Lecture Schedule

<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Lecture (M: 2:05p-3:50, 50-N202)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>23-Jul</td>
<td>Introduction + Representing Position &amp; Orientation &amp; State</td>
</tr>
<tr>
<td>2</td>
<td>30-Jul</td>
<td>Robot Kinematics: Frames, Transformation Matrices &amp; Affine Transformations</td>
</tr>
<tr>
<td>3</td>
<td>6-Aug</td>
<td>Robot Kinematics: Forward, Inverse &amp; Differential Kinematics</td>
</tr>
<tr>
<td>4</td>
<td>13-Aug</td>
<td>Robot Dynamics: Jacobians &amp; Joint Torque</td>
</tr>
<tr>
<td>5</td>
<td>20-Aug</td>
<td>Robot Sensing: Linear Observers</td>
</tr>
<tr>
<td>6</td>
<td>27-Aug</td>
<td>Robot Sensing: Camera Models &amp; Color &amp; Calibration</td>
</tr>
<tr>
<td>7</td>
<td>3-Sep</td>
<td>Robot Sensing: View Geometry</td>
</tr>
<tr>
<td>8</td>
<td>10-Sep</td>
<td>Robot Sensing: Feature Detection &amp; Direct Methods</td>
</tr>
<tr>
<td>9</td>
<td>17-Sep</td>
<td>Motion Planning (Deterministic + Sample-Based [Probabilistic])&lt;br&gt;PS 2 [20%, 21/Sept]</td>
</tr>
<tr>
<td>10</td>
<td>24-Sep</td>
<td>(Spring) Study Break</td>
</tr>
<tr>
<td>11</td>
<td>1-Oct</td>
<td>Queen's Birthday!</td>
</tr>
<tr>
<td>12</td>
<td>8-Oct</td>
<td>Probabilistic Robotics: Motion Planning &amp; Control (LQR, Value Functions, Q-Learning, etc.)</td>
</tr>
<tr>
<td>13</td>
<td>15-Oct</td>
<td>Probabilistic Robotics: Localization &amp; SLAM</td>
</tr>
<tr>
<td></td>
<td>22-Oct</td>
<td>The Future of Robotics/Automation + Open Challenges</td>
</tr>
</tbody>
</table>
Geometry of Camera View(s)
Image Formation: (Thin-Lens) Projection model

- “Thin Lens” ≅ pinhole
  \[ \frac{y}{f} = \frac{Y}{Z} \quad \text{and} \quad \frac{x}{f} = \frac{X}{Z} \]

- Thus:
  \[ x = \frac{fX}{Z}, \quad y = \frac{fY}{Z} \]

\( (X, Y, Z) \mapsto (x, y) \)

\( \mathbb{R}^3 \mapsto \mathbb{R}^2 \)

Image and Slide based on: Corke, Ch. 11
Image Formation – Single View Geometry [I]

\[
(X, Y, Z)^T \mapsto \left( f\frac{X}{Z}, f\frac{Y}{Z} \right)^T
\]

Hartly & Zisserman, Ch. 6

Image Formation – Single View Geometry [II]

<table>
<thead>
<tr>
<th>Camera Projection Matrix</th>
</tr>
</thead>
</table>
| \[
\begin{pmatrix}
X \\
Y \\
Z \\
1
\end{pmatrix}
\mapsto
\begin{pmatrix}
fX + Zp_x \\
fY + Zp_y \\
Z
\end{pmatrix}
= 
\begin{bmatrix}
f & p_x & 0 \\
f & p_y & 0 \\
1 & 0 & 1
\end{bmatrix}
\begin{pmatrix}
X \\
Y \\
Z \\
1
\end{pmatrix}
<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
</table>

- \( x \) = Image point
- \( X \) = World point
- \( K \) = Camera Calibration Matrix

\[ P = K[R \mid t] \]
where: \( P \) is 3x4 and of rank 3
Camera Calibration!

