Computer Vision: Image Alignment

Raquel Urtasun

TTI Chicago

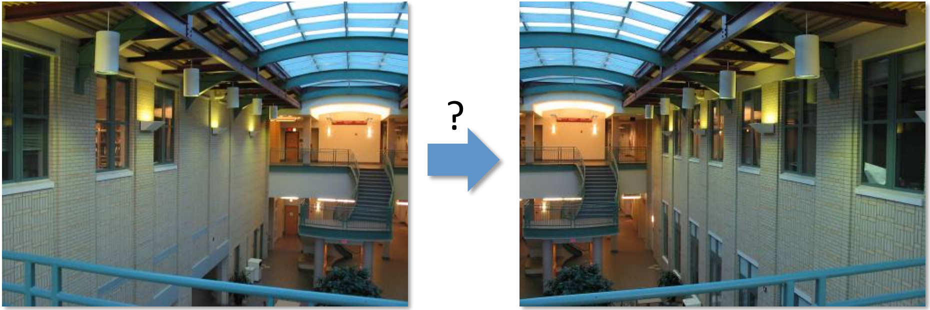
Jan 24, 2013

Readings

- Chapter 2.1, 3.6, 4.3 and 6.1 of Szeliski's book
- Chapter 1 of Forsyth & Ponce

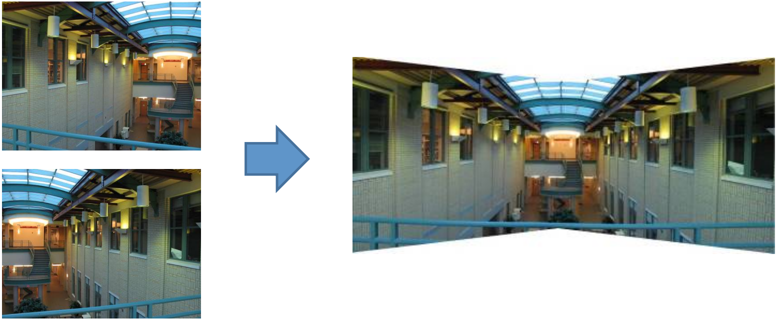
What did we see in class last week?

What is the geometric relationship between these images?



[Source: N. Snavely]

What is the geometric relationship between these images?



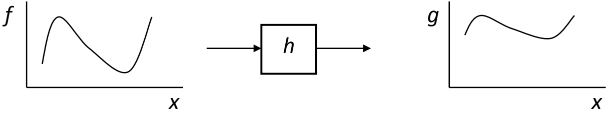
Very important for creating mosaics!

[Source: N. Snavely]

Image Warping

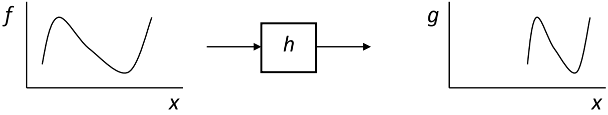
- **Image filtering**: change *range* of image

$$g(x) = h(f(x))$$

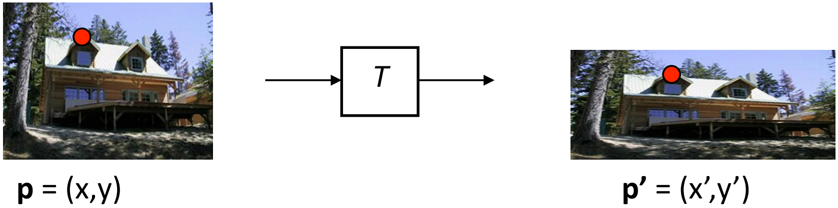


- **Image warping**: change *domain* of image

$$g(x) = f(h(x))$$



[Source: R. Szeliski]



- Transformation T is a coordinate-changing machine:

$$p' = T(p)$$

- What does it mean that T is global?
    - Is the same for any point p
    - Can be described by just a few numbers (parameters)

[Source: N. Snavely]

Forward and Inverse Warping

- **Forward Warping**: Send each pixel $f(x)$ to its corresponding location $(x', y') = T(x, y)$ in $g(x', y')$

> **procedure** *forwardWarp*$(f, \boldsymbol{h}, \textbf{out } g)$:
>
> For every pixel $\boldsymbol{x}$ in $f(\boldsymbol{x})$
>
> 1. Compute the destination location $\boldsymbol{x}' = \boldsymbol{h}(\boldsymbol{x})$.
> 2. Copy the pixel $f(\boldsymbol{x})$ to $g(\boldsymbol{x}')$.

- **Inverse Warping:** Each pixel at the destination is sampled from the original image

> **procedure** *inverseWarp*$(f, \boldsymbol{h}, \textbf{out } g)$:
>
> For every pixel $\boldsymbol{x}'$ in $g(\boldsymbol{x}')$
>
> 1. Compute the source location $\boldsymbol{x} = \hat{\boldsymbol{h}}(\boldsymbol{x}')$
> 2. Resample $f(\boldsymbol{x})$ at location $\boldsymbol{x}$ and copy to $g(\boldsymbol{x}')$

All 2D Linear Transformations

Linear transformations are combinations of

- Scale,
- Rotation
- Shear
- Mirror

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

[Source: N. Snavely]

All 2D Linear Transformations

Properties of linear transformations:

- Origin maps to origin
- Lines map to lines

All 2D Linear Transformations

Properties of linear transformations:

- Origin maps to origin
- Lines map to lines
- Parallel lines remain parallel

All 2D Linear Transformations

Properties of linear transformations:

- Origin maps to origin
- Lines map to lines
- Parallel lines remain parallel
- Ratios are preserved

All 2D Linear Transformations

Properties of linear transformations:

- Origin maps to origin
- Lines map to lines
- Parallel lines remain parallel
- Ratios are preserved
- Closed under composition

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} \begin{bmatrix} i & j \\ k & l \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

All 2D Linear Transformations

Properties of linear transformations:

- Origin maps to origin
- Lines map to lines
- Parallel lines remain parallel
- Ratios are preserved
- Closed under composition

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} \begin{bmatrix} i & j \\ k & l \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

What about the translation?

[Source: N. Snavely]

All 2D Linear Transformations

Properties of linear transformations:

- Origin maps to origin
- Lines map to lines
- Parallel lines remain parallel
- Ratios are preserved
- Closed under composition

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} \begin{bmatrix} i & j \\ k & l \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

What about the translation?

[Source: N. Snavely]

Affine Transformations

Affine transformations are combinations of

- Linear transformations, and
- Translations

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Properties of affine transformations:

- Origin does not necessarily map to origin
- Lines map to lines
- Parallel lines remain parallel
- Ratios are preserved
- Closed under composition

[Source: N. Snavely]

Projective Transformations

- Affine transformations and Projective warps

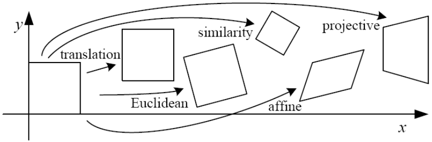
$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Properties of affine transformations:

- Origin does not necessarily map to origin
- Lines map to lines
- Parallel lines do not necessarily remain parallel
- Ratios are not preserved
- Closed under composition

[Source: N. Snavely]

