Structure from Motion

3D Vision
University of Illinois

Derek Hoiem
Structure from Motion (SfM)

Goal: Solve for camera poses and 3D points in scene
Example Application: Inspection

Enable inspection in hard to reach areas with drone photos and 3D reconstruction

- Create 3D model from images
- Provide tools to inspect on images and map interactions to 3D
Incremental SfM

1. Compute features

2. Match images

3. Reconstruct
   a) Solve for poses and 3D points in two cameras
   b) Solve for pose of additional camera(s) that observe reconstructed 3D points
   c) Solve for new 3D points that are viewed in at least two cameras
   d) Bundle adjust to minimize reprojection error
Incremental SFM: **detect features**

- Feature types: SIFT, ORB, Hessian-Laplacian, ...

Each circle represents a set of detected features
Incremental SFM: **match features and images**

For each pair of images:

1. Match feature descriptors via approximate nearest neighbor
2. Solve for F or E and find inlier feature correspondences

Points of same color have been matched to each other
Incremental SFM: create tracks graph

tracks graph: bipartite graph between observed 3D points and images
Incremental SFM: initialize reconstruction

1. Choose two images that are likely to provide a stable estimate of relative pose
   - E.g., \( \frac{\text{# inliers for } H}{\text{# inliers for } F} < 0.7 \) and many inliers for \( F \)
2. Get focal lengths from EXIF, estimate essential matrix using 5-point algorithm, extract pose \( R_2, t_2 \) with \( R_1 = I, t_1 = 0 \)
3. Solve for 3D points given poses
4. Perform bundle adjustment to refine points and poses

filled circles = “triangulated” points
filled rectangles = “resectioned” images (solved pose)
Bundle adjustment

- Non-linear method for refining structure and pose
- Minimizing reprojection error

\[ E(P, X) = \sum_{i=1}^{m} \sum_{j=1}^{n} D(x_{ij}, P_i X_j)^2 \]
Incremental SFM: **grow reconstruction**

1. **Resection:** solve pose for image(s) that have the most triangulated points
2. **Triangulate:** solve for any new points that have at least two cameras
3. **Bundle adjust**
4. Optionally, align with GPS from EXIF or ground control points (GCP)

Filled circles = “triangulated” points
Filled rectangles = “resectioned” images (solved pose)
Incremental SFM: **grow reconstruction**

1. Resection: solve pose for image(s) that have the most triangulated points
2. Triangulate: solve for any new points that have at least two cameras
3. Bundle adjust
4. Optionally, align with GPS from EXIF or ground control points (GCP)

filled circles = “triangulated” points  
filled rectangles = “resectioned” images (solved pose)
Why SfM is hard

- **Slow**
  - Matching $N^2$ pairs of images takes too long (~1-4s per pair)
  - Bundle adjustment takes longer with more images and needs to be repeated as images are added: up to $O(N^3)$
  - Grow reconstruction phase is not easy to parallelize

- **Bad feature matches are very common and cause misregistrations**

- **Insufficient feature matches cause incomplete reconstructions**

---

<table>
<thead>
<tr>
<th># Images</th>
<th># Registered</th>
<th>Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Theia</td>
<td>Bundler</td>
</tr>
<tr>
<td>Rome [14]</td>
<td>74,394</td>
<td>–</td>
</tr>
<tr>
<td>Quad [14]</td>
<td>6,514</td>
<td>–</td>
</tr>
<tr>
<td>Dubrovnik [36]</td>
<td>6,044</td>
<td>–</td>
</tr>
<tr>
<td>Alamo [61]</td>
<td>2,915</td>
<td>582</td>
</tr>
<tr>
<td>Ellis Island [61]</td>
<td>2,587</td>
<td>231</td>
</tr>
<tr>
<td>Gendarmenmarkt [61]</td>
<td>1,463</td>
<td>703</td>
</tr>
<tr>
<td>Madrid Metropolis [61]</td>
<td>1,344</td>
<td>351</td>
</tr>
<tr>
<td>Montreal Notre Dame [61]</td>
<td>2,298</td>
<td>2,546</td>
</tr>
<tr>
<td>NYC Library [61]</td>
<td>2,550</td>
<td>339</td>
</tr>
<tr>
<td>Piazza del Popolo [61]</td>
<td>2,251</td>
<td>335</td>
</tr>
<tr>
<td>Piccadilly [61]</td>
<td>7,381</td>
<td>2,270</td>
</tr>
<tr>
<td>Roman Forum [61]</td>
<td>2,364</td>
<td>1,074</td>
</tr>
<tr>
<td>Tower of London [61]</td>
<td>1,576</td>
<td>468</td>
</tr>
<tr>
<td>Trafalgar [61]</td>
<td>15,685</td>
<td>5,067</td>
</tr>
<tr>
<td>Union Square [61]</td>
<td>5,961</td>
<td>720</td>
</tr>
<tr>
<td>Vienna Cathedral [61]</td>
<td>6,288</td>
<td>858</td>
</tr>
<tr>
<td>Yorkminster [61]</td>
<td>3,368</td>
<td>429</td>
</tr>
</tbody>
</table>

