Computer Vision: Panorama

Raquel Urtasun

TTI Chicago

Feb 5, 2013
What did we see in class last week?
Image Alignment Algorithm

Given images $A$ and $B$

1. Compute image features for $A$ and $B$
2. Match features between $A$ and $B$
3. Compute homography between $A$ and $B$ using least squares on set of matches

Is there a problem with this?

[Source: N. Snavely]
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model

[Source: R. Raguram]
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function

[Source: R. Raguram]
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model

[Source: R. Raguram]
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize and verify loop

[Source: R. Raguram]
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize and verify loop

[Source: R. Raguram]
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize and verify loop
6. Choose model with largest set of inliers

[Source: R. Raguram]
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize and verify loop
6. Choose model with largest set of inliers

[Source: R. Raguram]
Hough Transform Algorithm

With the parameterization $x \cos \theta + y \sin \theta = r$

- Let $r \in [-R, R]$ and $\theta \in [0, \pi)$
- For each edge point $(x_i, y_i)$, calculate: $\hat{r} = x_i \cos \hat{\theta} + y_i \sin \hat{\theta}$ $\forall \hat{\theta} \in [0, \pi)$
- Increase accumulator $A(\hat{r}, \hat{\theta}) = A(\hat{r}, \hat{\theta}) + 1$

- Threshold the accumulator values to get parameters for detected lines

[Source: M. Kazhdan]
The coordinate system

- We will use the pinhole model as an approximation
- Put the optical center (Center Of Projection) at the origin
The coordinate system

- We will use the pinhole model as an approximation
- Put the **optical center** (Center Of Projection) at the origin
- Put the **image plane** (Projection Plane) in front of the COP. Why?
The coordinate system

- We will use the pinhole model as an approximation
- Put the **optical center** (Center Of Projection) at the origin
- Put the **image plane** (Projection Plane) in front of the COP. Why?
  - The camera looks down the negative z axis, for right-handed-coordinates

[Source: N. Snavely]
The **coordinate system**

- We will use the pinhole model as an approximation
- Put the **optical center** (Center Of Projection) at the origin
- Put the **image plane** (Projection Plane) in front of the COP. Why?
- The camera looks down the negative z axis, for right-handed-coordinates

[Source: N. Snavely]
Projection Equations

- Compute intersection with PP of ray from $(x, y, z)$ to COP. How?
- Derived using similar triangles

$$(x, y, z) \rightarrow (-d \frac{x}{z}, -d \frac{y}{z}, -d)$$
Modeling projection

Projection Equations

- Compute intersection with PP of ray from \((x, y, z)\) to COP. How?
- Derived using similar triangles

\[
(x, y, z) \rightarrow (-d \frac{x}{z}, -d \frac{y}{z}, -d)
\]

- Get the projection by throwing the last coordinate

[Source: N. Snavely]
Projection Equations

- Compute intersection with PP of ray from \((x, y, z)\) to COP. How?
- Derived using similar triangles

\[
(x, y, z) \rightarrow (-d \frac{x}{z}, -d \frac{y}{z}, -d)
\]

- Get the projection by throwing the last coordinate

[Source: N. Snavely]
Perspective

3D World

Perspective Projection
Variants of Orthographic

3D World

Orthographic Projection
Projection properties

- **Many-to-one**: any points along same ray map to same point in image
- Points $\rightarrow$ points
Projection properties

- **Many-to-one**: any points along same ray map to same point in image
  - Points $\rightarrow$ points

- Lines $\rightarrow$ lines

[Source: N. Snavely]
Projection properties

- **Many-to-one**: any points along same ray map to same point in image
  - Points $\rightarrow$ points
  - Lines $\rightarrow$ lines
  - But line through focal point projects to a point. Why?
Projection properties

- **Many-to-one**: any points along same ray map to same point in image
- Points $\rightarrow$ points
- Lines $\rightarrow$ lines
- But line through focal point projects to a point. Why?
- Planes $\rightarrow$ planes
Many-to-one: any points along same ray map to same point in image
Points → points
Lines → lines
But line through focal point projects to a point. Why?
Planes → planes
But plane through focal point projects to line. Why?

[Source: N. Snavely]
Projection properties

- **Many-to-one**: any points along same ray map to same point in image
  - Points $\rightarrow$ points
  - Lines $\rightarrow$ lines
  - But line through focal point projects to a point. Why?
  - Planes $\rightarrow$ planes
  - But plane through focal point projects to line. Why?

[Source: N. Snavely]
Camera Parameters

How many numbers do we need to describe a camera?

- We need to describe its **pose in the world**
How many numbers do we need to describe a camera?

- We need to describe its **pose in the world**
- We need to describe its **internal parameters**
How many numbers do we need to describe a camera?

- We need to describe its **pose in the world**
- We need to describe its **internal parameters**
- How many then?

[Source: N. Snavely]
How many numbers do we need to describe a camera?

- We need to describe its **pose in the world**
- We need to describe its **internal parameters**
- How many then?

[Source: N. Snavely]
The projection matrix models the cumulative effect of all intrinsic and extrinsic parameters:

\[ \mathbf{X} = \begin{bmatrix} ax \\ ay \\ a \end{bmatrix} = \mathbf{P} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \]

It can be computed as:

\[ \mathbf{P} = \begin{bmatrix} -f \cdot s_x & 0 & x'_c \\ 0 & -f \cdot s_y & y'_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{3 \times 3} & \mathbf{0}_{3 \times 1} \\ \mathbf{0}_{1 \times 3} & 1 \\ \mathbf{T}_{3 \times 1} \end{bmatrix} \]

- intrinsics
- projection
- rotation
- translation
The projection matrix models the cumulative effect of all intrinsic and extrinsic parameters.

\[
X = \begin{bmatrix}
ax \\
ay \\
a
\end{bmatrix} = P
\begin{bmatrix}
X \\
Y \\
Z \\
1
\end{bmatrix}
\]

It can be computed as

\[
P = \begin{bmatrix}
-f \cdot s_x & 0 & x'_c \\
0 & -f \cdot s_y & y'_c \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
R_{3 \times 3} & 0_{3 \times 1} \\
0_{1 \times 3} & 1 \\
0_{1 \times 3} & 1
\end{bmatrix}
\]

No standard definition of intrinsics and extrinsics.
The projection matrix models the cumulative effect of all intrinsic and extrinsic parameters:

\[
X = \begin{bmatrix}
ax \\
ay \\
a
\end{bmatrix}
= P
\begin{bmatrix}
X \\
Y \\
Z \\
1
\end{bmatrix}
\]

It can be computed as:

\[
P = \begin{bmatrix}
-f \cdot s_x & 0 & x'_c \\
0 & -f \cdot s_y & y'_c \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
R_{3 \times 3} & 0_{3 \times 1} & I_{3 \times 3} & T_{3 \times 1}
\end{bmatrix}
\]

No standard definition of intrinsics and extrinsics.
Extrinsics

How do we get the camera to canonical form?

Step 1: Translate by \(-c\)

[Source: N. Snavely]
Extrinsics

How do we get the camera to canonical form?