2D Image Tranformations



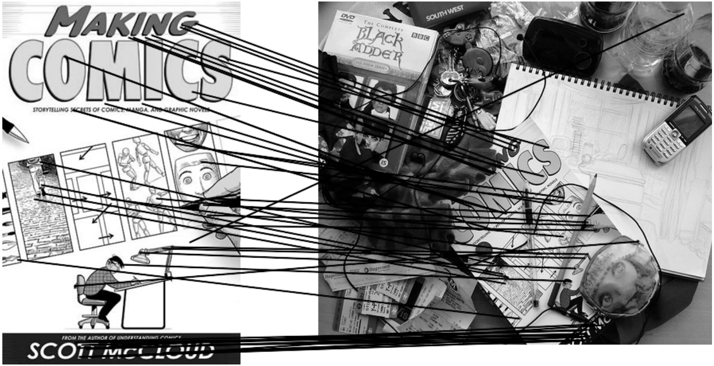
Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	3	lengths	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	4	angles	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times3}$	6	parallelism	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times3}$	8	straight lines	

- These transformations are a nested set of groups
- Closed under composition and inverse is a member

Computing transformations

Given a set of matches between images A and B

- How can we compute the transform T from A to B?
- Find transform T that best agrees with the matches



[Source: N. Snavely]

Least squares formulation

- For each point (x_i, y_i) we have

$$\begin{aligned} x_i + x_t &= x_i' \\ y_i + y_t &= y_i' \end{aligned}$$

- We define the residuals as

$$\begin{aligned} r_{x_i}(x_t) &= x_i + x_t - x_i' \\ r_{y_i}(y_t) &= y_i + y_t - y_i' \end{aligned}$$

Least squares formulation

- For each point (x_i, y_i) we have

$$\begin{aligned} x_i + x_t &= x_i' \\ y_i + y_t &= y_i' \end{aligned}$$

- We define the residuals as

$$\begin{aligned} r_{x_i}(x_t) &= x_i + x_t - x_i' \\ r_{y_i}(y_t) &= y_i + y_t - y_i' \end{aligned}$$

- **Goal:** minimize sum of squared residuals

$$C(x_t, y_t) = \sum_{i=1}^{n} (r_{x_i}(x_t)^2 + r_{y_i}(y_t)^2)$$

Least squares formulation

- For each point (x_i, y_i) we have

$$\begin{aligned} x_i + x_t &= x_i' \\ y_i + y_t &= y_i' \end{aligned}$$

- We define the residuals as

$$\begin{aligned} r_{x_i}(x_t) &= x_i + x_t - x_i' \\ r_{y_i}(y_t) &= y_i + y_t - y_i' \end{aligned}$$

- **Goal:** minimize sum of squared residuals

$$C(x_t, y_t) = \sum_{i=1}^{n} (r_{x_i}(x_t)^2 + r_{y_i}(y_t)^2)$$

- The solution is called the **least squares** solution

Least squares formulation

- For each point (x_i, y_i) we have

$$\begin{aligned} x_i + x_t &= x_i' \\ y_i + y_t &= y_i' \end{aligned}$$

- We define the residuals as

$$\begin{aligned} r_{x_i}(x_t) &= x_i + x_t - x_i' \\ r_{y_i}(y_t) &= y_i + y_t - y_i' \end{aligned}$$

- **Goal:** minimize sum of squared residuals

$$C(x_t, y_t) = \sum_{i=1}^{n} (r_{x_i}(x_t)^2 + r_{y_i}(y_t)^2)$$

- The solution is called the **least squares** solution
- For translations, is equal to mean displacement

Least squares formulation

- For each point (x_i, y_i) we have

$$\begin{aligned} x_i + x_t &= x_i' \\ y_i + y_t &= y_i' \end{aligned}$$

- We define the residuals as

$$\begin{aligned} r_{x_i}(x_t) &= x_i + x_t - x_i' \\ r_{y_i}(y_t) &= y_i + y_t - y_i' \end{aligned}$$

- **Goal:** minimize sum of squared residuals

$$C(x_t, y_t) = \sum_{i=1}^{n} (r_{x_i}(x_t)^2 + r_{y_i}(y_t)^2)$$

- The solution is called the **least squares** solution
- For translations, is equal to mean displacement

[Source: N. Snavely]

Least squares formulation

- For each point (x_i, y_i) we have

$$\begin{aligned} x_i + x_t &= x_i' \\ y_i + y_t &= y_i' \end{aligned}$$

- We define the residuals as

$$\begin{aligned} r_{x_i}(x_t) &= x_i + x_t - x_i' \\ r_{y_i}(y_t) &= y_i + y_t - y_i' \end{aligned}$$

- **Goal:** minimize sum of squared residuals

$$C(x_t, y_t) = \sum_{i=1}^{n} (r_{x_i}(x_t)^2 + r_{y_i}(y_t)^2)$$

- The solution is called the **least squares** solution
- For translations, is equal to mean displacement

[Source: N. Snavely]

Matrix Formulation

- We can also write as a matrix equation

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ & \vdots \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{bmatrix} x'_1 - x_1 \\ y'_1 - y_1 \\ x'_2 - x_2 \\ y'_2 - y_2 \\ \vdots \\ x'_n - x_n \\ y'_n - y_n \end{bmatrix}$$

$$\underset{2n \times 2}{\mathbf{A}} \quad \underset{2 \times 1}{\mathbf{t}} = \underset{2n \times 1}{\mathbf{b}}$$

- Solve for **t** by looking at the fixed-point equation

Affine Transformations

When we are dealing with an affine transformation

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- How many unknowns?

Affine Transformations

When we are dealing with an affine transformation

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- How many unknowns?
- How many equations per match?

Affine Transformations

When we are dealing with an affine transformation

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- How many unknowns?
- How many equations per match?
- How many matches do we need?

Affine Transformations

When we are dealing with an affine transformation

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- How many unknowns?
- How many equations per match?
- How many matches do we need?
- Why to use more?

[Source: N. Snavely]

Affine Transformations

When we are dealing with an affine transformation

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- How many unknowns?
- How many equations per match?
- How many matches do we need?
- Why to use more?

[Source: N. Snavely]

Affine Transformation Cost Function

- We can write the residuals as

$$\begin{aligned} r_{x_i}(a,b,c,d,e,f) &= (ax_i + by_i + c) - x'_i \\ r_{y_i}(a,b,c,d,e,f) &= (dx_i + ey_i + f) - y'_i \end{aligned}$$

- Cost function

$$C(a,b,c,d,e,f) = \sum_{i=1}^{N} \left(r_{x_i}(a,b,c,d,e,f)^2 + r_{y_i}(a,b,c,d,e,f)^2 \right)$$

Affine Transformation Cost Function

- We can write the residuals as

$$\begin{aligned} r_{x_i}(a,b,c,d,e,f) &= (ax_i + by_i + c) - x'_i \\ r_{y_i}(a,b,c,d,e,f) &= (dx_i + ey_i + f) - y'_i \end{aligned}$$

- Cost function

$$C(a,b,c,d,e,f) = \sum_{i=1}^{N} \left(r_{x_i}(a,b,c,d,e,f)^2 + r_{y_i}(a,b,c,d,e,f)^2 \right)$$

- And in matrix form ...

[Source: N. Snavely]

Affine Transformation Cost Function

- We can write the residuals as

$$\begin{aligned} r_{x_i}(a,b,c,d,e,f) &= (ax_i + by_i + c) - x'_i \\ r_{y_i}(a,b,c,d,e,f) &= (dx_i + ey_i + f) - y'_i \end{aligned}$$

- Cost function

$$C(a,b,c,d,e,f) = \sum_{i=1}^{N} \left(r_{x_i}(a,b,c,d,e,f)^2 + r_{y_i}(a,b,c,d,e,f)^2 \right)$$

- And in matrix form ...

