The 3D-Reconstruction Problem

In this talk, we consider how to compute the 3D structure of a static scene from its multiple images captured by a calibrated camera.

Two related sub-tasks:

1. camera motion estimation.
2. 3D structure computation.
The standard approach for 3D-Reconstruction

• The standard approach to 3D reconstruction is...
  
  - the “structure-FROM-motion” in its literal sense, meaning that,...
  
  • First, estimate camera motion via e.g. epipolar geometry or fundamental matrix.
  
  • Then, compute 3D structure via intersection and resectioning.
  
  • Finally, bundle adjustment.
This approach has been quite successful in practice

- Represents the state-of-the-art 3D reconstruction technique.
- Two representative systems.
  - **PhotoTourism** (Snavely, et. al.); Build Rome in a day (Agarwal, et al.)
  - **3D Urban Modeling** (Pollefeys et.al.)
Motion-Estimation is almost always the first step in the processing pipeline...

- While most existing approaches follow the "structure-from-motion" paradigm, ... in this work,
Motion-Estimation is almost always the first step in the processing pipeline...

• While most existing approaches follow the "structure-from-motion" paradigm, ..., in this work,

• We intend to do it differently ...

Video by Sameer Agarwal et.al.
Motion-Estimation is almost always the first step in the processing pipeline...

• While most existing approaches follow the “structure-from-motion” paradigm, ..., in this work,

• We intend to do it differently ...

  • Can we bypass the Motion-Estimation stage, and go directly to structure computation?

Figure taken from http://www.tnt.uni-hannover.de/print/project/motionestimation/index.php

Video by Sameer Agarwal et.al.
Motion-Estimation is almost always the first step in the processing pipeline...

- While most existing approaches follow the “structure-from-motion” paradigm, ..., in this work,

- We intend to do it differently ...
  - Can we bypass the Motion-Estimation stage, and go directly to structure computation?
  - Can we do it as “motion FROM structure”?

Figure taken from http://www.tnt.uni-hannover.de/print/project/motionestimation/index.php

Video by Sameer Agarwal et. al.
Motivations

1. Apart from intellectual curiosity, it may provide other benefits as well, e.g.,

2. By avoiding explicit motion computation, we may be able to avoid the well-known noise sensitivity/inherent ambiguity in motion estimation.

- Rotation-Translation (Bas-Relief) ambiguity;


Our goal

- To reconstruct (solely) the **3D Structure (3D Shape)** of a set of N point clouds from M calibrated camera measurements, but not the cameras’ motion (rotation and translation).
How do we describe the 3D shape of point clouds?

- X-Y-Z coordinates representation
- \((3N-7)\) independent parameters (modulo rotation, translation, scale).
How do we describe the 3D shape of point clouds?

- X-Y-Z coordinates representation
- \((3N-7)\) independent parameters (modulo rotation translation, scale).
- Inter-point distance representation.
- Point -> Graph vertex
- Distance -> Graph edges
Coordinate Representation \textbf{VERSUS} Distance Representation

- From x-y-z coordinate representation, to compute distances (edge length) is trivial:
  \[ l_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \]

- From distances to recover coordinates (up to Euclidean motion) is also possible:

- \textbf{known as:} \textit{graph embedding problem}. 

Wednesday, 23 June 2010
Graph-Embedding Problem

- a.k.a.
  - Graph Realization
  - Graph Drawing

- Closely relates to
  - MDS (multi-dimensional scaling)
  - Dimen-Reduction
  - IsoMap/LLE

- Has also been studied in Structural Chemistry/Biology.
The Molecule Problem

- Suppose that you given distance constraints on the inter-atom distances, determine a 3D shape of a molecule that satisfies those distance constraints.

Key message: the original 3D shape/structure can be faithfully recovered from a sufficient set of inter-point distances.
Our unique idea

• In doing 3D structure reconstruction,
• instead of directly recovering the 3D coordinates of the point clouds,
• we compute the inter-point distances between pairs of 3D points.

• why this is relevant to the proposed “motion from structure”, we shall see later ...
How to measure distance in 3-space?
How to measure distance in 3-space?

- We cannot directly measure distance in 3D with a camera.

\[ l, \theta \]
How to measure distance in 3-space?

- We cannot directly measure distance in 3D with a camera.
- We however, can measure the **viewing angle**, from a calibrated camera.

\[ l \]

\[ \theta \]
How to measure distance in 3-space?