Step 1: Translate by $-c$

How do we represent translation as a matrix multiplication?

$$
T = \begin{bmatrix}
I_{3 \times 3} & -c \\
0 & 0 & 0 & 1
\end{bmatrix}
$$

[Source: N. Snavely]
Extrinsics

How do we get the camera to canonical form?

Step 1: Translate by $-\mathbf{c}$
Step 2: Rotate by $\mathbf{R}$

$\mathbf{R} = \begin{bmatrix} \mathbf{u}^T \\ \mathbf{v}^T \\ \mathbf{w}^T \end{bmatrix}$

3x3 rotation matrix

[Source: N. Snavely]
Extrinsics

How do we get the camera to canonical form?

Step 1: Translate by $-c$
Step 2: Rotate by $R$

$R = \begin{bmatrix} u^T \\ v^T \\ w^T \end{bmatrix}$

[Source: N. Snavely]
Perspective Projection

\[
\begin{bmatrix}
-f & 0 & 0 \\
0 & -f & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\]

\( K \)

(intrinsics) (converts from 3D rays in camera coordinate system to pixel coordinates)

in general, \( K = \begin{bmatrix}
-f & s & c_x \\
0 & -\alpha f & c_y \\
0 & 0 & 1
\end{bmatrix} \)

(upper triangular matrix)

\( \alpha \): aspect ratio (1 unless pixels are not square)

\( s \): skew (0 unless pixels are shaped like rhombi/parallelograms)

\( c_x, c_y \): principal point ((0,0) unless optical axis doesn’t intersect projection plane at origin)

- Simplifications used in practice

[Source: N. Snavely]
Today’s Readings

- Chapter 9 of Szeliski’s book
Let’s look at panoramas again
Can we use homography to create a 360 panorama?

[Source: N Snavely]
Can we use homography to create a 360 panorama?

- Idea: projecting images onto a common plane

We’ll see what this homograph means later.

First -- Can’t create a 360 panorama this way...

[Source: N Snavely]
Creating Panoramas

- Before we can register and align images, we need mathematical relationships that **map pixel coordinates from one image to another**

- A variety of such **parametric motion models** are possible from
Before we can register and align images, we need mathematical relationships that **map pixel coordinates from one image to another**.

A variety of such **parametric motion models** are possible from:

- simple 2D transforms
Creating Panoramas

- Before we can register and align images, we need mathematical relationships that map pixel coordinates from one image to another.
- A variety of such parametric motion models are possible from:
  - simple 2D transforms
  - planar perspective models
Creating Panoramas

- Before we can register and align images, we need mathematical relationships that map pixel coordinates from one image to another.
- A variety of such parametric motion models are possible from:
  - simple 2D transforms
  - planar perspective models
  - 3D camera rotations
Creating Panoramas

- Before we can register and align images, we need mathematical relationships that map pixel coordinates from one image to another.

- A variety of such parametric motion models are possible from:
  - simple 2D transforms
  - planar perspective models
  - 3D camera rotations
  - lens distortions
Before we can register and align images, we need mathematical relationships that map pixel coordinates from one image to another.

A variety of such parametric motion models are possible from:

- simple 2D transforms
- planar perspective models
- 3D camera rotations
- lens distortions
- mapping to non-planar (e.g., cylindrical) surfaces
Creating Panoramas

Before we can register and align images, we need mathematical relationships that map pixel coordinates from one image to another.

A variety of such parametric motion models are possible from:
- simple 2D transforms
- planar perspective models
- 3D camera rotations
- lens distortions
- mapping to non-planar (e.g., cylindrical) surfaces

Deciding which model is a model selection problem.
Creating Panoramas

- Before we can register and align images, we need mathematical relationships that **map pixel coordinates from one image to another**.

- A variety of such **parametric motion models** are possible from:
  - simple 2D transforms
  - planar perspective models
  - 3D camera rotations
  - lens distortions
  - mapping to non-planar (e.g., cylindrical) surfaces

- Deciding which model is a model selection problem.

(a) translation [2 dof]  
(b) affine [6 dof]  
(c) perspective [8 dof]  
(d) 3D rotation [3+ dof]
Simple Motion Model

- Consists of 2D rotation and translation
- In a **panography**, images are translated, rotated and scaled.
- We saw the case of linear transformations, where we used least squares
- To be more robust we employed RANSAC or Hough transform
Consider, the problem of estimating a **rigid Euclidean 2D transformation** (translation plus rotation) between two sets of points.

If we parameterize this transformation by the translation \((t_x; t_y)\) and the rotation angle \(\theta\), the Jacobian of this transformation, depends on the current value of \(\theta\).

Is this problematic?

<table>
<thead>
<tr>
<th>Transform</th>
<th>Matrix</th>
<th>Parameters (p)</th>
<th>Jacobian (J)</th>
</tr>
</thead>
</table>
| translation | \[
\begin{bmatrix}
1 & 0 & t_x \\
0 & 1 & t_y
\end{bmatrix}
\]                     | \((t_x, t_y)\)   | \[
\begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}
\]                                |
| Euclidean   | \[
\begin{bmatrix}
c_\theta & -s_\theta & t_x \\
s_\theta & c_\theta & t_y
\end{bmatrix}
\]                   | \((t_x, t_y, \theta)\) | \[
\begin{bmatrix}
1 & 0 & -s_\theta x - c_\theta y \\
0 & 1 & c_\theta x - s_\theta y
\end{bmatrix}
\]                                      |
| similarity  | \[
\begin{bmatrix}
1 + a & -b & t_x \\
b & 1 + a & t_y
\end{bmatrix}
\]                     | \((t_x, t_y, a, b)\) | \[
\begin{bmatrix}
1 & 0 & x & -y \\
0 & 1 & y & x
\end{bmatrix}
\]                                |
| affine      | \[
\begin{bmatrix}
1 + a_{00} & a_{01} & t_x \\
a_{10} & 1 + a_{11} & t_y
\end{bmatrix}
\]                 | \((t_x, t_y, a_{00}, a_{01}, a_{10}, a_{11})\) | \[
\begin{bmatrix}
1 & 0 & x & y & 0 & 0 \\
0 & 1 & 0 & 0 & x & y
\end{bmatrix}
\]                                |
Minimizing the non-linear least-squares

- **Iteratively** update $\Delta p$ to the current parameter estimate $\Delta p$ by minimizing

  $$E_{NLS}(\Delta p) = \sum_{i} \| f(x_i; p + \Delta p) - x_i' \|^2$$

- We can approximate this by

  $$E_{NLS}(\Delta p) \approx \sum_{i} \| J(x_i; p) \Delta p - r_i' \|^2$$
Minimizing the non-linear least-squares

- **Iteratively** update $\Delta p$ to the current parameter estimate $\Delta p$ by minimizing

$$E_{NLS}(\Delta p) = \sum_i \| f(x_i; p + \Delta p) - x_i' \|^2_2$$

- We can approximate this by

$$E_{NLS}(\Delta p) \approx \sum_i \| J(x_i; p) \Delta p - r_i' \|^2_2$$

- Expanding this we have

$$E_{NLS}(\Delta p) \approx \Delta_p^T A \Delta p - 2 \Delta p^T b + c$$

with $A = \sum_i J^T J$ the Hessian and

$$b = \sum_i J^T(x_i)r_i$$

is a Jacobian-weighted sum of residual vectors.
Minimizing the non-linear least-squares