[Source: N. Snavely]

Matrix form

$$
\begin{bmatrix}
x_1 & y_1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & x_1 & y_1 & 1 \\
x_2 & y_2 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & x_2 & y_2 & 1 \\
 & & & \vdots & & \\
x_n & y_n & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & x_n & y_n & 1
\end{bmatrix}
\begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix}
=
\begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \\ x'_n \\ y'_n \end{bmatrix}
$$

$$
\underset{2n \times 6}{\mathbf{A}} \quad \underset{6 \times 1}{\mathbf{t}} = \underset{2n \times 1}{\mathbf{b}}
$$

[Source: N. Snavely]

General Formulation

- Let $x' = f(x; p)$ be a parametric transformation
- In the case of translation, similarity and affine, there is a linear relationship between the amount of motion $\Delta x = x' - x$ and the unknown parameters

$$\Delta x = x' - x = \mathbf{J}(x)\mathbf{p}$$

with $\mathbf{J} = \frac{\partial f}{\partial p}$ is the **Jacobian** of the transformation $\mathbf{f}$ with respect to the motion parameters $\mathbf{p}$

General Formulation

- Let $x' = f(x; p)$ be a parametric transformation
- In the case of translation, similarity and affine, there is a linear relationship between the amount of motion $\Delta x = x' - x$ and the unknown parameters

$$\Delta x = x' - x = \mathbf{J}(x)\mathbf{p}$$

with $\mathbf{J} = \frac{\partial f}{\partial p}$ is the **Jacobian** of the transformation **f** with respect to the motion parameters **p**

General Formulation

Transform	Matrix	Parameters p	Jacobian J
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, θ)	$\begin{bmatrix} 1 & 0 & -s_\theta x - c_\theta y \\ 0 & 1 & c_\theta x - s_\theta y \end{bmatrix}$
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	$\begin{bmatrix} 1+a_{00} & a_{01} & t_x \\ a_{10} & 1+a_{11} & t_y \end{bmatrix}$	$(t_x, t_y, a_{00}, a_{01}, a_{10}, a_{11})$	$\begin{bmatrix} 1 & 0 & x & y & 0 & 0 \\ 0 & 1 & 0 & 0 & x & y \end{bmatrix}$

- Let's do a couple on the board!

General Formulation

- The sum of square residuals is then

$$
\begin{aligned}
E_{LLS} &= \sum_i ||\mathbf{J}(\mathbf{x}_i)\mathbf{p} - \Delta\mathbf{x}_i||_2^2 \\
&= \mathbf{p}^T[\sum_i \mathbf{J}^T(\mathbf{x}_i)\mathbf{J}(\mathbf{x}_i)]\mathbf{p} - 2\mathbf{p}^T[\sum_i \mathbf{J}^T(\mathbf{x}_i)\Delta\mathbf{x}_i)] + \sum_i ||\Delta\mathbf{x}_i||_2
\end{aligned}
$$

General Formulation

- The sum of square residuals is then

$$\begin{aligned} E_{LLS} &= \sum_i ||\mathbf{J}(\mathbf{x}_i)\mathbf{p} - \Delta\mathbf{x}_i||_2^2 \\ &= \mathbf{p}^T[\sum_i \mathbf{J}^T(\mathbf{x}_i)\mathbf{J}(\mathbf{x}_i)]\mathbf{p} - 2\mathbf{p}^T[\sum_i \mathbf{J}^T(\mathbf{x}_i)\Delta\mathbf{x}_i)] + \sum_i ||\Delta\mathbf{x}_i||_2 \\ &= \mathbf{p}^T\mathbf{A}\mathbf{p} - 2\mathbf{p}^T\mathbf{b} + c \end{aligned}$$

General Formulation

- The sum of square residuals is then

$$
\begin{aligned}
E_{LLS} &= \sum_i ||\mathbf{J}(\mathbf{x}_i)\mathbf{p} - \Delta\mathbf{x}_i||_2^2 \\
&= \mathbf{p}^T[\sum_i \mathbf{J}^T(\mathbf{x}_i)\mathbf{J}(\mathbf{x}_i)]\mathbf{p} - 2\mathbf{p}^T[\sum_i \mathbf{J}^T(\mathbf{x}_i)\Delta\mathbf{x}_i)] + \sum_i ||\Delta\mathbf{x}_i||_2 \\
&= \mathbf{p}^T\mathbf{A}\mathbf{p} - 2\mathbf{p}^T\mathbf{b} + c
\end{aligned}
$$

- We can compute the solution by looking for a fixed point, yielding

$$\mathbf{A}\mathbf{p} = \mathbf{b}$$

with $\mathbf{A} = \sum_i \mathbf{J}^T(\mathbf{x}_i)\mathbf{J}(\mathbf{x}_i)$ the **Hessian** and $\mathbf{b} = \sum_i \mathbf{J}^T(\mathbf{x}_i)\Delta\mathbf{x}_i$

General Formulation

- The sum of square residuals is then

$$
\begin{aligned}
E_{LLS} &= \sum_i ||\mathbf{J}(\mathbf{x}_i)\mathbf{p} - \Delta\mathbf{x}_i||_2^2 \\
&= \mathbf{p}^T[\sum_i \mathbf{J}^T(\mathbf{x}_i)\mathbf{J}(\mathbf{x}_i)]\mathbf{p} - 2\mathbf{p}^T[\sum_i \mathbf{J}^T(\mathbf{x}_i)\Delta\mathbf{x}_i)] + \sum_i ||\Delta\mathbf{x}_i||_2 \\
&= \mathbf{p}^T\mathbf{A}\mathbf{p} - 2\mathbf{p}^T\mathbf{b} + c
\end{aligned}
$$

- We can compute the solution by looking for a fixed point, yielding

$$\mathbf{A}\mathbf{p} = \mathbf{b}$$

with $\mathbf{A} = \sum_i \mathbf{J}^T(\mathbf{x}_i)\mathbf{J}(\mathbf{x}_i)$ the **Hessian** and $\mathbf{b} = \sum_i \mathbf{J}^T(\mathbf{x}_i)\Delta\mathbf{x}_i$

Uncertainty Weighting

- The above solution assumes that all feature points are matched with same accuracy.
- If we associate a scalar variance σ_i^2 with each correspondence, we can minimize the **weighted least squares** problem

$$E_{WLS} = \sum_i \sigma_i^{-2} ||\mathbf{r}_i||_2^2$$

Uncertainty Weighting

- The above solution assumes that all feature points are matched with same accuracy.
- If we associate a scalar variance σ_i^2 with each correspondence, we can minimize the **weighted least squares** problem

$$E_{WLS} = \sum_i \sigma_i^{-2} ||\mathbf{r}_i||_2^2$$

- If the σ_i^2 are fixed, then the solution is simply

$$\mathbf{p} = (\Sigma^T \mathbf{A}^T \mathbf{A} \Sigma)^{-1} \Sigma^T \mathbf{A} \mathbf{b}$$

with Σ, the matrix containing for each observation the noise level

Uncertainty Weighting

- The above solution assumes that all feature points are matched with same accuracy.
- If we associate a scalar variance σ_i^2 with each correspondence, we can minimize the **weighted least squares** problem

$$E_{WLS} = \sum_i \sigma_i^{-2} ||\mathbf{r}_i||_2^2$$

- If the σ_i^2 are fixed, then the solution is simply

$$\mathbf{p} = (\Sigma^T \mathbf{A}^T \mathbf{A} \Sigma)^{-1} \Sigma^T \mathbf{A} \mathbf{b}$$

with Σ, the matrix containing for each observation the noise level

- What if we don't know Σ?

