Recap: A General Point

- Equations of the form
  \[ Ax = 0 \]
  - How do we solve them? (always!)
    - Apply SVD
      \[ A = U \Sigma V^T \]
      - Singular values \( \Sigma \)
      - Singular vectors \( U, V \)
        - Singular values of A = square roots of the eigenvalues of \( A^T A \)
        - The solution of \( Ax = 0 \) is the nullspace vector of A.
        - This corresponds to the smallest singular vector of A.

Recap: Camera Parameters

- Intrinsic parameters
  - Principal point coordinates
  - Focal length
  - Pixel magnification factors
  - Skew (non-rectangular pixels)
  - Radial distortion
- Extrinsic parameters
  - Rotation \( R \)
  - Translation \( t \)
    (both relative to world coordinate system)
- Camera projection matrix
  \[ P = K [R \mid t] \]
  - General pinhole camera: 9 DoF
  - CCD Camera with square pixels: 10 DoF
  - General camera: 11 DoF

Course Outline

- Image Processing Basics
- Segmentation & Grouping
- Object Recognition
- Local Features & Matching
- Object Categorization
- 3D Reconstruction
  - Epipolar Geometry and Stereo Basics
  - Camera calibration & Uncalibrated Reconstruction
  - Structure-from-Motion
- Motion and Tracking

Properties of SVD

- Frobenius norm
  - Generalization of the Euclidean norm to matrices
  \[ \| A \|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2} = \sqrt{\sum_{i=1}^m \sigma_i^2} \]
- Partial reconstruction property of SVD
  - Let \( \sigma_i = 1, \ldots, N \) be the singular values of A.
  - Let \( A_p = U_p \Sigma_p V_p^T \) be the reconstruction of A when we set \( \sigma_{p+1}, \ldots, \sigma_N \) to zero.
  - Then \( A_p = U_p \Sigma_p V_p^T \) is the best rank-p approximation of A in the sense of the Frobenius norm
    (i.e., the best least-squares approximation).

Recap: Calibrating a Camera

Goal

- Compute intrinsic and extrinsic parameters using observed camera data.

Main idea

- Place “calibration object” with known geometry in the scene
- Get correspondences
- Solve for mapping from scene to image: estimate \( P = P_{\text{ext}} P_{\text{int}} \)
Recap: Camera Calibration (DLT Algorithm)

\[
\begin{bmatrix}
0' & X_1' - y_1 X_1' \\
X_1' & 0' - x_1 X_1' \\
\vdots & \vdots \\
0' & X_n' - y_n X_n' \\
X_n' & 0' - x_n X_n'
\end{bmatrix} \begin{bmatrix}
P_1 \\
P_2 \end{bmatrix} = 0 \quad \text{Ap} = 0
\]

- P has 11 degrees of freedom.
- Two linearly independent equations per independent 2D/3D correspondence.
- Solve with SVD (similar to homography estimation)
- Solution corresponds to smallest singular vector.
- 5 ½ correspondences needed for a minimal solution.

Recap: Triangulation - Linear Algebraic Approach

\[
\begin{align*}
\lambda_1 x_1 &= P_1 x \\
\lambda_2 x_2 &= P_2 x
\end{align*}
\]

\[
\begin{align*}
x_1 \times P_1 x &= 0 \\
x_2 \times P_2 x &= 0
\end{align*}
\]

- Two independent equations each in terms of three unknown entries of X.
- Stack equations and solve with SVD.
- This approach nicely generalizes to multiple cameras.

Recap: Epipolar Geometry - Calibrated Case

Camera matrix: \([I|0]\)

\[
X = (u, v, w, 1)^T
\]

\[
x = (u, v)^T
\]

The vectors \(x, x'\), and \(Rx'\) are coplanar

Essential Matrix (Longuet-Higgins, 1981)

Recap: Epipolar Geometry - Uncalibrated Case

- The calibration matrices \(K\) and \(K'\) of the two cameras are unknown
- We can write the epipolar constraint in terms of unknown normalized coordinates:

\[
\hat{x}'^T E \hat{x}' = 0 \\
x = K \hat{x}, \quad x' = K' \hat{x}'
\]

