COMP3065
Computer Vision

Topic 4 – Image Stitching 1

Dr. Tianxiang Cui
2025 Spring

Outline

- Geometric Transformation
  - Translation
  - Euclidean
  - Similarity
  - Affine
  - Projective
- Image Stitching
- Compute Transformation
- RANSAC

Image Stitching

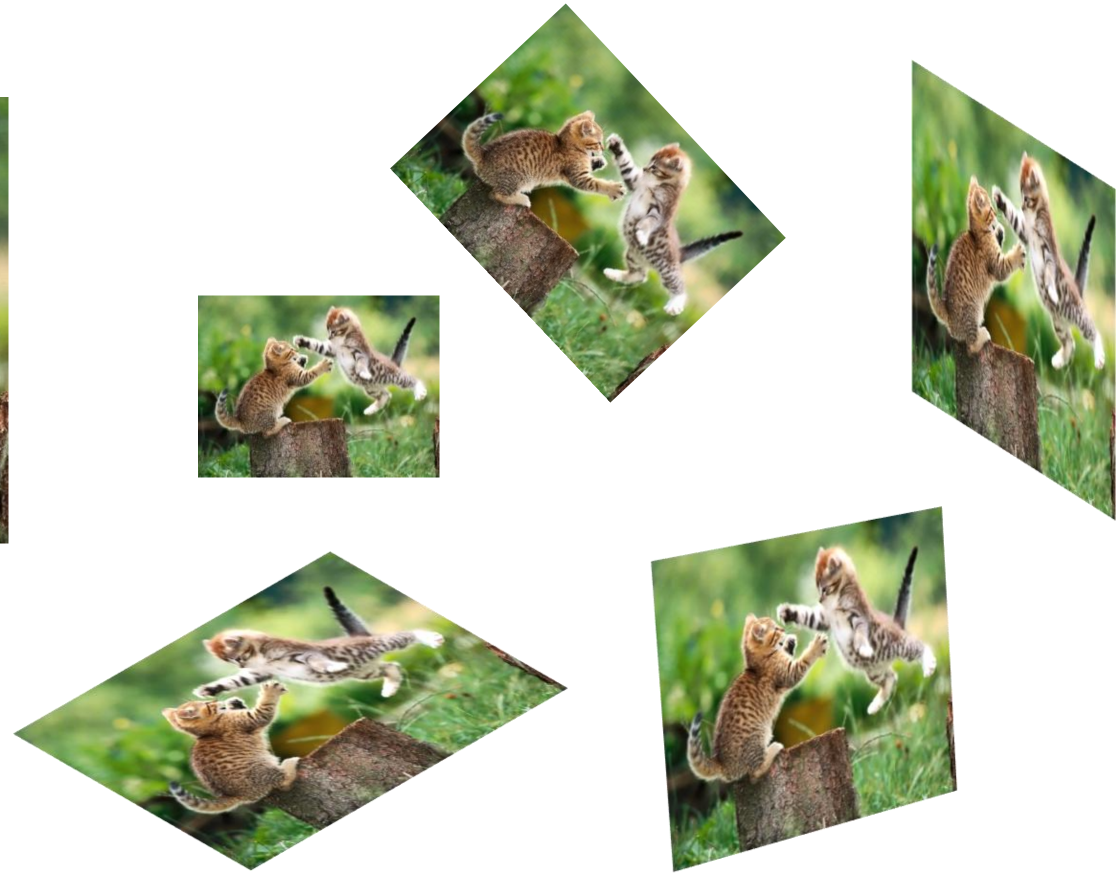
Combine two or more overlapping images to make one larger image

Slides credit: Shapino

Image Stitching: Geometric Transformation

We need to **transform** one image first in order to stitch the two images to together!

What Are Geometric Transformations?

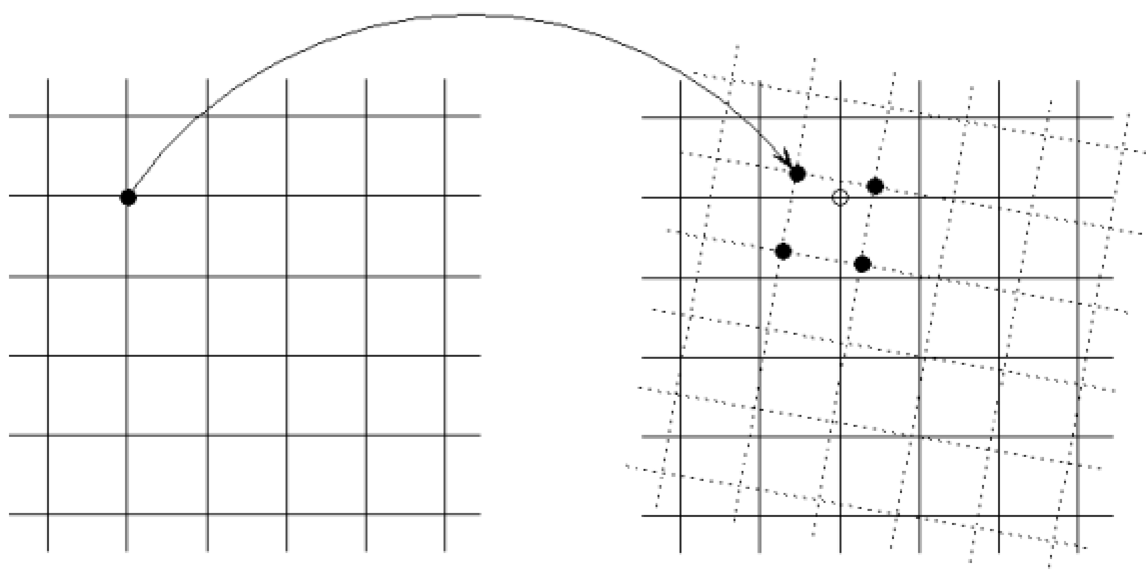


How do we do them?

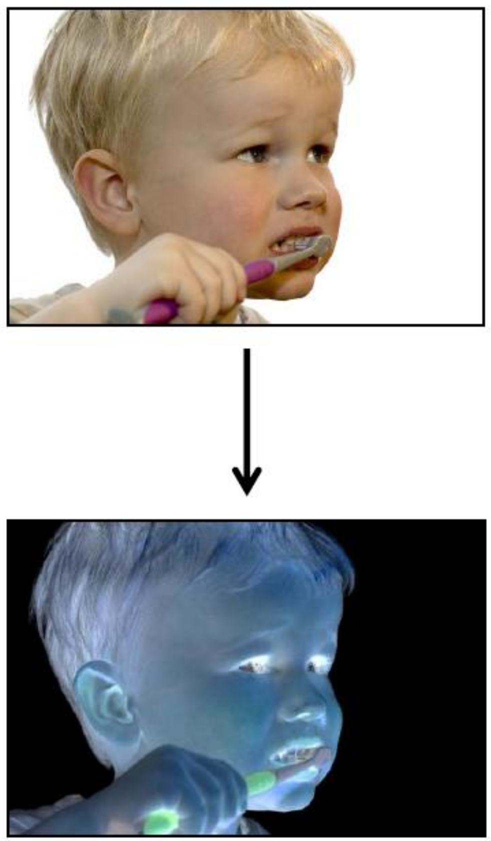
What Are Geometric Transformations?

- In a geometric transformation each point (x, y) of image A is mapped to a point (u, v) in a new coordinate system

$$u = f_1(x, y)$$
$$v = f_2(x, y)$$

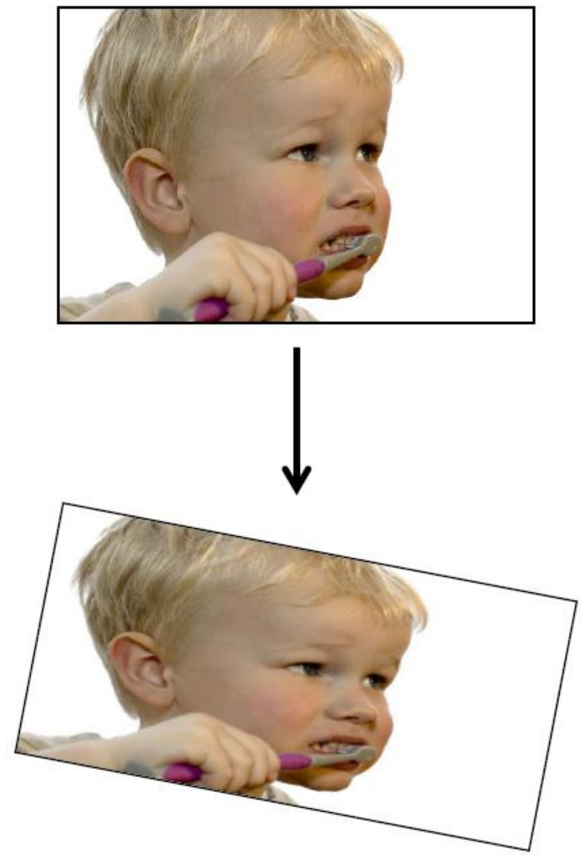


What Are Geometric Transformations?



changes pixel *values*

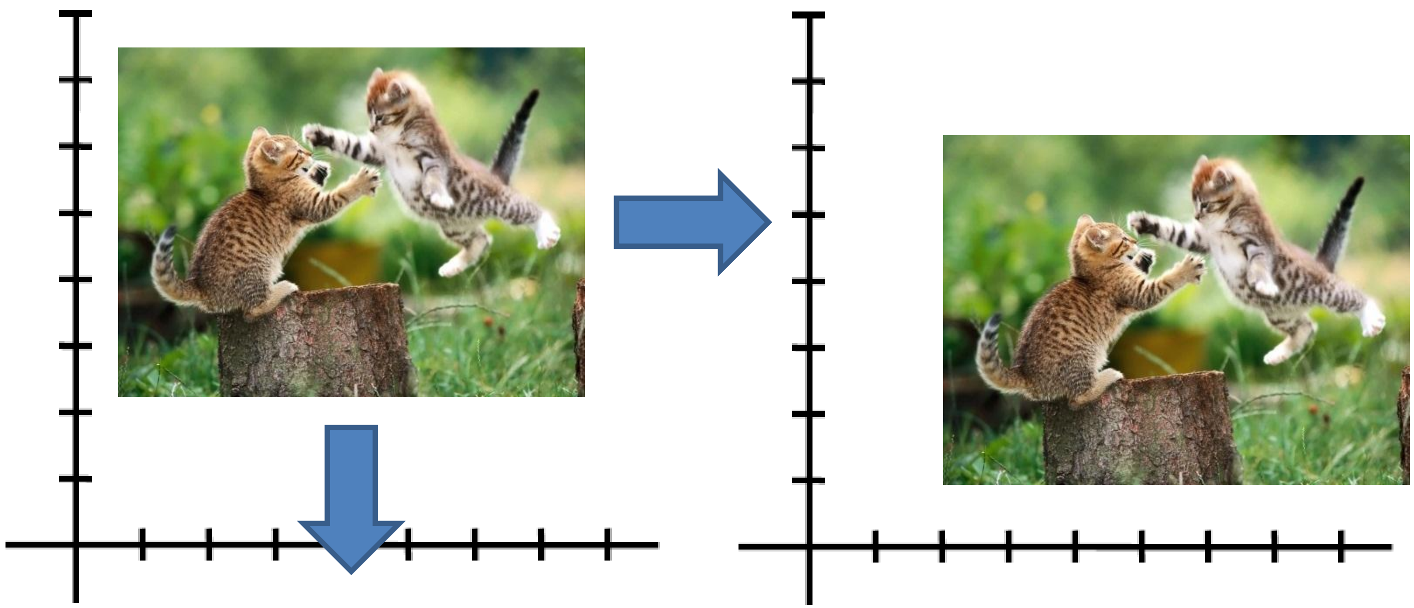
changes *range* of image function



changes pixel *locations*

changes *domain* of image function

Translation

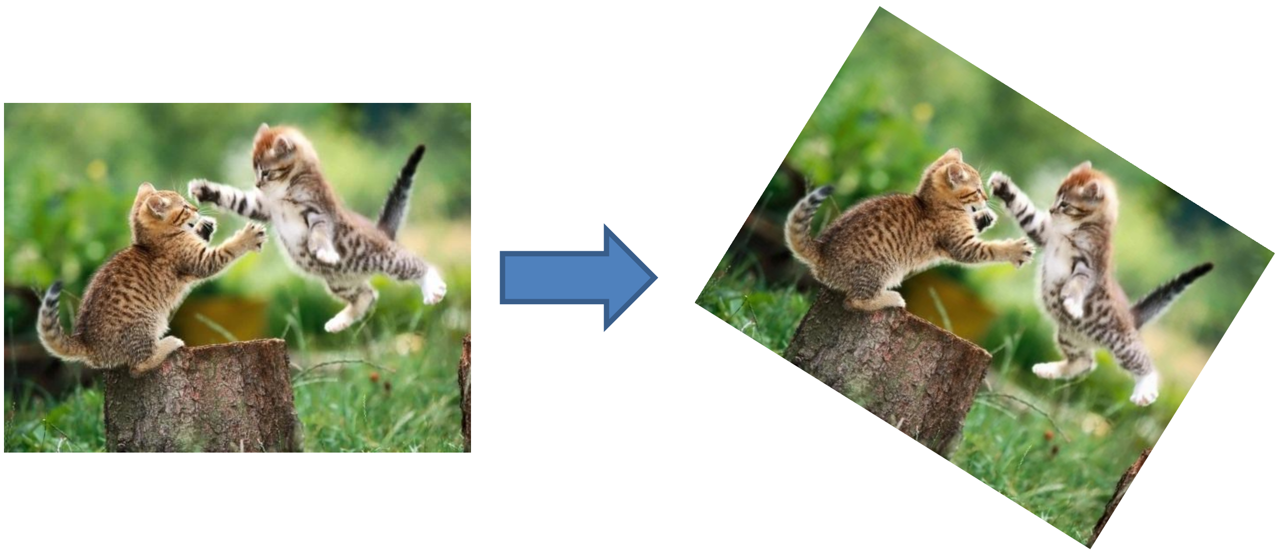


x and *y* are pixel location.

