Structure from motion

The SFM Problem
- Reconstruct scene geometry and camera motion from two or more images

**SFM Pipeline**
- Track 2D Features
- Estimate 3D
- Optimize (Bundle Adjustment)
- Fit Surfaces

---

**Structure from motion**

**Step 1: Track Features**
- Detect good features
  - corners, line segments
- Find correspondences between frames
  - Lucas & Kanade-style motion estimation
  - window-based correlation

**Structure from motion**

**Step 2: Estimate Motion and Structure**
- Simplified projection model, e.g., [Tomasi 92]
- 2 or 3 views at a time [Hartley 00]
**Structure from motion**

**Step 3: Refine Estimates**
- "Bundle adjustment" in photogrammetry

**Step 4: Recover Surfaces**
- Image-based triangulation [Morris 00, Baillard 99]
- Silhouettes [Fitzgibbon 98]
- Stereo [Pollefeys 99]

---

**Feature tracking**

**Problem**
- Find correspondence between $n$ features in $f$ images

**Issues**
- What’s a feature?
- What does it mean to “correspond”?
- How can correspondence be reliably computed?

---

**Feature detection**

What’s a good feature?
Good features to track

Recall Lucas-Kanade equation:
\[
\begin{bmatrix}
\sum I_x I_x & \sum I_x I_y \\
\sum I_x I_y & \sum I_y I_y
\end{bmatrix}
\begin{bmatrix}
u \\
v
\end{bmatrix} = -\begin{bmatrix}
\sum I_x I_t \\
\sum I_y I_t
\end{bmatrix}
\]

When is this solvable?
- \( A^T A \) should be invertible
- \( A^T A \) should not be too small due to noise
- \( \lambda_1 \) and \( \lambda_2 \) of \( A^T A \) should not be too small
- \( A^T A \) should be well-conditioned
- \( \lambda_1/\lambda_2 \) should not be too large \( \lambda_1 \) = larger eigenvalue

These conditions are satisfied when \( \min(\lambda_1, \lambda_2) > c \)

Feature correspondence

Correspondence Problem
- Given feature patch \( F \) in frame \( H \), find best match in frame \( I \)

Find displacement \((u, v)\) that minimizes SSD error over feature region
\[
\sum_{(x, y) \in F \cap J} [I(x + u, y + v) - H(x, y)]^2
\]

Solution
- Small displacement: Lucas-Kanade
\[
\begin{bmatrix}
\sum I_x I_x & \sum I_x I_y \\
\sum I_x I_y & \sum I_y I_y
\end{bmatrix}
\begin{bmatrix}
u \\
v
\end{bmatrix} = -\begin{bmatrix}
\sum I_x I_t \\
\sum I_y I_t
\end{bmatrix}
\]
- Large displacement: discrete search over \((u, v)\)
  - Choose match that minimizes SSD (or normalized correlation)

Feature distortion

Feature may change shape over time
- Need a distortion model to really make this work

Find displacement \((u, v)\) that minimizes SSD error over feature region
\[
\sum_{(x, y) \in F \cap J} [(W_x(x, y), W_y(x, y)) - J(x, y)]^2
\]

Minimize with respect to \( W_x \) and \( W_y \)
- Affine model is common choice [Shi & Tomasi 94]
  \[
  W_x(x, y) = ax + by + c \\
  W_y(x, y) = ex + fy + g
  \]

Tracking over many frames

So far we’ve only considered two frames

Basic extension to \( f \) frames
1. Select features in first frame
2. Given feature in frame \( i \), compute position/deformation in \( i+1 \)
3. Select more features if needed
4. \( i = i + 1 \)
5. If \( i < f \), go to step 2

Issues
- Discrete search vs. Lucas Kanade?
  - depends on expected magnitude of motion
  - discrete search is more flexible
- How often to update feature template?
  - update often enough to compensate for distortion
  - updating too often causes drift
- How big should search window be?
  - too small: lost features. Too large: slow
Incorporating dynamics

Idea

- Can get better performance if we know something about the way points move
- Most approaches assume constant velocity
  \[ x_{t+1} = x_t \]
  \[ x_{t+1} = 2x_t - x_{t-1} \]
  or constant acceleration
  \[ x_{t+1} = x_t \]
  \[ x_{t+1} = 3x_t - 3x_{t-1} + x_{t-2} \]
- Use above to predict position in next frame, initialize search

Modeling uncertainty

Kalman Filtering (http://www.cs.unc.edu/~welch/kalman/)

- Updates feature state and Gaussian uncertainty model
- Get better prediction, confidence estimate

CONDENSATION
(http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/ISARD1/condensation.html)

- Also known as “particle filtering”
- Updates probability distribution over all possible states
- Can cope with multiple hypotheses