from COLMAP SfM (Schonberger et al. 2016)

Bad matches in low texture, repetitive hallway cause COLMAP to fail to reconstruct loop (Kataria et al. 2020)
Incremental SfM, Take 2: improvements in green

1. Compute features

2. Match images

3. Reconstruct
   a) Solve for poses and 3D points in two cameras
   b) Solve for pose of additional camera(s) that observe reconstructed 3D points
   c) Solve for new 3D points that are viewed in at least two cameras
   d) Bundle adjust to minimize reprojection error
Incremental SFM: detect features

- Feature types: SIFT, ORB, Hessian-Laplacian, ...
- Use GPU for fast feature computation

Each circle represents a set of detected features
Incremental SFM: match features and images

Find match candidates:
• Match K closest images in GPS distance or time
• Use vocab tree on features to find K most similar images
• Potentially, add new candidates based on candidates that are already found

For each pair of candidate images:
1. Match feature descriptors via approximate nearest neighbor
   - GPU can be used for fast feature matching
   - Lowe’s ratio test used to reject some potentially bad matches
2. Solve for F or E and find inlier feature correspondences
   - Remove feature matches that have above threshold reprojection error according to F or E
   - Discard image pairs that have below threshold number of geometrically verified matches

Points of same color have been matched to each other
Incremental SFM: create tracks graph

tracks graph: bipartite graph between observed 3D points and images
Incremental SFM: initialize reconstruction

1. Choose two images that are likely to provide a stable estimate of relative pose
   - E.g., \( \frac{\text{# inliers for } H}{\text{# inliers for } F} < 0.7 \) and many inliers for \( F \)
2. Get focal lengths from EXIF, estimate essential matrix using \textbf{5-point algorithm}, extract pose \( R_2, t_2 \) with \( R_1 = I, t_1 = 0 \)
3. Solve for 3D points given poses
4. Perform bundle adjustment to refine points and poses

filled circles = “triangulated” points
filled rectangles = “resectioned” images (solved pose)
Triangulation: Linear Solution

Given $P, P', x, x'$

1. Precondition points and projection matrices
2. Create matrix $A$
3. $[U, S, V] = \text{svd}(A)$
4. $X = V(:, \text{end})$

Pros and Cons
• Works for any number of corresponding images
• Not projectively invariant

Code: [http://www.robots.ox.ac.uk/~vgg/hzbook/code/vgg_multiview/vgg_X_from_xP_lin.m](http://www.robots.ox.ac.uk/~vgg/hzbook/code/vgg_multiview/vgg_X_from_xP_lin.m)
Triangulation: Non-linear Solution

- Minimize projected error while satisfying

\[ \hat{x}'^T F \hat{x} = 0 \]

\[ cost(X) = \text{dist}(x, \hat{x})^2 + \text{dist}(x', \hat{x}')^2 \]

Figure source: Robertson and Cipolla (Chpt 13 of Practical Image Processing and Computer Vision)
Bundle adjustment