- We cannot directly measure distance in 3D with a camera.
- We however, can measure the viewing angle, from a calibrated camera.
- \( l \) is the unknown we want to estimate.

\[ \theta \]
How to measure distance in 3-space?

- We cannot directly measure distance in 3D with a camera.
- We however, can measure the viewing angle, from a calibrated camera.
- $l$ is the unknown we want to estimate.
- $\theta$ is what we can measure.
How to measure distance in 3-space?

- We cannot directly measure distance in 3D with a camera.
- We however, can measure the **viewing angle**, from a calibrated camera.
- \( l \) is the unknown we want to estimate.
- \( \theta \) is what we can measure.
- they are related by ...
Basic Equation System

- Viewing-triangle
- the law of cosine
- polynomial equation
- system of equations

Each viewing-triangle contributes to one basic equation. Collecting sufficient number of basic equations, and solve this equation system, then we compute the unknown \( l_s \).
Roadmap

• So far,
  ✓ we have presented the basic idea, and the basic equation system.

• Next, we study,
  ✦ [practice] *how to solve the system efficiently*?
  ✦ [theory] *when the system will admit unique solution*?
How to solve the system of basic equations efficiently?

\[
\begin{align*}
    x_i^2 + x_j^2 - 2x_i x_j \cos \theta_{ij} &= l_{ij}^2 \\
    x_k^2 + x_l^2 - 2x_k x_l \cos \theta_{kl} &= l_{kl}^2 \\
    \ldots \ldots
\end{align*}
\]
How to solve the system of basic equations efficiently?

\[
\begin{align*}
  x_i^2 + x_j^2 - 2x_i x_j \cos \theta_{ij} &= l_{ij}^2 \\
  x_k^2 + x_l^2 - 2x_k x_l \cos \theta_{kl} &= l_{kl}^2 \\
  \ldots \ldots
\end{align*}
\]

- a set of second order (quadratic) equations
How to solve the system of basic equations efficiently?

\[
\begin{align*}
&x_i^2 + x_j^2 - 2x_ix_j \cos \theta_{ij} = l_{ij}^2 \\
&x_k^2 + x_l^2 - 2x_kx_l \cos \theta_{kl} = l_{kl}^2 \\
\ldots &
\end{align*}
\]

- a set of second order (quadratic) equations
- in the unknowns of \{x’s, l’s\}. 
How to solve the system of basic equations efficiently?

\[
\begin{align*}
    x_i^2 + x_j^2 - 2x_i x_j \cos \theta_{ij} &= l_{ij}^2 \\
    x_k^2 + x_l^2 - 2x_k x_l \cos \theta_{kl} &= l_{kl}^2
\end{align*}
\]

- a set of second order (quadratic) equations
- in the unknowns of \{x’s, l’s\}.
- In theory, one could possibly solve it via, e.g.
How to solve the system of basic equations efficiently?

\[
\begin{align*}
\{ & x_i^2 + x_j^2 - 2x_i x_j \cos \theta_{ij} = l_{ij}^2 \\
& x_k^2 + x_l^2 - 2x_k x_l \cos \theta_{kl} = l_{kl}^2 \\
& \text{......}
\end{align*}
\]

- a set of second order (quadratic) equations
- in the unknowns of \{x’s, l’s\}.
- In theory, one could possibly solve it via, e.g.
  - Gröbner Basis
How to solve the system of basic equations efficiently?

\[
\begin{aligned}
    x_i^2 + x_j^2 - 2x_i x_j \cos \theta_{ij} &= l_{ij}^2 \\
    x_k^2 + x_l^2 - 2x_k x_l \cos \theta_{kl} &= l_{kl}^2 \\
    \ldots
\end{aligned}
\]

- a set of second order (quadratic) equations
- in the unknowns of \{x’s, l’s\}.
- In theory, one could possibly solve it via, e.g.
  - Gröbner Basis
  - Continuation
How to solve the system of basic equations efficiently?

\[
\begin{align*}
  x_i^2 + x_j^2 - 2x_i x_j \cos \theta_{ij} &= l_{ij}^2 \\
  x_k^2 + x_l^2 - 2x_k x_l \cos \theta_{kl} &= l_{kl}^2 \\
  \ldots &
\end{align*}
\]

- a set of second order (quadratic) equations
- in the unknowns of \{x’s, l’s\}.
- In theory, one could possibly solve it via, e.g.
  - Gröbner Basis
  - Continuation
  - Global optimization
How to solve the system of basic equations efficiently?