Calibration matrix

- Is this form of $K$ good enough?
- non-square pixels (digital video)
- skew
- radial distortion

\[
\begin{bmatrix}
  f & 0 & u_c \\
  0 & f & v_c \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  X_c \\
  Y_c \\
  Z_c
\end{bmatrix}
= K \begin{bmatrix}
  X_c \\
  Y_c \\
  Z_c
\end{bmatrix}
\]

\[
\begin{bmatrix}
  f & s & u_c \\
  0 & f & v_c \\
  0 & 0 & 1
\end{bmatrix}
= K
\]

From Szeliski, *Computer Vision: Algorithms and Applications*
Calibration


- **Intrinsic: Internal Parameters**
  - **Focal length**: The focal length in pixels.
  - **Principal point**: The principal point
  - **Skew coefficient**: The skew coefficient defining the angle between the x and y pixel axes.
  - **Distortions**: The image distortion coefficients (radial and tangential distortions)
    (typically two quadratic functions)

- **Extrinsics: Where the Camera (image plane) is placed**:
  - **Rotations**: A set of 3x3 rotation matrices for each image
  - **Translations**: A set of 3x1 translation vectors for each image

---

Camera calibration

- Determine camera parameters from known 3D points or calibration object(s)
- internal or intrinsic parameters such as focal length, optical center, aspect ratio:
  what kind of camera?
- external or extrinsic (pose) parameters:
  where is the camera?
- How can we do this?

From Szeliski, *Computer Vision: Algorithms and Applications*
Complete camera model

\[
\begin{bmatrix}
\tilde{x} \\
\tilde{y} \\
\tilde{z}
\end{bmatrix} = \begin{bmatrix}
f & 0 & 0 & 0 \\
0 & f & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
X \\
Y \\
Z \\
1
\end{bmatrix}
\]

\[
\begin{bmatrix}
\tilde{u} \\
\tilde{v} \\
\tilde{w}
\end{bmatrix} = \begin{bmatrix}
\frac{1}{\rho_u} & 0 & u_0 \\
0 & \frac{1}{\rho_v} & v_0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
f & 0 & 0 & 0 \\
0 & f & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
R \\
t
\end{bmatrix}^{-1}
\begin{bmatrix}
X \\
Y \\
Z \\
1
\end{bmatrix}
\]

Some “Non-Perspective” Factors to Calibrate For:
Camera Image Lens Distortions

- Barrel
- Pincushion
- Fisheye

⇒ Explore these with visualize_distortions in the
Camera Calibration Toolbox

Fig. 2.1.3 from Szeliski, Computer Vision: Algorithms and Applications
Transformations of Single View Geometry

Image Rectification ➔ Transformation

To unwarp (rectify) an image
- solve for \( H \) given \( p'' \) and \( p \)
- solve equations of the form: \( sp'' = Hp \)
  - linear in unknowns: \( s \) and coefficients of \( H \)
  - need at least 4 points

Slide from Szeliski, *Computer Vision: Algorithms and Applications*
Transformations

- **\( x' \): New Image & \( \mathbf{x} \): Old Image

- **Euclidean:**
  (Distances preserved)
  \[ x' = \begin{bmatrix} R & t \end{bmatrix} \mathbf{x} \]

- **Similarity (Scaled Rotation):**
  (Angles preserved)
  \[ x' = \begin{bmatrix} sR & t \end{bmatrix} \mathbf{x} \]

Fig. 2.4 from Szeliski, *Computer Vision: Algorithms and Applications*

Transformations [2]

- **Affine:**
  (\( \| \) lines remain \( \| \))
  \[ x' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \mathbf{x} \]

- **Projective:**
  (straight lines preserved)
  \[ \mathbf{x}' = \mathbf{H} \mathbf{x} \]
  \[ x' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}} \]
  \[ y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}} \]

Fig. 2.4 from Szeliski, *Computer Vision: Algorithms and Applications*
Planar Projective Transformations

- Perspective projection of a plane
  - lots of names for this:
    - homography, colineation, planar projective map
  - Easily modeled using homogeneous coordinates

\[
\begin{bmatrix}
px' \\
py' \\
zs
\end{bmatrix} =
\begin{bmatrix}
*s & * & 0 & x \\
*s & * & 0 & y \\
*s & * & 1 & 0
\end{bmatrix}
\begin{bmatrix}
p_x \\
p_y \\
p_z \\
1
\end{bmatrix}
\]

To apply a homography \(H\)

- compute \(p' = Hp\)
- \(p'' = p'/s\) normalize by dividing by third component

A Take Away Lesson From This:
Measurements on Planes
Normally, one cannot just add a tape measure!