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}$$

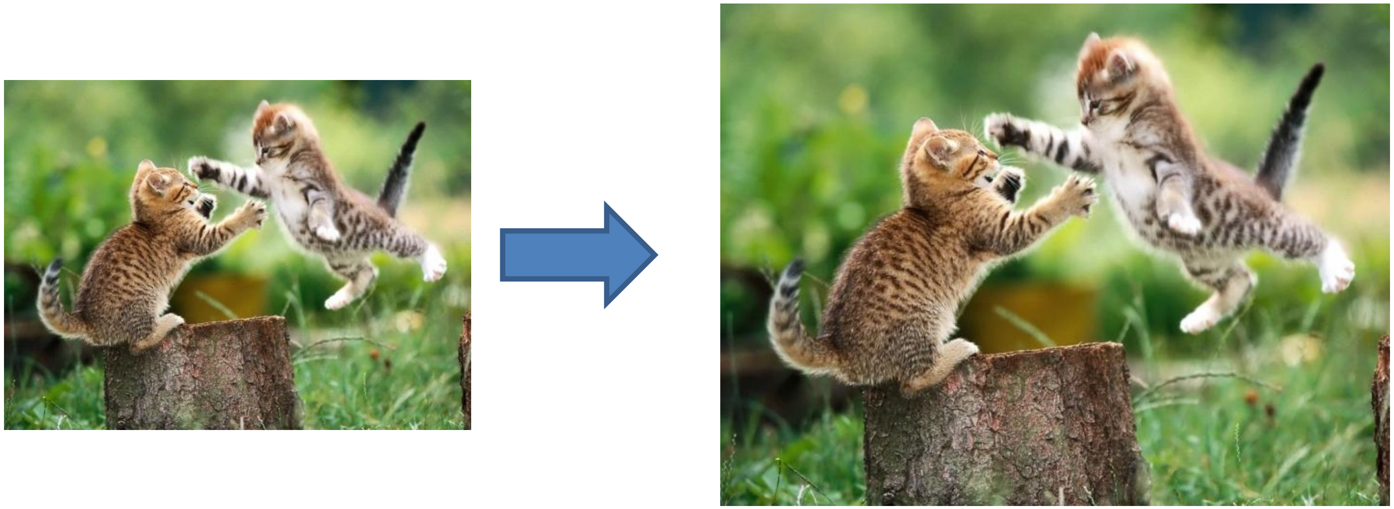
Ignore last row for now

Rotation



$$\begin{bmatrix} cos(\theta) & -sin(\theta) & t_x \\ sin(\theta) & cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

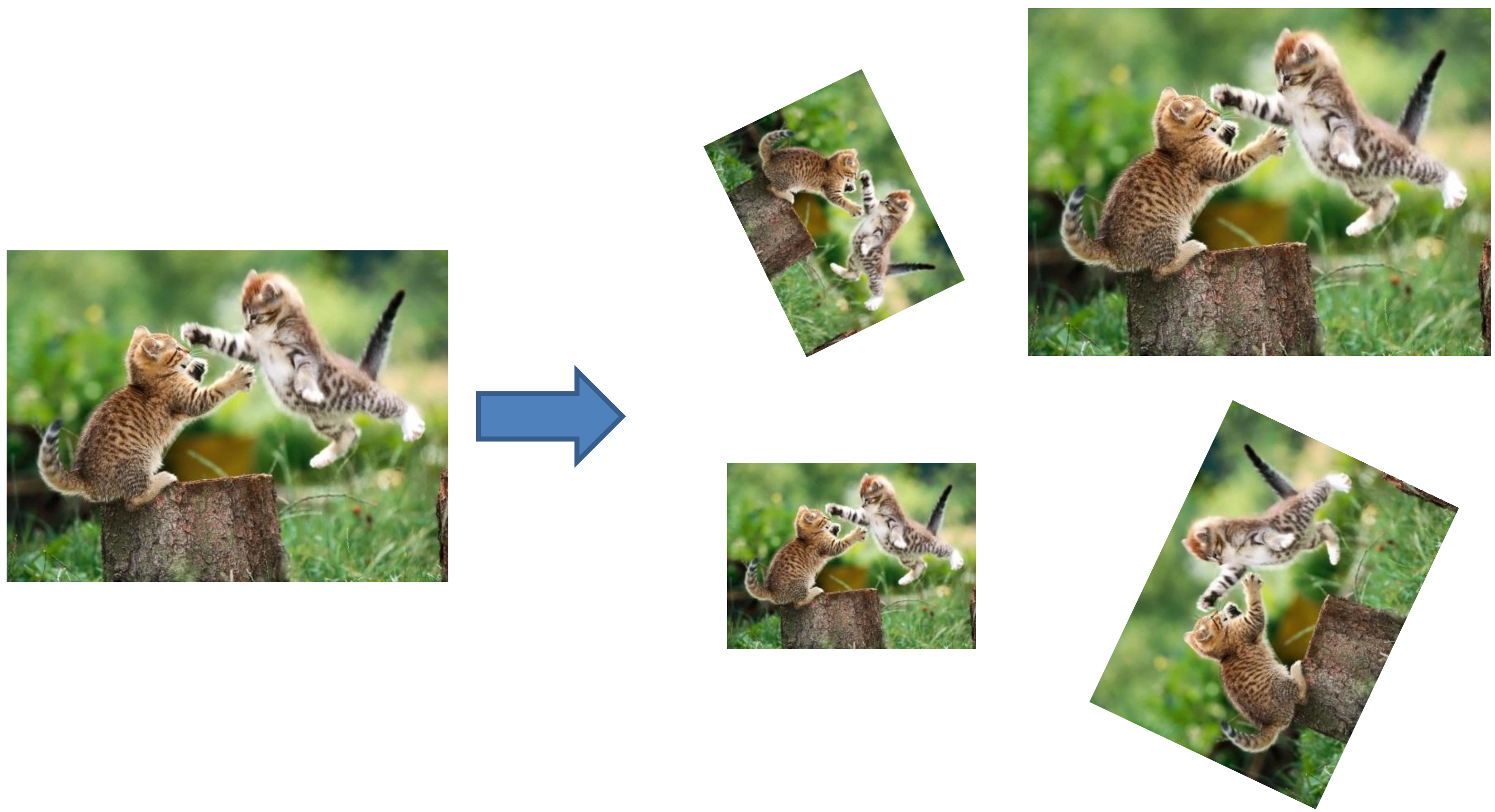
Scale



Each component is multiplied by a scalar

$$\begin{bmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

Similarity



Euclidean transform = translation + rotation

Similarity transform = translation + rotation + scale

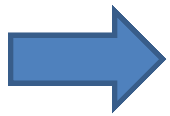
Similarity

- Any transform of the form:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ -b & a & d \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- 4 degrees of freedom (DOF)

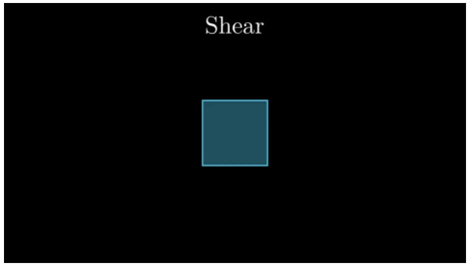
Aspect Ratio



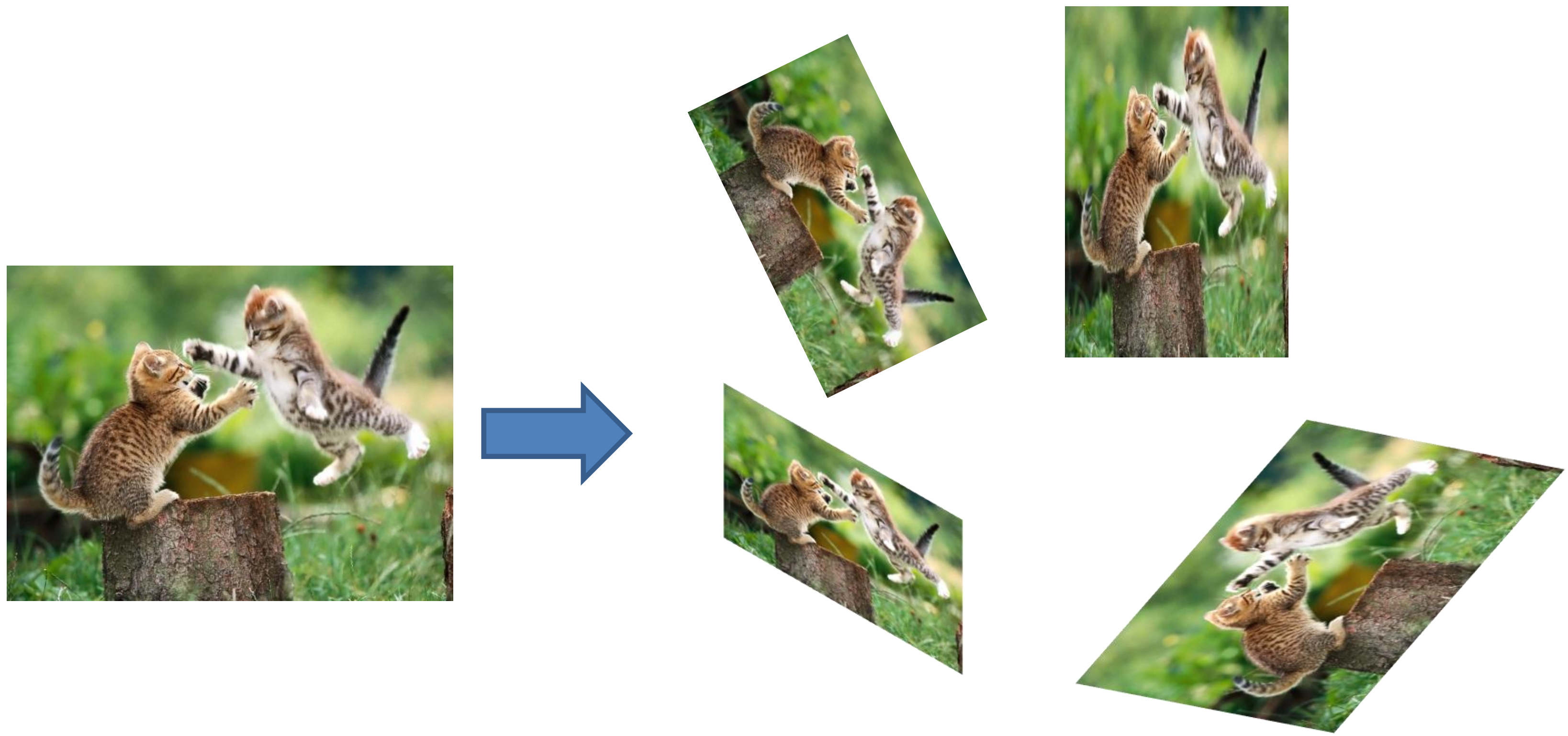
$$\begin{bmatrix} a & 0 & 0 \\ 0 & \frac{1}{a} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

Shear

$$\begin{bmatrix} 1 & a & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$



Affine



Affine transform = translation + rotation +
scale + aspect ratio + shear

Preserves: Parallelism

Affine

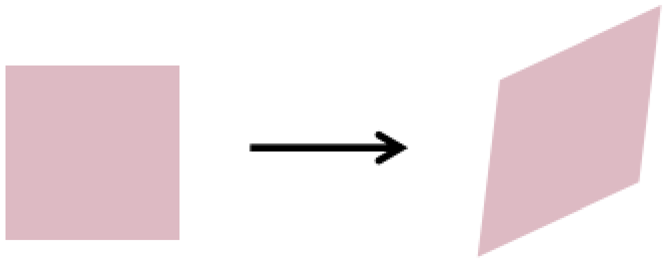
- Any transform of the form:

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

- ? degrees of freedom (DOF)

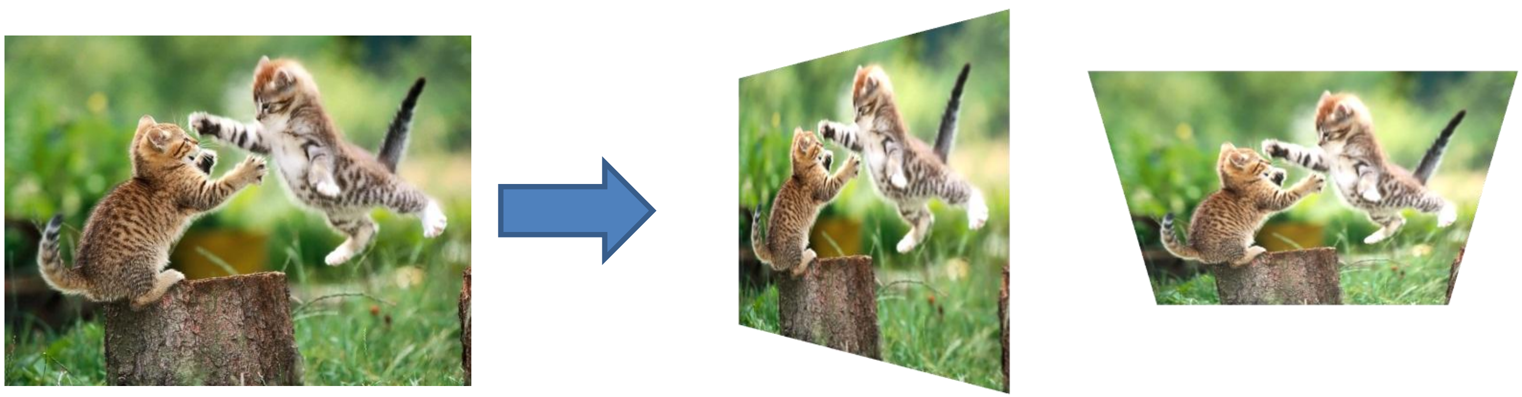
Properties of Affine

- Origin **does not** necessarily **map** to origin
- Lines **map** to lines
- Parallel lines **map** to parallel lines
- Ratios (lengths and areas) are **preserved**
- **Compositions** of affine transforms are also affine transforms



Anything Else?

- Are there any other planar transformation?



Homogeneous Coordinates

heterogeneous coordinates		homogeneous coordinates
$\begin{bmatrix} x \\ y \end{bmatrix}$	$\Rightarrow$	$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

add a 1 here

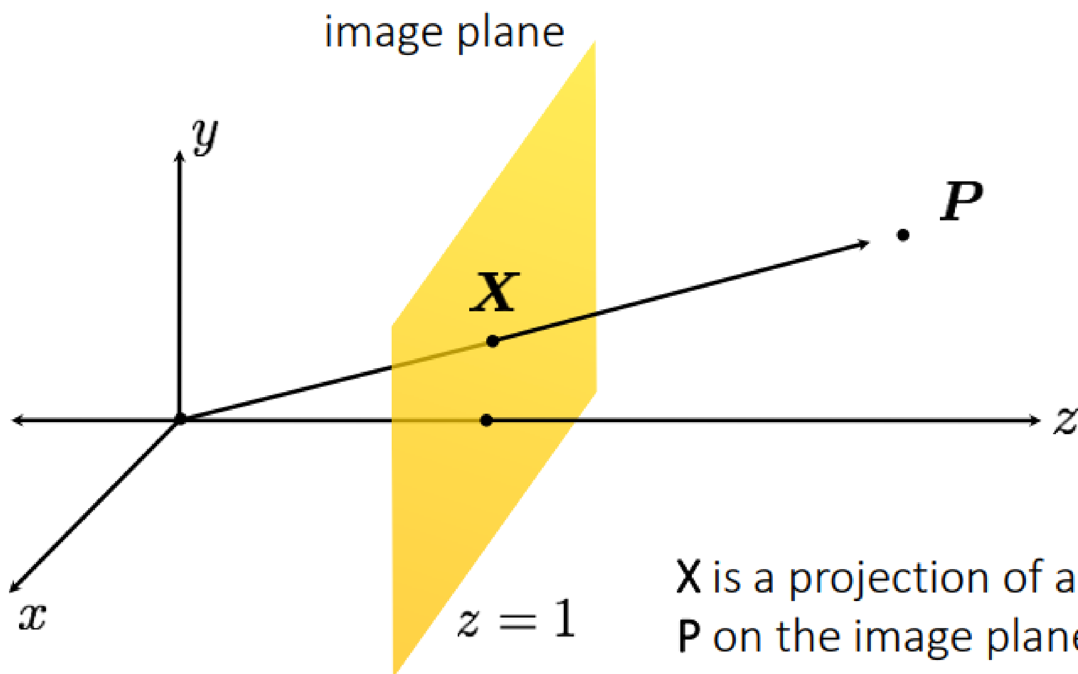
- Represent 2D point with a 3D vector

Projective Geometry

image point in pixel coordinates $\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$\Updownarrow$$

image point in homogeneous coordinates $\boldsymbol{X} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$



X is a projection of a point **P** on the image plane

- Model an image as a plane in space, and project it onto any other image

Homogeneous Coordinates

homogeneous → heterogeneous

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow \begin{bmatrix} x/w \\ y/w \end{bmatrix}$$

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

One extra step:

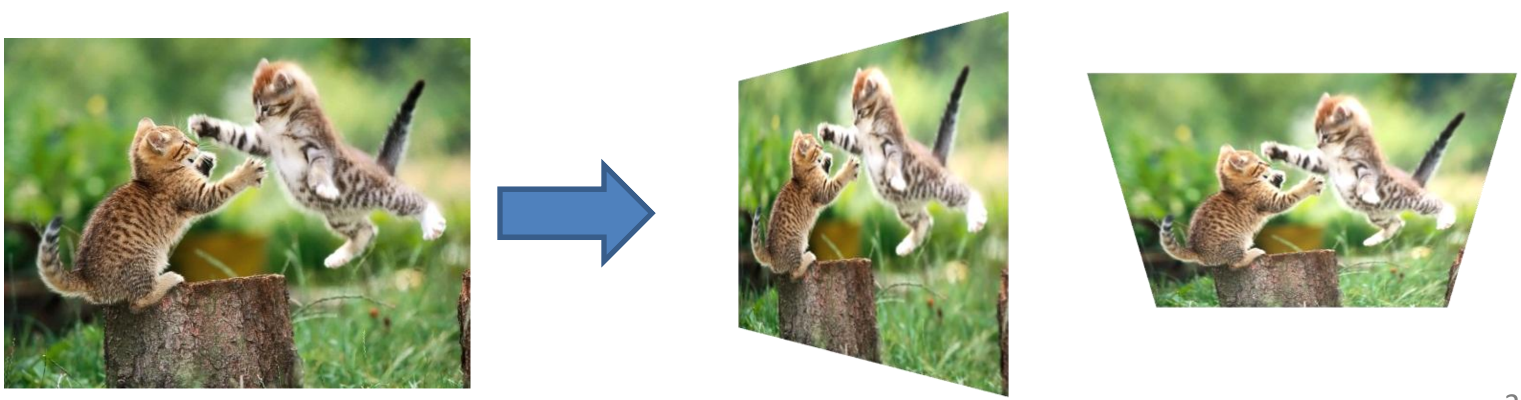
$$x' = u/w$$

$$y' = v/w$$

Projective Transformation

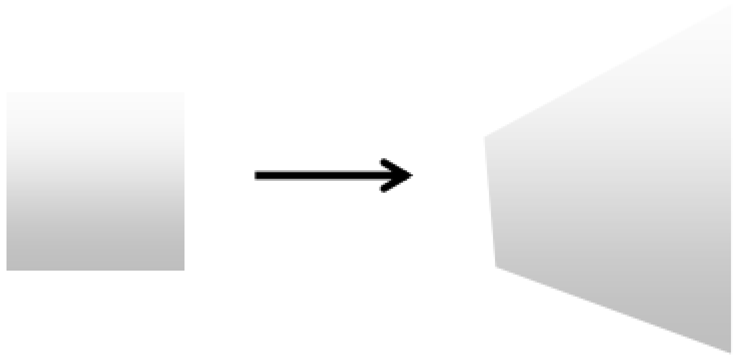
$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

$$x' = u/w$$
$$y' = v/w$$



Properties of Projective

- Origin **does not** necessarily **map** to origin
- Lines **map** to lines
- Parallel lines **do not** necessarily **map** to parallel lines
- Ratios (lengths and areas) are **not** necessarily **preserved**
- **Compositions** of projective transforms are also projective transforms



Classification of 2D transformations

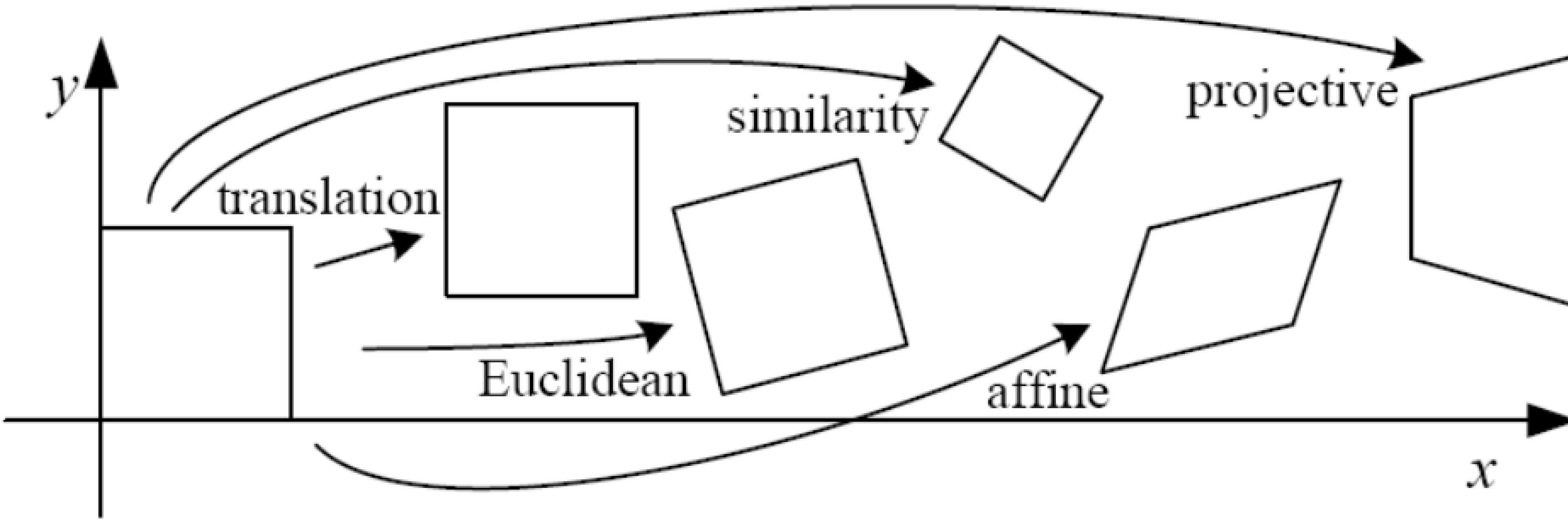


Image Stitching: The Idea

1. Take a sequence of images from the same position.
2. To stitch two images: compute transformation between second image and first.
3. Shift (warp) the second image to overlap with the first.
4. Blend the two together to create a mosaic.
5. If there are more images, repeat step 2 to 4.

The Idea

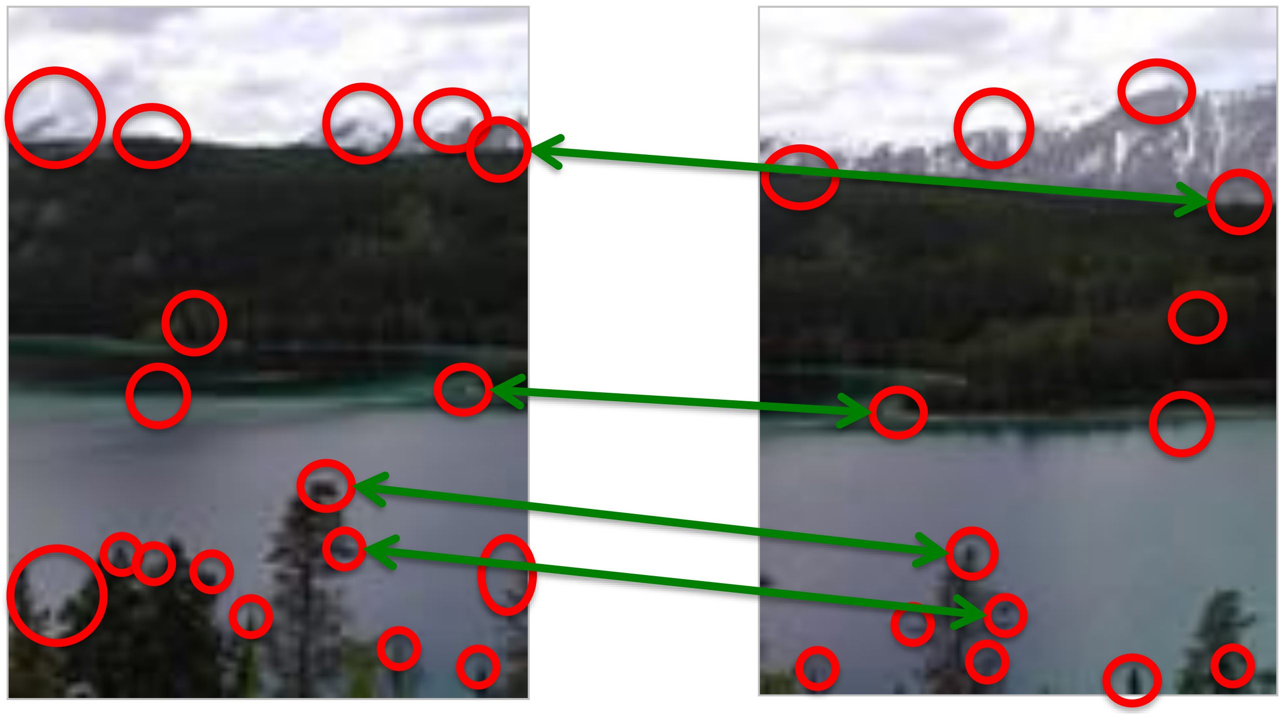
1. Take a sequence of images from the same position.
   - Rotate the camera about its optical center, (more accurate with camera mounted on a tripod).
   - No tripods? Hold the camera and turn the body without changing the position.

The Idea

2. Compute transformation between images.
    - Extract interest points
    - Find Matches
    - Compute transformation

The Idea

2. Compute transformation between images.
   - Extract interest points
   - Find Matches
   - Compute transformation



The Idea

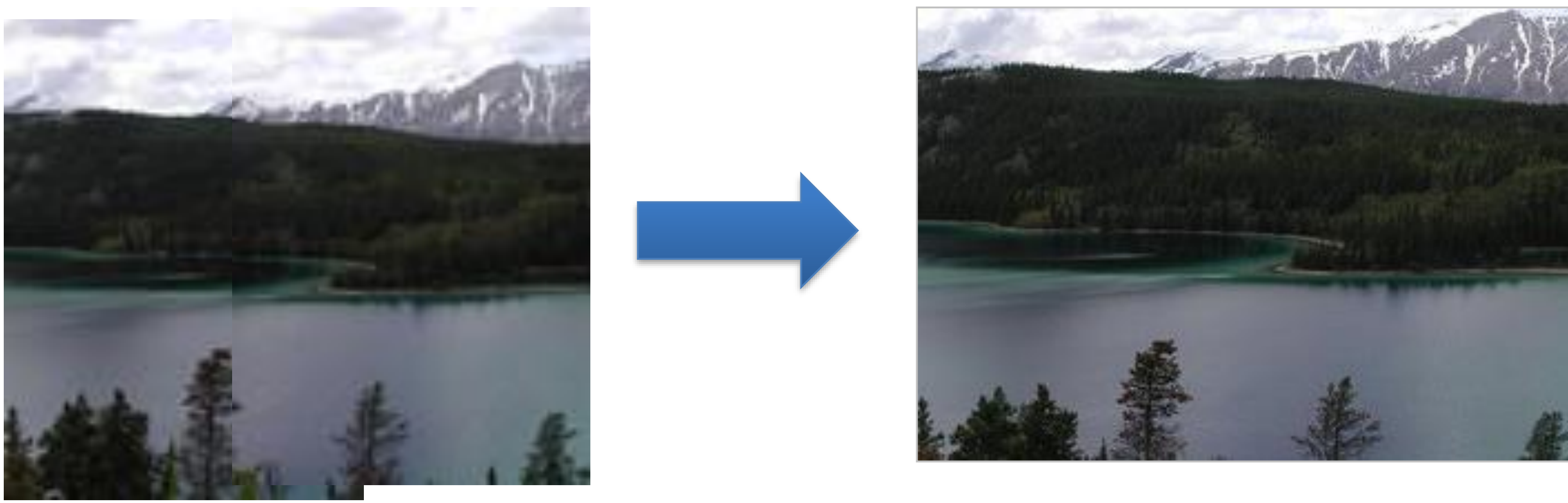
3. Shift image to overlap.

The Idea

3. Shift image to overlap.

The Idea

4. Blend together to create mosaic.



The Idea

5. Repeat for all images.

The Idea

1. Take a sequence of images from the same position.
2. To stitch two images: compute transformation between second image and first.
3. Shift (warp) the second image to overlap with the first.
4. Blend the two together to create a mosaic.
5. If there are more images, repeat step 2 to 4.

The details …

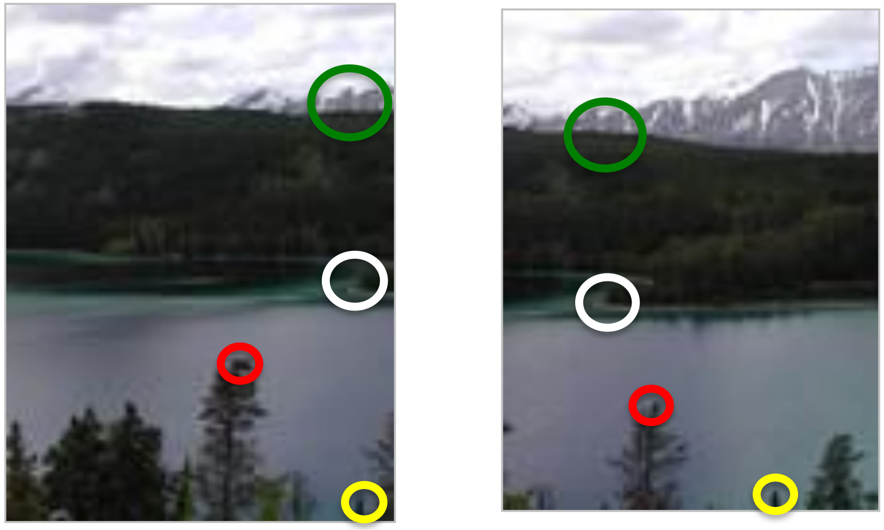
Compute Transformation

- Extract feature points (e.g. SIFT)
- Find good matches (e.g. compare feature vectors)

Compute Transformation

- Extract feature points (e.g. SIFT)
- Find good matches (e.g. compare feature vectors)
- **Compute Transformation**

Let’s assume we are given a set of good matching interest points



The Underline Theory: Image Reprojection

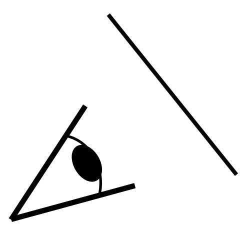
The mosaic has a natural interpretation in 3D.

- The images are reprojected onto a common plane.
- The mosaic is formed on this plane.

The Underline Theory: Image Reprojection

The mosaic has a natural interpretation in 3D.

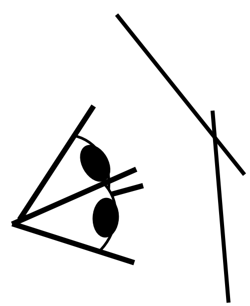
- The images are reprojected onto a common plane.
- The mosaic is formed on this plane.



The Underline Theory: Image Reprojection

The mosaic has a natural interpretation in 3D.

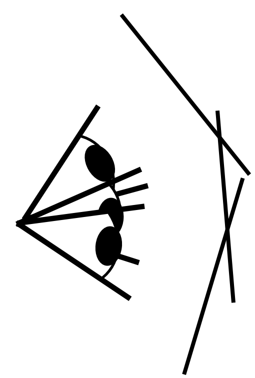
- The images are reprojected onto a common plane.
- The mosaic is formed on this plane.



The Underline Theory: Image Reprojection

The mosaic has a natural interpretation in 3D.

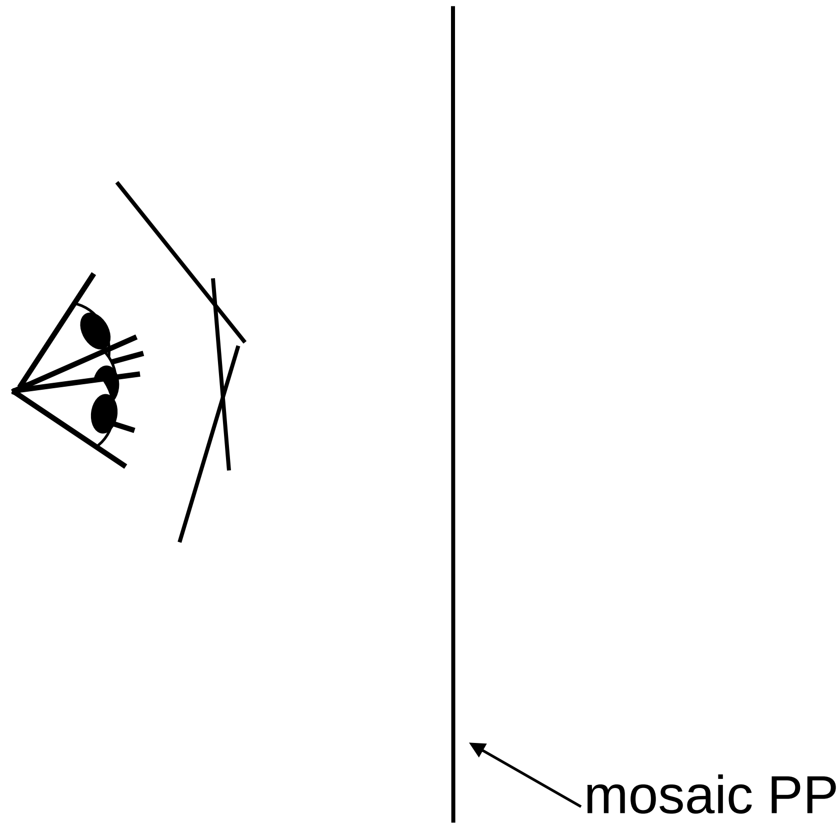
- The images are reprojected onto a common plane.
- The mosaic is formed on this plane.