Uncertainty Weighting

- The above solution assumes that all feature points are matched with same accuracy.
- If we associate a scalar variance σ_i^2 with each correspondence, we can minimize the **weighted least squares** problem

$$E_{WLS} = \sum_i \sigma_i^{-2} ||\mathbf{r}_i||_2^2$$

- If the σ_i^2 are fixed, then the solution is simply

$$\mathbf{p} = (\Sigma^T \mathbf{A}^T \mathbf{A} \Sigma)^{-1} \Sigma^T \mathbf{A} \mathbf{b}$$

  with Σ, the matrix containing for each observation the noise level
- What if we don't know Σ?
- Solve using **iteratively reweighted least squares (IRLS)**

Uncertainty Weighting

- The above solution assumes that all feature points are matched with same accuracy.
- If we associate a scalar variance σ_i^2 with each correspondence, we can minimize the **weighted least squares** problem

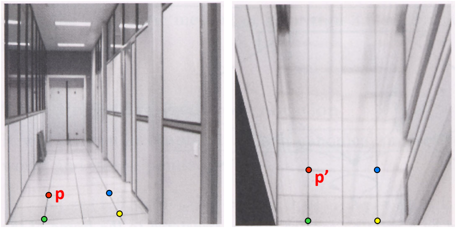
$$E_{WLS} = \sum_i \sigma_i^{-2} ||\mathbf{r}_i||_2^2$$

- If the σ_i^2 are fixed, then the solution is simply

$$\mathbf{p} = (\Sigma^T \mathbf{A}^T \mathbf{A} \Sigma)^{-1} \Sigma^T \mathbf{A} \mathbf{b}$$

  with Σ, the matrix containing for each observation the noise level
- What if we don't know Σ?
- Solve using **iteratively reweighted least squares (IRLS)**

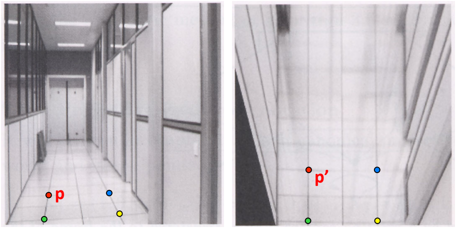
Homographies



To unwarp (rectify) and image

- solve for homography H given p and p'

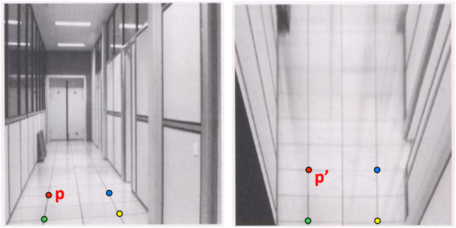
Homographies



To unwarp (rectify) and image

- solve for homography H given p and p'
- solve equations of the form: $\mathbf{wp}' = \mathbf{Hp}$

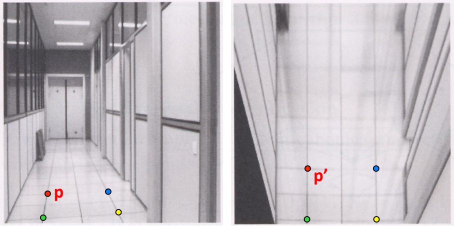
Homographies



To unwarp (rectify) and image

- solve for homography H given p and p'
- solve equations of the form: $\mathbf{wp}' = \mathbf{Hp}$
  - linear in unknowns: $\mathbf{w}$ and coefficients of $\mathbf{H}$

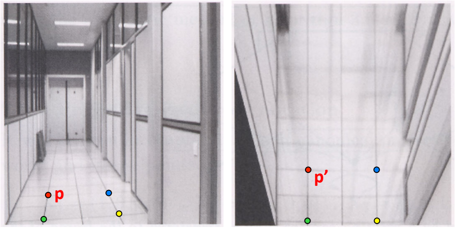
Homographies



To unwarp (rectify) and image

- solve for homography H given p and p'
- solve equations of the form: $\mathbf{w}\mathbf{p}' = \mathbf{H}\mathbf{p}$
  - linear in unknowns: $\mathbf{w}$ and coefficients of $\mathbf{H}$
  - $\mathbf{H}$ is defined up to an arbitrary scale factor

Homographies

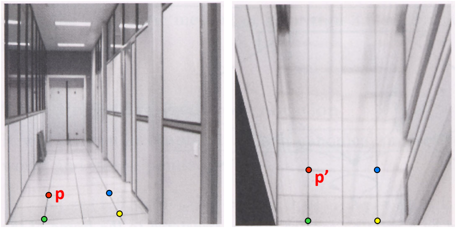


To unwarp (rectify) and image

- solve for homography H given p and p'
- solve equations of the form: $\mathbf{w}\mathbf{p}' = \mathbf{H}\mathbf{p}$
  - linear in unknowns: $\mathbf{w}$ and coefficients of $\mathbf{H}$
  - $\mathbf{H}$ is defined up to an arbitrary scale factor
  - how many points are necessary to solve for $\mathbf{H}$?

[Source: N. Snavely]

Homographies



To unwarp (rectify) and image

- solve for homography H given p and p'
- solve equations of the form: $\mathbf{wp}' = \mathbf{Hp}$
  - linear in unknowns: $\mathbf{w}$ and coefficients of $\mathbf{H}$
  - $\mathbf{H}$ is defined up to an arbitrary scale factor
  - how many points are necessary to solve for $\mathbf{H}$?

[Source: N. Snavely]

Solving for Homographies

$$\begin{bmatrix} ax'_i \\ ay'_i \\ a \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

- To get to non-homogenous coordinates

$$x'_i = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

$$y'_i = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

Solving for Homographies

$$\begin{bmatrix} ax_i' \\ ay_i' \\ a \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

- To get to non-homogenous coordinates

$$x_i' = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

$$y_i' = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

- **Warning**: This is non-linear!!!

Solving for Homographies

$$\begin{bmatrix} ax'_i \\ ay'_i \\ a \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

- To get to non-homogenous coordinates

$$\begin{aligned} x'_i &= \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}} \\ y'_i &= \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}} \end{aligned}$$

- **Warning**: This is non-linear!!!
- But wait a minute!

$$\begin{aligned} x'_i(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{00}x_i + h_{01}y_i + h_{02} \\ y'_i(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{10}x_i + h_{11}y_i + h_{12} \end{aligned}$$

Solving for Homographies

$$\begin{bmatrix} ax'_i \\ ay'_i \\ a \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

- To get to non-homogenous coordinates

$$\begin{aligned} x'_i &= \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}} \\ y'_i &= \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}} \end{aligned}$$

- **Warning**: This is non-linear!!!
- But wait a minute!

$$\begin{aligned} x'_i\left(h_{20}x_i + h_{21}y_i + h_{22}\right) &= h_{00}x_i + h_{01}y_i + h_{02} \\ y'_i\left(h_{20}x_i + h_{21}y_i + h_{22}\right) &= h_{10}x_i + h_{11}y_i + h_{12} \end{aligned}$$

Solving for homographies

$$\begin{aligned} x_i' \left(h_{20}x_i + h_{21}y_i + h_{22}\right) &= h_{00}x_i + h_{01}y_i + h_{02} \\ y_i' \left(h_{20}x_i + h_{21}y_i + h_{22}\right) &= h_{10}x_i + h_{11}y_i + h_{12} \end{aligned}$$

- This is still linear in the unknowns

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_i'x_i & -x_i'y_i & -x_i' \\ 0 & 0 & 0 & x_i & y_i & 1 & -y_i'x_i & -y_i'y_i & -y_i' \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Solving for homographies

- Taking all the observations into account

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ & & & & \vdots & & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix}}_{\mathbf{A}\ (2n \times 9)} \underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\mathbf{h}\ (9)} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_{\mathbf{0}\ (2n)}$$