- **Iteratively** update $\Delta p$ to the current parameter estimate $\Delta p$ by minimizing

$$E_{NLS}(\Delta p) = \sum_i \| f(x_i; p + \Delta p) - x_i' \|^2$$

- We can approximate this by

$$E_{NLS}(\Delta p) \approx \sum_i \| J(x_i; p) \Delta p - r_i' \|^2$$

- Expanding this we have

$$E_{NLS}(\Delta p) \approx \Delta^T p A \Delta p - 2\Delta p^T b + c$$

with $A = \sum_i J^T J$ the Hessian and

$$b = \sum_i J^T(x_i) r_i$$

is a Jacobian-weighted sum of residual vectors.
Minimizing the non-linear least-squares

- The parameters are pulled in the direction of the prediction error with strength proportional to the Jacobian.

- Once $A$ and $b$ are computed, one solves for $\Delta p$ by solving

\[(A + \lambda \text{diag}(A))\Delta p = b\]

with $\lambda$ a damping parameter.
Minimizing the non-linear least-squares

- The parameters are pulled in the direction of the prediction error with strength proportional to the Jacobian.
- Once $A$ and $b$ are computed, one solves for $\Delta p$ by solving:

$$\begin{align*}
(A + \lambda \text{diag}(A))\Delta p &= b \\
\end{align*}$$

with $\lambda$ a damping parameter.
- Thus the algorithm looks like:

```plaintext
repeat
  1. Compute $A$ and $b$ at current solution
  2. Solve for $\Delta p$
  3. $p \leftarrow p + \Delta p$
end
```

How to initialize?
The parameters are pulled in the direction of the prediction error with strength proportional to the Jacobian

Once $A$ and $b$ are computed, one solves for $\Delta p$ by solving

$$(A + \lambda \text{diag}(A))\Delta p = b$$

with $\lambda$ a damping parameter

Thus the algorithm looks like

repeat

1. Compute $A$ and $b$ at current solution
2. Solve for $\Delta p$,
3. $p \leftarrow p + \Delta p$

end

How to initialize?
Minimizing the non-linear least-squares

- The parameters are pulled in the direction of the prediction error with strength proportional to the Jacobian.
- Once \( \mathbf{A} \) and \( \mathbf{b} \) are computed, one solves for \( \Delta \mathbf{p} \) by solving

\[
(\mathbf{A} + \lambda \text{diag} (\mathbf{A})) \Delta \mathbf{p} = \mathbf{b}
\]

with \( \lambda \) a damping parameter.
- Thus the algorithm looks like

```plaintext
repeat
    1. Compute \( \mathbf{A} \) and \( \mathbf{b} \) at current solution
    2. Solve for \( \Delta \mathbf{p} \),
    3. \( \mathbf{p} \leftarrow \mathbf{p} + \Delta \mathbf{p} \)
end
```

- How to initialize?
For the case of our 2D translation+rotation, we end up with a $3 \times 3$ set of normal equations in the unknowns $\delta t_x, \delta t_y, \delta \theta$.

An initial guess for translation can be obtained by fitting a four-parameter similarity transform in $(t_x; t_y; c; s)$ and then setting $\theta = \tan^{-1}(s/c)$. 

For the case of our 2D translation+rotation, we end up with a $3 \times 3$ set of normal equations in the unknowns $\delta t_x, \delta t_y, \delta \theta$.

An initial guess for translation can be obtained by fitting a four-parameter similarity transform in $(t_x; t_y; c; s)$ and then setting $\theta = \tan^{-1}(s/c)$.

An alternative approach is to estimate the translation parameters using the centroids of the 2D points and to then estimate the rotation angle using polar coordinates.
For the case of our 2D translation+rotation, we end up with a $3 \times 3$ set of normal equations in the unknowns $\delta t_x, \delta t_y, \delta \theta$

An initial guess for translation can be obtained by fitting a four-parameter similarity transform in $(t_x; t_y; c; s)$ and then setting $\theta = \tan^{-1}(s/c)$.

An alternative approach is to estimate the translation parameters using the centroids of the 2D points and to then estimate the rotation angle using polar coordinates.
The mapping between two cameras viewing a common plane can be described with a $3 \times 3$ homography.

Consider $M_{10}$, the matrix that arises from mapping a pixel in one image to a 3D point and then back onto the second image:

$$\hat{x}_1 \sim \hat{P}_1 \hat{P}_0^{-1} \hat{x}_0 = M_{10} \hat{x}_0$$
Planar Perspective Motion

- The mapping between two cameras viewing a common plane can be described with a $3 \times 3$ homography.

- Consider $\mathbf{M}_{10}$, the matrix that arises from mapping a pixel in one image to a 3D point and then back onto the second image:

  $$\hat{x}_1 \sim \hat{\mathbf{P}}_1 \hat{\mathbf{P}}_0^{-1} \hat{x}_0 = \mathbf{M}_{10} \hat{x}_0$$

- When the last row of the $\hat{\mathbf{P}}_0$ matrix is replaced with a plane equation $\hat{n}_0 \cdot \mathbf{p} + c_0$ and points are assumed to lie on this plane, i.e., their disparity is $d = 0$ we can ignore the last column of $\mathbf{M}_{10}$ and also its last row, since we do not care about the final z-buffer depth:

  $$\hat{x}_1 \sim \hat{\mathbf{H}}_{10} \hat{x}_0$$
Planar Perspective Motion

- The mapping between two camera viewing a common plane can be described with a $3 \times 3$ homography.

- Consider $M_{10}$, the matrix that arises from mapping a pixel in one image to a 3D point and then back onto the second image

$$\hat{x}_1 \sim \hat{P}_1 \hat{P}_0^{-1} \hat{x}_0 = M_{10} \hat{x}_0$$

- When the last row of the $\hat{P}_0$ matrix is replaced with a plane equation $\hat{n}_0 \cdot p + c_0$ and points are assumed to lie on this plane, i.e., their disparity is $d = 0$ we can ignore the last column of $M_{10}$ and also its last row, since we do not care about the final z-buffer depth

$$\hat{x}_1 \sim \hat{H}_{10} \hat{x}_0$$

- You will show this in an exercise
Planar Perspective Motion

- The mapping between two camera viewing a common plane can be described with a $3 \times 3$ homography.

- Consider $M_{10}$, the matrix that arises from mapping a pixel in one image to a 3D point and then back onto the second image

\[
\hat{x}_1 \sim \hat{P}_1 \hat{P}_0^{-1} \hat{x}_0 = M_{10} \hat{x}_0
\]

- When the last row of the $\hat{P}_0$ matrix is replaced with a plane equation $\hat{n}_0 \cdot p + c_0$ and points are assumed to lie on this plane, i.e., their disparity is $d = 0$ we can ignore the last column of $M_{10}$ and also its last row, since we do not care about the final z-buffer depth

\[
\hat{x}_1 \sim \hat{H}_{10} \hat{x}_0
\]

- You will show this in an exercise

- More recent algorithms use robust methods such as RANSAC
Planar Perspective Motion

- The mapping between two camera viewing a common plane can be described with a $3 \times 3$ homography.