Probabilistic Tracking

Treat tracking problem as a Markov process

- Estimate \( p(x_t | z_t, x_{t-1}) \) – prob of being in state \( x_t \) given measurement \( z_t \) and previous state \( x_{t-1} \)
- Combine Markov assumption with Bayes Rule
  \[ p(x_t|z_t, x_{t-1}) \propto p(z_t|x_t) p(x_t|x_{t-1}) \]

Approach

- Predict position at time \( t \): \( p(x_t|x_{t-1}) \)
- Measure (perform correlation search or Lukas-Kanade) and compute likelihood \( p(z_t|x_t) \)
- Combine to obtain (unnormalized) state probability \( p(X_t|z_t, x_{t-1}) \)

Key

- \( s = x \) (position)
- \( o = z \) (sensor)

Robot figures courtesy of Dieter Fox
Modeling probabilities with samples

Allocate samples according to probability
- Higher probability—more samples

CONDENSATION [Isard & Blake]

Initialization: unknown position (uniform)

Prediction:
- draw new samples from the PDF
- use the motion model to move the samples
Monte Carlo robot localization

Particle Filters [Fox, Dellaert, Thrun and collaborators]

CONDENSATION Contour Tracking

Red: smooth drawing
Green: scribble
Blue: pause

Training a tracker

CONDENSATION Contour Tracking

The SFM Problem
- Reconstruct scene geometry and camera positions from two or more images

Assume
- Pixel correspondence
  - via tracking
- Projection model
  - classic methods are orthographic
  - newer methods use perspective
  - practically any model is possible with bundle adjustment
**SFM under orthographic projection**

\[ u = \Pi X + t \]

2×1  2×3  3×1  2×1

image point projection scene image

**Matrix**

**Scene**

**Image**

**Offset**

More generally: weak perspective, para-perspective, affine

Trick
- Choose scene origin to be centroid of 3D points
- Choose image origins to be centroid of 2D points
- Allows us to drop the camera translation:

\[ u = \Pi X \]

2×1  2×3  3×1

**Shape by factorization** [Tomasi & Kanade, 92]

**Projection of \( n \) features in one image:**

\[
\begin{bmatrix}
    u_1, u_2, \ldots, u_n
\end{bmatrix} = \Pi \begin{bmatrix}
    X_1, X_2, \ldots, X_n
\end{bmatrix}
\]

2×n  2×3  3×n

**Projection of \( n \) features in \( f \) images**

\[
\begin{bmatrix}
    u_1^t, u_2^t, \ldots, u_n^t
\end{bmatrix} = \Pi^t \begin{bmatrix}
    X_1, X_2, \ldots, X_n
\end{bmatrix}
\]

2f×n  2f×3

\( W \) measurement  \( M \) motion  \( S \) shape

Key Observation: \( \text{rank}(W) \leq 3 \)

**Shape by factorization** [Tomasi & Kanade, 92]

**Factorization Technique**
- \( W \) is at most rank 3 (assuming no noise)
- We can use singular value decomposition to factor \( W \):

\[
W = M' S' \]

2f×n  2f×3  3×n

**Singular value decomposition (SVD)**

SVD decomposes any \( m \times n \) matrix \( A \) as

\[
A = U \Sigma V^T
\]

Properties
- \( \Sigma \) is a diagonal matrix containing the eigenvalues of \( A^T A \)
- known as “singular values” of \( A \)
- diagonal entries are sorted from largest to smallest
- columns of \( U \) are eigenvectors of \( A A^T \)
- columns of \( V \) are eigenvectors of \( A^T A \)

If \( A \) is singular (e.g., has rank 3)
- only first 3 singular values are nonzero
- we can throw away all but first 3 columns of \( U \) and \( V \)

\[
A = U' \Sigma' V'^T
\]

3×n  3×3  3×n

- Choose \( M' = U' \), \( S' = \Sigma' \)
**Factorization Technique**

- \( W \) is at most rank 3 (assuming no noise)
- We can use singular value decomposition to factor \( W \):
  \[
  W = MS' \quad 2f \times n \quad 2f \times 3 \quad 3 \times n
  \]
- \( S' \) differs from \( S \) by a linear transformation \( A \):
  \[
  W = MS' = (MA^{-1})(AS)
  \]
- Solve for \( A \) by enforcing metric constraints on \( M \)

**Orthographic Camera**

- Rows of \( \Pi \) are orthonormal:
  \[
  \Pi \Pi^t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}
  \]

**Weak Perspective Camera**

- Rows of \( \Pi \) are orthogonal:
  \[
  \Pi \Pi^t = \begin{bmatrix} * & 0 \\ 0 & * \end{bmatrix}
  \]

