
<tr>
<th></th>
<th>Keyframe</th>
<th>SGM</th>
<th>DTAM</th>
<th>DeMoN</th>
<th>Ours</th>
<th>GT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUN3D</td>
<td><img src="image1" alt="Keyframe" /></td>
<td><img src="image2" alt="SGM" /></td>
<td><img src="image3" alt="DTAM" /></td>
<td><img src="image4" alt="DeMoN" /></td>
<td><img src="image5" alt="Ours" /></td>
<td><img src="image6" alt="GT" /></td>
</tr>
<tr>
<td>SUNCG</td>
<td><img src="image7" alt="Keyframe" /></td>
<td><img src="image8" alt="SGM" /></td>
<td><img src="image9" alt="DTAM" /></td>
<td><img src="image10" alt="DeMoN" /></td>
<td><img src="image11" alt="Ours" /></td>
<td><img src="image12" alt="GT" /></td>
</tr>
<tr>
<td>MVS</td>
<td><img src="image13" alt="Keyframe" /></td>
<td><img src="image14" alt="SGM" /></td>
<td><img src="image15" alt="DTAM" /></td>
<td><img src="image16" alt="DeMoN" /></td>
<td><img src="image17" alt="Ours" /></td>
<td><img src="image18" alt="GT" /></td>
</tr>
</tbody>
</table>
Take-home message - DeepTAM

• Key to success
  • Map based approach
  • Map provides reference to be used for pose estimation
  • Synthesize frames to compare with current prediction and actual input

• U-net architecture + Image pyramid
  • To handle large motions effectively
  • Way better than the single U-net for pose estimation

• It is not scalable
  • Only builds and updates single depth map
  • We need to design efficient localization and mapping algorithms
In This Lecture

• Addressing two practical problems
  • Camera tracking / pose estimation
  • Geometric Feature from Point Clouds
3D Representation

- Multi-view RGB(D) Images
- Volumetric
- Polygonal mesh
- Point cloud
- Premitive-based CAD models
RGB-D Camera

Color camera

Depth camera (triangulated as a point cloud)
<table>
<thead>
<tr>
<th></th>
<th>Image</th>
<th>3D geometry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boundary</td>
<td>Fixed</td>
<td>Varying</td>
</tr>
<tr>
<td>Signal</td>
<td>Dense</td>
<td>Very sparse in 3D domain</td>
</tr>
<tr>
<td>Convolution</td>
<td>Well-defined</td>
<td>Questionable</td>
</tr>
</tbody>
</table>
Properties of Point Sets

• **Unordered.**
  • point cloud is a set of points without specific order.
  • A network that consumes N 3D point sets needs to be invariant to N! permutations of the input set in data feeding order.

• **Interaction among points.**
  • The model needs to be able to capture local structures from nearby points

• **Invariance under transformations.**
  • The learned representation of the point set should be invariant to certain transformations.
  • Rotating and translating points all together should not modify the global point cloud category nor the segmentation of the points.
Literature survey: PointNet

Charles R. Qi et al., PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, CVPR 2017
PointNet for point set understanding

- Classification
- Part Segmentation
- Semantic Segmentation
PointNet: Architecture
PointNet: Architecture

Point-wise, independent operation

Multilayer Perceptron
Multilayer perceptron (MLP)
Multilayer perceptron (MLP)
Multilayer perceptron (MLP)

- Simple way to make a use of trainable function that *can express any mapping function*
- Except for the input nodes, each node is a neuron that uses a nonlinear activation function
- MLP consists of multiple (at least three) fully connected layers
- Easy to adjust input-output dimensions
- Downside: a lot of parameters ($MxN+NxK$)
PointNet: Architecture
PointNet: Architecture

- An **affine transformation matrix** by a mini-network and directly apply this transformation to the coordinates of input points.
- **Transformation matrix** in the feature space: greatly increases the difficulty of optimization.
  - Adding a **regularization term** to softmax training loss.
  - Constrain the feature transformation matrix to be close to orthogonal matrix:

\[
L_{reg} = \| I - AA^T \|^2_F
\]
Theorical Analysis

• Universal approximation
  • Approximation ability of our neural network to continuous set functions
  • Worst case the network can learn to convert a point cloud into a volumetric representation, by partitioning the space into equal-sized voxels.

