# Image Stitching - First Principles of Computer Vision

Michel Liao

January 2024

## Contents



# 1 Introduction

The following notes were taken based off Columbia University Professor Shree Nayar's lecture series on image stitching.

The fundamental question is how can we combine multiple photos to create a larger photo? (Panoramas)

Derivation logic behind certain formulas are detailed in Prof. Nayar's videos.

# 2 2x2 Image Transformations

## 2.1 Image Manipulation

- There are two classes of image manipulation:

- **Image filtering:** changing range (brightness)

$$g(x, y) = T_r(f(x, y)).$$

- **Image warping:** changing domain (location)

$$g(x, y) = f(T_d(x, y))$$

  * The transformation $T_d$ is the same over the entire image.
  * Examples include translation, rotation, scaling and aspect ratio, etc.

## 2.2 2x2 Linear Transformations

- We have a pixel $\boldsymbol{p}_1 = (x_1, y_1)$ in the original image that transforms into $\boldsymbol{p}_2 = (x_2, y_2)$, where

$$\boldsymbol{p}_2 = T\boldsymbol{p}_1.$$

  –

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}.$$

- Note: We can find the inverses of all the transformation matrices because $T$ is invertible.

- We can **scale** the image, where $x_2 = ax_1$ and $y_2 = by_1$ with the transformation matrix

$$S = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}.$$

- We can **rotate** the image by $\theta$ in the counterclockwise direction with the matrix

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

- We can **horizontally skew** with the matrix

$$S_x = \begin{bmatrix} 1 & m_x \\ 0 & 1 \end{bmatrix}$$

  or **vertically skew with the matrix**

$$S_y = \begin{bmatrix} 1 & 0 \\ m_y & 1 \end{bmatrix}.$$

- We can **mirror** about the y-axis with

$$S_x = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}.$$

- Remark: Any transformation of the form

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

  - Maps the origin to the origin
  - Maps lines to lines
  - Maintain parallel lines as parallel
  - Are closed under composition.
    * If $T_{21}$ transforms $\boldsymbol{p}_1$ to $\boldsymbol{p}_2$ and $T_{32}$ transforms $\boldsymbol{p}_2$ to $\boldsymbol{p}_3$, then

$$T_{31} = T_{32}T_{21}.$$

# 3 3x3 Image Transformations

- We can't represent a translation with a 2x2 matrix.
- The **homogeneous** representation of a 2D point $\boldsymbol{p} = (x, y)$ is a 3D point $\tilde{p} = (\tilde{x}, \tilde{y}, \tilde{z})$. The third coordinate $\tilde{z} \neq 0$ such that

$$x = \frac{\tilde{x}}{\tilde{z}}, y = \frac{\tilde{y}}{\tilde{z}}.$$

  -
$$\boldsymbol{p} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{z}x \\ \tilde{z}y \\ \tilde{z} \end{bmatrix} \equiv \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{bmatrix} \equiv \tilde{p}.$$

- Then, we represent the translation $x_2 = x_1 + t_x$ and $y_2 = y_1 + t_y$ as

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{x}_2 \\ \tilde{y}_2 \\ \tilde{z}_2 \end{bmatrix} \equiv \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}.$$

- All of the 2x2 transformations can be represented as 3x3 transformations.
- Scaling, rotation, skew, and translation are all **affine transformations.** They satisfy the form

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{x}_2 \\ \tilde{y}_2 \\ \tilde{z}_2 \end{bmatrix} \equiv \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{y}_1 \\ \tilde{z}_1 \end{bmatrix}.$$

- Remark: Affine transformations
  - Don't necessarily map the origin to the origin
  - Maps lines to lines

- Maintain parallel lines as parallel
  - Are closed under composition.

- If the last row of the transformation matrix isn't $0, 0, 1$ (can be anything), then we have a **projective transformation,** also known as a **homography.**
  - Homographies map one plane to another through a point, like imaging a plane through a pinhole (lens).
  - Homographies can only be defined up to a scale:

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{y}_1 \\ \tilde{z}_1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{x}_2 \\ \tilde{y}_2 \\ \tilde{z}_2 \end{bmatrix} \equiv k \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{y}_1 \\ \tilde{z}_1 \end{bmatrix}.$$

    We can fix $k$ such that $\sqrt{\sum (h_{ij})^2} = 1$.

- Remark: Projective transformations
  - Don't necessarily map the origin to the origin
  - Map lines to lines
  - Don't necessarily maintain parallel lines as parallel
  - Are closed under composition.

# 4 Computing Homography

- If two planes share the same center of projection, you can compute the homography between the two images to map them to the same plane.
- Given a set of matching points between a source image (that gets warped to the destination) and destination image, find the homography $H$ that best "agrees" with the matches.
  - Only possible if:
    * You capture a 3D scene from the same viewpoint
    * You capture a plane from any viewpoint
    * You capture a 3D scene that's far away (acts as if you're capturing from the same viewpoint)
- $$\begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{x}_d \\ \tilde{y}_d \\ \tilde{z}_d \end{bmatrix} \equiv \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix}.$$
  - There are 9 unknowns, but 8 degrees of freedom since homographies are equivalent up to a scale factor $k$.
  - We need at least 4 matching points, but more is always better.
  - We use this system to created an overdetermined system to find $h$.

# 5 Dealing with Outliers: RANSAC

- Not all pairs that match aren't the same point in 3D (**outliers**).
- If the number of outlier pairs is less than half of the total pairs, then you can use **Random Sample Consensus (RANSAC)**.
- General RANSAC Algorithm:
  - Randomly choose $s$ samples. Typically, $s$ is the minimum samples needed to fit a model.
    - For a homography, $s = 4$.
  - Fit the model to the randomly chosen samples.
  - Count the number $M$ inliers that fit the model within a measure error of $\epsilon$.
    - $\epsilon$ is the acceptable alignment error in pixels.
  - Repeat steps above $N$ times.
  - Choose the model that has the largest number $M$ of inliers.
    - Optional: Recompute the homography matrix with the new inliers.

# 6 Warping and Blending Images

## 6.1 Warping

- If a pixel in $f(x, y)$ is sent to its corresponding location $g(x, y) = f(T(x, y))$, (**forward warping**) the pixel can land unaligned with the center of a pixel and result in not all pixels in $g(x, y)$ being filled.
- Solution: **backward warping.**
  - Use forward warping to find the four corners of $f(x, y)$ in $g(x, y)$.
  - Apply the inverse transformation to a pixel in $g(x, y)$ to find its corresponding location in $f(x, y)$ and use that brightness value. If the pixel lands between pixels, use the nearest neighbor or interpolate (use neighbors in a small surrounding box to estimate the brightness value).

## 6.2 Blending

- After warping, you'll see hard seams between images because of vignetting and exposure differences.

- Use weighting functions:

$$I_{\text{blend}} = \frac{w_1 I_1 + w_2 I_2}{w_1 + w_2},$$

  where $I_{\text{blend}}$ is the intensity of a pixel after being blended.

- Use a weighting function that gives pixels closer to the edge a lower weight.