Fundamental Matrix (Faugeras and Luong, 1992)
Recap: The Eight-Point Algorithm

\[ x = (u, v, 1)^T, \quad x' = (u', v', 1)^T \]

\[
\begin{bmatrix}
F_{11} & F_{12} & F_{13} \\
F_{21} & F_{22} & F_{23} \\
F_{31} & F_{32} & F_{33}
\end{bmatrix}
\begin{bmatrix}
u' \\
v' \\
1
\end{bmatrix} = 0
\]

\[
\begin{bmatrix}
u' u, u', v, u, v, u, v, 1
\end{bmatrix}
\begin{bmatrix}
F_{11} & F_{12} & F_{13} \\
F_{21} & F_{22} & F_{23} \\
F_{31} & F_{32} & F_{33}
\end{bmatrix}
\begin{bmatrix}
u' \\
v' \\
1
\end{bmatrix} = 0
\]

This minimizes:

\[ \sum_{i=1}^{N} (x_i^T F x_i')^2 \]

Solve using... SVD

Recap: Normalized Eight-Point Algorithm

1. Center the image data at the origin, and scale it so the mean squared distance between the origin and the data points is 2 pixels.
2. Use the eight-point algorithm to compute \( F \) from the normalized points.
3. Enforce the rank-2 constraint using SVD.

\[ F = U D V^T = U \begin{bmatrix} d_{11} & & & \\ & d_{22} & & \\ & & \ddots & \\ & & & d_{N,N} \end{bmatrix} V^T \]

4. Transform fundamental matrix back to original units: if \( T \) and \( T' \) are the normalizing transformations in the two images, then the fundamental matrix in original coordinates is \( T^T F T' \).

Recap: Comparison of Estimation Algorithms

<table>
<thead>
<tr>
<th>Method</th>
<th>Error (avg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eight point</td>
<td>2.0 pixels</td>
</tr>
<tr>
<td>Normalized eight</td>
<td>2.33 pixels</td>
</tr>
<tr>
<td>Nonlinear least squares</td>
<td>3.00 pixel</td>
</tr>
<tr>
<td>Eight point</td>
<td>2.16 pixels</td>
</tr>
<tr>
<td>Normalized eight</td>
<td>0.85 pixel</td>
</tr>
<tr>
<td>Nonlinear least squares</td>
<td>0.80 pixel</td>
</tr>
</tbody>
</table>

Recap: Active Stereo with Structured Light

- Optical triangulation
  - Project a single stripe of laser light
  - Scan it across the surface of the object
  - This is a very precise version of structured light scanning

Recap: Epipolar Transfer

- Assume the epipolar geometry is known
- Given projections of the same point in two images, how can we compute the projection of that point in a third image?

\[ x_1 \quad x_2 \quad x_3 \]

\[ l_{ij} = F_{ij} x_i \]

Topics of This Lecture

- Structure from Motion (SfM)
  - Motivation
  - Ambiguity
- Affine SfM
  - Affine cameras
  - Affine factorization
  - Euclidean upgrade
  - Dealing with missing data
- Projective SfM
  - Two-camera case
  - Projective factorization
  - Bundle adjustment
  - Practical considerations
- Applications
Structure from Motion

- Given: $m$ images of $n$ fixed 3D points
  
  $$x_{ij} = P_i X_j, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n$$

- Problem: estimate $m$ projection matrices $P_i$ and $n$ 3D points $X_j$ from the $mn$ correspondences $x_{ij}$.

Structure from Motion Ambiguity

- If we scale the entire scene by some factor $k$ and, at the same time, scale the camera matrices by the factor of $1/k$, the projections of the scene points in the image remain exactly the same:

  $$x = PX = \left(\frac{1}{k}P\right)(kX)$$

  ⇒ It is impossible to recover the absolute scale of the scene!

Reconstruction Ambiguity: Similarity

$$X = \left(\frac{1}{k}S\right)QX$$

Reconstruction Ambiguity: Affine

$$X = \left(\frac{1}{k}A\right)QX$$

What Can We Use This For?

