2D — Transformations (warping) 扭曲

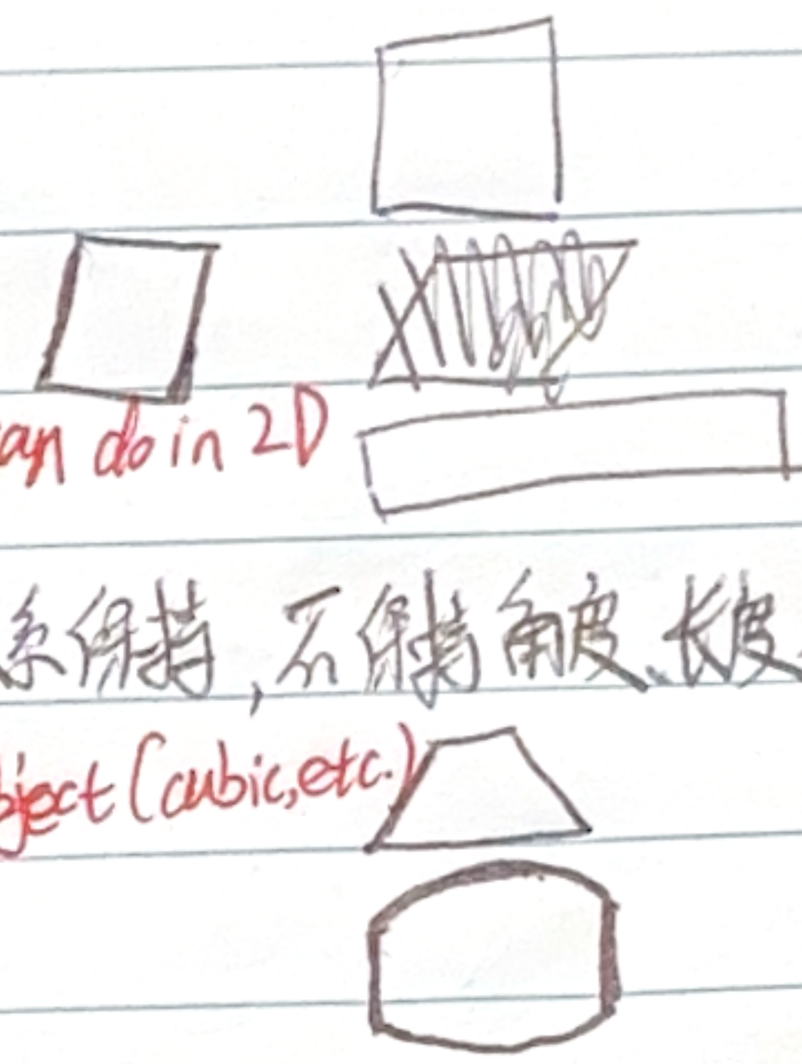
original: $f(x)$

difference
- filtering : change range $g(x) = T(f(x))$
- warping : change domain $g(x) = f(T(x))$

linear
- ① translation 平移 $\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$
- ② rotation $\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$ counterclockwise
- ③ aspect/scaling 纵横比例缩放 $\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$ } can do in 2D
- ④ affine 仿射变换 (直线,平行关系保持,不保持角度,长度) for 2D object (post, etc.) = translation + rotation + aspect + shear $\begin{bmatrix} 1 & sh_x \\ sh_y & 1 \end{bmatrix}$ $\begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix}$
- ⑤ perspective/homography 透视变换 for 3D object (cubic, etc.)

non-linear
- ⑥ cylindrical 柱面变换

aspect
a, d 缩放
b, c shear 斜拉 矩形→平行四边形



$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

normalization

→ homogeneous coordinates 齐次坐标 (使 + 常数项, aka 平移, 可用矩阵表示)

$(x, y) \leftrightarrow (w\cdot x, w\cdot y, w)$

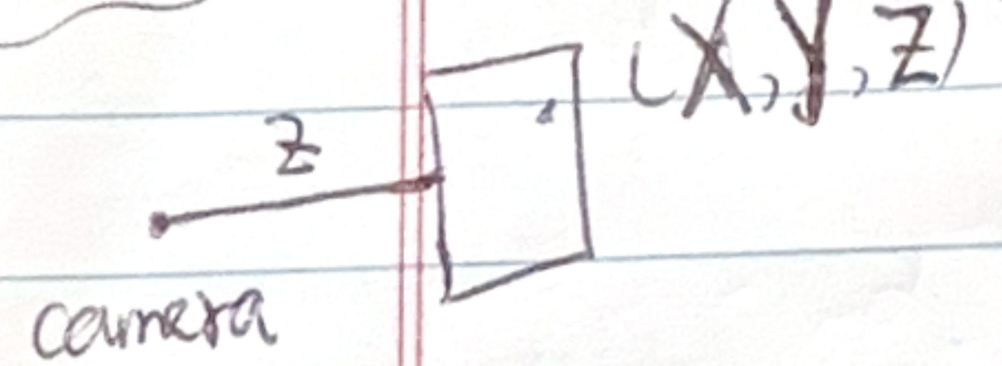
$w = a_{31}x + a_{32}y + 1$

$x'' = \frac{x'}{w}$

$y'' = \frac{y'}{w}$

definition of linear:
① additive : $T(x+y) = T(x) + T(y)$
② homogeneous : $T(a\cdot x) = a \cdot T(x)$