The Underline Theory: Image Reprojection

The mosaic has a natural interpretation in 3D.

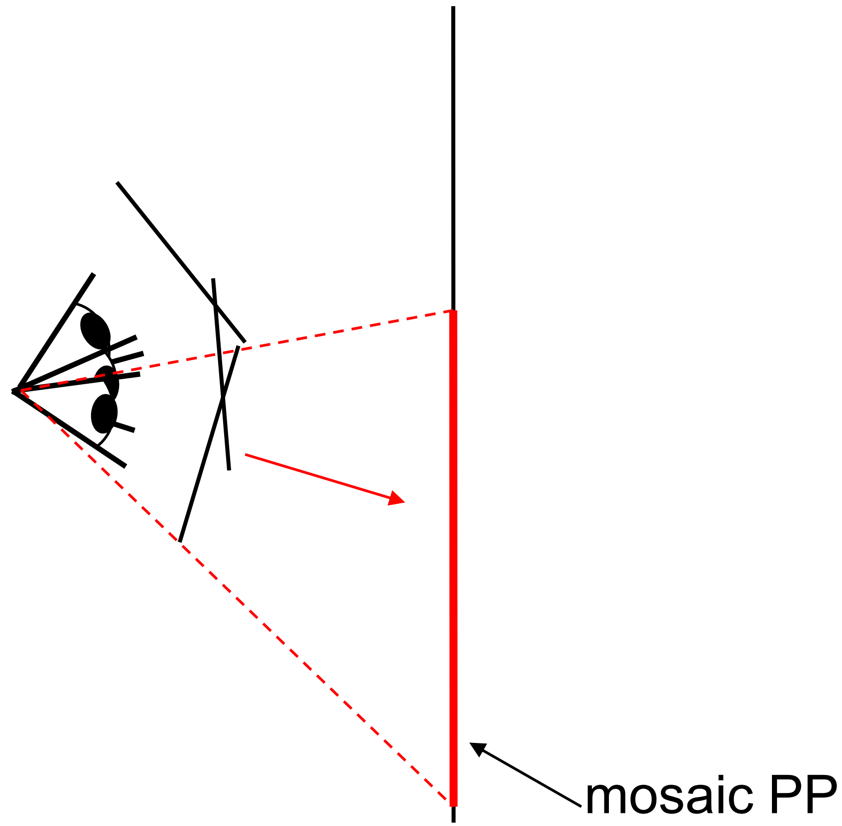
- The images are reprojected onto a common plane.
- The mosaic is formed on this plane.



The Underline Theory: Image Reprojection

The mosaic has a natural interpretation in 3D.

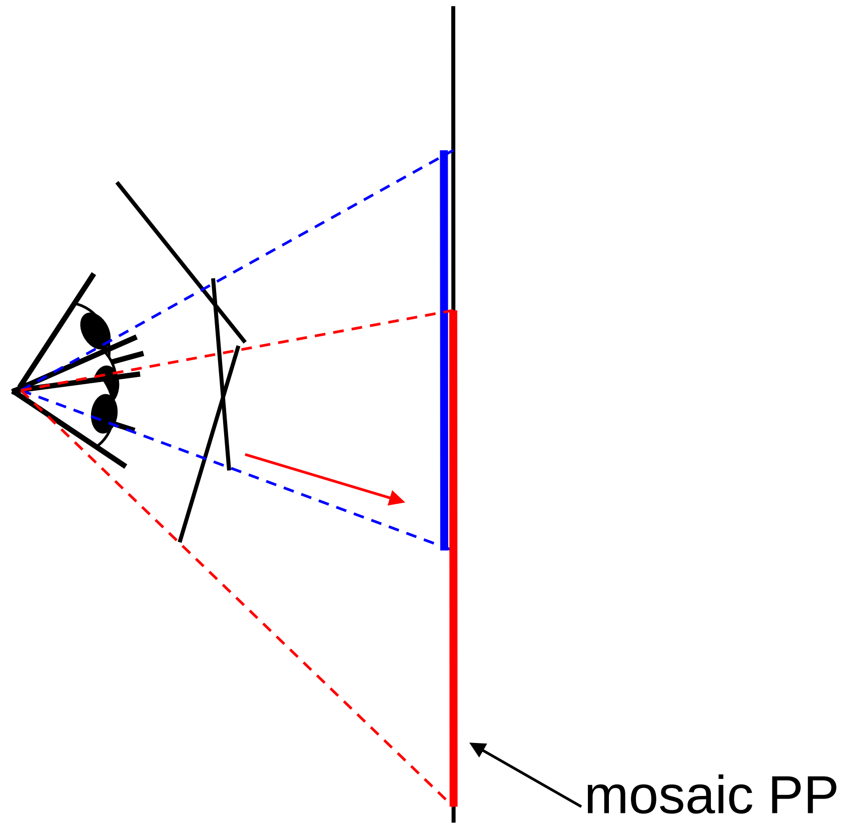
- The images are reprojected onto a common plane.
- The mosaic is formed on this plane.



The Underline Theory: Image Reprojection

The mosaic has a natural interpretation in 3D.

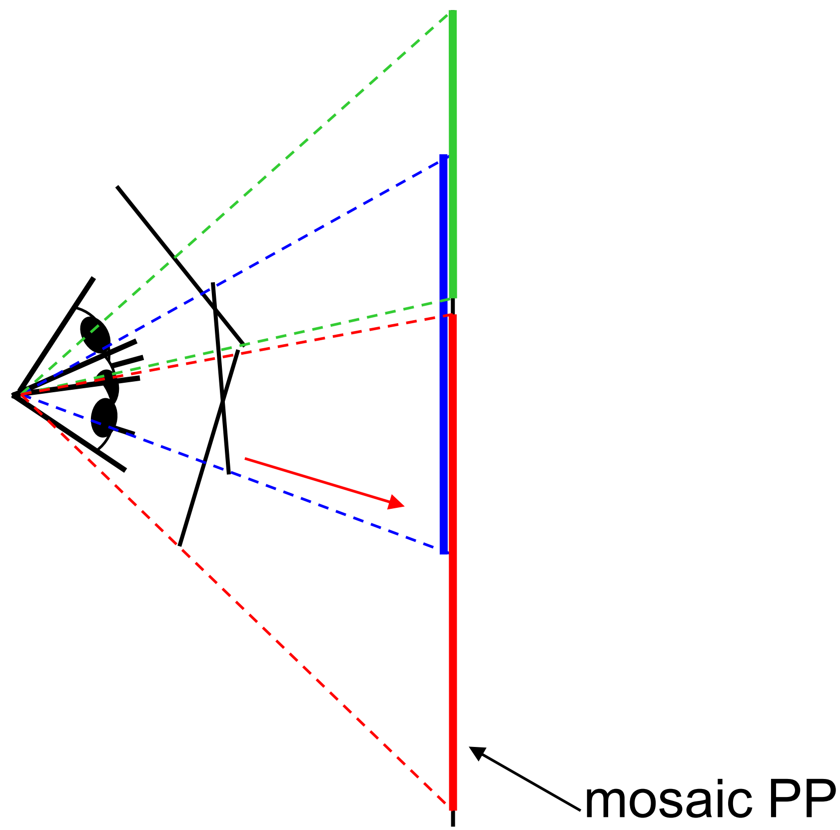
- The images are reprojected onto a common plane.
- The mosaic is formed on this plane.



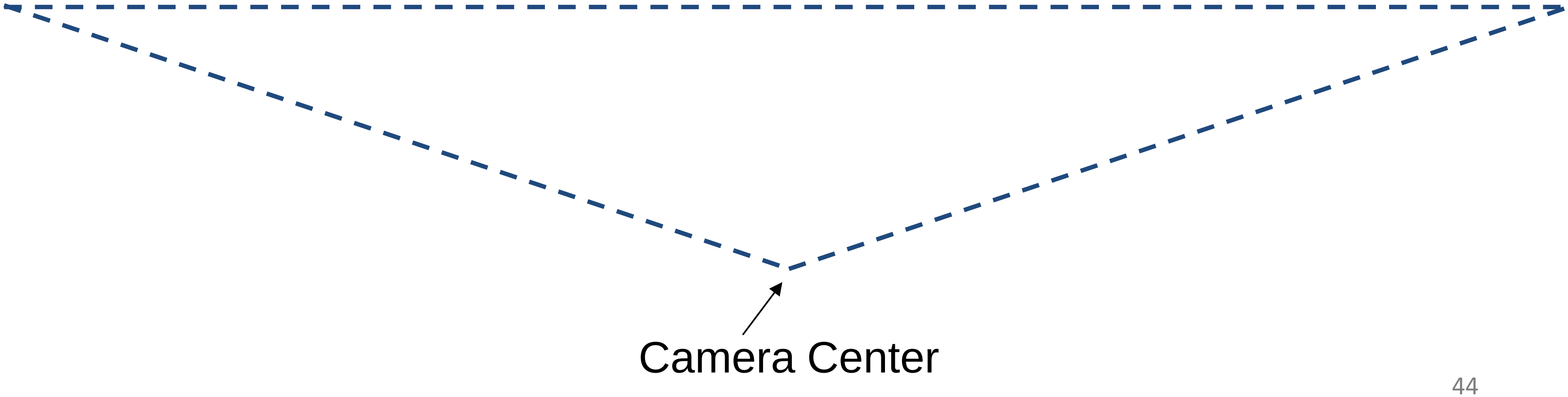
The Underline Theory: Image Reprojection

The mosaic has a natural interpretation in 3D.

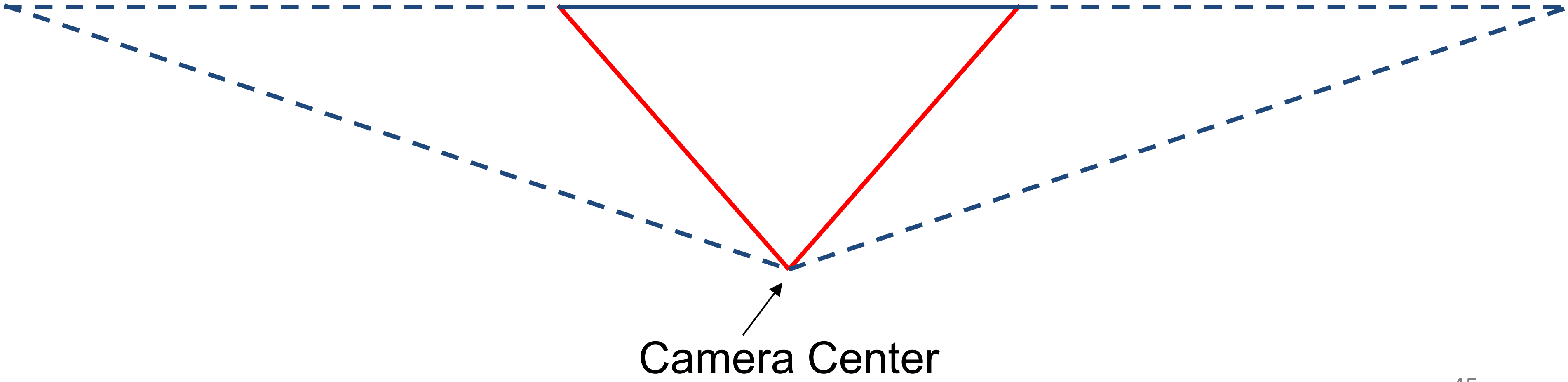
- The images are reprojected onto a common plane.
- The mosaic is formed on this plane.