**Enforcing “Metric” Constraints**

- Compute \( A \) such that rows of \( M \) have these properties
  \[
  M' = M
  \]
- Trick (not in original Tomasi/Kanade paper, but in followup work)
  - Constraints are linear in \( AA^t \):
    \[
    \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \Pi \Pi^t = \Pi' AA^t \Pi'^t = \Pi' G \Pi'^t \quad \text{where} \quad G = AA^t
    \]
  - Solve for \( G \) first by writing equations for every \( \Pi_i \) in \( M \)
  - Then \( G = AA^t \) by SVD (since \( U = V \))

**Factorization with noisy data**

- \( W = MS + E \quad 2f \times n \quad 2f \times 3 \quad 3 \times n \quad 2f \times n \)

Once again: use SVD of \( W \)
- Set all but the first three singular values to 0
- Yields new matrix \( W' \)
- \( W' \) is optimal rank 3 approximation of \( W \)
  \[
  W = W' + E \quad 2f \times n \quad 2f \times n
  \]

**Many extensions**

- Independently Moving Objects
- Perspective Projection
- Outlier Rejection
- Subspace Constraints
- SFM Without Correspondence
Extending factorization to perspective

Several Recent Approaches
- [Christy 96], [Triggs 96], [Han 00], [Mahamud 01]
- Initialize with ortho/weak perspective model then iterate

**Christy & Horaud**
- Derive expression for weak perspective as a perspective projection plus a correction term:
  \[ u_i = (1 + \epsilon_i) u_x \]
- where \[ \epsilon_i = \frac{k \cdot X}{t_z} \]
- and \([k \ t_z]\) is third row of projection matrix
- Basic procedure:
  - Run Tomasi-Kanade with weak perspective
  - Solve for \(\epsilon_i\) (different for each row of \(M\))
  - Add correction term to \(W\), solve again (until convergence)

**Bundle adjustment**
- 3D \(\rightarrow\) 2D mapping
  - a function of intrinsics \(K\), extrinsics \(R\) & \(t\)
  - measurement affected by noise
  \[ u_i = f(K, R, t, x_i) + \epsilon_i = \tilde{u}_i + v_i, \quad v_i \sim N(0, \sigma) \]
  \[ v_i = g(K, R, t, x_i) + m_i = \tilde{v}_i + m_i, \quad m_i \sim N(0, \sigma) \]
  - Log likelihood of \(K, R, t\) given \(\{(u_i, v_i)\}\)
    \[ C = -\log L = \sum (u_i - \tilde{u}_i)^2 / \sigma_i^2 + (v_i - \tilde{v}_i)^2 / \sigma_i^2 \]
  - Minimized via nonlinear least squares regression
  - called “Bundle Adjustment”
  - e.g., Levenberg-Marquardt
    - described in Press et al., Numerical Recipes

**Match Move**
- Film industry is a heavy consumer
  - composite live footage with 3D graphics
  - known as “match move”

Commercial products
- 2D3
  - [http://www.2d3.com/](http://www.2d3.com/)
- RealVis

Show video

**Closing the loop**

Problem
- requires good tracked features as input
Can we use SFM to help track points?
- basic idea: recall form of Lucas-Kanade equation:
  \[
  \begin{bmatrix}
  a & b & w_0 \\
  c & d & w_1 \\
  e & f & w_2 
  \end{bmatrix}
  \begin{bmatrix}
  u_1 \\
  v_1 \\
  1 
  \end{bmatrix} = \begin{bmatrix}
  u_0 \\
  v_0 \\
  1 
  \end{bmatrix}
  
  \]
  - with \(n\) points in \(f\) frames, we can stack into a big matrix
  \[
  \begin{bmatrix}
  A & B & U \\
  C & D & V \\
  1 & 1 & f 
  \end{bmatrix} = \begin{bmatrix}
  G \\
  H \\
  f 
  \end{bmatrix}
  \]
  - Matrix on RHS has rank \(<= 3\) !!
  - use SVD to compute a rank 3 approximation
  - has effect of filtering optical flow values to be consistent
  - [Irani 99]
From [Irani 99]

Figure 1: Real image sequence (the KTH collection sequence). (a) One frame from a 35-frame sequence of a formal drape moving in a 3-D scene. (b) The 3-D geometry with the motion lines & normals superimposed. Since the images are right-hand ones, there is depth information (positive in front of mirror), as well as the yawning position. (c) The same for the corresponding frame generated by the cubic spline time parameterization. Note the good recovery of flow in this region. (d) The false magnitudes are overly large. This display provides a higher resolution display of the error. Note the clear depth discontinuities in the cubic spline flow images. The false values in the edge are very small, because the centers H10 is in that area.