- Consider $\mathbf{M}_{10}$, the matrix that arises from mapping a pixel in one image to a 3D point and then back onto the second image

$$
\hat{x}_1 \sim \hat{\mathbf{P}}_1 \hat{\mathbf{P}}_0^{-1} \hat{x}_0 = \mathbf{M}_{10} \hat{x}_0
$$

- When the last row of the $\hat{\mathbf{P}}_0$ matrix is replaced with a plane equation \( \hat{\mathbf{n}}_0 \cdot \mathbf{p} + c_0 \) and points are assumed to lie on this plane, i.e., their disparity is $d = 0$ we can ignore the last column of $\mathbf{M}_{10}$ and also its last row, since we do not care about the final z-buffer depth

$$
\hat{x}_1 \sim \hat{\mathbf{H}}_{10} \hat{x}_0
$$

- You will show this in an exercise

- More recent algorithms use robust methods such as RANSAC

- How do we align multiple images? Is there a problem?
Planar Perspective Motion

- The mapping between two camera viewing a common plane can be described with a $3 \times 3$ homography.

- Consider $M_{10}$, the matrix that arises from mapping a pixel in one image to a 3D point and then back onto the second image

$$\hat{x}_1 \sim \hat{P}_1 \hat{P}_0^{-1} \hat{x}_0 = M_{10} \hat{x}_0$$

- When the last row of the $\hat{P}_0$ matrix is replaced with a plane equation $\hat{n}_0 \cdot p + c_0$ and points are assumed to lie on this plane, i.e., their disparity is $d = 0$ we can ignore the last column of $M_{10}$ and also its last row, since we do not care about the final z-buffer depth

$$\hat{x}_1 \sim \hat{H}_{10} \hat{x}_0$$

- You will show this in an exercise

- More recent algorithms use robust methods such as RANSAC

- How do we align multiple images? Is there a problem?
Rotational Panoramas

- Assume the camera is doing **pure 3D rotation**
- The most common panoramic image stitching, e.g., when taking images of the Grand Canyon
Rotational Panoramas

- Assume the camera is doing **pure 3D rotation**
- The most common panoramic image stitching, e.g., when taking images of the Grand Canyon
- Assumes that all points are very far from the camera
Rotational Panoramas

- Assume the camera is doing **pure 3D rotation**
- The most common panoramic image stitching, e.g., when taking images of the Grand Canyon
- Assumes that all points are very far from the camera

\[ \tilde{x}_0 = (x_0, y_0, f_0) \quad \tilde{x}_1 = (x_1, y_1, f_1) \]
In this case simplified homography

\[ \hat{H}_{10} = K_1 R_1 R_0^{-1} K^{-1}_0 = K_1 R_{10} K^{-1}_0 \]

with \( K \) the camera intrinsic matrix assuming \( c_x = c_y = 0 \)
Rotational Panoramas

- In this case simplified homography
  \[
  \hat{H}_{10} = K_1 R_1 R_0^{-1} K_0^{-1} = K_1 R_{10} K_0^{-1}
  \]
  
  with \( K \) the camera intrinsic matrix assuming \( c_x = c_y = 0 \)

- This can be rewritten as

\[
\begin{bmatrix}
  x_1 \\
y_1 \\
1
\end{bmatrix}
\sim
\begin{bmatrix}
f_1 \\
f_1 \\
1
\end{bmatrix}
R_{10}
\begin{bmatrix}
f_0^{-1} \\
f_0^{-1} \\
1
\end{bmatrix}
\begin{bmatrix}
x_0 \\
y_0 \\
1
\end{bmatrix}
\]
Rotational Panoramas

- In this case simplified homography
  \[ \hat{H}_{10} = K_1 R_1 R_0^{-1} K_0^{-1} = K_1 R_{10} K_0^{-1} \]
  with \( K \) the camera intrinsic matrix assuming \( c_x = c_y = 0 \)

- This can be rewritten as

\[
\begin{bmatrix}
  x_1 \\
y_1 \\
1
\end{bmatrix} \sim \begin{bmatrix}
f_1 \\
f_1 \\
1
\end{bmatrix} R_{10} \begin{bmatrix}
f_0^{-1} \\
f_0^{-1} \\
1
\end{bmatrix} \begin{bmatrix}
x_0 \\
y_0 \\
1
\end{bmatrix}
\]

- Or more explicitly

\[
\begin{bmatrix}
x_1 \\
y_1 \\
f_1
\end{bmatrix} \sim R_{10} \begin{bmatrix}
x_0 \\
y_0 \\
f_0
\end{bmatrix}
\]
Rotational Panoramas

- In this case simplified homography
  \[
  \hat{H}_{10} = K_1 R_1 R_0^{-1} K_0^{-1} = K_1 R_{10} K_0^{-1}
  \]

  with \( K \) the camera intrinsic matrix assuming \( c_x = c_y = 0 \)

- This can be rewritten as

  \[
  \begin{bmatrix}
  x_1 \\
  y_1 \\
  1
  \end{bmatrix}
  \sim
  \begin{bmatrix}
  f_1 \\
  f_1 \\
  1
  \end{bmatrix}
  R_{10}
  \begin{bmatrix}
  f_0^{-1} \\
  f_0^{-1} \\
  1
  \end{bmatrix}
  \begin{bmatrix}
  x_0 \\
  y_0 \\
  1
  \end{bmatrix}
  \]

- Or more explicitly

  \[
  \begin{bmatrix}
  x_1 \\
  y_1 \\
  f_1
  \end{bmatrix}
  \sim
  R_{10}
  \begin{bmatrix}
  x_0 \\
  y_0 \\
  f_0
  \end{bmatrix}
  \]

- We have 3, 4 or 5 parameters depending if the focal length is known, fixed or variable
Rotational Panoramas

- In this case simplified homography

\[ \hat{H}_{10} = K_1 R_1 R_0^{-1} K_0^{-1} = K_1 R_{10} K_0^{-1} \]

with \( K \) the camera intrinsic matrix assuming \( c_x = c_y = 0 \)

- This can be rewritten as

\[
\begin{bmatrix}
  x_1 \\
  y_1 \\
  1
\end{bmatrix}
\sim
\begin{bmatrix}
  f_1 \\
  f_1 \\
  1
\end{bmatrix}
R_{10}
\begin{bmatrix}
  f_0^{-1} \\
  f_0^{-1} \\
  1
\end{bmatrix}
\begin{bmatrix}
  x_0 \\
  y_0 \\
  1
\end{bmatrix}
\]

- Or more explicitly

\[
\begin{bmatrix}
  x_1 \\
  y_1 \\
  f_1
\end{bmatrix}
\sim
R_{10}
\begin{bmatrix}
  x_0 \\
  y_0 \\
  f_0
\end{bmatrix}
\]

- We have 3, 4 or 5 parameters depending if the focal length is known, fixed or variable
Figure: Four images taken with a hand-held camera registered using a 3D rotation motion model (Szeliski and Shum 1997)
What if you want a 360 field of view?