- Non-linear method for refining structure and motion
- Minimizing reprojection error

\[ E(P, X) = \sum_{i=1}^{m} \sum_{j=1}^{n} D(x_{ij}, P_i X_j)^2 \]

Ceres Solver

Use robust loss for reprojection error, such as Huber
Incremental SFM: grow reconstruction

1. Sort images, e.g. by number of triangulated points
   a. Resection: solve pose for image(s) that have the most triangulated points
   b. Triangulate: solve for any new points viewed by at least two reconstructed cameras
   c. Remove 3D points that do not have enough baseline or too high reprojection error in any camera (optionally, split into multiple tracks)
   d. Bundle adjust
      • Only do full bundle adjust after some percent of new images are resectioned (huge time savings for large reconstructions)

2. Optionally, align with GPS from EXIF or ground control points (GCP)

filled circles = “triangulated” points
filled rectangles = “resectioned” images (solved pose)
Incremental SFM: **grow reconstruction**

1. Sort images, e.g. by number of triangulated points
   a. Resection: solve pose for image(s) that have the most triangulated points
   b. Triangulate: solve for any new points viewed by at least two reconstructed cameras
   c. Remove 3D points that do not have enough baseline or too high reprojection error in any camera (optionally, split into multiple tracks)
   d. Bundle adjust
      • Only do full bundle adjust after some percent of new images are resectioned (huge time savings for large reconstructions)

2. Optionally, align with GPS from EXIF or ground control points (GCP)

Filled circles = “triangulated” points
Filled rectangles = “resectioned” images (solved pose)
Improving Structure from Motion with Reliable Resectioning
False matches on repeated structures cause catastrophic failures
Resectioning is a critical step

1. Select image that views the most triangulated points

2. Estimate pose of image using all the triangulated points (PnP algorithm using RANSAC)
Ambiguity-adjusted match score (AAM): Discount longer tracks that are more likely to correspond to duplicate structures.
Local resectioning order uses most similar image

We use points from a smaller set of reliable images to determine **resectioning order** and **pose estimation**
Local pose estimation uses reliable images only

We use points from a smaller set of reliable images to determine resectioning order and pose estimation.
Local pose estimation uses reliable images only

Use points from a smaller set of reliable images to determine resectioning order and pose estimation.
Our method improves standard pipelines

Local resectioning using ambiguity-adjusted matches compared against baselines (standard OpenSfM and COLMAP pipelines)

<table>
<thead>
<tr>
<th>Duplicate Structures Dataset</th>
<th>OpenSfM</th>
<th>OpenSfM w/ Our Resectioning</th>
<th>COLMAP</th>
<th>COLMAP w/ Our Resectioning</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 Failures</td>
<td>6 Successes</td>
<td>6 Failures</td>
<td>6 Failures</td>
<td>2 Failures</td>
</tr>
<tr>
<td>9 Failures</td>
<td>3 Failures</td>
<td>7 Failures</td>
<td>7 Failures</td>
<td>5 Failures</td>
</tr>
<tr>
<td>3 Partial Successes</td>
<td>13 Successes</td>
<td>2 Partial Successes</td>
<td>3 Partial Successes</td>
<td>8 Successes</td>
</tr>
<tr>
<td>4 Successes</td>
<td>6 Successes</td>
<td>7Successes</td>
<td>8 Successes</td>
<td>7 Successes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UIUCTag Dataset</th>
<th>OpenSfM</th>
<th>OpenSfM w/ Our Resectioning</th>
<th>COLMAP</th>
<th>COLMAP w/ Our Resectioning</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 Failures</td>
<td>3 Failures</td>
<td>7 Failures</td>
<td>7 Failures</td>
<td>5 Failures</td>
</tr>
<tr>
<td>3 Partial Successes</td>
<td>13 Successes</td>
<td>2 Partial Successes</td>
<td>3 Partial Successes</td>
<td>8 Successes</td>
</tr>
<tr>
<td>4 Successes</td>
<td>6 Successes</td>
<td>7 Successes</td>
<td>8 Successes</td>
<td>7 Successes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TanksAndTemples Dataset</th>
<th>OpenSfM</th>
<th>OpenSfM w/ Our Resectioning</th>
<th>COLMAP</th>
<th>COLMAP w/ Our Resectioning</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Failures</td>
<td>1 Failure</td>
<td>1 Failure</td>
<td>1 Failure</td>
<td>7 Successes</td>
</tr>
<tr>
<td>3 Successes</td>
<td>6 Successes</td>
<td>6 Successes</td>
<td>6 Successes</td>
<td>7 Successes</td>
</tr>
</tbody>
</table>
Successful reconstruction of Cereal (DuplicateStructures)
Successful reconstruction of ece_floor3_loop_cw (UIUCTag)
Successful reconstruction of Courthouse (TanksAndTemples)
Successful reconstruction of TempleOfHeaven (Internet)
Robust Global Translations with 1DSfM