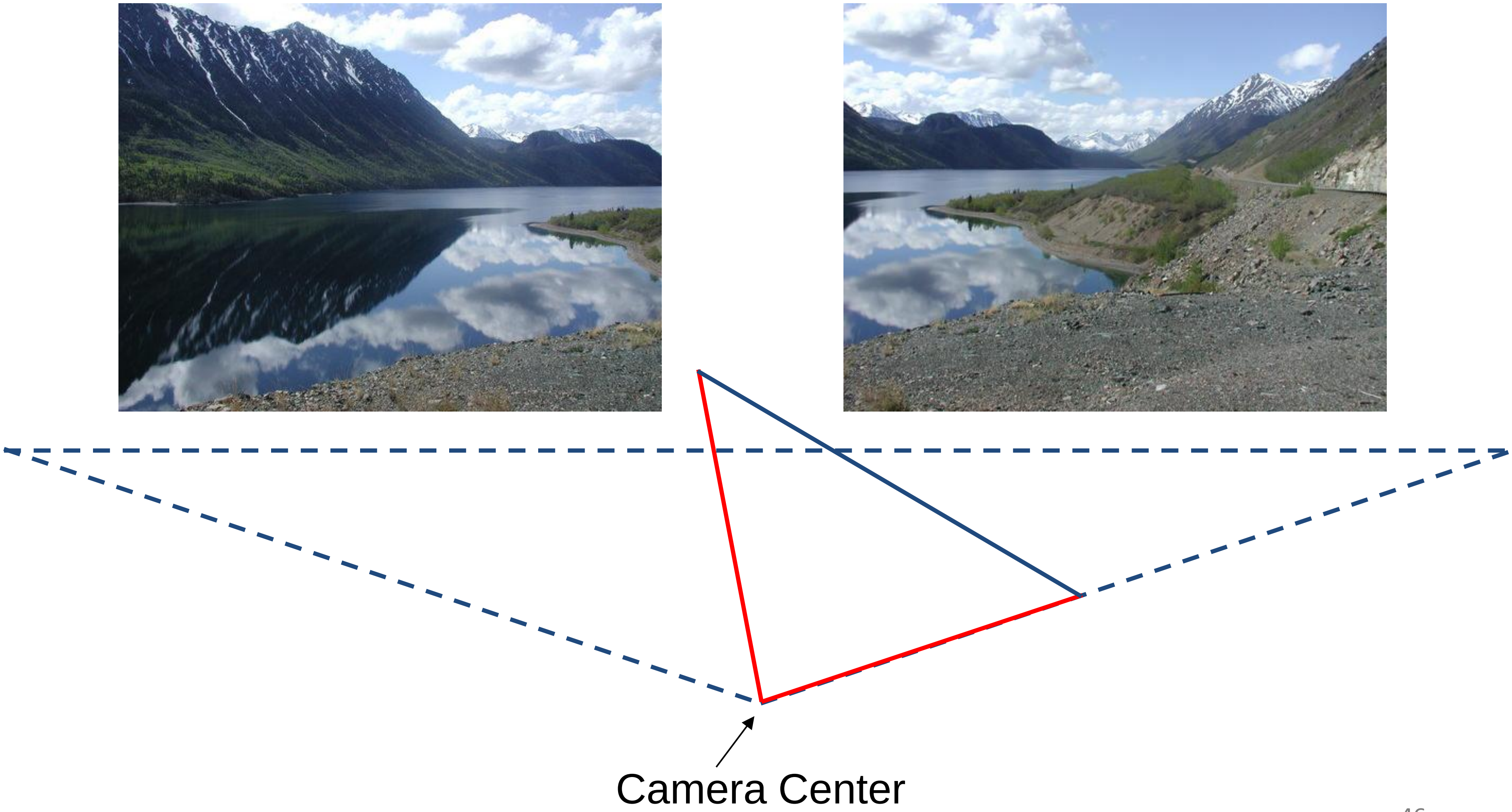
Example



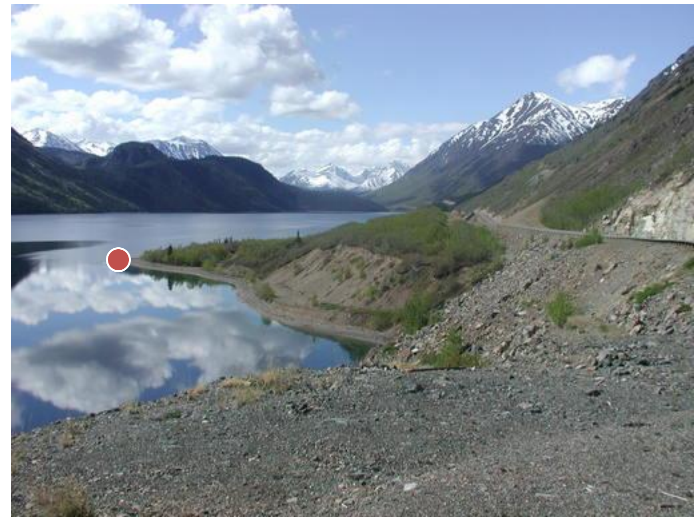
Example

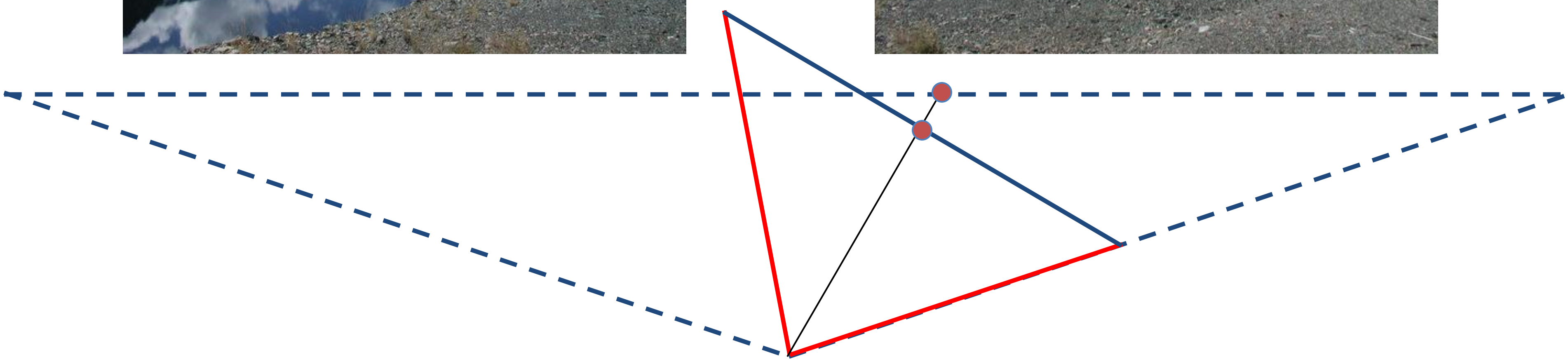


Example



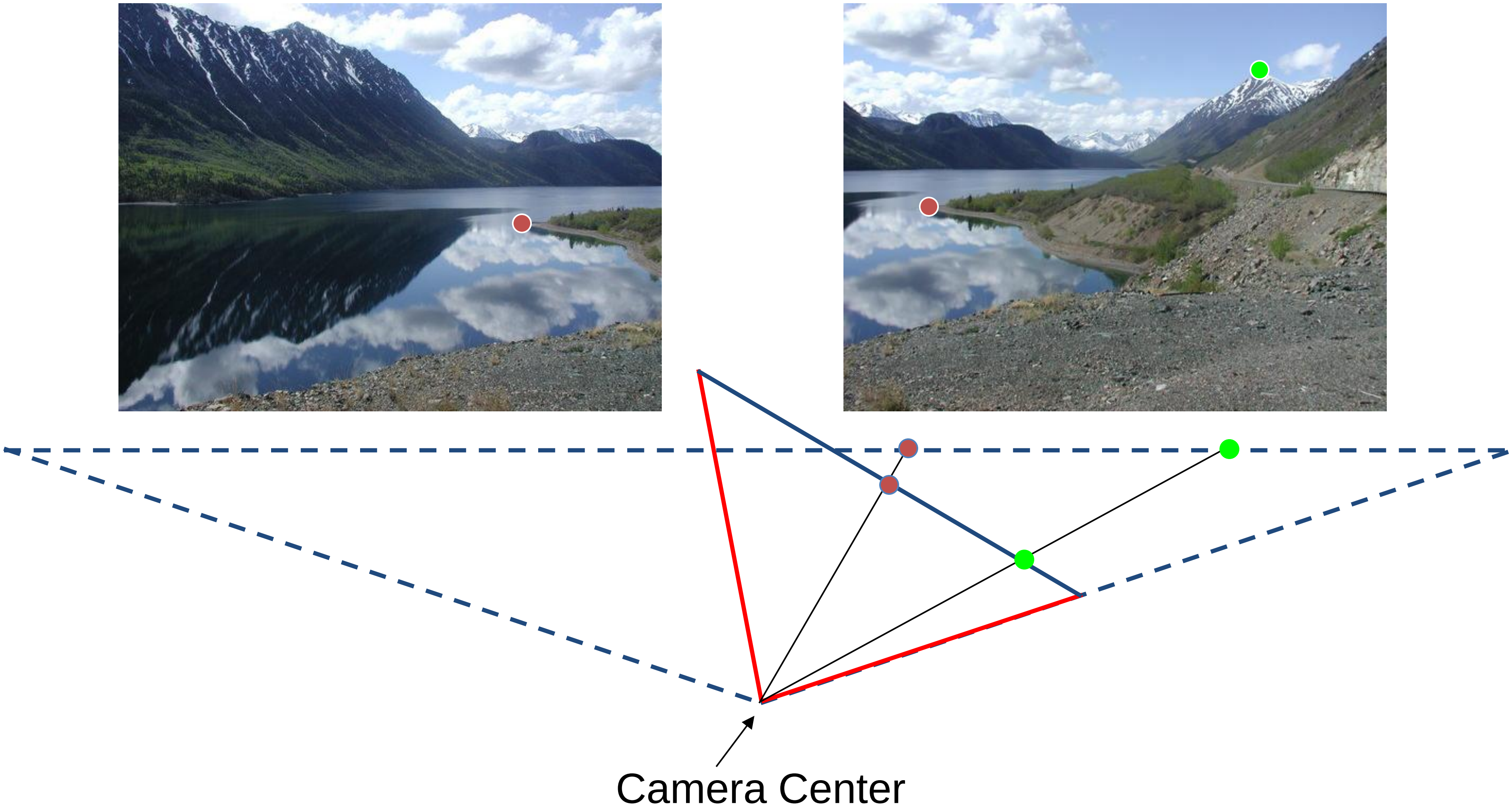
Example



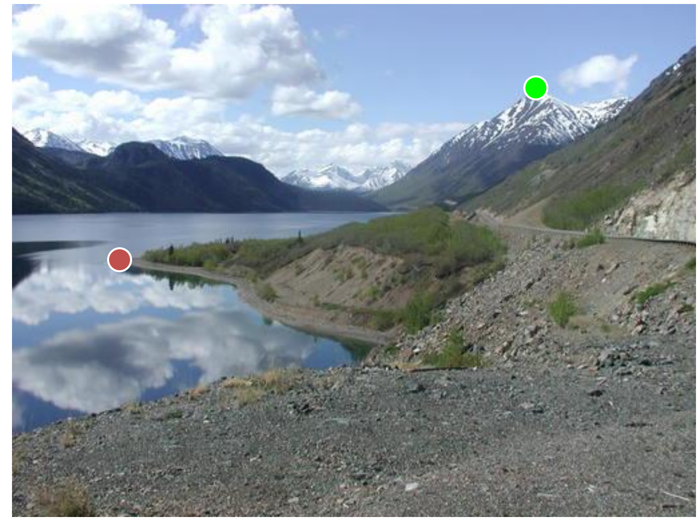


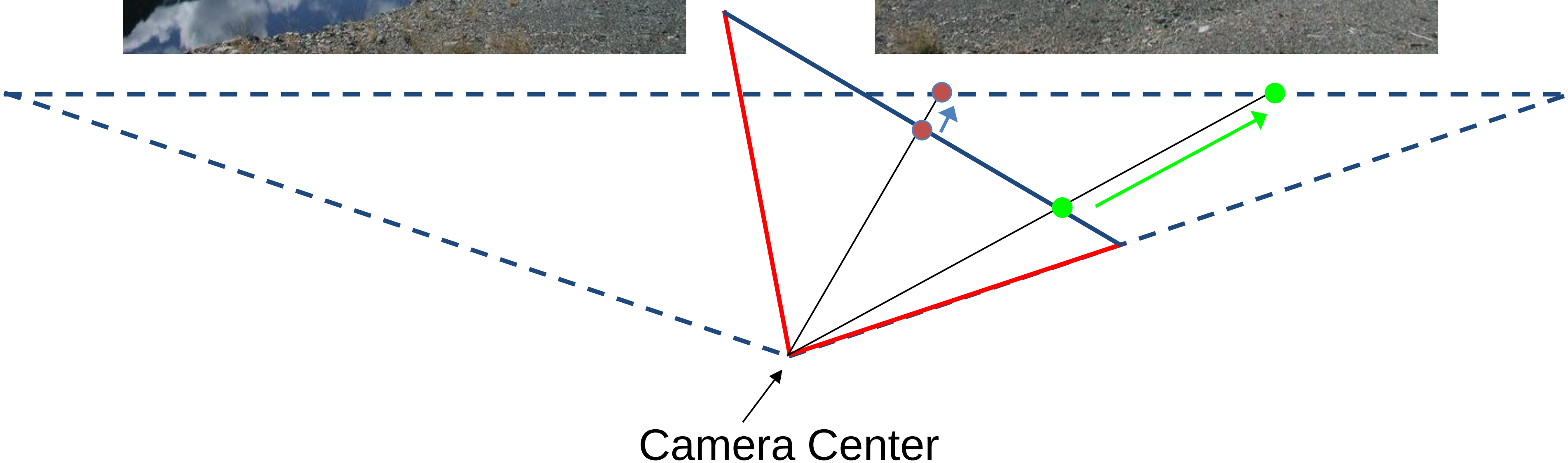
Camera Center

Example

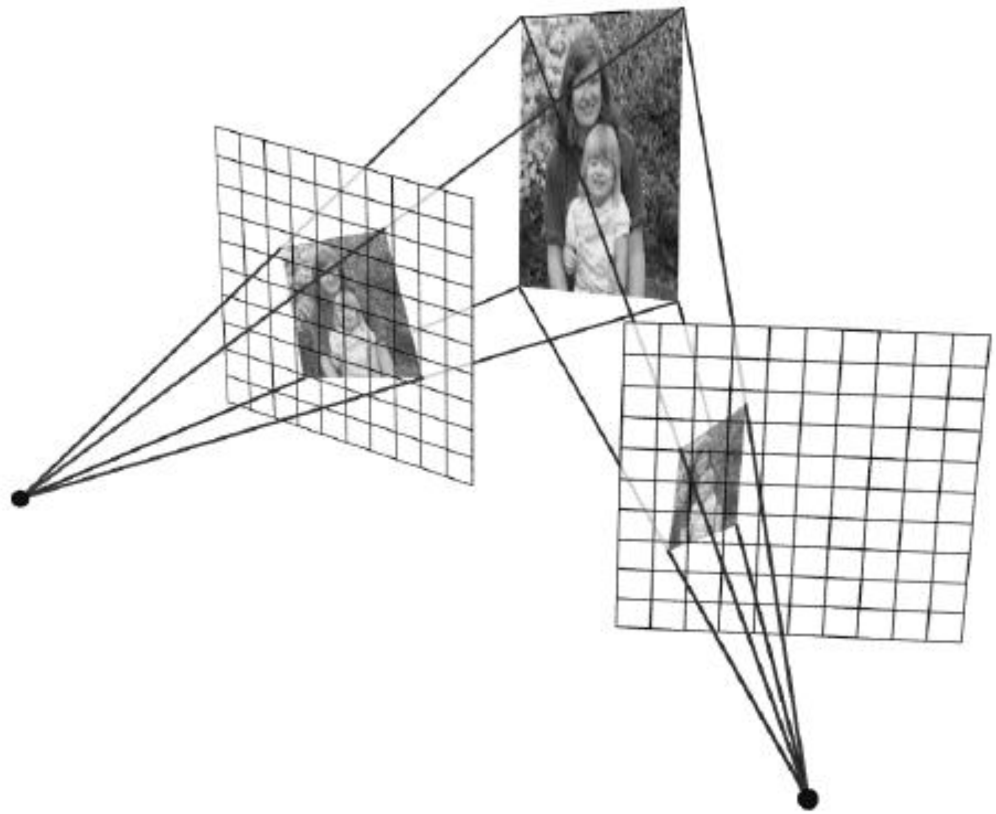


Example





The Underline Theory: Image Reprojection



- Although the theory says the process is a 3D reprojection.
- <u>We can also think of it as a **2D image warp** from one image to another.</u>

This is how we actually do it!

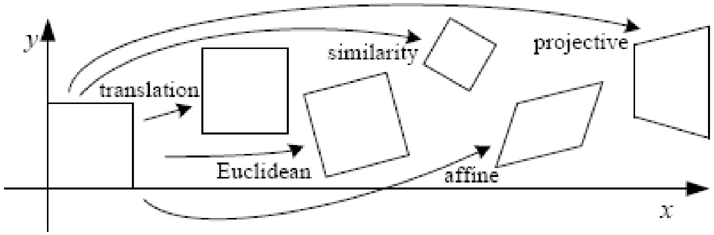
Compute Transformation: Motion Models

- What happens when we take two images with a camera and try to align them?

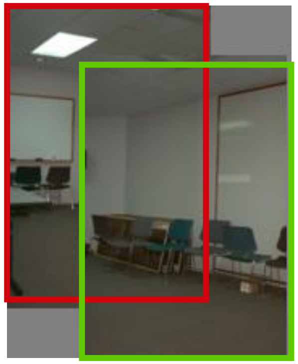
 

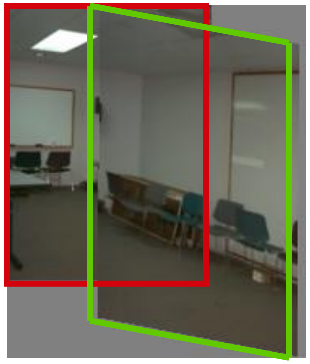
Translation? Rotation? Scale? Affine? Projective?