- E.g. movie special effects

Structure from Motion Ambiguity

- If we scale the entire scene by some factor $k$ and, at the same time, scale the camera matrices by the factor of $1/k$, the projections of the scene points in the image remain exactly the same.

- More generally: if we transform the scene using a transformation $Q$ and apply the inverse transformation to the camera matrices, then the images do not change:

  $$X = \left(\frac{1}{k}S\right)QX$$

Perceptual and Sensory Augmented Computing

Topics of This Lecture
- Structure from Holons (SfH)
  - Motivation
  - Ambiguity
- Affine SfM
  - Affine cameras
  - Affine factorization
  - Euclidean upgrade
  - Dealing with missing data
- Projective SfM
  - Two-camera case
  - Projective factorization
  - Bundle adjustment
  - Practical considerations
- Applications

Similarity

From Affine to Similarity

From Projective to Affine

Hierarchy of 3D Transformations
- Projective 15dof
- Affine 12dof
- Similarity 7dof
- Euclidean 6dof

- 7dof: Preserves angles, ratios of length
- 6dof: Preserves angles, lengths
- 12dof: Preserves parallelism, volume ratios
- 15dof: Preserves intersection and tangency

- With no constraints on the camera calibration matrix or on the scene, we get a projective reconstruction.
- Need additional information to upgrade the reconstruction to affine, similarity, or Euclidean.
**Affine Structure from Motion**

- Given: $m$ images of $n$ fixed 3D points:
  - $x_i = A_i X_j + b_i$, $i = 1, \ldots, m$, $j = 1, \ldots, n$
- Problem: use the $mn$ correspondences $x_{ij}$ to estimate $m$ projection matrices $A_i$ and translation vectors $b_i$, and $n$ points $X_j$
- The reconstruction is defined up to an arbitrary affine transformation $Q$ (12 degrees of freedom):
  \[
  \begin{pmatrix}
  A \\
  b
  \end{pmatrix} \rightarrow \begin{pmatrix}
  A & b \\
  0 & 1
  \end{pmatrix} Q^{-1},
  \begin{pmatrix}
  X \\
  1
  \end{pmatrix} \rightarrow Q \begin{pmatrix}
  X \\
  1
  \end{pmatrix}
  \]
- We have $2mn$ knowns and $8m + 3n$ unknowns (minus 12 dof for affine ambiguity).
  - Thus, we must have $2mn > 8m + 3n - 12$.
  - For two views, we need four point correspondences.

**Affine Cameras**

- A general affine camera combines the effects of an affine transformation of the 3D space, orthographic projection, and an affine transformation of the image:
  \[
P = [3 \times 3 \text{affine}] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ b \end{bmatrix}
\]
- Affine projection is a linear mapping + translation in inhomogeneous coordinates
  \[
x = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = AX + b
\]

**Orthographic Projection**

- Special case of perspective projection
  - Distance from center of projection to image plane is infinite
  
  \[
  \begin{bmatrix}
  1 & 0 & 0 & 0 \\
  0 & 0 & 1 & 0 \\
  0 & 0 & 0 & 1
  \end{bmatrix}
  \begin{bmatrix}
  x \\
  y
  \end{bmatrix}
  \mapsto
  \begin{bmatrix}
  x/1 \\
  y/1
  \end{bmatrix} = \begin{bmatrix}
  x \\
  y
  \end{bmatrix}
  \]

**Affine Structure from Motion**

- Centering: subtract the centroid of the image points
  \[
  \hat{x}_i = x_i - \frac{1}{m} \sum_{j=1}^{m} x_j = A_i X_j + b_i - \frac{1}{m} \sum_{j=1}^{m} (A_i X_j + b_i)
  = A_i \left( X_j - \frac{1}{m} \sum_{j=1}^{m} X_j \right) = A_i \tilde{X}_j
  \]
- For simplicity, assume that the origin of the world coordinate system is at the centroid of the 3D points.
- After centering, each normalized point $x_{ij}$ is related to the 3D point $X_j$ by
  \[
  \hat{x}_{ij} = A_j X_j
  \]
Affine Structure from Motion