Problem Statement
Incremental SFM is expensive and error-prone. We explore global methods to solve the problem in one shot.

Goal:
Build a 3D model in one shot given many two-view models. We use Chatterjee and Govindu [1] to solve for rotations, and focus only on translations.

Challenges:
- Many formulations of the translations problem are non-convex. A solver must find a good solution reliably.
- Translations problems generally contain outliers. Those bad measurements can reduce solution quality and make it harder for solvers to converge.

Contributions:
- **1DSfM**: A simple way to detect outlier translation measurements using 1D subproblems.


Takeaway:
We pose a translations problem as a standard nonlinear optimization, which, coupled with outlier removal yields good results even when initialized randomly.

---

**Contribution 1: Outlier Removal with 1DSfM**

**Left:** An example translations problem
**Right:** Solution space

An outlier edge is shown in red. Given the output embedding, we can tell it is an outlier. But how can we detect it efficiently?

**Input:**
- Two projection directions

**Output:**
- Absolute canonical positions

**1D subproblems:**
- Easier: we project the problem onto a single unit vector, so each edge becomes a simple plus-minus sign (due to the unknown scale of each edge) which we can represent in a directed graph.

Two projection directions

1D subproblem (a)

1D subproblem (b)

---

**Contribution 2: New Translations Solver**

We want to solve problems of this general form.

**Given:**
- A directed graph $G = (V,E)$
- 3D translation directions $f : E \rightarrow \mathbb{R}^3$

**Compute:**
- An embedding $X : V \rightarrow \mathbb{R}^3$ (up to scale and translation)

**Such that:**
- The translation directions induced by $X$ are close to $f$.

We compute poses in the measurement space of unit vectors with the squared chordal distance.

$\hat{X} = \arg\min_X \sum_{e=(u,v)} d_{ch} \left( t_{e}, (X_{e} - X_{v}) \right)^2$

$d_{ch}(u,v) = \|u - v\|

**Properties:**
- Nonlinear Least Squares problem (NLLS)—we use Ceres [3]
- Well-behaved error surface, especially after 1DSfM
- Can additionally use a Huber loss for even greater robustness
- Geometrically meaningful: MLE of the error model below

$f(t_{e}) \propto \exp(-\frac{d_{ch}^2}{\sigma^2})$

**Convergence:**
- NLL is a local optimizer—global convergence not guaranteed

**Surprisingly, we find good solutions, even from random initializations.

**Plausibility:** For a noise-free problem, the error surface is decreasing towards the global optimum. It deviates from this behavior slowly as noise increases:

$d_{ch}(u,v) \leq d_{ch}(u,X_{v}) + d_{ch}(X_{v}, X_{e})$ where $X_{v} = \lambda X_{e} + (1 - \lambda) X_{e}^{opt}$, $0 \leq \lambda \leq 1$

---

**Results**
- 13 large datasets—all new (except Notre Dame, from [7])
- State-of-the-art results
- Datasets and code available

We evaluate our results by robustly rigidly aligning solutions to models produced by BundleNet in incremental SFM solver [5].