- Defines a least squares problem:

$$\min_{\mathbf{h}} ||\mathbf{Ah}||_2^2$$

Solving for homographies

- Taking all the observations into account

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1 x_1 & -x'_1 y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1 x_1 & -y'_1 y_1 & -y'_1 \\ & & & & \vdots & & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_n x_n & -x'_n y_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_n x_n & -y'_n y_n & -y'_n \end{bmatrix}}_{\substack{\mathbf{A} \\ 2n \times 9}} \underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\substack{\mathbf{h} \\ 9}} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_{\substack{\mathbf{0} \\ 2n}}$$

- Defines a least squares problem:

$$\min_{\mathbf{h}} ||\mathbf{Ah}||_2^2$$

- Since $\mathbf{h}$ is only defined up to scale, solve for unit vector

Solving for homographies

- Taking all the observations into account

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1 x_1 & -x'_1 y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1 x_1 & -y'_1 y_1 & -y'_1 \\ & & & & \vdots & & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_n x_n & -x'_n y_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_n x_n & -y'_n y_n & -y'_n \end{bmatrix}}_{\mathbf{A}_{2n \times 9}} \underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\mathbf{h}_9} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_{\mathbf{0}_{2n}}$$

- Defines a least squares problem:

$$\min_{\mathbf{h}} ||\mathbf{A}\mathbf{h}||_2^2$$

- Since **h** is only defined up to scale, solve for unit vector
- Solution: $\hat{\mathbf{h}}$ = eigenvector of $\mathbf{A}^T\mathbf{A}$ with smallest eigenvalue

Solving for homographies

- Taking all the observations into account

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1 x_1 & -x'_1 y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1 x_1 & -y'_1 y_1 & -y'_1 \\ & & & & \vdots & & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_n x_n & -x'_n y_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_n x_n & -y'_n y_n & -y'_n \end{bmatrix}}_{\mathbf{A}\ (2n \times 9)} \underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\mathbf{h}\ (9)} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_{\mathbf{0}\ (2n)}$$

- Defines a least squares problem:

$$\min_{\mathbf{h}} ||\mathbf{Ah}||_2^2$$

- Since $\mathbf{h}$ is only defined up to scale, solve for unit vector
- Solution: $\hat{\mathbf{h}}$ = eigenvector of $\mathbf{A}^T\mathbf{A}$ with smallest eigenvalue
- Works with 4 or more points

Solving for homographies

- Taking all the observations into account

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1 x_1 & -x'_1 y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1 x_1 & -y'_1 y_1 & -y'_1 \\ & & & & \vdots & & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_n x_n & -x'_n y_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_n x_n & -y'_n y_n & -y'_n \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

$$\underset{2n \times 9}{\mathbf{A}} \qquad \underset{9}{\mathbf{h}} \qquad \underset{2n}{\mathbf{0}}$$

- Defines a least squares problem:

$$\min_{\mathbf{h}} ||\mathbf{A}\mathbf{h}||_2^2$$

- Since $\mathbf{h}$ is only defined up to scale, solve for unit vector
- Solution: $\hat{\mathbf{h}}$ = eigenvector of $\mathbf{A}^T\mathbf{A}$ with smallest eigenvalue
- Works with 4 or more points

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]

Image Alignment Algorithm

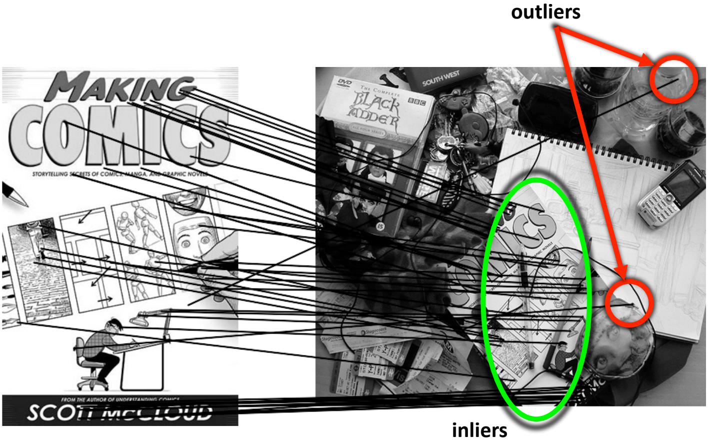
Given images A and B

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]

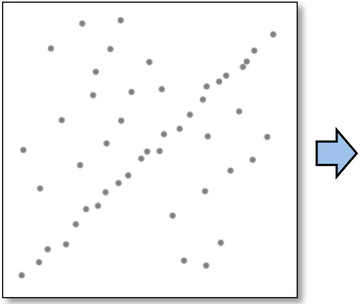
Robustness



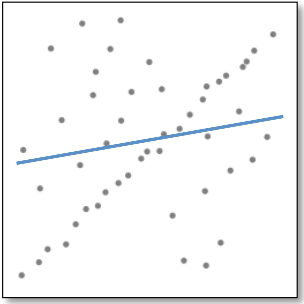
[Source: N. Snavely]

Simple case

- Lets consider a simpler example ... linear regression



Problem: Fit a line to these datapoints

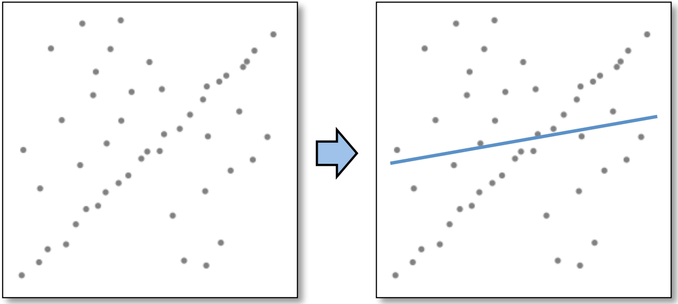


Least squares fit

- How can we fix this?

Simple case

- Lets consider a simpler example ... linear regression



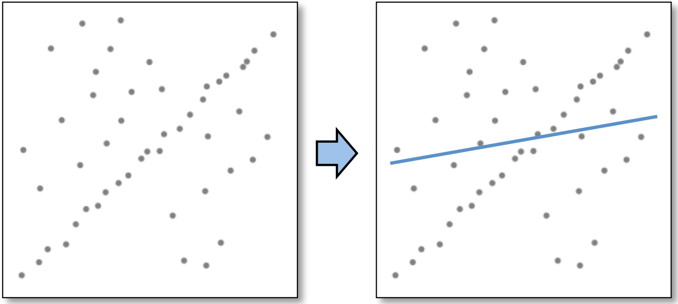
Problem: Fit a line to these datapoints — Least squares fit

- How can we fix this?
- We need a better cost function

[Source: N. Snavely]

Simple case

- Lets consider a simpler example ... linear regression



Problem: Fit a line to these datapoints

Least squares fit

- How can we fix this?
- We need a better cost function

[Source: N. Snavely]

More Robust Least-squares

- Least-squares assumes that the noise follows a Gaussian distribution
- **M-estimators** are use to make least-squares more robust

More Robust Least-squares

- Least-squares assumes that the noise follows a Gaussian distribution
- **M-estimators** are use to make least-squares more robust
- They involve applying a robust penalty function $\rho(\mathbf{r})$ to the residuals

$$E_{RLS}(\Delta \mathbf{p}) = \sum_i \rho(||\mathbf{r}_i||)$$

  instead of taking the square of the residual

More Robust Least-squares

- Least-squares assumes that the noise follows a Gaussian distribution
- **M-estimators** are use to make least-squares more robust
- They involve applying a robust penalty function $\rho(\mathbf{r})$ to the residuals

$$E_{RLS}(\Delta \mathbf{p}) = \sum_i \rho(||\mathbf{r}_i||)$$

  instead of taking the square of the residual
- We can take the derivative with respect to $\mathbf{p}$ and set it to 0

$$\sum_i \psi(||\mathbf{r}_i||) \frac{\partial ||\mathbf{r}_i||}{\partial \mathbf{p}} = \sum_i \frac{\psi(||\mathbf{r}_i||)}{||\mathbf{r}_i||} \mathbf{r}_i^T \frac{\partial \mathbf{r}_i}{\partial \mathbf{p}} = 0$$

  where $\psi(\mathbf{r}) = \rho'(\mathbf{r})$ is the derivative, called **influence function**