- Let's create a $2m \times n$ data (measurement) matrix:

$$D = \begin{bmatrix} \hat{x}_{11} & \hat{x}_{12} & \cdots & \hat{x}_{1n} \\ \hat{x}_{21} & \hat{x}_{22} & \cdots & \hat{x}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{x}_{m1} & \hat{x}_{m2} & \cdots & \hat{x}_{mn} \end{bmatrix}$$

Cameras (2m)

Points (n)


Slide credit: Svetlana Lazebnik

Factorizing the Measurement Matrix

- Singular value decomposition of $D$:

$$D = U \Sigma V^T$$

$U$ is $2m \times 3$

$\Sigma$ is $3 \times n$

$V^T$ is $n \times 3$

Obtaining a factorization from SVD:

$$D = U_3 \Sigma_3 V_3^T$$
Factorizing the Measurement Matrix

- Obtaining a factorization from SVD:
  \[ D = U_L \times \begin{pmatrix} W_1 & W_2 \end{pmatrix} \times V_I \]

Possible decomposition:
\[ M = U_L W_1 \times V_I \]

This decomposition minimizes \(|D-MS|^2\).

Affine Ambiguity

- The decomposition is not unique. We get the same \( D \) by using any \( 3 \times 3 \) matrix \( C \) and applying the transformations \( M \rightarrow MC, S \rightarrow C^{-1}S \).
- That is because we have only an affine transformation and we have not enforced any Euclidean constraints (like forcing the image axes to be perpendicular, for example). We need a Euclidean upgrade.

Estimating the Euclidean Upgrade

- Orthographic assumption: image axes are perpendicular and scale is 1.
  \[ a_1 \cdot a_2 = 0 \]
  \[ |a_1|^2 = |a_2|^2 = 1 \]

- This can be converted into a system of \( 3m \) equations:
  \[
  \begin{align*}
  \hat{a}_1 \cdot \hat{a}_3 &= 0 \\
  \hat{a}_2 \cdot \hat{a}_3 &= 0 \\
  |\hat{a}_1|^2 &= 1 \\
  |\hat{a}_2|^2 &= 1 \\
  \end{align*}
  \]

- Let \( L = CC^T \)
  \[ A = \begin{bmatrix} \hat{a}_1^T \\ \hat{a}_2^T \end{bmatrix}, \quad i = 1, \ldots, m \]

Then this translates to \( 3m \) equations in \( L \)
  \[ A_L A_L^T = I, \quad i = 1, \ldots, m \]

- Solve for \( L \)
- Recover \( C \) from \( L \) by Cholesky decomposition: \( L = CC^T \)
- Update \( M \) and \( S \): \( M = MC, S = C^{-1}S \)

Algorithm Summary

- Given: \( m \) images and \( n \) features \( x_{ij} \)
- For each image \( i \), center the feature coordinates.
- Construct a \( 2m \times n \) measurement matrix \( D \):
  - Column \( j \) contains the projection of point \( j \) in all views
  - Row \( i \) contains one coordinate of the projections of all the \( n \) points in image \( i \)
- Factorize \( D \):
  - Compute SVD: \( D = U W V^T \)
  - Create \( U_L \) by taking the first 3 columns of \( U \)
  - Create \( W_L \) by taking the first 3 columns of \( W \)
  - Create \( W_L \) by taking the upper left 3 \( \times \) 3 block of \( W \)
  - Create the motion and shape matrices:
    - \( M = U_L W_L \) and \( S = W_L V_I \) (or \( M = U_L \) and \( S = W_L V_I \))
  - Eliminate affine ambiguity

Reconstruction Results
Dealing with Missing Data

- So far, we have assumed that all points are visible in all views.
- In reality, the measurement matrix typically looks something like this:

```
<table>
<thead>
<tr>
<th>Cameras</th>
<th>Points</th>
</tr>
</thead>
</table>
```


Slide credit: Svetlana Lazebnik

Dealing with Missing Data

- Possible solution: decompose matrix into dense sub-blocks, factorize each sub-block, and fuse the results.
- Finding dense maximal sub-blocks of the matrix is NP-complete (equivalent to finding maximal cliques in a graph).
- Incremental bilinear refinement

```
(1) Perform factorization on a dense sub-block
(2) Solve for a new 3D point visible by at least two known cameras (linear least squares)
(3) Solve for a new camera that sees at least three known 3D points (linear least squares)
```


Slide credit: Svetlana Lazebnik

Comments: Affine SfM

- Affine SfM was historically developed first.
- It is valid under the assumption of affine cameras.
  - Which does not hold for real physical cameras...
  - ...but which is still tolerable if the scene points are far away from the camera.
- For good results with real cameras, we typically need projective SfM.
  - Harder problem, more ambiguity
  - Math is a bit more involved...
  - (Here, only basic ideas. If you want to implement it, please look at the H&Z book for details).