The numbers below are errors in meters after a final bundle adjustment:

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>X</th>
<th>1DSfM</th>
<th>Y</th>
<th>1DSfM</th>
<th>Z</th>
<th>1DSfM</th>
<th>Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>Piccadilly</td>
<td>252</td>
<td>0.3</td>
<td>0.3</td>
<td>0.7</td>
<td>0.7</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Union Square</td>
<td>789</td>
<td>0.3</td>
<td>0.3</td>
<td>1.6</td>
<td>1.6</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Roman Forum</td>
<td>1084</td>
<td>2.7</td>
<td>5.0</td>
<td>3.0</td>
<td>3.0</td>
<td>27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vienna Cathedral</td>
<td>836</td>
<td>0.7</td>
<td>0.7</td>
<td>0.4</td>
<td>0.4</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Piazza del Popolo</td>
<td>528</td>
<td>1.0</td>
<td>1.0</td>
<td>2.2</td>
<td>2.2</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NYC Library</td>
<td>322</td>
<td>0.2</td>
<td>0.2</td>
<td>0.1</td>
<td>0.1</td>
<td>1.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alamo</td>
<td>777</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>2.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Metropolis</td>
<td>341</td>
<td>0.6</td>
<td>0.6</td>
<td>0.6</td>
<td>0.6</td>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yorkminster</td>
<td>437</td>
<td>0.3</td>
<td>0.3</td>
<td>0.1</td>
<td>0.1</td>
<td>6.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Montreal N.D.</td>
<td>650</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>9.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tower of London</td>
<td>572</td>
<td>0.2</td>
<td>0.2</td>
<td>1.0</td>
<td>1.0</td>
<td>44</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ellis Island</td>
<td>277</td>
<td>0.5</td>
<td>0.5</td>
<td>3.0</td>
<td>3.0</td>
<td>8.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Notre Dame</td>
<td>553</td>
<td>0.8</td>
<td>0.8</td>
<td>1.9</td>
<td>1.9</td>
<td>2.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Dataset sizes are given in both meters and number of cameras. The table shows median and mean camera error.

We significantly outperform an existing method [4]. 1DSfM often results in a smaller median error, but a greatly improved average. Times are 3-12x faster than [5].

**References**

Problem Statement

Incremental SfM is expensive and error-prone. We explore global methods to solve the problem in one shot.

Goal:
Build a 3D model in one shot given many two-view models. We use Chatterjee and Govindu [1] to solve for rotations, and focus only on translations.

Challenges:

- Many formulations of the translations problem are non-convex. A solver must find a good solution reliably.
- Translations problems generally contain outliers. These bad measurements can reduce solution quality and make it harder for solvers to converge.

Contributions:

1DSfM: a simple way to detect outlier translation measurements using 1D subproblems

Solver: a new approach to solving translations problems using nonlinear optimization
Contribution 1: Outlier Removal with 1DSfM

Left: an example translations problem
Right: the correct solution

An outlier edge is shown in red. Given the output embedding, we can tell it is an outlier. But how can we detect it upfront?

1D subproblems are easier: we project the problem onto a single unit vector, so each edge becomes a simple plus/minus sign (due to the unknown scale of each edge) which we can represent as a directed graph.

These 1D problems are instances of Minimum Feedback Arc Set [2]. Solving them means choosing a best ordering. Outlier edges may not be consistent with the others.

Outliers won’t be detected in some projections. We project in many random directions and reject edges that are frequently inconsistent.
Contribution 2: New Translations Solver

We want to solve problems of this general form:

| Given:      | a directed graph $G = (V, E)$ |
|            | 3D translation directions $t : E \to S^2$ |
| Compute:   | an embedding $X : V \to \mathbb{R}^3$ |
|            | (up to scale and translation) |
| Such that: | the translation directions induced by $X$ |
|            | are close to $t$ |

We compare poses in the **measurement space** of unit vectors with the squared chordal distance.

\[
\hat{X} = \arg\min_X \sum_{(i,j) \in E} d_{ch} \left( t_{ij}, \frac{X_j - X_i}{\|X_j - X_i\|} \right)^2 \\
\]

\[
d_{ch}(u, v) = \|u - v\|
\]

Properties:

- Nonlinear Least Squares problem (**NLLS**)—we use Ceres [3]
- Well-behaved error surface, especially after 1DSfM
- Can additionally use a Huber loss for even greater robustness
Results

- 13 large datasets—all new (except Notre Dame, from [5])
- state of the art results
- datasets and code available

We evaluate our results by robustly rigidly aligning solutions to models produced by Bundler, in incremental SfM solver [5].