➔ Though An Alternative Approach: unwarp & then measure

Slide from Szeliski, *Computer Vision: Algorithms and Applications*
3D Projective Geometry

- These concepts generalize naturally to 3D
  - Homogeneous coordinates
    - Projective 3D points have four coords: \( P = (X, Y, Z, W) \)
  - Duality
    - A plane \( L \) is also represented by a 4-vector
    - Points and planes are dual in 3D: \( L P = 0 \)
  - Projective transformations
    - Represented by 4x4 matrices \( T: P' = TP, \quad L' = L T^{-1} \)
  - Lines are a special case…

3D → 2D Perspective Projection (Image Formation Equations)

\[
\begin{bmatrix}
X_c \\
Y_c \\
Z_c
\end{bmatrix} = 
\begin{bmatrix}
R
\end{bmatrix}_{3 \times 3} 
\begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix} + t
\]

\[
\begin{bmatrix}
u \\
v \\
1
\end{bmatrix} \sim 
\begin{bmatrix}
U \\
V \\
W
\end{bmatrix} = 
\begin{bmatrix}
f & 0 & u_c \\
0 & f & v_c \\
0 & 0 & 1
\end{bmatrix} 
\begin{bmatrix}
X_c \\
Y_c \\
Z_c
\end{bmatrix}
\]
3D → 2D Perspective Projection

• Matrix Projection (camera matrix):

\[ \mathbf{p} = \begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \mathbf{\Pi}_{2D} \mathbf{p} \]

It’s useful to decompose \( \mathbf{\Pi} \) into \( \mathbf{T} \rightarrow \mathbf{R} \rightarrow \text{project} \rightarrow \mathbf{A} \)

\[ \mathbf{\Pi} = \begin{bmatrix} s_x & 0 & -t_x \\ 0 & s_y & -t_y \\ 0 & 0 & 1/f \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I}_{3\times3} \\ \mathbf{T}_{3\times1} \end{bmatrix} \]

3D Transformations

<table>
<thead>
<tr>
<th>Name</th>
<th>Matrix</th>
<th># D.O.F.</th>
<th>Preserves</th>
<th>Icon</th>
</tr>
</thead>
<tbody>
<tr>
<td>translation</td>
<td>( \begin{bmatrix} I &amp; t \end{bmatrix}_{3\times4} )</td>
<td>3</td>
<td>orientation + \cdots</td>
<td>( \square )</td>
</tr>
<tr>
<td>rigid (Euclidean)</td>
<td>( \begin{bmatrix} R &amp; t \end{bmatrix}_{3\times4} )</td>
<td>6</td>
<td>lengths + \cdots</td>
<td>( \diamond )</td>
</tr>
<tr>
<td>similarity</td>
<td>( \begin{bmatrix} sR &amp; t \end{bmatrix}_{3\times4} )</td>
<td>7</td>
<td>angles + \cdots</td>
<td>( \diamond )</td>
</tr>
<tr>
<td>affine</td>
<td>( \begin{bmatrix} A \end{bmatrix}_{3\times4} )</td>
<td>12</td>
<td>parallelism + \cdots</td>
<td>( \bigtriangleup )</td>
</tr>
<tr>
<td>projective</td>
<td>( \begin{bmatrix} H \end{bmatrix}_{4\times4} )</td>
<td>15</td>
<td>straight lines</td>
<td>( \square )</td>
</tr>
</tbody>
</table>
Compare to 2D Transformations

Projection Models

- Orthographic

- Weak Perspective

- Affine

- Perspective

- Projective

Slide from Szeliski, Computer Vision: Algorithms and Applications
The Projective Plane

- Why do we need homogeneous coordinates?
  - Represent points at infinity, homographies, perspective projection, multi-view relationships
- What is the geometric intuition?
  - A point in the image is a ray in projective space

Each point \((x, y)\) on the plane is represented by a ray \((sx, sy, s)\)
- all points on the ray are equivalent: \((x, y, 1) \equiv (sx, sy, s)\)