[Source: N Snavely]
Cylindrical and Spherical Coordinates

- An alternative to using homographies or 3D motions to align images is to first warp the images into **cylindrical coordinates** and then use a **pure translational model** to align them.

- This only works if the images are all taken with a level camera or with a known tilt angle.
Cylindrical and Spherical Coordinates

- An alternative to using homographies or 3D motions to align images is to first warp the images into **cylindrical coordinates** and then use a **pure translational model** to align them.

- This only works if the images are all taken with a level camera or with a known tilt angle.

- Assume for now that the camera is in its canonical position, i.e., \( R = I \) and the optical axis is aligned with the z axis and the y axis is aligned vertically.
Cylindrical and Spherical Coordinates

- An alternative to using homographies or 3D motions to align images is to first warp the images into **cylindrical coordinates** and then use a **pure translational model** to align them.

- This only works if the images are all taken with a level camera or with a known tilt angle.

- Assume for now that the camera is in its canonical position, i.e., $\mathbf{R} = \mathbf{I}$ and the optical axis is aligned with the z axis and the y axis is aligned vertically.

- We wish to project this image onto a cylindrical surface of unit radius.
Cylindrical and Spherical Coordinates

- An alternative to using homographies or 3D motions to align images is to first warp the images into cylindrical coordinates and then use a pure translational model to align them.
- This only works if the images are all taken with a level camera or with a known tilt angle.
- Assume for now that the camera is in its canonical position, i.e., $R = I$ and the optical axis is aligned with the $z$ axis and the $y$ axis is aligned vertically.
- We wish to project this image onto a cylindrical surface of unit radius.
- Points on this surface are parameterized by an angle $\theta$ and a height $h$ with the 3D cylindrical given by $(\sin \theta, h, \cos \theta) \propto (x, y, f)$.
An alternative to using homographies or 3D motions to align images is to first warp the images into **cylindrical coordinates** and then use a **pure translational model** to align them.

This only works if the images are all taken with a level camera or with a known tilt angle.

Assume for now that the camera is in its canonical position, i.e., $\mathbf{R} = \mathbf{I}$ and the optical axis is aligned with the $z$ axis and the $y$ axis is aligned vertically.

We wish to project this image onto a cylindrical surface of unit radius.

Points on this surface are parameterized by an angle $\theta$ and a height $h$ with the 3D cylindrical given by $(\sin \theta, h, \cos \theta) \propto (x, y, f)$.

$$p = (X, Y, Z)$$

$$x = (\sin \theta, h, \cos \theta)$$
Cylindrical and Spherical Coordinates

We can compute the correspondence between **warped** and **mapped** coordinates

\[ x' = s\theta = s\tan^{-1} \frac{x}{f'}, \]
\[ y' = sh = s\frac{y}{\sqrt{x^2 + f'^2}}, \]
\[ x = f\tan\theta = f\tan \frac{x'}{s}, \]
\[ y = h\sqrt{x^2 + f'^2} = \frac{y'}{s} f\sqrt{1 + \tan^2 x'/s} = \frac{y'}{s} \sec \frac{x'}{s} \]
Cylindrical Panorama

- Cylindrical is used if the camera is level and we have only rotation around its vertical axis
- Then we only need to estimate a translation

Figure: A cylindrical panorama (Szeliski and Shum 1997)
Spherical Projection

- Map 3D point \((X,Y,Z)\) onto sphere
  \[
  (\tilde{x}, \tilde{y}, \tilde{z}) = \frac{1}{\sqrt{X^2 + Y^2 + Z^2}} (X, Y, Z)
  \]

- Convert to spherical coordinates
  \[
  (\sin \theta \cos \phi, \sin \phi, \cos \theta \cos \phi) = (\tilde{x}, \tilde{y}, \tilde{z})
  \]

- Convert to spherical image coordinates
  \[
  (\tilde{x}, \tilde{y}) = (s \theta, s \phi) + (\tilde{x}_c, \tilde{y}_c)
  \]
  - \(s\) defines size of the final image
    » often convenient to set \(s = \text{camera focal length in pixels}\)

Source: N Snavely

Raquel Urtasun (TTI-C)
Spherical Projection

\[
p = (X, Y, Z)
\]

\[
x' = s\theta = s \tan^{-1} \frac{x}{f},
\]

\[
y' = s\phi = s \tan^{-1} \frac{y}{\sqrt{x^2 + f^2}},
\]

while the inverse is given by

\[
x = f \tan \theta = f \tan \frac{x'}{s},
\]

\[
y = \sqrt{x^2 + f^2} \tan \phi = \tan \frac{y'}{s} f \sqrt{1 + \tan^2 \frac{x'}{s}} = f \tan \frac{y'}{s} \sec \frac{x'}{s}
\]
It is desirable if the global motion model is translation.

For a pure panning motion, if we convert two images to their cylindrical maps with known $f$, the relationship between them is a translation.

Similarly, we can map an image to its longitude/latitude spherical coordinates as well if $f$ is given.
Modeling Distortion with Panoramas

- Project point to normalized image coordinates:
  \[ x_n = \frac{x}{z} \]
  \[ y_n = \frac{y}{z} \]

- Apply radial distortion:
  \[ r^2 = x_n^2 + y_n^2 \]
  \[ x_d = x_n(1 + \kappa_1 r^2 + \kappa_2 r^4) \]
  \[ y_d = y_n(1 + \kappa_1 r^2 + \kappa_2 r^4) \]
Modeling Distorsion with Panoramas

- Project point to normalized image coordinates
  \[ x_n = \frac{x}{z} \]
  \[ y_n = \frac{y}{z} \]

- Apply radial distortion
  \[ r^2 = x_n^2 + y_n^2 \]
  \[ x_d = x_n(1 + \kappa_1 r^2 + \kappa_2 r^4) \]
  \[ y_d = y_n(1 + \kappa_1 r^2 + \kappa_2 r^4) \]

- Apply focal length and translate image center
  \[ x' = fx_d + x_c \]
  \[ y' = fy_d + y_c \]
Modeling Distortion with Panoramas

- Project point to normalized image coordinates
  \[ x_n = \frac{x}{z} \]
  \[ y_n = \frac{y}{z} \]

- Apply radial distorsion
  \[ r^2 = x_n^2 + y_n^2 \]
  \[ x_d = x_n(1 + \kappa_1 r^2 + \kappa_2 r^4) \]
  \[ y_d = y_n(1 + \kappa_1 r^2 + \kappa_2 r^4) \]

- Apply focal length and translate image center
  \[ x' = fx_d + x_c \]
  \[ y' = fy_d + y_c \]

To model lens distortion with panoramas, use above projection operation after projecting onto a sphere.
Modeling Distorsion with Panoramas

- Project point to normalized image coordinates
  \[ x_n = \frac{x}{z}, \quad y_n = \frac{y}{z} \]

- Apply radial distortion
  \[ r^2 = x_n^2 + y_n^2 \]
  \[ x_d = x_n(1 + \kappa_1 r^2 + \kappa_2 r^4) \]
  \[ y_d = y_n(1 + \kappa_1 r^2 + \kappa_2 r^4) \]

- Apply focal length and translate image center
  \[ x' = fx_d + x_c \]
  \[ y' = fy_d + y_c \]

- To model lens distortion with panoramas, use above projection operation after projecting onto a sphere

[Source: N. Snavely]
Aligning spherical images

Suppose we rotate the camera by $\theta$ about the vertical axis.