More Robust Least-squares

- Least-squares assumes that the noise follows a Gaussian distribution
- **M-estimators** are use to make least-squares more robust
- They involve applying a robust penalty function $\rho(\mathbf{r})$ to the residuals

$$E_{RLS}(\Delta \mathbf{p}) = \sum_i \rho(||\mathbf{r}_i||)$$

  instead of taking the square of the residual

- We can take the derivative with respect to $\mathbf{p}$ and set it to 0

$$\sum_i \psi(||\mathbf{r}_i||) \frac{\partial ||\mathbf{r}_i||}{\partial \mathbf{p}} = \sum_i \frac{\psi(||\mathbf{r}_i||)}{||\mathbf{r}_i||} \mathbf{r}_i^T \frac{\partial \mathbf{r}_i}{\partial \mathbf{p}} = 0$$

  where $\psi(\mathbf{r}) = \rho'(\mathbf{r})$ is the derivative, called **influence function**

- If we introduce a weight $w(r) = \psi(r)/r$, we observe that finding the stationary point is equivalent to minimizing the **iteratively reweighted least squares (IRLS)**

$$E_{IRLS} = \sum_i w(||\mathbf{r}_i||)||\mathbf{r}_i||^2$$

More Robust Least-squares

- Least-squares assumes that the noise follows a Gaussian distribution
- **M-estimators** are use to make least-squares more robust
- They involve applying a robust penalty function $\rho(\mathbf{r})$ to the residuals

$$E_{RLS}(\Delta \mathbf{p}) = \sum_i \rho(||\mathbf{r}_i||)$$

  instead of taking the square of the residual
- We can take the derivative with respect to **p** and set it to 0

$$\sum_i \psi(||\mathbf{r}_i||) \frac{\partial ||\mathbf{r}_i||}{\partial \mathbf{p}} = \sum_i \frac{\psi(||\mathbf{r}_i||)}{||\mathbf{r}_i||} \mathbf{r}_i^T \frac{\partial \mathbf{r}_i}{\partial \mathbf{p}} = 0$$

  where $\psi(\mathbf{r}) = \rho'(\mathbf{r})$ is the derivative, called **influence function**
- If we introduce a weight $w(r) = \psi(r)/r$, we observe that finding the stationary point is equivalent to minimizing the **iteratively reweighted least squares (IRLS)**

$$E_{IRLS} = \sum_i w(||\mathbf{r}_i||)||\mathbf{r}_i||^2$$

Iterative reweighted least-squares

- We want to minimize

$$E_{IRLS} = \sum_i w(||\mathbf{r}_i||)||\mathbf{r}_i||^2$$

- A simple algorithm works by iterating between
  1. Solving for the parameters $\mathbf{p}$
  2. Solving for the weights w

Iterative reweighted least-squares

- We want to minimize

$$E_{IRLS} = \sum_i w(||\mathbf{r}_i||)||\mathbf{r}_i||^2$$

- A simple algorithm works by iterating between
  1. Solving for the parameters **p**
  2. Solving for the weights w
- When the number of outliers is very high, IRLS does not work well (will not converge to the global optima)

Iterative reweighted least-squares

- We want to minimize

$$E_{IRLS} = \sum_i w(||\mathbf{r}_i||)||\mathbf{r}_i||^2$$

- A simple algorithm works by iterating between
  1. Solving for the parameters **p**
  2. Solving for the weights w
- When the number of outliers is very high, IRLS does not work well (will not converge to the global optima)

Simple Idea

- Given a hypothesized line, count the number of points that agree with the line
- Agree = within a small distance of the line i.e., the inliers to that line

Simple Idea

- Given a hypothesized line, count the number of points that agree with the line
- Agree = within a small distance of the line i.e., the inliers to that line
- For all possible lines, select the one with the largest number of inliers

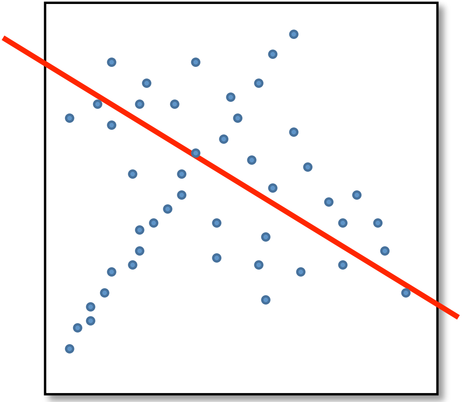
[Source: N. Snavely]

Simple Idea

- Given a hypothesized line, count the number of points that agree with the line
- Agree = within a small distance of the line i.e., the inliers to that line
- For all possible lines, select the one with the largest number of inliers

[Source: N. Snavely]

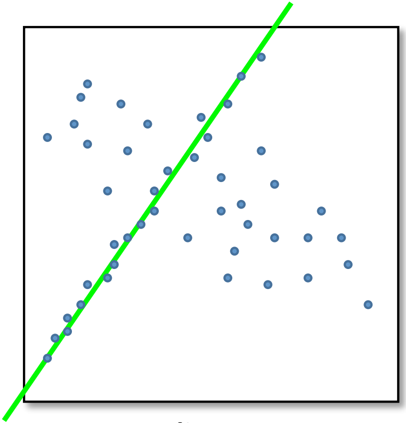
Counting Inliers



Inliers: 3

[Source: N. Snavely]

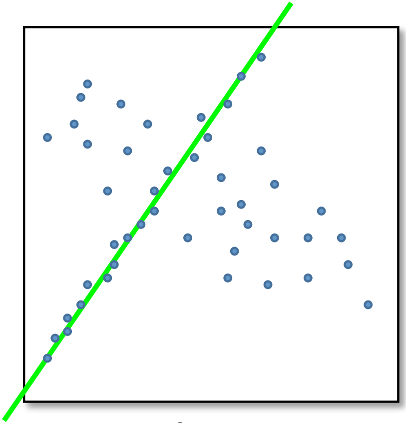
Counting Inliers



Inliers: 20

[Source: N. Snavely]

Counting Inliers



Inliers: 20

What's the problem with this approach?

How do we find the best line?

- Unlike least-squares, no simple closed-form solution
- Hypothesize-and-test

How do we find the best line?

- Unlike least-squares, no simple closed-form solution
- Hypothesize-and-test
- Try out many lines, keep the best one

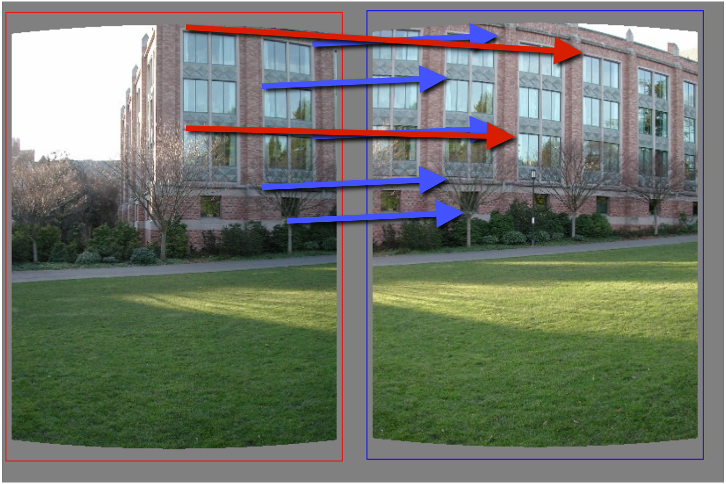
How do we find the best line?