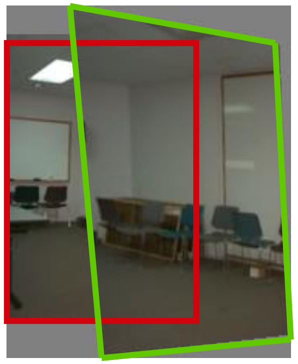
The Transformation: Motion Models



Translation | **Affine** | **Projective**







2 unknowns | **6 unknowns** | **8 unknowns**

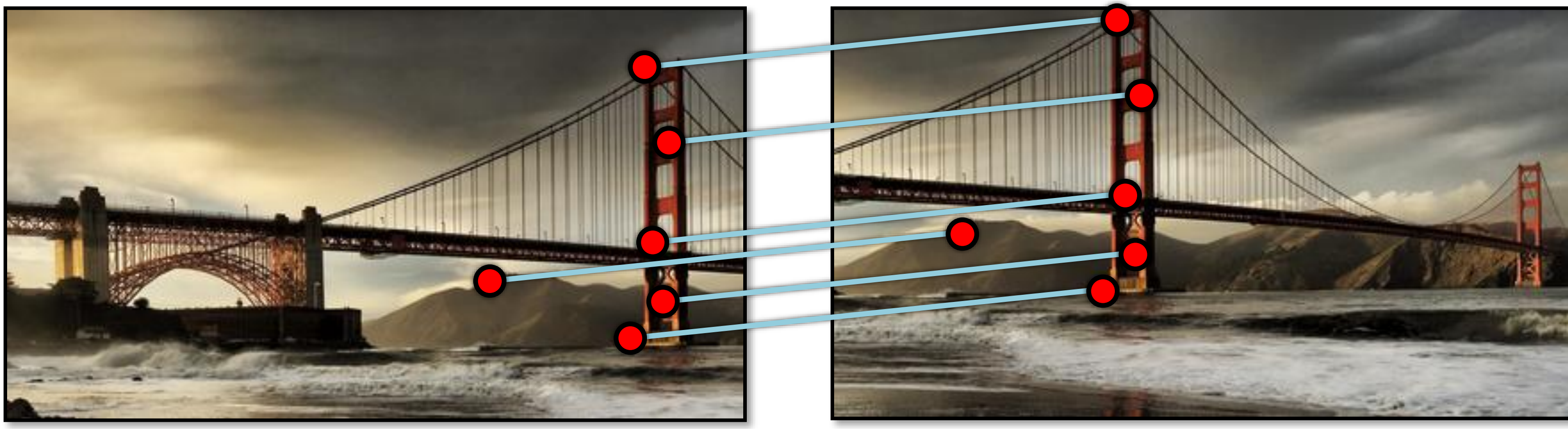
Finding the Transformation

- Translation = 2 degrees of freedom
- Similarity = 4 degrees of freedom
- Affine = 6 degrees of freedom
- Projective (Homography) = 8 degrees of freedom

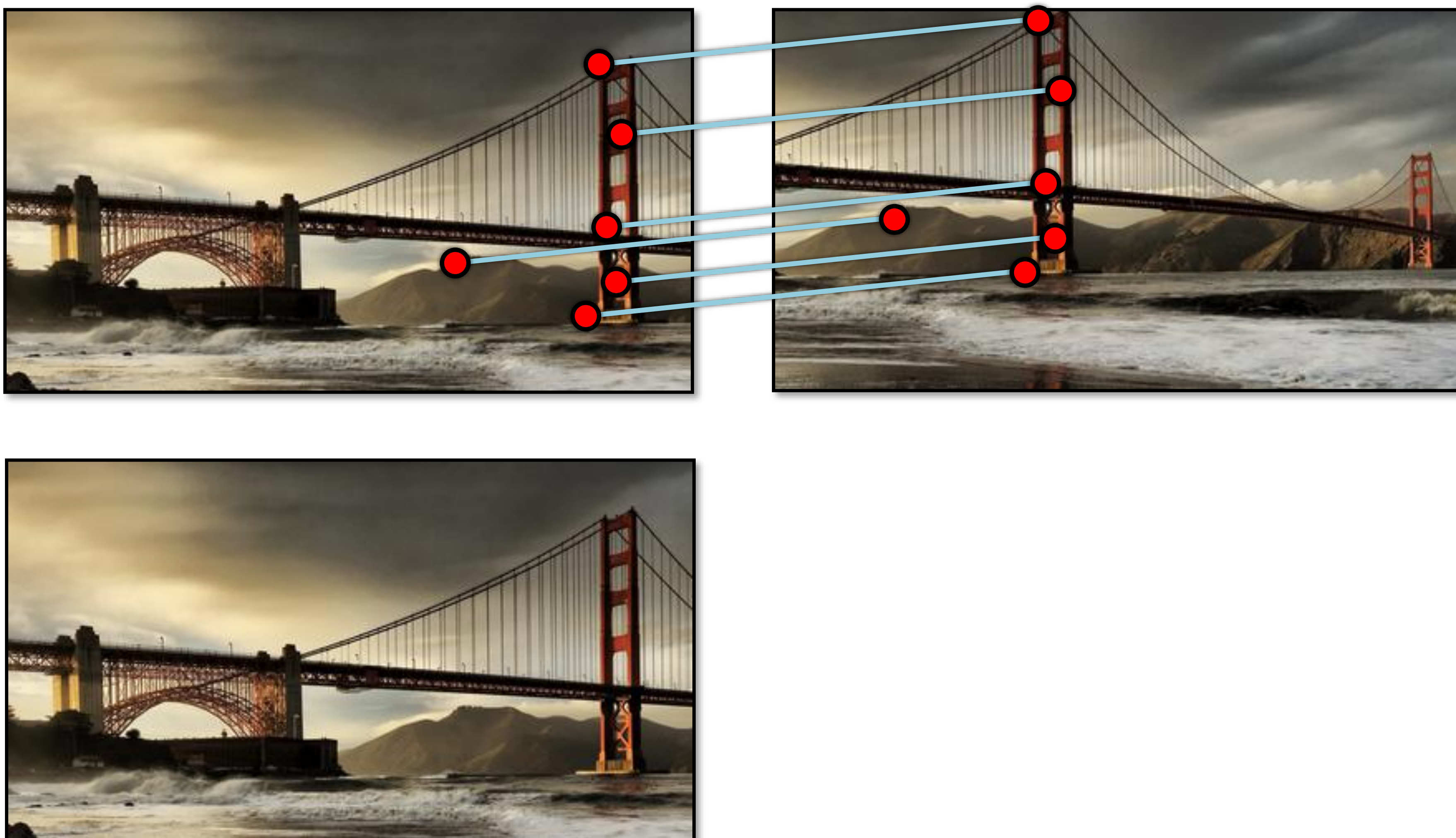
- How many corresponding points do we need to solve?

Simple Case: Translations

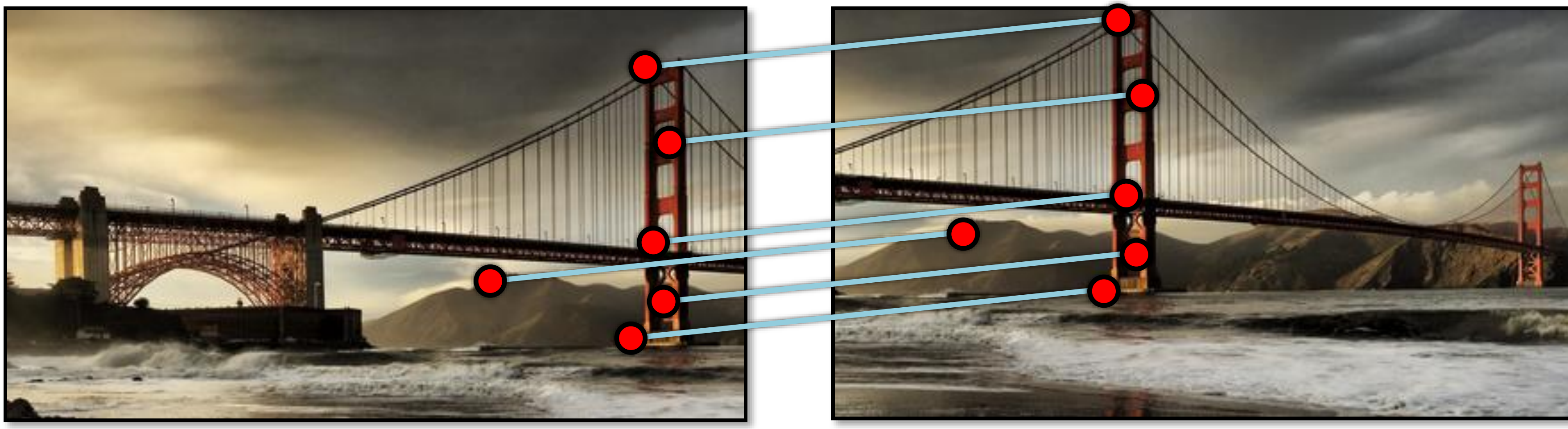
Simple Case: Translations



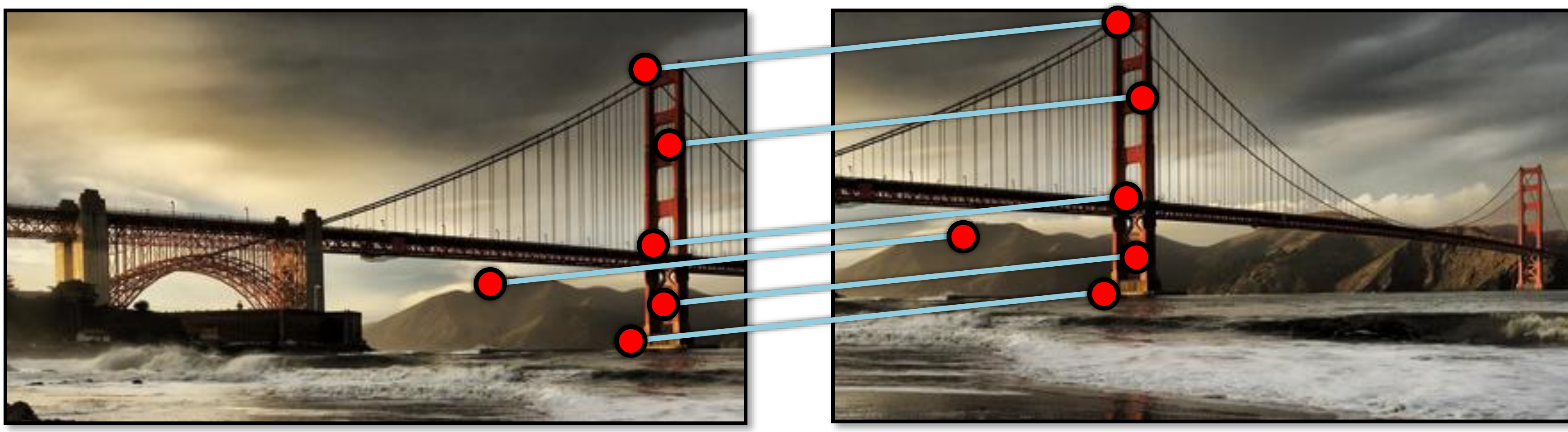
Simple Case: Translations



Simple Case: Translations

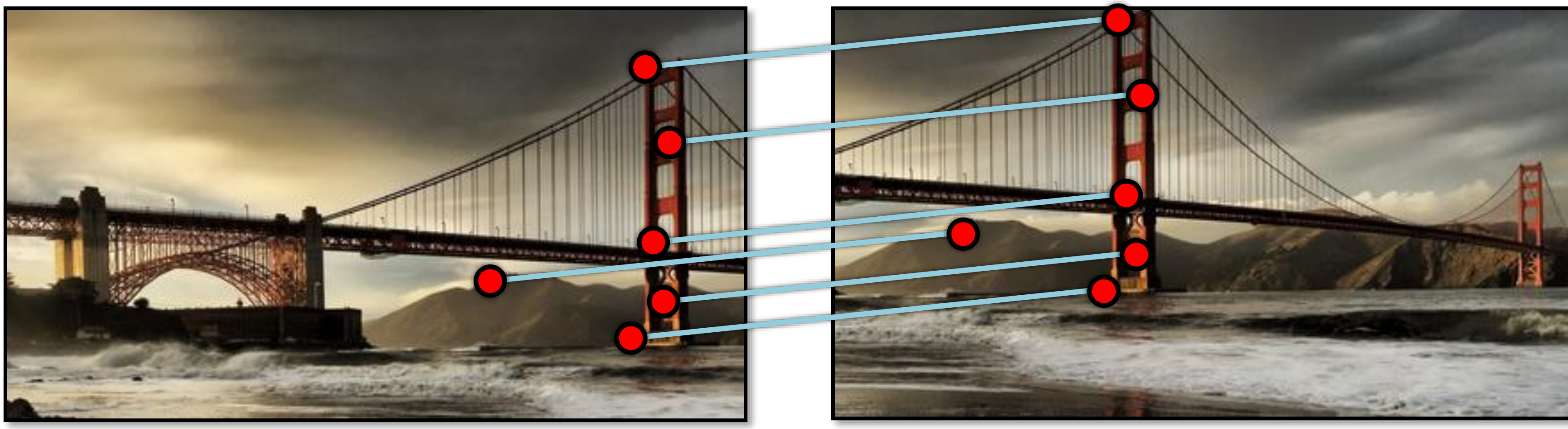


Simple Case: Translations



$(\mathbf{x}_t, \mathbf{y}_t)$

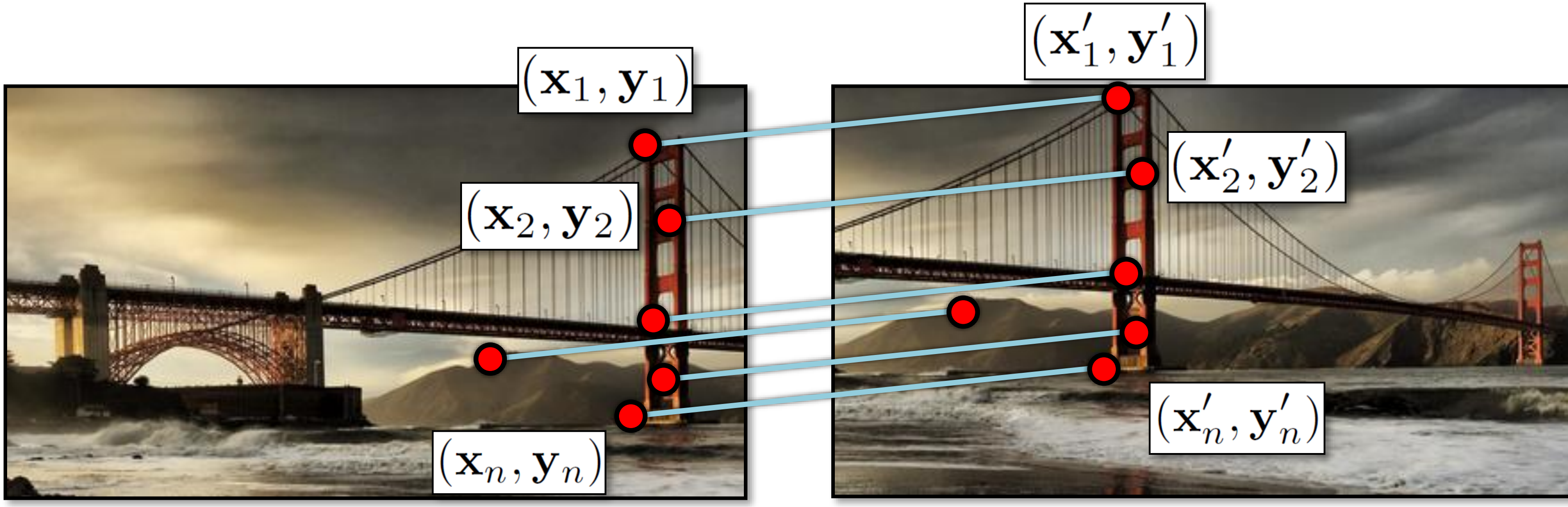
Simple Case: Translations



$(\mathbf{x}_t, \mathbf{y}_t)$

How do we solve for $(\mathbf{x}_t, \mathbf{y}_t)$?

Simple Case: Translations



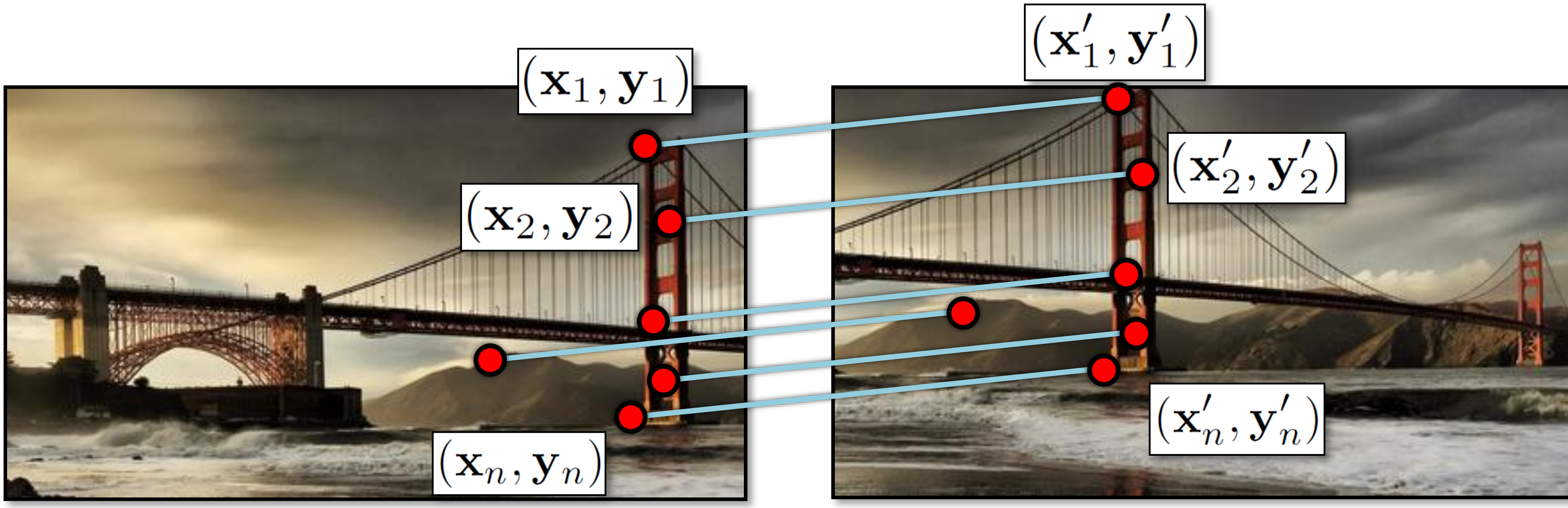
For each pair of matching points, we have

$$\mathbf{x}_i + \mathbf{x_t} = \mathbf{x}'_i$$

$$\mathbf{y}_i + \mathbf{y_t} = \mathbf{y}'_i$$

- System of linear equations
  - What are the knowns? Unknowns?
  - How many unknowns? How many equations (per match)?