How does this change the spherical image?

[Source: N. Snavely]
Suppose we rotate the camera by $\theta$ about the vertical axis.

How does this change the spherical image?

This means that we can align spherical images by translation.

[Source: N. Snavely]
Assembling the panorama

- Stitch pairs together, blend, then crop

[Source: N. Snavely]
Problem: Drift

- Small errors accumulate over time

[Source: N. Snavely]
Solutions to Drift

- Add another copy of first image at the end, giving a constraint: $y_n = y_1$
- There are a bunch of ways to solve this problem
  - add displacement of $(y_1 - y_n)/(n - 1)$ to each image after the first
Solutions to Drift

- Add another copy of first image at the end, giving a constraint: \( y_n = y_1 \)
- There are a bunch of ways to solve this problem
  - add displacement of \( (y_1 - y_n)/(n - 1) \) to each image after the first
  - apply an affine warp: \( y' = y + ax \)
Solutions to Drift

- Add another copy of first image at the end, giving a constraint: $y_n = y_1$
- There are a bunch of ways to solve this problem
  - add displacement of $(y_1 - y_n)/(n - 1)$ to each image after the first
  - apply an affine warp: $y' = y + ax$
  - Bundle Adjustment: run a big optimization problem, incorporating this constraint

[Source: N. Snavely]
Solutions to Drift

- Add another copy of first image at the end, giving a constraint: \( y_n = y_1 \)
- There are a bunch of ways to solve this problem
  - add displacement of \( (y_1 - y_n) / (n - 1) \) to each image after the first
  - apply an affine warp: \( y' = y + ax \)
  - Bundle Adjustment: run a big optimization problem, incorporating this constraint

[Source: N. Snavely]
Dealing with multiple images

- Extend the pairwise matching criteria to deal with multiple images

- Typical pipeline include
  - **Panorama recognition**: Decide which images to align
  - **Global alignment**
  - **Local adjustments**
Bundle Adjustment

- **Goal**: Find a globally consistent set of alignment parameters that minimize the mis-registration between all pairs of images.

- The process of simultaneously adjusting pose parameters for a large collection of overlapping images is called **bundle adjustment**.
Bundle Adjustment

- **Goal**: Find a globally consistent set of alignment parameters that minimize the mis-registration between all pairs of images.

- The process of simultaneously adjusting pose parameters for a large collection of overlapping images is called **bundle adjustment**

- In the case of a single pair of images, we have feature-based alignment problem:

  \[
  E_{\text{pairwise-LS}} = \sum_i \| \mathbf{r}_i \|^2 = \| \mathbf{\tilde{x}}'_i (\mathbf{x}_i; \mathbf{p}) - \hat{\mathbf{x}}_i \|^2
  \]
Goal: Find a globally consistent set of alignment parameters that minimize the mis-registration between all pairs of images.

The process of simultaneously adjusting pose parameters for a large collection of overlapping images is called bundle adjustment.

In the case of a single pair of images, we have feature-based alignment problem

\[ E_{\text{pairwise-LS}} = \sum_i \|r_i\|^2 = \|\tilde{x}'_i(x_i; p) - \hat{x}_i\|^2 \]

For multi-alignment, instead of \( n \) correspondences \( \{x_i, \hat{x}_i\} \), we have \( n_{jk} \) correspondences for every pair of images.
Bundle Adjustment

- **Goal:** Find a globally consistent set of alignment parameters that minimize the mis-registration between all pairs of images.

- The process of simultaneously adjusting pose parameters for a large collection of overlapping images is called **bundle adjustment**.

- In the case of a single pair of images, we have feature-based alignment problem:

  \[
  E_{\text{pairwise-LS}} = \sum_i \|r_i\|_2^2 = \|\tilde{x}_i'(x_i; p) - \hat{x}_i\|_2^2
  \]

- For multi-alignment, instead of \( n \) correspondences \( \{x_i, \hat{x}_i'\} \), we have \( n_{jk} \) correspondences for every pair of images.

- We will look into the case of pose expressed by rotation.
Goal: Find a globally consistent set of alignment parameters that minimize the mis-registration between all pairs of images.

The process of simultaneously adjusting pose parameters for a large collection of overlapping images is called **bundle adjustment**.

In the case of a single pair of images, we have feature-based alignment problem:

\[
E_{\text{pairwise-LS}} = \sum_i \| r_i \|^2 = \| \tilde{x}'_i(x_i; p) - \hat{x}_i \|^2
\]

For multi-alignment, instead of \( n \) correspondences \( \{ x_i, \hat{x}'_i \} \), we have \( n_{jk} \) correspondences for every pair of images.

We will look into the case of pose expressed by rotation.

Look at (Szeliski and Shum, 97) for the case of homographies.
**Bundle Adjustment**

- **Goal**: Find a globally consistent set of alignment parameters that minimize the mis-registration between all pairs of images.

- The process of simultaneously adjusting pose parameters for a large collection of overlapping images is called **bundle adjustment**.

- In the case of a single pair of images, we have feature-based alignment problem:

\[
E_{\text{pairwise-LS}} = \sum_l \| \mathbf{r}_l \|^2 = \| \hat{x}'_i(\mathbf{x}_i; \mathbf{p}) - \hat{x}_i \|^2
\]

- For multi-alignment, instead of \( n \) correspondences \( \{ \mathbf{x}_i, \hat{x}'_i \} \), we have \( n_{jk} \) correspondences for every pair of images.

- We will look into the case of pose expressed by rotation.