The numbers below are errors in meters after a final bundle adjustment.

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>$N_c$</th>
<th>$\tilde{x}$</th>
<th>$\bar{x}$</th>
<th>$\tilde{x}$</th>
<th>$\bar{x}$</th>
<th>$\tilde{x}$</th>
<th>$\bar{x}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Piccadilly</td>
<td>80</td>
<td>2152</td>
<td>0.3</td>
<td>9e3</td>
<td>0.7</td>
<td>7e2</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Union Square</td>
<td>300</td>
<td>789</td>
<td>3.2</td>
<td>2e2</td>
<td>3.4</td>
<td>9e1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Roman Forum</td>
<td>200</td>
<td>1084</td>
<td>2.7</td>
<td>9e5</td>
<td>0.2</td>
<td>3e0</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>Vienna Cathedral</td>
<td>120</td>
<td>836</td>
<td>0.7</td>
<td>7e4</td>
<td>0.4</td>
<td>2e4</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>Piazza del Popolo</td>
<td>60</td>
<td>328</td>
<td>1.6</td>
<td>9e1</td>
<td>2.2</td>
<td>2e2</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>NYC Library</td>
<td>130</td>
<td>332</td>
<td>0.2</td>
<td>8e1</td>
<td>0.4</td>
<td>1e0</td>
<td>1.4</td>
<td></td>
</tr>
<tr>
<td>Alamo</td>
<td>70</td>
<td>577</td>
<td>0.2</td>
<td>7e5</td>
<td>0.3</td>
<td>2e7</td>
<td>2.4</td>
<td></td>
</tr>
<tr>
<td>Metropolis</td>
<td>200</td>
<td>341</td>
<td>0.6</td>
<td>3e1</td>
<td>0.5</td>
<td>7e1</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>Yorkminster</td>
<td>150</td>
<td>437</td>
<td>0.4</td>
<td>9e3</td>
<td>0.1</td>
<td>5e2</td>
<td>6.7</td>
<td></td>
</tr>
<tr>
<td>Montreal N.D.</td>
<td>30</td>
<td>450</td>
<td>0.1</td>
<td>4e-1</td>
<td>0.4</td>
<td>1e0</td>
<td>9.8</td>
<td></td>
</tr>
<tr>
<td>Tower of London</td>
<td>300</td>
<td>572</td>
<td>0.2</td>
<td>3e4</td>
<td>1.0</td>
<td>4e1</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>Ellis Island</td>
<td>180</td>
<td>227</td>
<td>0.3</td>
<td>3e0</td>
<td>0.3</td>
<td>3e0</td>
<td>8.0</td>
<td></td>
</tr>
<tr>
<td>Notre Dame</td>
<td>300</td>
<td>553</td>
<td>0.8</td>
<td>7e4</td>
<td>1.9</td>
<td>7e0</td>
<td>2.1</td>
<td></td>
</tr>
</tbody>
</table>

Dataset sizes are given in both meters and number of cameras. The table shows median and mean camera error.
Incremental vs. Global SfM

• Incremental includes more outlier checks and generates more precise results but take much longer

• Global is much faster but does not as effectively remove outliers and provides an approximate solution that is not precise enough (in my experience) for MVS
Open problems / research ideas

• Improved matching
  – Learned features, especially for handling large viewpoint, scale, or time differences, or features for low-texture regions

• Improved outlier rejection
  – Perhaps global SfM outlier checks can benefit incremental SfM

• Improved speed
  – Hybrid global/incremental and hierarchical systems
  – Online SfM / MVS

• Improved standard evaluations
  – More real-world scenarios like inspection instead of internet collections
Summary

• Structure-from-Motion usually works (95% of the time)
  – But it matters when it doesn’t work

• Incremental SfM is most precise, but Global SfM is faster

• Main practical challenges (beyond speed) stem from feature matching in poor light environments, textureless surfaces, and large baselines and scale differences