Simple Case: Translations



For each pair of matching points, we have

$$\begin{aligned} \mathbf{x}_i + \mathbf{x_t} &= \mathbf{x}'_i \\ \mathbf{y}_i + \mathbf{y_t} &= \mathbf{y}'_i \end{aligned}$$

- The problem: more equations than unknowns.
  - "Overdetermined" system of equations.
  - **We find the *least square* solution.**

Least Squares Formulation

- For each point $(\mathbf{x}_i, \mathbf{y}_i)$

$$
\begin{aligned}
\mathbf{x}_i + \mathbf{x_t} &= \mathbf{x}_i' \\
\mathbf{y}_i + \mathbf{y_t} &= \mathbf{y}_i'
\end{aligned}
$$

- we define the *residuals* as

$$
\begin{aligned}
r_{\mathbf{x}_i}(\mathbf{x}_t) &= (\mathbf{x}_i + \mathbf{x}_t) - \mathbf{x}_i' \\
r_{\mathbf{y}_i}(\mathbf{y}_t) &= (\mathbf{y}_i + \mathbf{y}_t) - \mathbf{y}_i'
\end{aligned}
$$

Least Squares Formulation

- Goal: minimize sum of squared residuals.

$$C(\mathbf{x}_t, \mathbf{y}_t) = \sum_{i=1}^{n} \left(r_{\mathbf{x}_i}(\mathbf{x}_t)^2 + r_{\mathbf{y}_i}(\mathbf{y}_t)^2 \right)$$

- "Least squares" solution.

Solving for Translations

- Using least squares

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

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

Least Squares

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

- Find **t** that minimizes

$$||\mathbf{A}\mathbf{t} - \mathbf{b}||^2$$

- To solve, form the *normal equations*

$$\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{t} = \mathbf{A}^{\mathrm{T}}\mathbf{b}$$

$$\mathbf{t} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b}$$

Least Squares: Example

- $(x_1,y_1) = (1,1)$, $(x_1',y_1') = (4.3,14.5)$, $(x_2,y_2) = (2,2)$, $(x_2',y_2') = (1.9,-3)$

```
A =

     1     0
     0     1
     1     0
     0     1
```

```
b =

    3.3000
   13.5000
   -0.1000
   -5.0000
```

- Least square solution is:

```
>> inv(A'*A)*A'*b

ans =

    1.6000
    4.2500
```

Affine Transformations

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

- How many unknowns?

- How many matches do we need?

Affine Transformations

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + c \\ dx + ey + f \\ 1 \end{bmatrix}$$

- How many unknowns?
  - 6 unknowns
- How many matches do we need?
  - At least 6 equations, so 3 matches.

Affine Transformations

- Residuals:

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

- 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 Transformations

- Matrix form

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

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

Solving for Homographies

$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

Why is this now a variable and not just 1?

- A homography is a projective object, in that it has no scale. It is represented by the above matrix, up to scale.
- One way of fixing the scale is to **set one of the coordinates to 1**, though that choice is arbitrary.
- But that's what most people do.

Solving for Homographies

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

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

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

Why the division?

$$x_i'(h_{20}x_i + h_{21}y_i + h_{22}) = h_{00}x_i + h_{01}y_i + h_{02}$$

$$y_i'(h_{20}x_i + h_{21}y_i + h_{22}) = h_{10}x_i + h_{11}y_i + h_{12}$$

Solving for Homographies

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

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

This is just for one pair of points.

Direct Linear Transforms (n points)

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

Defines a least squares problem: minimize $\|\mathbf{A}\mathbf{h} - \mathbf{0}\|^2$

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

Direct Linear Transforms

- Why could we not solve for the homography in exactly the same way we did for the affine transform, ie.

$$\mathbf{t} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b}$$

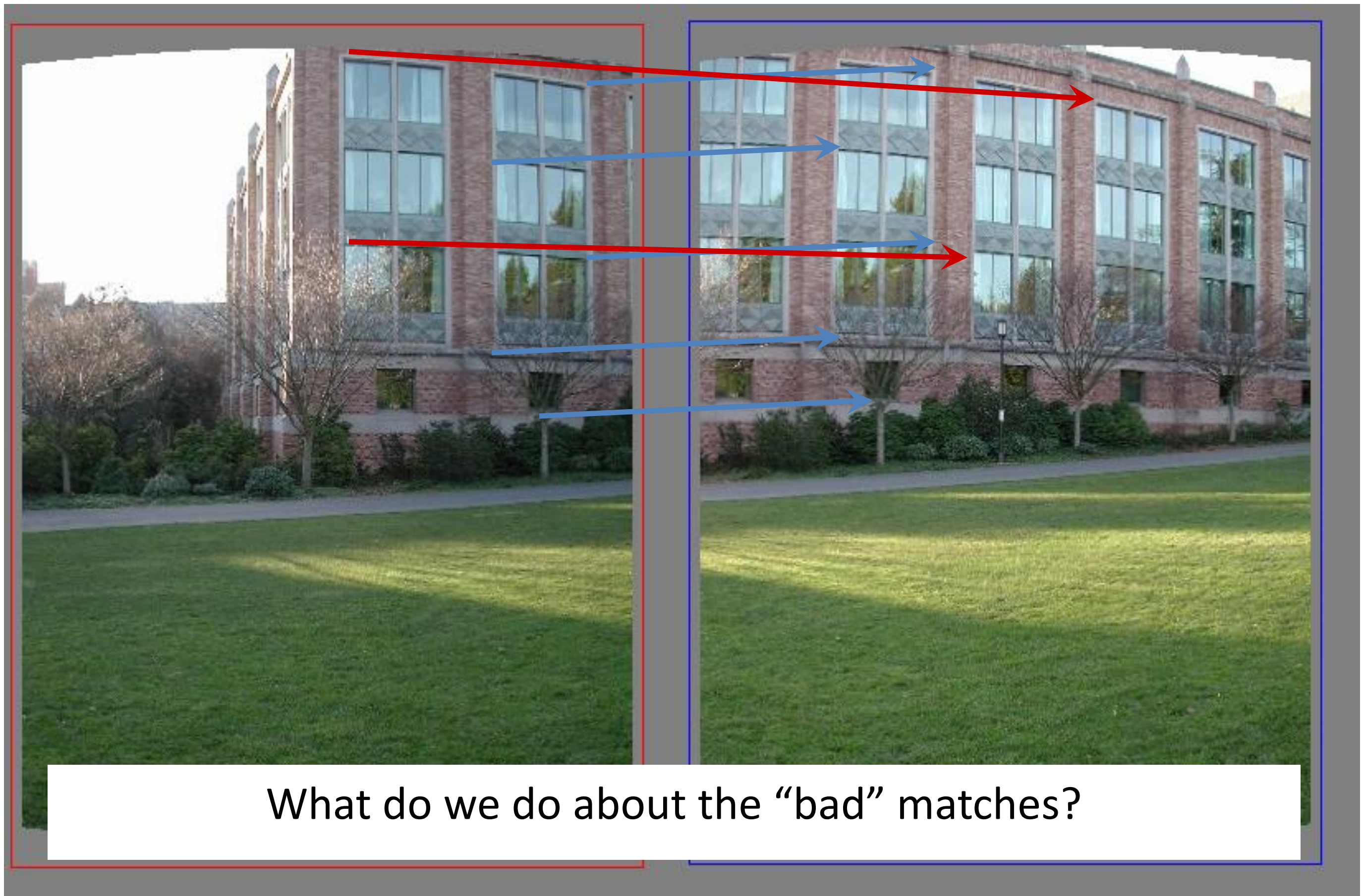
The Answer

- For an affine transform, we have equations of the form $Ax_i + b = y_i$, solvable by linear regression.
- For the homography, the equation is of the form

  $H\tilde{x}_i \sim \tilde{y}_i$ (homogeneous coordinates)

and the ~ means it holds only up to scale. The affine solution does not hold.

Matching features



What do we do about the “bad” matches?

RAndom SAmple Consensus

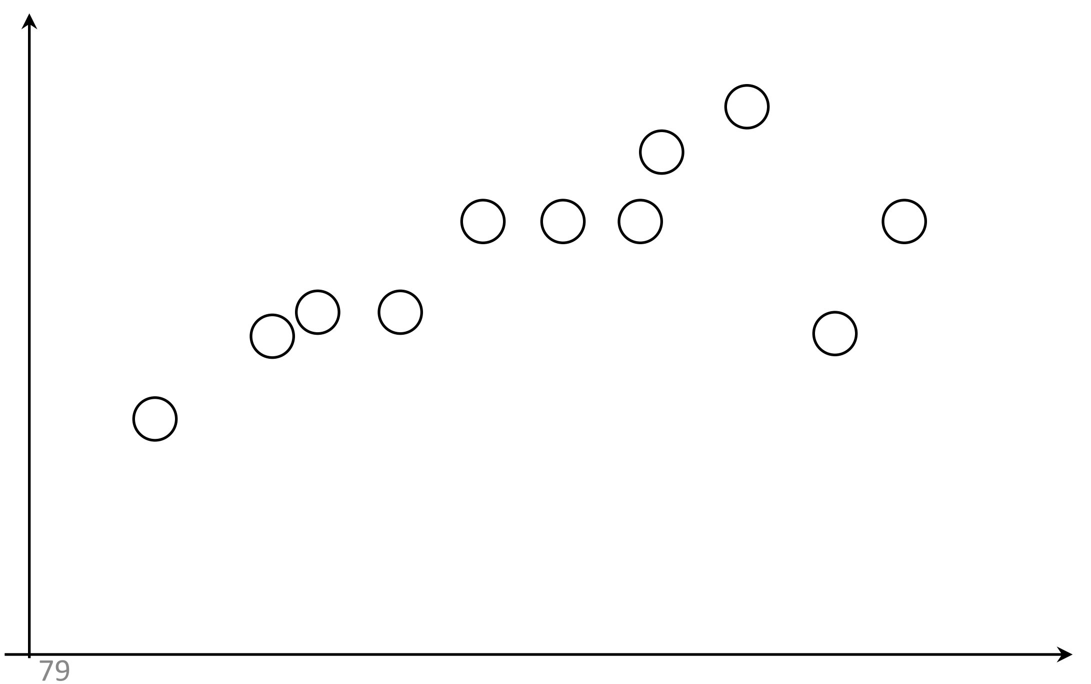
- RANSAC loop for estimating homography:

1. Select four feature pairs (at random)
2. Compute homography $\boldsymbol{H}$ (exact)
3. Compute inliers where $\|p_i', \boldsymbol{H}\, p_i\| < \varepsilon$

- Keep the largest set of inliers
- Re-compute least-squares $\boldsymbol{H}$ estimate using all of the inliers

Simple example: fit a line

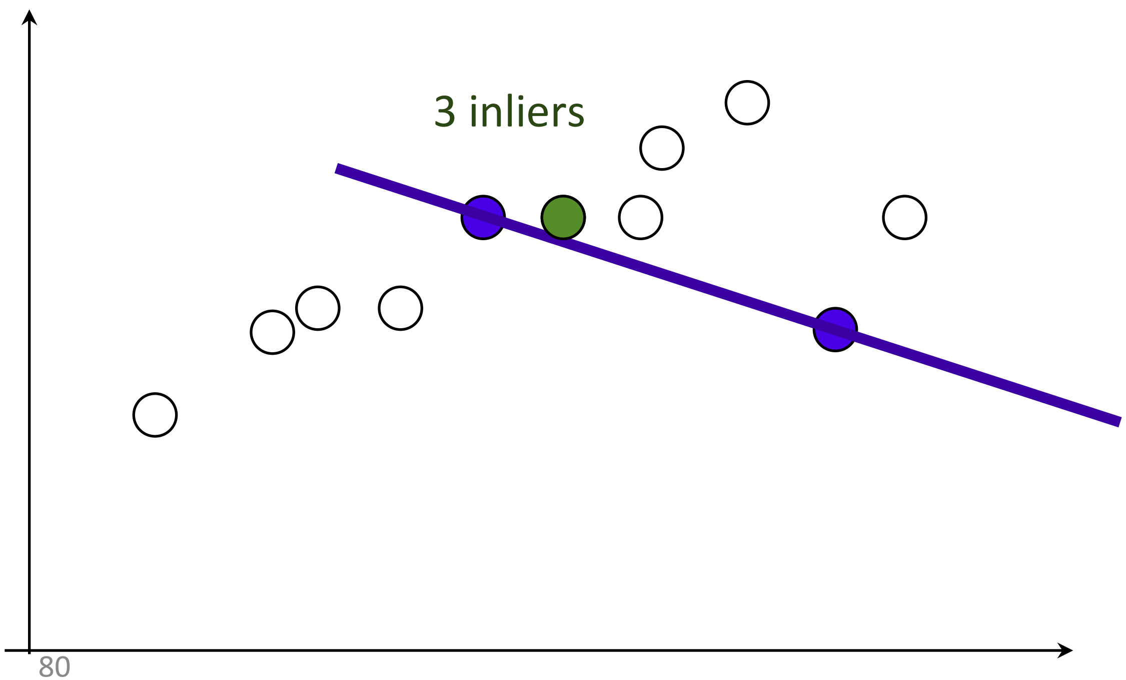
- Rather than homography H (8 numbers) fit y=ax+b (2 numbers a, b) to 2D pairs