• Bottleneck dimension and stability
  • the expressiveness of our network is strongly affected by the dimension of the max pooling layer, i.e., $K$
  • $f(S)$ is determined by a finite subset $C$ of less or equal to $K$ elements.
  • learns to summarize a shape by a sparse set of key points.
PointNet: Architecture
PointNet Segmentation Network
Indoor Semantic Segmentation
PointNet: Comparison with other approaches achieving over invariance

Charles R. Qi et al., PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, CVPR 2017
Approaches to achieve order invariance

1) Sequential Model (such as LSTM)
Approaches to achieve order invariance

2) MLP with sorted/unsorted input

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP (unsorted input)</td>
<td>24.2</td>
</tr>
<tr>
<td>MLP (sorted input)</td>
<td>45.0</td>
</tr>
<tr>
<td>LSTM</td>
<td>78.5</td>
</tr>
</tbody>
</table>
3) MLP with symmetry function (PointNet architecture)

<table>
<thead>
<tr>
<th></th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP (unsorted input)</td>
<td>24.2</td>
</tr>
<tr>
<td>MLP (sorted input)</td>
<td>45.0</td>
</tr>
<tr>
<td>LSTM</td>
<td>78.5</td>
</tr>
<tr>
<td>Attention sum</td>
<td>83.0</td>
</tr>
<tr>
<td>Average pooling</td>
<td>83.8</td>
</tr>
<tr>
<td>Max pooling</td>
<td><strong>87.1</strong></td>
</tr>
</tbody>
</table>
Take-home message

- **PointNet**
  - Made for unordered 3D coordinates
  - Achieves state-of-the-art performance (at the time of publication)
  - Lightweight, consists of shared parameters
  - Dealing with point coordinate directly

- **Does PointNet solved the 3D shape understanding problem?**
  - Squeezing local context globally
  - Implicit approach to learn local geometry
  - Not attended for large scale data
Take-home message

• PointNet (2016)
  • Handling unordered point set
  • Invariant to local transformations
  • Limited receptive field

• PointNet++ (2017)
  • Structured PointNet to be more robust to uneven point set
  • Manual interpolation required

• PointNet variants (2017~Present)
  • Widely used for local point set structure learning
  • Applied to other problems that requires to handle unordered set
Sparse Convolution for 3D Data

B. Graham, Sparse 3D convolutional neural networks, BMVC 2015
C. Choy et al., 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks, CVPR 2019
Image classification
Image classification
Image classification
2D convolution

![2D Convolution Diagram](image-url)
Semantic Segmentation for Images

Zhao et al., Pyramid Scene Parsing Network, CVPR 2017
Semantic Segmentation for Images

*Noh et al., Learning Deconvolution Network for Semantic Segmentation, ICCV 2015*
Motivation

• What about large scaled scenes?
• Sliding window based local region analysis
• Limited receptive field size
Sparse convolution

How we analyze sparse data?

PointNet

Global feature

Symmetric function

(x, y, z) → MLP

(x, y, z) → MLP

(x, y, z) → MLP

(x, y, z) → MLP

(x, y, z) → MLP

(x, y, z) → MLP
PointNet - Questions

• Can it be extendable to the larger 3D scenes?
• Can we design the pipeline that can **utilize spatial relationship more efficiently**?
• Could the network could be **fully convolutional** in 3D space?
• Can we use **U-shape Network with downsampling and upsampling**?
• How to make the network to be **rotation and translation invariance**?
  • Like T-Net? Introduced in PointNet?
Timeline

• PointNet (2016)
  • Handling unordered point set
  • Invariant to local transformations
  • Limited receptive field

• PointNet++ (2017)
  • Structured PointNet to be more robust to uneven point set
  • Manual interpolation required

• PointNet variants (2017~Present)
  • Widely used for local point set structure learning
  • Applied to other problems that requires to handle unordered set

• Can we do this better?
Revisiting Volumetric Representation

• Required to put multiple voxels from multiple view points
  • Network maybe not deep enough
  • Not enough layers
  • No downsampling and upsampling
  • Not enough features to learn rotation invariant features

Charles R. Qi et al., Volumetric and Multi-View CNNs for Object Classification on 3D Data, CVPR 2016
Key observations

• Volumetric convolution?

• Point Sets in real-world are often very sparse
• Don’t have to consider dense voxel grid
Key observations

• Volumetric convolution?

Apply convolution only on the occupied voxels!

It is Sparse Convolution
U-Shape Network Architecture

Increase ability of spatial reasoning

Core elements: Sparse Convolution, Transposed Convolution, Pooling, Unpooling

*Each Sparse Convolution operator accompanies nonlinear functions
Sparse convolution

How we analyze sparse data?

PointNet

Symmetric function

Global feature

(x,y,z) MLP

(x,y,z) MLP

(x,y,z) MLP

(x,y,z) MLP

(x,y,z) MLP
Generalized Sparse Convolution

Step 1) Coordinate Quantization
Generalized Sparse Convolution

Step 2) Convolution

\[ \square = ? \]
Generalized Sparse Convolution

\[
\text{Conv} \left( \begin{array}{cccc}
\vdots & 
\vdots & 
\vdots & \\
\vdots & 
\vdots & 
\vdots & \\
\vdots & 
\vdots & 
\vdots & \\
\end{array} \right)
\]