Projective Lines

- What is a line in projective space?
  - A line is a plane of rays through origin
  - all rays \((x, y, z)\) satisfying: \(ax + by + cz = 0\)

  \[
  \begin{bmatrix}
  a & b & c \\
  x & y & z
  \end{bmatrix}
  \]

  in vector notation: \(0 = \begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}\)

  \(l^T p\)

  A line is represented as a homogeneous 3-vector \(l\)
Ideal points and lines

- **Ideal point** ("point at infinity")
  - \( p \cong (x, y, 0) \) – parallel to image plane
  - It has infinite image coordinates

![Diagram of ideal point](image)

**Line at infinity**

- \( l_\infty \cong (0, 0, 1) \) – parallel to image plane
- Contains all ideal points

![Diagram of line at infinity](image)

Point and Line Duality

- A line \( l \) is a homogeneous 3-vector (a ray)
- It is \( \perp \) to every point (ray) \( p \) on the line: \( l^T p = 0 \)

![Diagram of point and line duality](image)

- What is the line \( l \) spanned by rays \( p_1 \) and \( p_2 \)?
  - \( l \) is \( \perp \) to \( p_1 \) and \( p_2 \) \( \Rightarrow \) \( l = p_1 \times p_2 \) (\( l \) is the plane normal)
- What is the intersection of two lines \( l_1 \) and \( l_2 \)?
  - \( p \) is \( \perp \) to \( l_1 \) and \( l_2 \) \( \Rightarrow \) \( p = l_1 \times l_2 \)
- Points and lines are *dual in projective space*
  - every property of points also applies to lines
Point and Line Duality [II]

Homogeneous ⇔ Cartesian

- **Point:**
  \[ \mathbf{P} = (\bar{x}, \bar{y}, \bar{z}) \mid \mathbf{P} = (x, y) \quad x = \frac{\bar{x}}{\bar{z}}, y = \frac{\bar{y}}{\bar{z}} \]

- **Line:**
  - Is such that \( \mathbf{l}^T \mathbf{p} = 0 \)
  - Point Eq of a line is: \( y = mx + b \)

---

Point and Line Duality [III]

- 2 Points Make a Line

  \[ \mathbf{l} = \mathbf{p}_1 \times \mathbf{p}_2 \]

- 2 Lines Make Point!

  \[ \mathbf{p} = \mathbf{l}_1 \times \mathbf{l}_2 \]

Image/Notation from: Corke, Ch. 11
“Fundamental”
Multi-View Geometry

(aka: “Notorious MVG” 😊)

Image Formation – Two-View Geometry [Stereopsis] ➞ Fundamental Matrix

\[(R,T)\]
Stereo Geometry $\rightarrow$ Epipolar Geometry

- Match features along epipolar lines

Epipolar lines := are the projection of the pencil of planes passing through the centers

- For 2 images (or images with collinear camera centers):
  We can find epipolar lines that intersect and thus “simplify” the stereo feature matching and correspondence problem

- Rectification := warping the input images (perspective transformation) so that epipolar lines are horizontal
Two-View Geometry: Epipolar Plane

- **Epipole**: the point of intersection of the line joining the camera centres (the baseline) with the image plane. Equivalently, the epipole is the image in one view of the camera centre of the other view.

- **Epipolar plane**: a plane containing the baseline. There is a one-parameter family (a pencil) of epipolar planes.

- **Epipolar line**: the intersection of an epipolar plane with the image plane. All epipolar lines intersect at the epipole. An epipolar plane intersects the left and right image planes in epipolar lines, and defines the correspondence between the lines.

Key Correlator: Vanishing Points

- **Vanishing Points** can be fun…

  - They also hold a key to correlating views!

Slide from Szeliski, *Computer Vision: Algorithms and Applications*
Vanishing Points (2D)

- Any two parallel lines have the same vanishing point
- The ray from C through v point is parallel to the lines
- An image may have more than one vanishing point
Vanishing Points

- Vanishing point
  - projection of a point at infinity
  - whiteboard capture, architecture,…

Vanishing Lines

- Multiple Vanishing Points
  - Any set of parallel lines on the plane define a vanishing point
  - The union of all of these vanishing points is the horizon line
Back to Two-frames

Two classes of two-frame main variants:

I. **Calibrated**: \textit{“Essential matrix”} $E$
   - Use ray directions ($x_i, x'_i$)

II. **Uncalibrated**: \textit{“Fundamental matrix”} $F$

THE reference: [Hartley & Zisserman, Chapter 9]