79

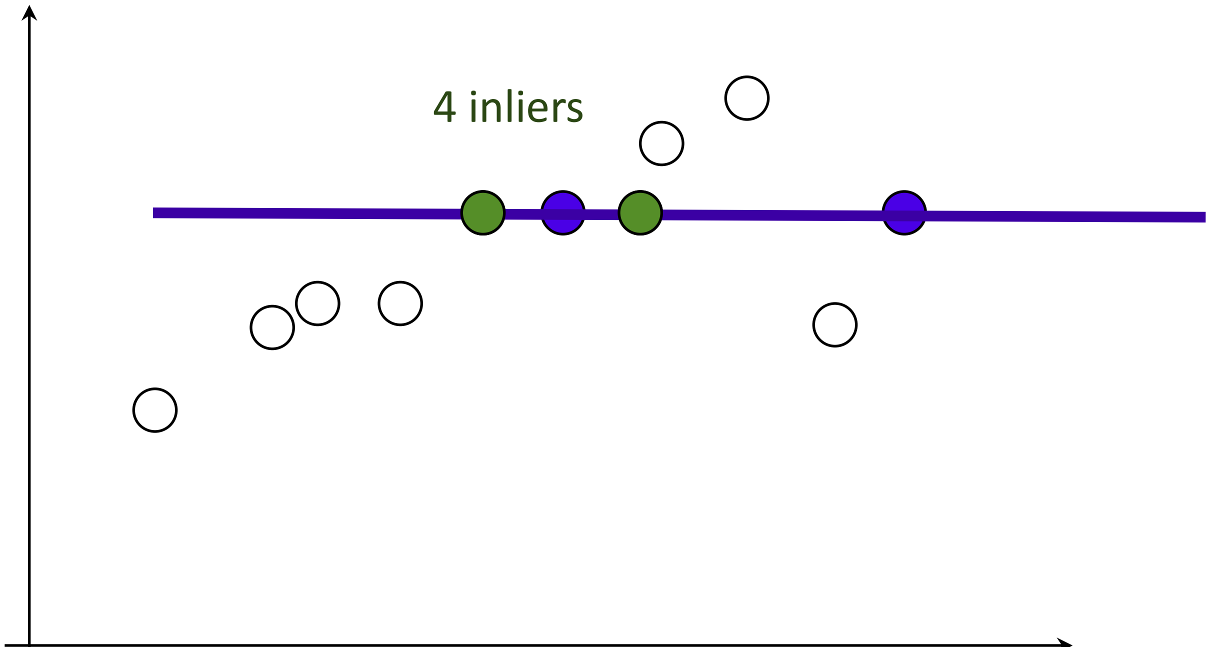
Simple example: fit a line

- Pick 2 points
- Fit line
- Count inliers



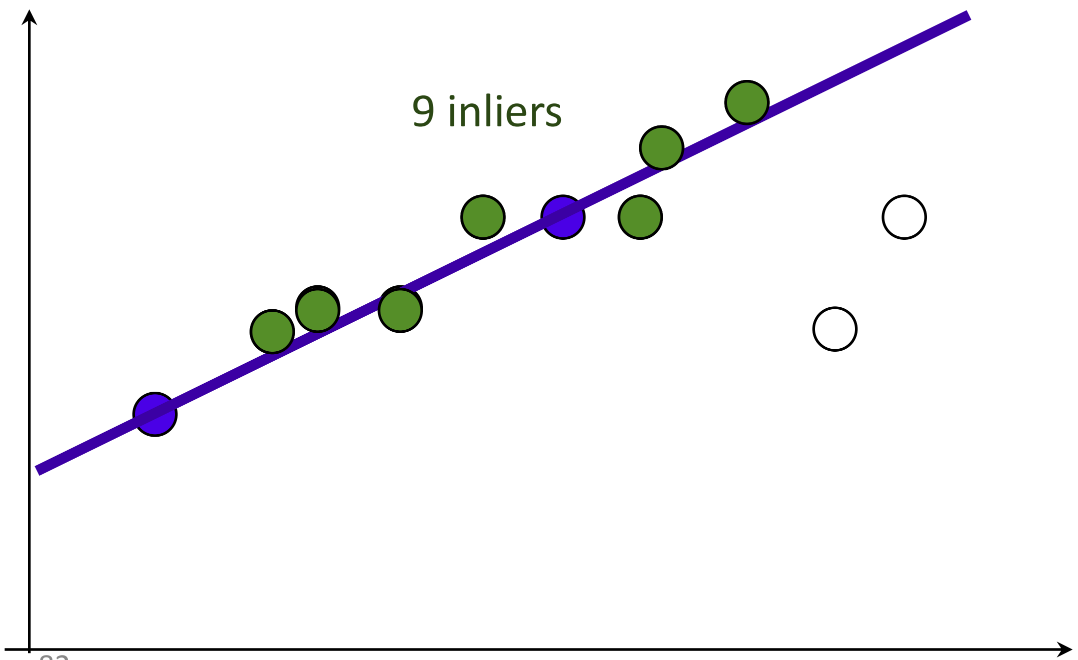
Simple example: fit a line

- Pick 2 points
- Fit line
- Count inliers



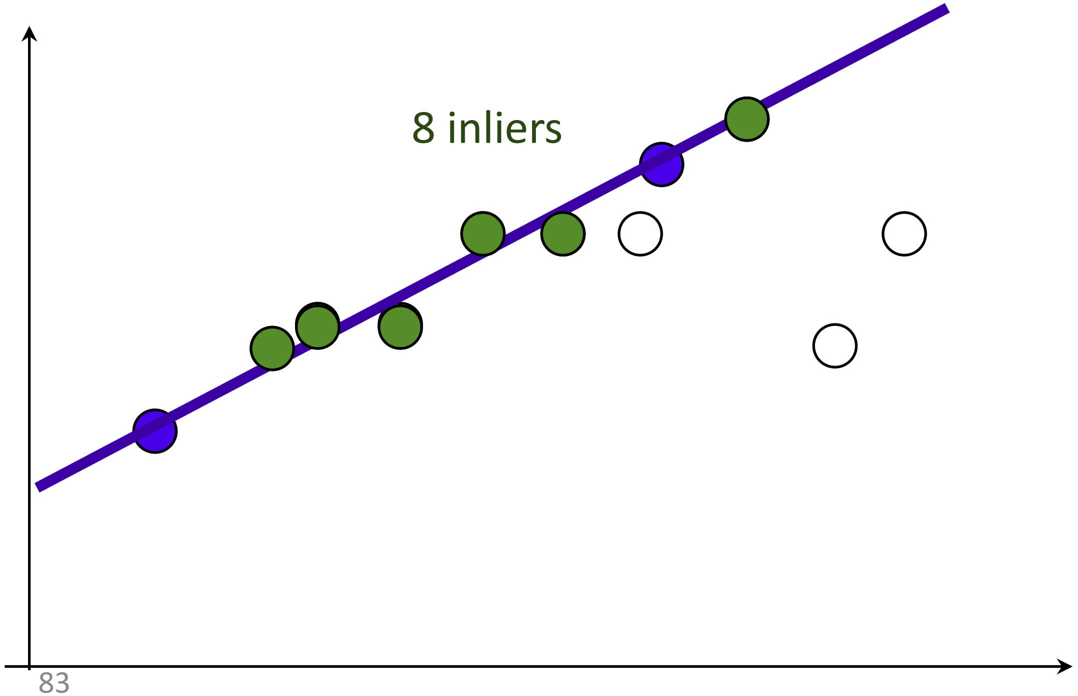
Simple example: fit a line

- Pick 2 points
- Fit line
- Count inliers



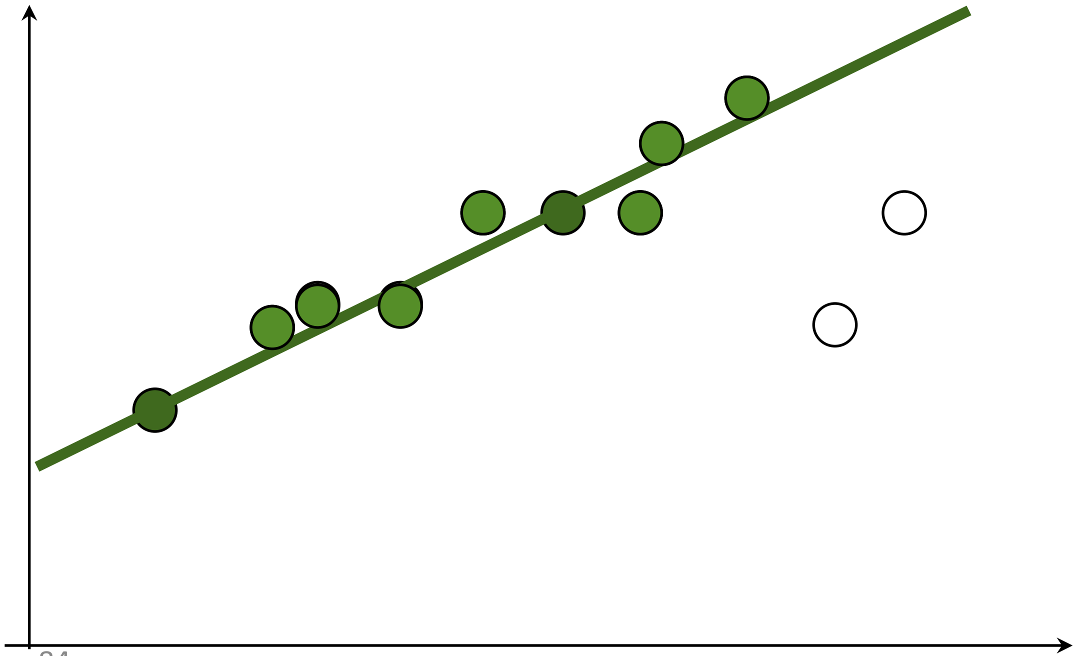
Simple example: fit a line

- Pick 2 points
- Fit line
- Count inliers

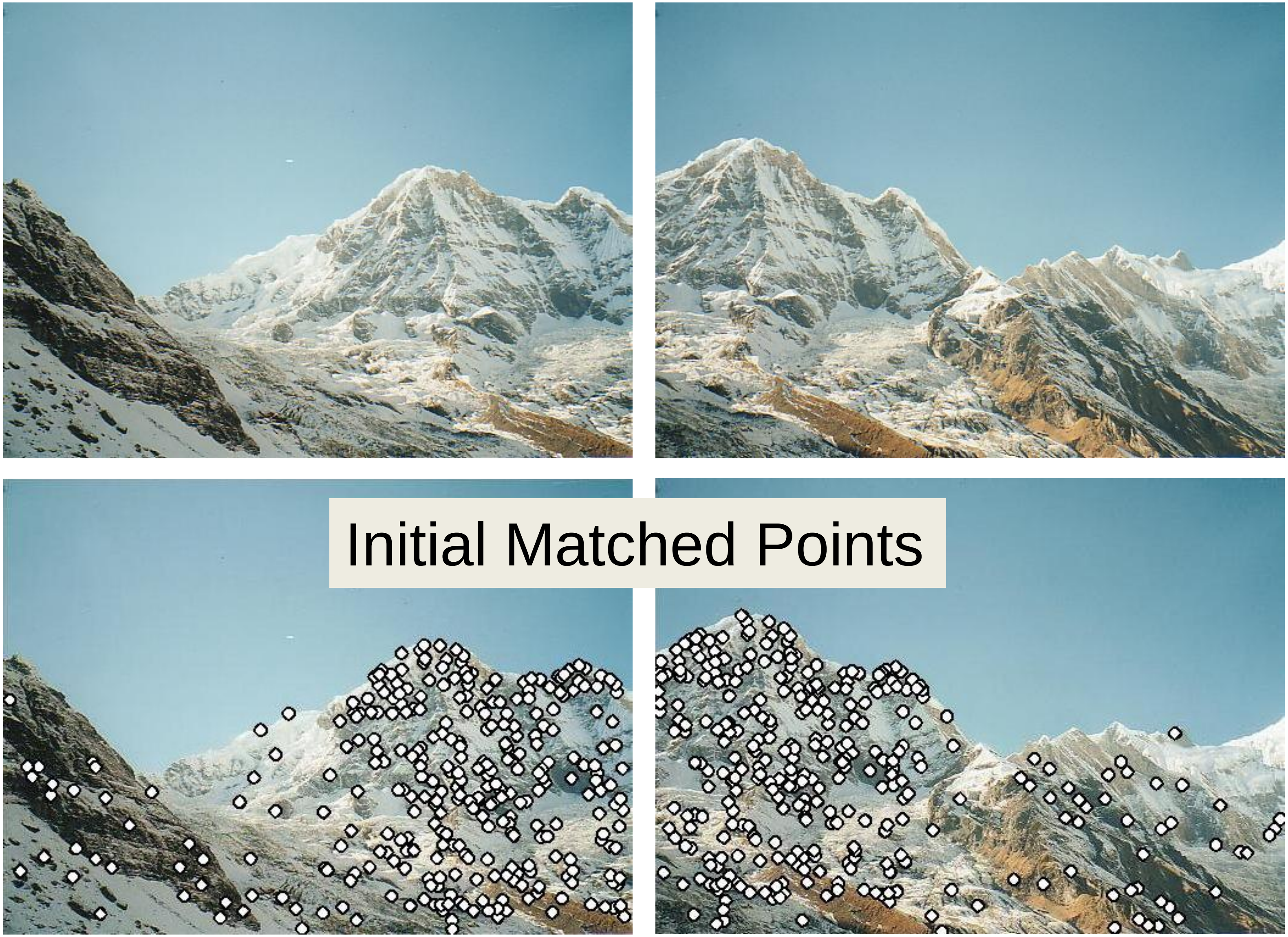


Simple example: fit a line

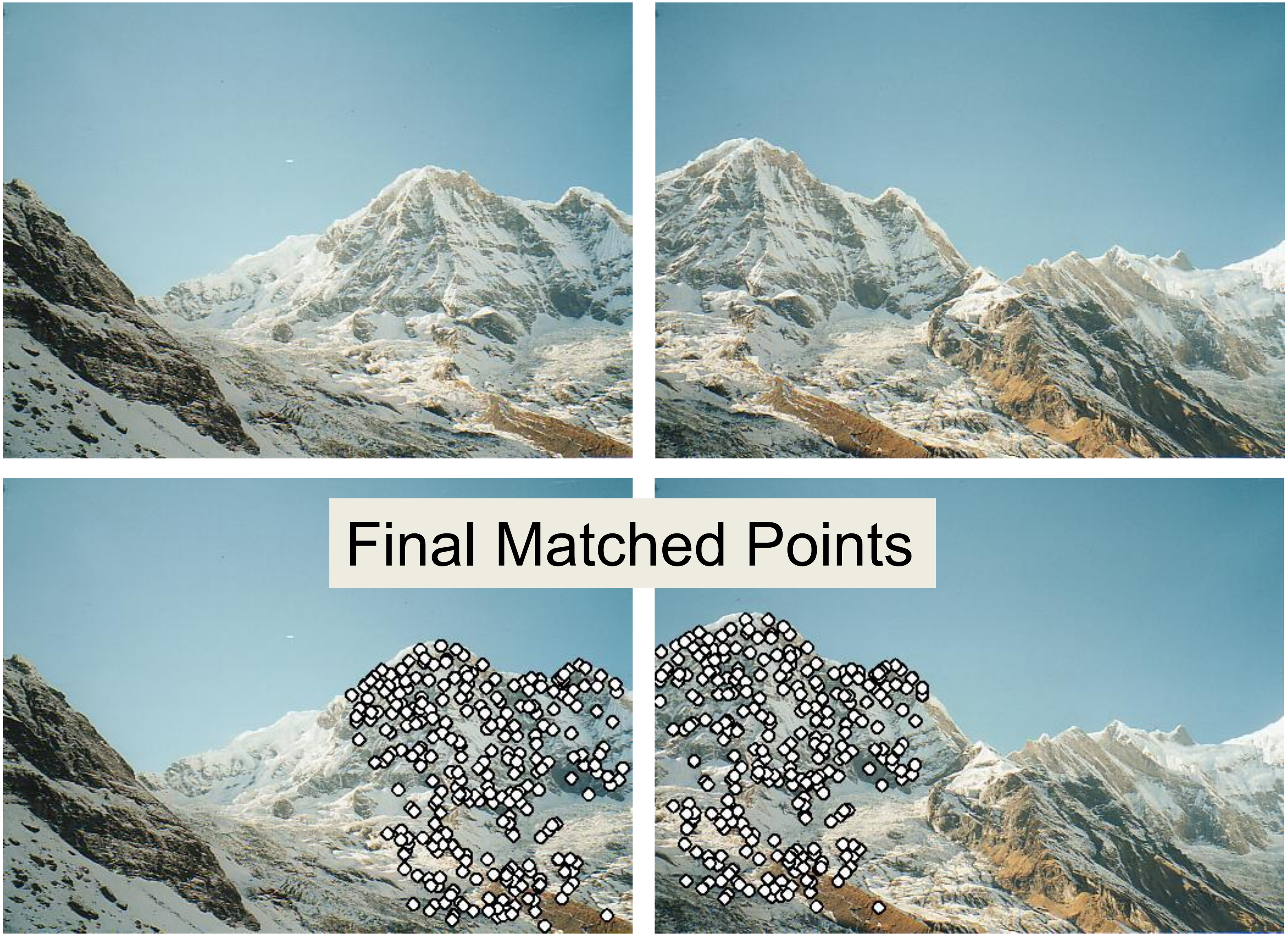
- Use biggest set of inliers
- Do least-square fit



RANSAC for Homography



RANSAC for Homography



Conclusion

- Geometric transformation
  - Translation, rotation, scale, affine, homography…
- The idea of image stitching
  - The overall process.
  - Finding transformation given an image pair.
  - Least square solution, etc.
  - RANSAC.