B. Leibe

Topics of This Lecture

- Structure from Holani (SfH)
  - Motivation
  - Ambiguity
- Affine SfM
  - Affine cameras
  - Affine factorization
  - Bundle upgrade
  - Dealing with missing data
- Projective SfM
  - Two-camera case
  - Projective factorization
  - Bundle adjustment
  - Practical considerations
- Applications

B. Leibe

Slide credit: Svetlana Lazebnik
**Projective Structure from Motion**

- Given: $m$ images of $n$ fixed 3D points
  \[ x_j = P_j X_j, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n \]
- Problem: estimate $m$ projection matrices $P_j$ and $n$ 3D points $X_j$ from the $mn$ correspondences $x_j$

**Problem:** estimate $P_j$ and $n$ 3D points $X_j$ from the $mn$ correspondences $x_j$

- With no calibration info, cameras and points can only be recovered up to a $4 \times 4$ projective transformation $Q$:
  \[ X \rightarrow QX, \quad P \rightarrow PQ^{-1} \]
- We can solve for structure and motion when $2mn \gg 11m + 3n - 15$
- For two cameras, at least 7 points are needed.

**Sequential Structure from Motion**

- Initialize motion from two images using fundamental matrix
- Initialize structure
- For each additional view:
  - Determine projection matrix of new camera using all the known 3D points that are visible in its image - calibration

**Projective SfM: Two-Camera Case**

- Assume fundamental matrix $F$ between the two views
  - First camera matrix: $[I|0]Q^T$
  - Second camera matrix: $[Ab|0]Q$
- Let $X = QX_j$, then
  \[ z = [I|0]X, \quad z' = [Ab]X \]
- And
  \[ z' = A[I|0]X + b = A[X] + b \]
  \[ z' = Ax + b \]
- So we have
  \[ F = [b]A \]
  \[ b: \text{epipole} (F^Tb = 0), \quad A = [b]F \]

**Projective Factorization**

\[
D = \begin{bmatrix}
    z_{11} X_{11} & z_{12} X_{12} & \cdots & z_{1n} X_{1n} \\
    z_{21} X_{21} & z_{22} X_{22} & \cdots & z_{2n} X_{2n} \\
    \vdots & \vdots & \ddots & \vdots \\
    z_{m1} X_{m1} & z_{m2} X_{m2} & \cdots & z_{mn} X_{mn}
\end{bmatrix} = \begin{bmatrix} P_1 & \ldots & P_m \end{bmatrix}
\begin{bmatrix} X_1 & X_2 & \cdots & X_n \end{bmatrix}
\]

$D = MS$ has rank 4

- If we knew the depths $z$, we could factorize $D$ to estimate $M$ and $S$.
- If we knew $M$ and $S$, we could solve for $z$.
- Solution: iterative approach (alternate between above two steps).
Self-Calibration

It can become an extremely large minimization problem.

Projective Ambiguity

• If we don’t know anything about the camera or the scene, the best we can get with this is a reconstruction up to a projective ambiguity Q.
  - This can already be useful.
  - E.g., we can answer questions like "at what point does a line intersect a plane?"
• If we want to convert this to a “true” reconstruction, we need a Euclidean upgrade.
  - Need to put in additional knowledge about the camera (calibration) or about the scene (e.g., from markers).
  - Several methods available (see F&P Chapter 13.5 or H&BZ Chapter 19)

Sequential Structure from Motion

• Initialize motion from two images using fundamental matrix
• Initialize structure
• For each additional view:
  - Determine projection matrix of new camera using all the known 3D points that are visible in its image - calibration
  - Refine and extend structure: compute new 3D points, re-optimize existing points that are also seen by this camera - triangulation