---

I. Essential matrix

- Co-planarity constraint:
  - $x' \approx Rx + t$
  - $[t] \times x' \approx [t] \times Rx$
  - $x' [t] \times x' \approx x' [t] \times Rx$
  - $x' E x = 0$ with $E = [t] \times R$

- Solve for $E$ using least squares (SVD)
- $t$ is the least singular vector of $E$
- $R$ obtained from the other two s.v.s

From Szeliski, \textit{Computer Vision: Algorithms and Applications}
II. Fundamental Matrix

• The fundamental matrix is the algebraic representation of epipolar geometry.

Fig. 9.5. A point $x$ in one image is transferred via the plane $\pi$ to a matching point $x'$ in the second image. The epipolar line through $x'$ is obtained by joining $x'$ to the epipole $e'$. In symbols one may write $x' = H_\pi x$ and $y' = [e']', x' = [e']$, $H_\pi x = F x$ where $F = [e]'$, $H_\pi$ is the fundamental matrix.

Reference: [Hartley & Zisserman, Chapter 9, § 9.2-9.4]

Fundamental matrix

• Camera calibrations are unknown

\[ x' F x = 0 \text{ with } F = [e] \quad H = K'[t] \quad R K^{-1} \]

• Solve for $F$ using least squares (SVD)
  
  – re-scale $(x_i, x_i')$ so that $|x_i| \approx \frac{1}{2}$ [Hartley]

• $e$ (epipole) is still the least singular vector of $F$

• $H$ obtained from the other two s.v.s

• “plane + parallax” (projective) reconstruction

• use self-calibration to determine $K$ [Pollefeys]

Reference: [Hartley & Zisserman, Chapter 9, p. 246]
Fundamental Matrix Example

- Suppose the camera matrices are those of a calibrated stereo rig with the world origin at the first camera
  \[ P = K[I \mid 0] \quad P' = K'[R \mid t]. \]
- Then:
  \[ P^+ = \begin{bmatrix} K^{-1} \\ 0^T \end{bmatrix} \quad C = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \]
- Epipoles are at:
  \[ e = P' \begin{pmatrix} -R^T t \\ 1 \end{pmatrix} = KR^T t \quad e' = P' \begin{pmatrix} 0 \\ 1 \end{pmatrix} = K't. \]

\[ F = [e']_x K'R^{-1} = K'-T[R^T t]_x R^{-1} = K'-T[R[R^T t]_x K^{-1} = K'-T R K^T[e]_x \]

Reference: [Hartley & Zisserman, Chapter 9, § 9.2-9.4]

Summary of fundamental matrix properties

- \( F \) is a rank 2 homogeneous matrix with 7 degrees of freedom.
- **Point correspondence**: If \( x \) and \( x' \) are corresponding image points, then
  \[ x'^T F x = 0. \]
- **Epipolar lines**:
  - \( l' = F x \) is the epipolar line corresponding to \( x \).
  - \( l = F^T x' \) is the epipolar line corresponding to \( x' \).
- **Epipoles**:
  - \( F o = 0. \)
  - \( F^T e' = 0. \)
- **Computation from camera matrices** \( P, P' \):
  - General cameras,
    \[ F = [e']_x P' P^+, \]  where \( P^+ \) is the pseudo-inverse of \( P \), and \( e' = F'C \), with \( FC = 0. \)
  - Canonical cameras, \( P = [I \mid 0], \) \( P' = [M \mid m] \).
    \[ F = [e']_x M = M^{-1}[e]_x, \]  where \( e' = m \) and \( e = m^{-1} m. \)
  - Cameras not at infinity \( P = K[I \mid 0], \) \( P' = K'[R \mid t] \).
    \[ F = K'-T[R^T(t)]_x R^{-1} = [K't]_x K'R^{-1} = K'-T R K^T K[R^T t]_x. \]

Reference: [Hartley & Zisserman, Chapter 9, p. 246]
Fundamental Matrix & Motion

- Under a pure translational camera motion, 3D points appear to slide along parallel rails. The images of these parallel lines intersect in a vanishing point corresponding to the translation direction. The epipole \( \mathbf{e} \) is the vanishing point.

\[
x'^T F x = 0.
\]

Cool Robotics Share: Fundamental Matrix Song

D. Wedge, The Fundamental Matrix Song [https://youtu.be/DgGV3l82NTk]
SFM: Structure from Motion

Structure [from] Motion

- Given a set of feature tracks, estimate the 3D structure and 3D (camera) motion.