\[
\begin{align*}
x_i^2 + x_j^2 - 2x_ix_j \cos \theta_{ij} &= l_{ij}^2 \\
x_k^2 + x_l^2 - 2x_kx_l \cos \theta_{kl} &= l_{kl}^2 \\
\ldots
\end{align*}
\]

- a set of second order (quadratic) equations
- in the unknowns of \( \{x's, l's\} \).
- In theory, one could possibly solve it via, e.g.
  - Gröbner Basis
  - Continuation
  - Global optimization
- In practice, this is an extremely difficult problem, even for moderate size problems.
Strategies for solving it

\[
\begin{align*}
    x_i^2 + x_j^2 - 2x_i x_j \cos \theta_{ij} &= l_{ij}^2 \\
    x_k^2 + x_l^2 - 2x_k x_l \cos \theta_{kl} &= l_{kl}^2 \\
    \ldots
\end{align*}
\]

- a set of quadratic non-convex equalities
- reduce/relax it ==> 
- a set of linear equations, subject to convex constraints.
Matrix trace form

Define vector $\mathbf{x} = [x_1, x_2, \ldots, x_k]^T$, and unit basis vector $\mathbf{e}_i = [0, 0, \ldots, 1, \ldots, 0, 0, 0]^T$, and

$$a_{ij} = (\mathbf{e}_i + \mathbf{e}_j)^T \mathbf{C} (\mathbf{e}_i + \mathbf{e}_j)$$

then each $x_i^2 + x_j^2 - 2x_ix_j \cos \theta = l_k^2$ becomes,

$$\text{Tr}(a_{ij} \mathbf{x} \mathbf{x}^T) = l_k^2$$

still quadratic.
Further relaxations

• quadratic to linear:

\[ Y = xx^T \]

not convex

• non-convex equality to convex inequality:

\[ Y \succeq xx^T \]

convex

• clearly \( Y \) is of rank(1), and we replace this to

\[ \min Tr(Y) \]

• add regularization term to avoid all-zero solutions.
The geometry intuition of $Y = xx^T$ $\Rightarrow$ $Y \preceq xx^T$

- **Intersecting point of quadratic curves $\Rightarrow$ Common intersection of convex conical regions.**
The final formulation: finding edges

\[
\min_l \left( \text{Tr}(Y) - \|x\|_1 \right) \quad \text{such that:}
\]

\[
\text{Tr} \left( a_{ij} Y \right) = l_k^2, \quad \forall \triangle (i,j,k),
\]

\[
Y \succeq xx^T, \quad \|l\|_2^2 = 1, \quad x \geq 0, \quad l \geq 0.
\]

- minimize linear objective function, subject to,
- linear and semi-definite constraints.
The final formulation: finding edges

\[ \min_{l} \left( \text{Tr}(Y) - \|x\|_1 \right) \quad \text{such that:} \]

\[ \text{Tr} \left( a_{ij} Y \right) = l^2_k, \quad \forall \triangle(i,j,k), \]

\[ Y \succeq xx^T, \quad \|l\|_2^2 = 1, \quad x \geq 0, \quad l \geq 0. \]

- minimize linear objective function, subject to,
- linear and semi-definite constraints.

SDP (Semi-definite programming)
How to address the solution uniqueness question?

- Motivation:
  - which set of inter-point distances to choose?
  - how many edges (inter-point) distances are necessary, or sufficient?
  - how to guarantee the established polynomial system (or the relaxed SDP) has unique solution?

\[
\begin{align*}
  x_i^2 + x_j^2 - 2x_i x_j \cos \theta_{ij} &= l_{ij}^2, \\
  x_k^2 + x_l^2 - 2x_k x_l \cos \theta_{kl} &= l_{kl}^2, \\
  &\ldots
\end{align*}
\]

\[
\min \left( \text{Tr}(Y) - \|x\|_1 \right) \quad \text{such that:}
\]

\[
\begin{align*}
  \text{Tr}(a_{ij} Y) &= l_{ij}^2, \\
  Y &\succeq xx^T, \\
  \|l\|_2 = 1, \\
  x &\geq 0, \\
  l &\geq 0.
\end{align*}
\]
Resort to graph rigidity theory

• To answer this question rigorously, we resort to the graph rigidity theory, in particular the property of “global rigidity”.

• Graph rigidity theory studies ...

  ● when is a given framework rigid?
Graph Rigidity Theory: some definitions

- A (bar-and-joint) **framework** is a graph $G$ together with a finite set of vectors in $d$-space, denoted by $p = (p_1, p_2, \ldots, p_n)$, where each $p_i$ corresponds to a node of $G$, and the edges of $G$ correspond to fixed length bars connecting the nodes.