Bundle Adjustment

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

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

Self-Calibration

• Self-calibration (auto-calibration) is the process of determining intrinsic camera parameters directly from uncalibrated images.
• For example, when the images are acquired by a single moving camera, we can use the constraint that the intrinsic parameter matrix remains fixed for all the images.
  - Compute initial projective reconstruction and find 3D projective transformation matrix Q such that all camera matrices are in the form $P_i = K_i [R_i | t_i]$. 
  - Can use constraints on the form of the calibration matrix: square pixels, zero skew, fixed focal length, etc.
Practical Considerations (1)

1. Role of the baseline
   - Small baseline: large depth error
   - Large baseline: difficult search problem

Solution
   - Track features between frames until baseline is sufficient.

Practical Considerations (2)

2. There will still be many outliers
   - Incorrect feature matches
   - Moving objects
   ⇒ Apply RANSAC to get robust estimates based on the inlier points.

3. Estimation quality depends on the point configuration
   - Points that are close together in the image produce less stable solutions.
   ⇒ Subdivide image into a grid and try to extract about the same number of features per grid cell.

General Guidelines

- Use calibrated cameras wherever possible.
  - It makes life so much easier, especially for SfM.
- SfM with 2 cameras is far more robust than with a single camera.
  - Triangulate feature points in 3D using stereo.
  - Perform 2D-3D matching to recover the motion.
  - More robust to loss of scale (main problem of 1-camera SfM).
- Any constraint on the setup can be useful
  - E.g. square pixels, zero skew, fixed focal length in each camera
  - E.g. fixed baseline in stereo SfM setup
  - E.g. constrained camera motion on a ground plane
  - Making best use of those constraints may require adapting the algorithms (some known results are described in H&Z).

Structure-from-Motion: Limitations

- Very difficult to reliably estimate metric SfM unless
  - Large (x or y) motion or
  - Large field-of-view and depth variation
- Camera calibration important for Euclidean reconstruction
- Need good feature tracker

Topics of This Lecture

- Structure from Motion (SfM)
  - Extration
  - Ambiguity
- Affine SfM
  - Affine cameras
  - Affine factorization
  - Bundle adjustment
  - Dealing with missing data
- Projective SfM
  - Two-camera case
  - Projective factorization
  - Bundle adjustment
  - Practical considerations
- Applications

Commercial Software Packages

- boujou (http://www.2d3.com/)
- PFTTrack (http://www.thepixelfarm.co.uk/)
- MatchMover (http://www.realviz.com/)
- SynthEyes (http://www.ssontech.com/)
- Icarus (http://aig.cs.man.ac.uk/research/reveal/icarus/)
- Voodoo Camera Tracker (http://www.digilab.uni-hannover.de/)
boujou demo

(We have a license available, so if you want to try it for interesting projects, contact us.)

Applications: Matchmoving

- Putting virtual objects into real-world videos

<table>
<thead>
<tr>
<th>Original sequence</th>
<th>SfM results</th>
<th>Final video</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracked features</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Applications: 3D City Modeling

- Putting virtual objects into real-world videos

<table>
<thead>
<tr>
<th>Original sequence</th>
<th>SfM results</th>
<th>Final video</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracked features</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Applications: Large-Scale SfM from Flickr

- Putting virtual objects into real-world videos

<table>
<thead>
<tr>
<th>Original sequence</th>
<th>SfM results</th>
<th>Final video</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracked features</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Another Example: The Campanile Movie

- Putting virtual objects into real-world videos

<table>
<thead>
<tr>
<th>Original sequence</th>
<th>SfM results</th>
<th>Final video</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracked features</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

References and Further Reading

- A (relatively short) treatment of affine and projective SfM and the basic ideas and algorithms can be found in Chapters 12 and 13 of


- More detailed information (if you really want to implement this) and better explanations can be found in Chapters 10, 18 (factorization) and 19 (self-calibration) of

  R. Hartley, A. Zisserman Multiple View Geometry in Computer Vision 2nd Ed., Cambridge Univ. Press, 2004