- Unlike least-squares, no simple closed-form solution
- Hypothesize-and-test
- Try out many lines, keep the best one
- Which lines?

How do we find the best line?

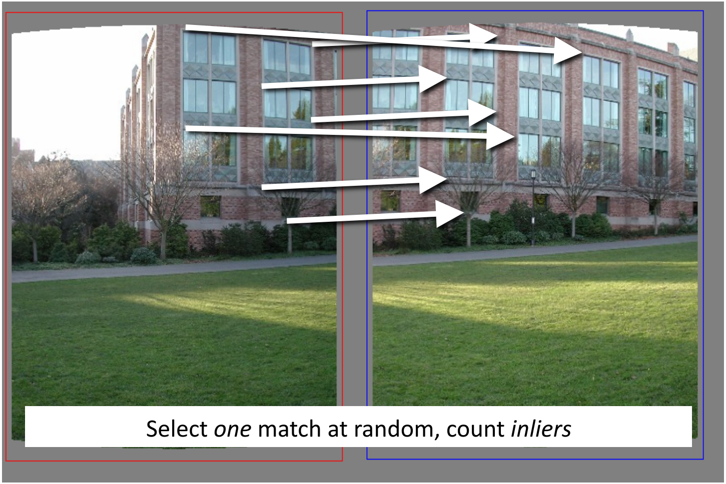
- Unlike least-squares, no simple closed-form solution
- Hypothesize-and-test
- Try out many lines, keep the best one
- Which lines?

Translations



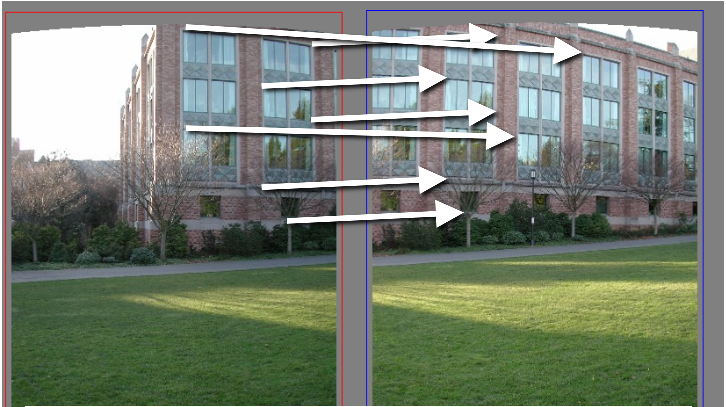
[Source: N. Snavely]

RAndom SAmple Consensus



[Source: N. Snavely]

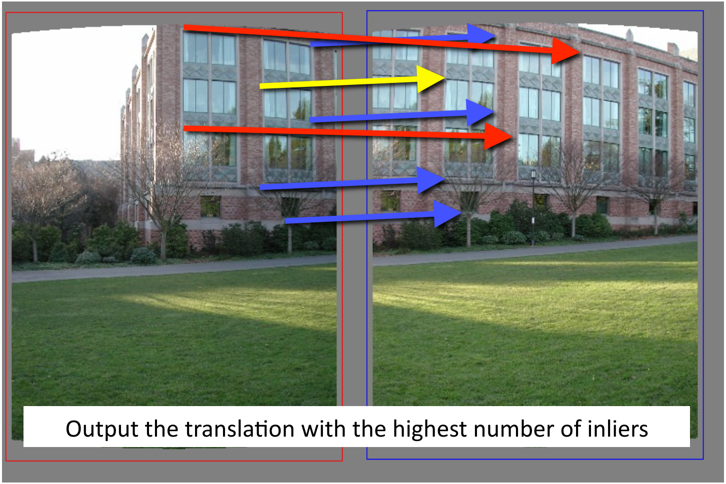
RAndom SAmple Consensus



Select another match at random, count *inliers*

[Source: N. Snavely]

RAndom SAmple Consensus



[Source: N. Snavely]

RANSAC

- All the inliers will agree with each other on the translation vector; the (hopefully small) number of outliers will (hopefully) disagree with each other
- RANSAC only has guarantees if there are $< 50\%$ outliers

RANSAC

- All the inliers will agree with each other on the translation vector; the (hopefully small) number of outliers will (hopefully) disagree with each other
- RANSAC only has guarantees if there are $< 50\%$ outliers
- "All good matches are alike; every bad match is bad in its own way." – [Tolstoy via Alyosha Efros]

[Source: N. Snavely]

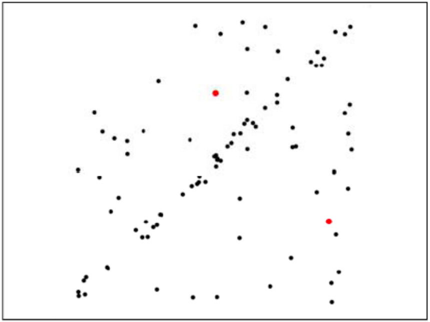
RANSAC

- All the inliers will agree with each other on the translation vector; the (hopefully small) number of outliers will (hopefully) disagree with each other
- RANSAC only has guarantees if there are $< 50\%$ outliers
- "All good matches are alike; every bad match is bad in its own way." – [Tolstoy via Alyosha Efros]

[Source: N. Snavely]

RANSAC for line fitting example

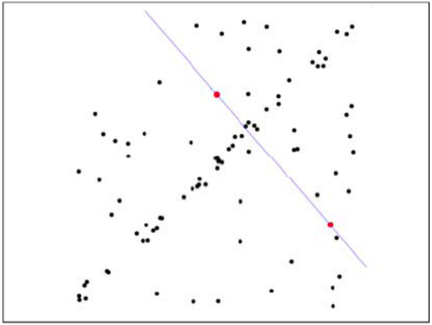
1. Randomly select minimal subset of points
2. Hypothesize a model



[Source: R. Raguram]

RANSAC for line fitting example

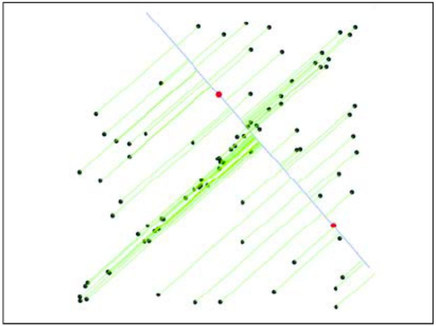
1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function



[Source: R. Raguram]

RANSAC for line fitting example

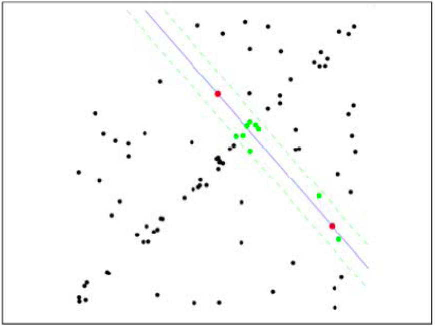
1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model



[Source: R. Raguram]