- Look at (Szeliski and Shum, 97) for the case of homographies.
Bundle Adjustment

- We can relate a 3D point $\mathbf{x}_i$ into a point $\mathbf{x}_{ij}$ in frame $j$ as
  $$\tilde{\mathbf{x}}_{ij} \sim K_j R_j \mathbf{x}_i \quad \text{and} \quad \mathbf{x}_i \sim R_{j}^{-1} K_{j}^{-1} \tilde{\mathbf{x}}_{ij}$$
  with $K_j = \text{diag}(f_j, f_j, 1)$

- The motion mapping a point $\mathbf{x}_{ij}$ from frame $j$ into a point $\mathbf{x}_{ik}$ in frame $k$ is similarly given by
  $$\tilde{\mathbf{x}}_{ik} \sim \tilde{\mathbf{H}} \tilde{\mathbf{x}}_{ij} = K_k R_k R_{j}^{-1} K_{j}^{-1} \tilde{\mathbf{x}}_{ij}$$
Bundle Adjustment

- We can relate a 3D point $\mathbf{x}_i$ into a point $\mathbf{x}_{ij}$ in frame $j$ as
  \[ \tilde{\mathbf{x}}_{ij} \sim K_j R_j \mathbf{x}_i \quad \text{and} \quad \mathbf{x}_i \sim R_j^{-1} K_j^{-1} \tilde{\mathbf{x}}_{ij} \]
  with $K_j = \text{diag}(f_j, f_j, 1)$

- The motion mapping a point $\mathbf{x}_{ij}$ from frame $j$ into a point $\mathbf{x}_{ik}$ in frame $k$ is similarly given by
  \[ \tilde{\mathbf{x}}_{ik} \sim \tilde{\mathbf{H}} \tilde{\mathbf{x}}_{ij} = K_k R_k R_j^{-1} K_j^{-1} \tilde{\mathbf{x}}_{ij} \]

- Given an initial set of $\{(R_j, f_j)\}$ estimates obtained from chaining pairwise alignments, how do we refine these estimates?
Bundle Adjustment

- We can relate a 3D point $\mathbf{x}_i$ into a point $\mathbf{x}_{ij}$ in frame $j$ as
  \[ \tilde{x}_{ij} \sim K_j R_j x_i \quad \text{and} \quad x_i \sim R_j^{-1} K_j^{-1} \tilde{x}_{ij} \]
  with $K_j = \text{diag}(f_j, f_j, 1)$

- The motion mapping a point $\mathbf{x}_{ij}$ from frame $j$ into a point $\mathbf{x}_{ik}$ in frame $k$ is similarly given by
  \[ \tilde{x}_{ik} \sim \tilde{H} \tilde{x}_{ij} = K_k R_k R_j^{-1} K_j^{-1} \tilde{x}_{ij} \]

- Given an initial set of $\{(R_j, f_j)\}$ estimates obtained from chaining pairwise alignments, how do we refine these estimates?

- We can extend the pairwise energy to the multiview formulation
  \[ E_{all-pairs-2D} = \sum_i \sum_{jk} c_{ij} c_{ik} \| \tilde{x}_{ik}(\tilde{x}_{ij}', R_j, f_j, R_k, f_k) - \hat{x}_{ik} \|^2 \]
  with $\tilde{x}_{ij}'$ the predicted location of feature $i$ in frame $k$, $\hat{x}_{ij}$ observed location.
Bundle Adjustment

- We can relate a 3D point \( x_i \) into a point \( x_{ij} \) in frame \( j \) as
  \[
  \tilde{x}_{ij} \sim K_j R_j x_i \quad \text{and} \quad x_i \sim R_j^{-1} K_j^{-1} \tilde{x}_{ij}
  \]
  with \( K_j = \text{diag}(f_j, f_j, 1) \)

- The motion mapping a point \( x_{ij} \) from frame \( j \) into a point \( x_{ik} \) in frame \( k \) is similarly given by
  \[
  \tilde{x}_{ik} \sim \tilde{H} \tilde{x}_{ij} = K_k R_k R_j^{-1} K_j^{-1} \tilde{x}_{ij}
  \]

Given an initial set of \( \{(R_j, f_j)\} \) estimates obtained from chaining pairwise alignments, how do we refine these estimates?

- We can extend the pairwise energy to the multiview formulation
  \[
  E_{all\text{-pairs}-2D} = \sum_i \sum_{jk} c_{ij} c_{ik} \| \tilde{x}_{ik}(\tilde{x}'_{ij}; R_j, f_j, R_k, f_k) - \hat{x}_{ik} \|^2_2
  \]
  with \( \tilde{x}'_{ij} \) the predicted location of feature \( i \) in frame \( k \), \( \hat{x}_{ij} \) observed location.

- The 2D subscript indicates that we minimize the image-plane error.
Bundle Adjustment

- We can relate a 3D point $x_i$ into a point $x_{ij}$ in frame $j$ as
  \[ \tilde{x}_{ij} \sim K_j R_j x_i \quad \text{and} \quad x_i \sim R_j^{-1} K_j^{-1} \tilde{x}_{ij} \]
  with $K_j = \text{diag}(f_j, f_j, 1)$

- The motion mapping a point $x_{ij}$ from frame $j$ into a point $x_{ik}$ in frame $k$ is similarly given by
  \[ \tilde{x}_{ik} \sim \tilde{H} \tilde{x}_{ij} = K_k R_k R_j^{-1} K_j^{-1} \tilde{x}_{ij} \]

- Given an initial set of $\{(R_j, f_j)\}$ estimates obtained from chaining pairwise alignments, how do we refine these estimates?

- We can extend the pairwise energy to the multiview formulation
  \[ E_{all-pairs-2D} = \sum_i \sum_{jk} c_{ij} c_{ik} \| \tilde{x}_{ik}(\tilde{x}_{ij}', R_j, f_j, R_k, f_k) - \hat{x}_{ik} \|_2^2 \]
  with $\tilde{x}_{ij}'$ the predicted location of feature $i$ in frame $k$, $\hat{x}_{ij}$ observed location.

- The 2D subscript indicates that we minimize the image-plane error

- We can use non-linear least squares if we have enough features
We can relate a 3D point $\mathbf{x}_i$ into a point $\mathbf{x}_{ij}$ in frame $j$ as

$$\tilde{\mathbf{x}}_{ij} \sim K_j R_j x_i \quad \text{and} \quad x_i \sim R_j^{-1} K_j^{-1} \tilde{\mathbf{x}}_{ij}$$

with $K_j = \text{diag}(f_j, f_j, 1)$

The motion mapping a point $\mathbf{x}_{ij}$ from frame $j$ into a point $\mathbf{x}_{ik}$ in frame $k$ is similarly given by

$$\tilde{\mathbf{x}}_{ik} \sim \tilde{H} \tilde{\mathbf{x}}_{ij} = K_k R_k R_j^{-1} K_j^{-1} \tilde{\mathbf{x}}_{ij}$$

Given an initial set of $\{(R_j, f_j)\}$ estimates obtained from chaining pairwise alignments, how do we refine these estimates?

We can extend the pairwise energy to the multiview formulation

$$E_{all-pairs-2D} = \sum_i \sum_{jk} c_{ij} c_{ik} \| \tilde{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}'_{ij}; R_j, f_j, R_k, f_k) - \hat{\tilde{\mathbf{x}}}_{ik} \|^2$$

with $\tilde{\mathbf{x}}'_{ij}$ the predicted location of feature $i$ in frame $k$, $\hat{\tilde{\mathbf{x}}}_{ij}$ observed location.

The 2D subscript indicates that we minimize the image-plane error.

We can use non-linear least squares if we have enough features.
Problems

The multiview formulation

\[ E_{all-pairs-2D} = \sum_i \sum_{jk} c_{ij} c_{ik} \| \hat{x}_{ik}(\hat{x}'_{ij}, R_j, f_j, R_k, f_k) - \hat{x}_{ik} \|_2^2 \]

has two potential disadvantages:

- Since a summation is taken over all pairs with corresponding features, features that are observed many times are overweighted in the final solution (a feature observed \( m \) times gets counted \( \binom{m}{2} \) instead of \( m \) times).