- Consider a straight-line embedding of a graph in Euclidean space. When any other embedding of the graph with the same edge lengths is **congruent** to the original, we say it is **globally rigid**.

- $G(V,E)$ is called **redundantly rigid** if $G(V, E-e)$ is rigid for all $e$, i.e. the removal of a single edge $e$ from the rigid graph $G$ does not destroy rigidity.
Eg: Not globally-rigid graph
Eg: **Not** globally-rigid graph
Eg: Not globally-rigid graph
Eg: Not globally-rigid graph
Eg: Globally-rigid graph

- $p$ is globally-rigid in $\mathbb{R}^d$, if there is no second framework in $\mathbb{R}^d$ with the same edge lengths.
In our 3D reconstruction context...

- Our aim is to construct a redundantly **globally-rigid**, spanning graph involving both 3D points and camera centers.
How to tell when a given framework is globally-rigid?

- In general, to exactly test global rigidity is very hard! (NP-Complete).
- We instead propose using a randomized algorithm to test the generic global rigidity approximately.
- This randomized algorithm is due to Gortler-Healy-Thurston’s recent result on characterizing generic global rigidity.


- [Theorem] A weighted graph is **globally-rigid** if and only if the rank of the associated **stress matrix** is N-d+1.
Algorithm Sketch

1. From all available point matches, incrementally construct a spanning graph by adding viewpoint triangles, until it is redundantly globally rigid.

   • *To check graph-rigidity, run the randomized algorithm of Gortler et al.*

2. Run the edge-finding SDP to compute the edge distances of the graph.

3. Do a graph embedding to recover 3D coordinates of the point clouds.
Experimental validation

• Purposes:
  • Validate the effectiveness of the SDP relaxation for edge finding.
  • Verify the reconstructed 3D coordinates compared with ground truth.
  • Test performance versus #(points), #(views), noise-levels, and time-complexity, etc.
  • Preliminary tests on toy-size problems.
• Generate 30 random 3D points in $[0,1]^3$, project to $M$ images, add Gaussian noise with std 0.001.

• Construct a complete graph.

• Solve two SDPs.

• Reconstruction result is shown on the right.

• RMSE coordinate error much less than noise std.
Reconstruction error v/s noise level

Figure 7. Error vs. Noise level: RMS error for estimated edge-length vs. Level of noise.
Worst case time-complexity (running time)

- Test on a modest 2.2G Intel Laptop.
- Using matlab (SEDUMI) as the SDP solver.
- Worst-case (i.e. complete graph): $O(N^4)$, where $N=$ #points.

Figure 8. Timing result: Computational time vs. Number of points; (Note: this is for a 3-view complete graph case, i.e. worst-case complexity.)
Error v/s Number of points

Figure 3. Error vs. Number of points: RMS error for the estimated edge lengths Versus number of points (2-view case).

Figure 5. Error in Coordinates vs. points: RMS error for estimated 3D point coordinates Versus number of points (3-view case).

- Minimal cases: 2-view 5-point, 3-view 4-points.
Figure 6. Error vs. Number of Views: RMS error for estimated edge-length vs. Number of views (15-point case).
Tests on real images (1):
with sparse, non-complete graph

Figure 9. Box-book—a toy Sfm problem. Left: images; right: reconstructed 3D points.


Figure 10. Oxford Corridor. Left: one of the input images; right: reconstructed 3D points.
Test on real image (2):
with Delaunay mesh
Conclusions
Conclusions

- A novel, unconventional 3D reconstruction paradigm.
Conclusions

• A novel, unconventional 3D reconstruction paradigm.
• No need to learn epipolar geometry, or fundamental matrix.
Conclusions

• A novel, unconventional 3D reconstruction paradigm.

• No need to learn epipolar geometry, or fundamental matrix.

• Finding viewing-triangle is largely a local operation, making the method potentially suitable for sparse, large scale input.
Conclusions

• A novel, unconventional 3D reconstruction paradigm.

• No need to learn epipolar geometry, or fundamental matrix.

• Finding viewing-triangle is largely a local operation, making the method potentially suitable for sparse, large scale input.

• Limitations:
  - Rely on calibrated camera views.
  - Sensitive to outliers.
  - Computational complexity due to SDP.
  - Lack of quantitative comparisons with conventional methods (i.e. which one is better? not yet tested).
Thank you!