\[
= \sum \left( \begin{array}{cccc}
\vdots & 
\vdots & 
\vdots & \\
\vdots & 
\vdots & 
\vdots & \\
\vdots & 
\vdots & 
\vdots & \\
\end{array} \right) \odot \left( \begin{array}{cccc}
\vdots & 
\vdots & 
\vdots & \\
\vdots & 
\vdots & 
\vdots & \\
\vdots & 
\vdots & 
\vdots & \\
\end{array} \right) \times \left( \begin{array}{cccc}
\vdots & 
\vdots & 
\vdots & \\
\vdots & 
\vdots & 
\vdots & \\
\vdots & 
\vdots & 
\vdots & \\
\end{array} \right) + \left( \begin{array}{cccc}
\vdots & 
\vdots & 
\vdots & \\
\vdots & 
\vdots & 
\vdots & \\
\vdots & 
\vdots & 
\vdots & \\
\end{array} \right)
\]

Indicator matrix \quad Input feature \quad weight \quad bias
Generalized Sparse Convolution

\[ \text{Indicator matrix} \oplus (\text{Input feature} \times \text{weight}) + \text{bias} \]

\[ = \sum \left( \begin{array}{c}
\text{Indicator matrix} \\
\text{Input feature} \\
\text{weight} \\
\text{bias}
\end{array} \right) \]
Generalized Sparse Convolution

\[
\text{Conv} \left( \begin{array}{ccc}
\textcolor{red}{x} & \textcolor{yellow}{y} & \textcolor{green}{z} \\
\textcolor{yellow}{y} & \textcolor{yellow}{y} & \textcolor{yellow}{y} \\
\textcolor{green}{z} & \textcolor{green}{z} & \textcolor{green}{z}
\end{array} \right) = \begin{array}{c}
\textcolor{blue}{\text{output}}
\end{array}
\]

\[
\text{f}_{\text{non-linear}} \left( \begin{array}{c}
\textcolor{blue}{\text{input}}
\end{array} \right) = \begin{array}{c}
\textcolor{brown}{\text{output}}
\end{array}
\]
Transposed Convolution

Pooling

Unpooling

Convolution

Transposed convolution
Transposed Convolution

• The convolution mask is transposed, weighted, and attached.

Image courtesy by Vincent Dumoulin
Efficiency

• Hash structure for storing occupancy

Idea: Use a quantized coordinate as a hash key to store occupancy
Efficiency

- Hash structure for fast inferencing

Task: What are adjacent voxels around (5,5,5)?

Query:
- (4,4,5)
- (5,4,5)
- (6,4,5)
- (4,5,5)
- (5,5,5)
- ...
- (6,6,5)

Result:
- Yes
- Yes
- Yes
- Yes
- ...
- No

Time complexity:
\( O(1) \)
Revisiting Volumetric Representation

<table>
<thead>
<tr>
<th></th>
<th>PointNet variants</th>
<th>Sparse Convolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large scene?</td>
<td>Sliding window</td>
<td>Yes. It’s easy!</td>
</tr>
<tr>
<td>Spatial Relationship?</td>
<td>Hand-crafted grouping</td>
<td>Yes, via hashing!</td>
</tr>
<tr>
<td>Fully convolutional</td>
<td>No</td>
<td>Yes!</td>
</tr>
<tr>
<td>U-shape Network?</td>
<td>Implicit</td>
<td>Yes it is!</td>
</tr>
<tr>
<td>Rotation and translation?</td>
<td>Yes</td>
<td>Yes, via deep network</td>
</tr>
</tbody>
</table>
Fully Convolutional Geometric Features

C. Choy et al., Fully Convolutional Geometric Features, ICCV 2019
U-Shape Network Architecture

Increase ability of spatial reasoning

Core elements: Sparse Convolution, Transposed Convolution, Pooling, Unpooling
Architecture

- Follows the design of standard U-Shape Network
  - Consists of 3D Conv (sparse convolution in 3D) and transposed convolution
  - U-shape network with skip connections
  - Residual block for better performance
  - Batch normalization
Performance

Hand crafted features (histogram of surface normal, and so on)

Now we are familiar with this

Learn to compress SHOT feature

Variants of PointNet using different orientation, encoding schemes, learning schemes

Local Volumetric Features (uses TSDF)

Fully Convolutional, Large receptive field
Performance

Maximum and Reliable Pareto optimality
• Trade-off between speed and accuracy
Indoor Scenes (Sun3D)

Geometric features were colored with T-SNE
Outdoor Scenes (KITTI Lidar Dataset)
Take-home message

• Fully Convolutional Network in 3D
  • Accurate, efficient
  • Can be configured with U-shape Net, Residual Net
  • Utilizes spatial relationship
  • Large field of view
  • Better than manual configurations used for PointNet variants
    • No preprocessing without O(1) hashing and quantization
    • Does not require TSDF volume, grouping, and so on.

• Just got accepted
  • Try for your own project
  • Code is available: https://github.com/chrischoy/fcgf