- Second, the derivatives of \( \hat{x}_{ij} \) with respect to \{\( R_j, f_j \)\} are a little cumbersome
Use **true bundle adjustment** solving for pose \( \{R_j, f_j\} \) and 3D positions \( \{x_i\} \)

\[
E_{BA-2D} = \sum_i \sum_j c_{ij} ||\tilde{x}_{ij}(x_i; R_j, f_j) - \hat{x}_{ij}||_2^2
\]

The disadvantage is that there are more variables to solve for.
Alternative Formulation

- Use **true bundle adjustment** solving for pose \( \{ R_j, f_j \} \) and 3D positions \( \{ x_i \} \)

\[
E_{BA-2D} = \sum_i \sum_j c_{ij} \| \tilde{x}_{ij}(x_i; R_j, f_j) - \hat{x}_{ij} \|_2^2
\]

- The disadvantage is that there are more variables to solve for

- Another alternative is to minimize the error in 3D

\[
E_{BA-3D} = \sum_i \sum_j c_{ij} \| \tilde{x}_i(\hat{x}_{ij}; R_j, f_j) - x_i \|_2^2
\]

with \( \tilde{x}_i = R_j^{-1}K_j^{-1}x_{ij} \)
Alternative Formulation

- Use **true bundle adjustment** solving for pose \( \{R_j, f_j\} \) and 3D positions \( \{x_i\} \)

\[
E_{BA-2D} = \sum_i \sum_j c_{ij} \| \tilde{x}_{ij}(x_i; R_j, f_j) - \hat{x}_{ij} \|^2
\]

- The disadvantage is that there are more variables to solve for

- Another alternative is to minimize the error in 3D

\[
E_{BA-3D} = \sum_i \sum_j c_{ij} \| \tilde{x}_i(\tilde{x}_{ij}; R_j, f_j) - x_i \|^2
\]

with \( \tilde{x}_i = R_j^{-1}K_j^{-1}x_{ij} \)

- This has bias towards longer focal lengths since the angles between rays become smaller as \( f \) increases
Alternative Formulation

- Use **true bundle adjustment** solving for pose \( \{R_j, f_j\} \) and 3D positions \( \{x_i\} \)

\[
E_{BA-2D} = \sum_i \sum_j c_{ij} \| \hat{x}_{ij}(x_i; R_j, f_j) - \hat{x}_{ij} \|^2_2
\]

- The disadvantage is that there are more variables to solve for
- Another alternative is to minimize the error in 3D

\[
E_{BA-3D} = \sum_i \sum_j c_{ij} \| \tilde{x}_i(\hat{x}_{ij}; R_j, f_j) - x_i \|^2_2
\]

with \( \tilde{x}_i = R_j^{-1}K_j^{-1}x_{ij} \)

- This has bias towards longer focal lengths since the angles between rays become smaller as \( f \) increases
- We can eliminate the 3D rays \( x_i \) and derive a 3D pairwise energy

\[
E_{all-pairs-3D} = \sum_i \sum_{jk} c_{ij} c_{ik} \| \tilde{x}_i(\hat{x}_{ij}, R_j, f_j) - \tilde{x}_i(\hat{x}_{ik}, R_k, f_k) \|^2_2
\]

- This is the simplest
Alternative Formulation

- Use **true bundle adjustment** solving for pose \( \{R_j, f_j\} \) and 3D positions \( \{x_i\} \)

\[
E_{BA-2D} = \sum_i \sum_j c_{ij} \| \tilde{x}_{ij}(x_i; R_j, f_j) - \hat{x}_{ij}\|_2^2
\]

- The disadvantage is that there are more variables to solve for
- Another alternative is to minimize the error in 3D

\[
E_{BA-3D} = \sum_i \sum_j c_{ij} \| \tilde{x}_i(\hat{x}_{ij}; R_j, f_j) - x_i\|_2^2
\]

with \( \tilde{x}_i = R_j^{-1}K_j^{-1}x_{ij} \)

- This has bias towards longer focal lengths since the angles between rays become smaller as \( f \) increases
- We can eliminate the 3D rays \( x_i \) and derive a 3D pairwise energy

\[
E_{all-pairs-3D} = \sum_i \sum_{jk} c_{ij} c_{ik} \| \tilde{x}_i(\hat{x}_{ij}, R_j, f_j) - \tilde{x}_i(\hat{x}_{ik}, R_k, f_k)\|_2^2
\]

- This is the simplest
Unwrapping a sphere

Credit: JHT’s Planetary Pixel Emporium
Spherical panoramas

Microsoft Lobby: http://www.acm.org/pubs/citations/proceedings/graph/258734/p251-szeliski
Different projections are possible

[Source: N. Snavely]
We want to seamlessly blend them together

[Source: N. Snavely]
Blending

- We want to seamlessly blend them together

[Source: N. Snavely]
Image Blending

[Source: N. Snavely]
Feathering

Take the average value at each pixel

[Source: N. Snavely]
Effect of window size

Use window to do average

[Source: N. Snavely]
Effect of window size

Use window to do average

[Source: N. Snavely]
Good window size

- Optimal window: smooth but not ghosted
- It doesn’t always work

[Source: N. Snavely]
Pyramid Blending

Create a Laplacian pyramid, blend each level


[Source: N. Snavely]
Laplacian Pyramid

\[ L_i = G_i - \text{expand}(G_{i+1}) \]

\[ G_i = L_i + \text{expand}(G_{i+1}) \]

Gaussian Pyramid

Laplacian Pyramid

\[ L_n = G_n \]

\[ L_2 \]

\[ L_1 \]

\[ L_0 \]

Source: N. Snavely
Encoding blend weights: \( I(x,y) = (\alpha R, \alpha G, \alpha B, \alpha) \)

\[
\text{color at } p = \frac{(\alpha_1 R_1, \alpha_1 G_1, \alpha_1 B_1) + (\alpha_2 R_2, \alpha_2 G_2, \alpha_2 B_2) + (\alpha_3 R_3, \alpha_3 G_3, \alpha_3 B_3)}{\alpha_1 + \alpha_2 + \alpha_3}
\]

Implement this in two steps:

1. accumulate: add up the (\(\alpha\) premultiplied) RGB\(\alpha\) values at each pixel
2. normalize: divide each pixel’s accumulated RGB by its \(\alpha\) value

Q: what if \(\alpha = 0\)?
Gradient domain reconstruction can be used to do object insertion in image editing applications.

Figure: Perez et al. SIGGRAPH 2003
Panorama Examples

- Every image on Google Streetview

[Source: N. Snavely]
Ghost Removal

Figure: Uyttendaele et al. ICCV01

[Source: N. Snavely]
Ghost Removal

Figure: Uyttendaele et al. ICCV01

[Source: N. Snavely]
Other Types

- Can mosaic onto any surface if you know the geometry
- See NASAs Visible Earth project for some stunning earth mosaics

[Source: N. Snavely]