RANSAC for line fitting example

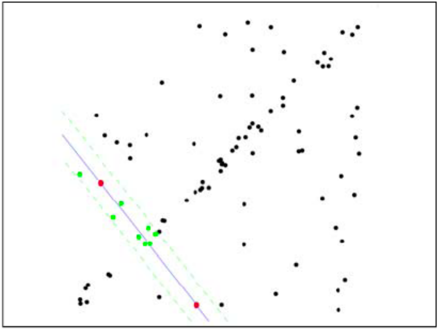
1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize and verify loop



[Source: R. Raguram]

RANSAC for line fitting example

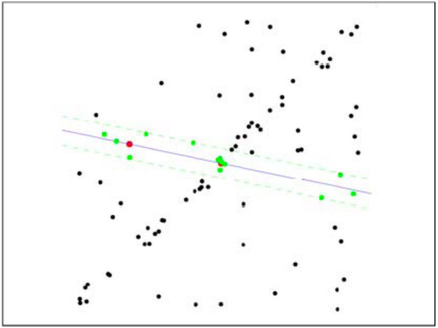
1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize and verify loop



[Source: R. Raguram]

RANSAC for line fitting example

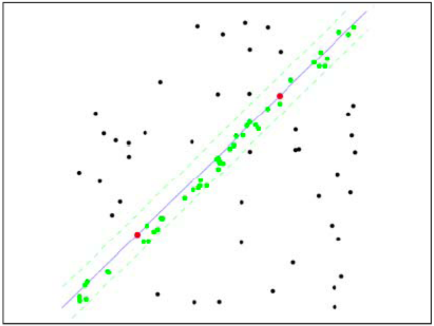
1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize and verify loop
6. Choose model with largest set of inliers



[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]

RANSAC

- **Inlier threshold** related to the amount of noise we expect in inliers
- Often model noise as Gaussian with some standard deviation (e.g., 3 pixels)

RANSAC

- **Inlier threshold** related to the amount of noise we expect in inliers
- Often model noise as Gaussian with some standard deviation (e.g., 3 pixels)
- **Number of rounds** related to the percentage of outliers we expect, and the probability of success we'd like to guarantee

RANSAC

- **Inlier threshold** related to the amount of noise we expect in inliers
- Often model noise as Gaussian with some standard deviation (e.g., 3 pixels)
- **Number of rounds** related to the percentage of outliers we expect, and the probability of success we'd like to guarantee
- Suppose there are 20% outliers, and we want to find the correct answer with 99% probability

RANSAC

- **Inlier threshold** related to the amount of noise we expect in inliers
- Often model noise as Gaussian with some standard deviation (e.g., 3 pixels)
- **Number of rounds** related to the percentage of outliers we expect, and the probability of success we'd like to guarantee
- Suppose there are 20% outliers, and we want to find the correct answer with 99% probability
- How many rounds do we need?

RANSAC

- **Inlier threshold** related to the amount of noise we expect in inliers
- Often model noise as Gaussian with some standard deviation (e.g., 3 pixels)
- **Number of rounds** related to the percentage of outliers we expect, and the probability of success we'd like to guarantee
- Suppose there are 20% outliers, and we want to find the correct answer with 99% probability
- How many rounds do we need?

How many rounds?

- Sufficient number of trials S must be tried.
- Let p be the probability that any given correspondence is valid and P be the total probability of success after S trials.

How many rounds?

- Sufficient number of trials S must be tried.
- Let p be the probability that any given correspondence is valid and P be the total probability of success after S trials.
- The likelihood in one trial that all k random samples are inliers is p^k

How many rounds?

- Sufficient number of trials S must be tried.
- Let p be the probability that any given correspondence is valid and P be the total probability of success after S trials.
- The likelihood in one trial that all k random samples are inliers is p^k
- The likelihood that S such trials will all fail is

$$1 - P = (1 - p^k)^S$$

How many rounds?

- Sufficient number of trials S must be tried.
- Let p be the probability that any given correspondence is valid and P be the total probability of success after S trials.
- The likelihood in one trial that all k random samples are inliers is p^k
- The likelihood that S such trials will all fail is

$$1 - P = (1 - p^k)^S$$

- The required minimum number of trials is

$$S = \frac{\log(1 - P)}{\log(1 - p^k)}$$

How many rounds?

- Sufficient number of trials S must be tried.
- Let p be the probability that any given correspondence is valid and P be the total probability of success after S trials.
- The likelihood in one trial that all k random samples are inliers is p^k
- The likelihood that S such trials will all fail is

$$1 - P = (1 - p^k)^S$$

- The required minimum number of trials is

$$S = \frac{\log(1 - P)}{\log(1 - p^k)}$$

- The number of trials grows quickly with the number of sample points used.

How many rounds?

- Sufficient number of trials S must be tried.
- Let p be the probability that any given correspondence is valid and P be the total probability of success after S trials.
- The likelihood in one trial that all k random samples are inliers is p^k
- The likelihood that S such trials will all fail is

$$1 - P = (1 - p^k)^S$$

- The required minimum number of trials is

$$S = \frac{\log(1 - P)}{\log(1 - p^k)}$$

- The number of trials grows quickly with the number of sample points used.
- Use the minimum number of sample points k possible for any given trial

How many rounds?

- Sufficient number of trials S must be tried.
- Let p be the probability that any given correspondence is valid and P be the total probability of success after S trials.
- The likelihood in one trial that all k random samples are inliers is p^k
- The likelihood that S such trials will all fail is

$$1 - P = (1 - p^k)^S$$

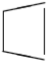
- The required minimum number of trials is

$$S = \frac{\log(1 - P)}{\log(1 - p^k)}$$

- The number of trials grows quickly with the number of sample points used.
- Use the minimum number of sample points k possible for any given trial

How big is the number of samples?

- For alignment, depends on the motion model
- Each sample is a correspondence (pair of matching points)

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	3	lengths	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	4	angles	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times 3}$	6	parallelism	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times 3}$	8	straight lines	

RANSAC pros and cons

Pros

- Simple and general
- Applicable to many different problems

RANSAC pros and cons

Pros

- Simple and general
- Applicable to many different problems
- Often works well in practice

RANSAC pros and cons

Pros

- Simple and general
- Applicable to many different problems
- Often works well in practice

Cons

- Parameters to tune

RANSAC pros and cons

Pros

- Simple and general
- Applicable to many different problems
- Often works well in practice

Cons

- Parameters to tune
- Sometimes too many iterations are required

RANSAC pros and cons

Pros

- Simple and general
- Applicable to many different problems
- Often works well in practice

Cons

- Parameters to tune
- Sometimes too many iterations are required
- Can fail for extremely low inlier ratios

RANSAC pros and cons

Pros

- Simple and general
- Applicable to many different problems
- Often works well in practice

Cons

- Parameters to tune
- Sometimes too many iterations are required
- Can fail for extremely low inlier ratios
- We can often do better than brute-force sampling

[Source: N. Snavely]

RANSAC pros and cons

Pros

- Simple and general
- Applicable to many different problems
- Often works well in practice

Cons

- Parameters to tune
- Sometimes too many iterations are required
- Can fail for extremely low inlier ratios
- We can often do better than brute-force sampling

[Source: N. Snavely]

RANSAC as Voting

- An example of a "voting"-based fitting scheme
- Each hypothesis gets voted on by each data point, best hypothesis wins

RANSAC as Voting

- An example of a "voting"-based fitting scheme
- Each hypothesis gets voted on by each data point, best hypothesis wins
- There are many other types of voting schemes, e.g., Hough transforms

[Source: N. Snavely]

RANSAC as Voting

- An example of a "voting"-based fitting scheme
- Each hypothesis gets voted on by each data point, best hypothesis wins
- There are many other types of voting schemes, e.g., Hough transforms

[Source: N. Snavely]

Next class ... more on cameras and projection