- Assumption: orthographic projection

- Tracks: \((u_{fp}, v_{fp}), f: \text{frame}, p: \text{point}\)

- Subtract out mean 2D position…

  \[ i_f: \text{rotation}, \ s_p: \text{position} \]

  \[ u_{fp} = i_f^T s_p, \ v_{fp} = j_f^T s_p \]

From Szeliski, *Computer Vision: Algorithms and Applications*
Structure from motion

• How many points do we need to match?
• 2 frames:
  – (R,t): 5 dof + 3n point locations ≤ \( \hat{u}_{ij} = f(K, R_j, t_j, x_i) \)
  – 4n point measurements ⇒ \( \hat{v}_{ij} = g(K, R_j, t_j, x_i) \)
  – \( n \geq 5 \)
• k frames:
  – 6(k–1)-1 + 3n ≤ 2kn
• always want to use many more

From Szeliski, *Computer Vision: Algorithms and Applications*

Measurement equations

• Measurement equations
  \( u_{fp} = i_f^T s_p \)
  \( j_f^T s_p \)

• Stack them up…
  \( W = R S \)
  \( R = (i_1,...,i_F, j_1,...,j_F)^T \)
  \( S = (s_1,...,s_p) \)

From Szeliski, *Computer Vision: Algorithms and Applications*
Factorization

\[ W = R_{2F \times 3} S_{3 \times P} \]

SVD

\[ W = U \Lambda V \quad \Lambda \text{ must be rank 3} \]

\[ W' = (U \Lambda^{1/2})(\Lambda^{1/2} V) = U' V' \]

Make \( R \) orthogonal

\[ R = QU', \quad S = Q^{-1}V' \]

\[ i_f^T Q^T i_f = 1 \]

From Szeliski, *Computer Vision: Algorithms and Applications*

Results

- Look at paper figures…

Figure 4.1: A view of the computed shape from approximately above the building (compare with figure 4.6).

Figure 4.7: For a quantitative evaluation, distances between the features shown in the picture were measured on the actual model, and compared with the computed results. The comparison is shown in figure 4.6.

From Szeliski, *Computer Vision: Algorithms and Applications*
Bundle Adjustment

- What makes this non-linear minimization hard?
  - many more parameters: potentially slow
  - poorer conditioning (high correlation)
  - potentially lots of outliers
  - gauge (coordinate) freedom

\[
\tilde{u}_{ij} = f(K, R_j, t_j, x_i) \\
\tilde{v}_{ij} = g(K, R_j, t_j, x_i)
\]

From Szeliski, *Computer Vision: Algorithms and Applications*

Lots of parameters: sparsity

- Only a few entries in Jacobian are non-zero

\[
\tilde{u}_{ij} = f(K, R_j, t_j, x_i) \\
\tilde{v}_{ij} = g(K, R_j, t_j, x_i)
\]

\[
\frac{\partial \tilde{u}_{ij}}{\partial K}, \quad \frac{\partial \tilde{u}_{ij}}{\partial R_j}, \quad \frac{\partial \tilde{u}_{ij}}{\partial t_j}, \quad \frac{\partial \tilde{u}_{ij}}{\partial x_i},
\]

\[
J = \begin{bmatrix} ... \end{bmatrix} \quad H = \begin{bmatrix} ... \end{bmatrix}
\]

From Szeliski, *Computer Vision: Algorithms and Applications*
Sparse Cholesky (skyline)

- First used in finite element analysis
- Applied to SfM by [Szeliski & Kang 1994]

Conditioning and gauge freedom

- Poor conditioning:
  - use 2nd order method
  - use Cholesky decomposition

- Gauge freedom
  - fix certain parameters (orientation) or
  - zero out last few rows in Cholesky decomposition
Cool Robotics Share²: Photosynth & Bundler

Cool Robotics Share³:
Handheld Monocular Object Reconstruction