① $T(0) = 0$ & ② $T^n(x) = A^n \cdot x$



3D: (X, Y, Z)
2D: $(\frac{X}{Z}, \frac{Y}{Z})$

$Z\uparrow$, $\frac{X}{Z}\downarrow$ farer, smaller
$\downarrow$ $\uparrow$ closer, bigger

$\sin(\alpha+\beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta$
$\cos(\alpha+\beta) = \cos\alpha\cos\beta - \sin\alpha\sin\beta$

$\det(R) = ad - bc$
orthogonal: $R^TR = RR^T = I \Longrightarrow R^T = R^{-1}$

X: order of multiplication (matrix operation) matter!

D.O.F:

	D.O.F
translation	2
rigid (欧 Euclidean)	3
similarity	4
affine	6
projective	8

$\begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$

Homography: forward warp (splatting) : holes
backward warp (interpolation) no holes

Homography

- plane-to-plane (object is a plane)
- need 4 points to solve (from same plane, different lines)

全景图 panoramas $\xleftarrow{\text{stitch}}$ ~~translation-only~~ ? ✗, because angle changes
homography ✓

formula: $p' \cong H p$, how to solve H ?

The Direct Linear Transform (DLT)

$x' = \sim\sim$, $y' = \sim\sim$, $1 = \sim\sim$ $\xRightarrow{\text{to make linear}}$ $x'(\sim\sim) = \cdots$, $y'(\sim\sim) = \cdots$ $\xRightarrow{\text{expend}}$ $A_i h = 0$

$$A_i = \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}$$ for every correspondence points pair.

将A_i stack $\Rightarrow$ homography $\boxed{A \cdot h = 0}$, use SVD (singular value decomposition) to solve it.

Reminder: Affine as example.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow \underbrace{\begin{bmatrix} x' \\ y' \end{bmatrix}}_{b} = \underbrace{\begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_6 \end{bmatrix}}_{x}$$

stack $\Rightarrow$ Affine $\boxed{Ax = b}$

what if points $((x,y) \to (x',y'))$ are not accurate, there would not be exact solution.
how to find approximate solution? aka. Linear least-squares problem. 线性最小二乘

$\boxed{E_{LLS} = \|\underbrace{Ax}_{\text{anticipate}} - \underbrace{b}_{\text{target}}\|^2}$, minimize it. $\Rightarrow$ $E_{LLS} = \min_x \|Ax - b\|^2$

note:
- $\|v\|^2 = v^T v$
- $(a+b)^T = a^T + b^T$
- $(AB)^T = B^T A^T$
- $\frac{d}{dx}(x^T C x) = 2Cx$ (if C对称 $C^T = C$)
- $\frac{d}{dx}(x^T C) = C$
- 若C不对称: $\frac{d}{dx}(x^T C x) = (C + C^T) \cdot x$

$\Rightarrow E_{LLS} = (Ax-b)^T(Ax-b) = [(Ax)^T - b^T](Ax-b) = x^T A^T A x \underline{- x^T A^T b - b^T A x} + b^T b$

note. $(x^T A^T b)^T = b^T A x$, (列向量x)^T · 列向量 = 标量 scalar, 相等

$= x^T A^T A x - 2x^T A^T b + b^T b$.

Then $\nabla E_{LLS} = 2A^T A x - 2A^T b$, set $\nabla E_{LLS} = 0$ to get min value

$\Rightarrow \boxed{A^T A x = A^T b \text{ (Normal Equation)}} \xRightarrow{\text{if } A^T A \text{可逆}} x = (A^T A)^{-1} A^T b$

$x = (A^TA)^{-1}A^Tb$ Don't try to solve inverse matrix !

unstable, unefficient

Instead,

np.linalg.lstsq(A,b)	最小二乘求解	stable
or scipy.linalg.qr	QR分解	stable + efficient
or np.linalg.svd	SVD分解	most stable, but slowest

note: what is singular matrix? non-invertible aka. A^{-1} not exist.

singular matrix properties:
① A^{-1} not exist
② $\det(A) = 0$
③ 存在线性相关列/行
④ 存在 $\lambda = 0$
⑤ $Ax=b$ 无唯一解
⑥ 无法做 Normal Equation 求解 $A^TAx = A^Tb$,
因为 $(A^TA)^{-1}$ 不存在.

overdetermined system —— number of equations > number of variables

unsolvable inconsistent system —— there is not x can satisfy $Ax=b$
↳ E_{LLS}

SVD: $A = U \Sigma V^T$

left · diagonal matrix · right singular matrix orthogonal column

h = last column of V $h = V[:,-1]$

reshape h → H H = h.reshape((3,3))

Use homography to make panoramas:

① The img corresponding pipeline

$E_{WSSD}(u,v)$

1. Feature point ~~dect~~ detection (Harris / Lowe's DoG) — reminder: Feature points are on the corners?
2. Feature point description (SIFT)
3. Feature matching: RANSAC (Random Sample Consensus)

RANSAC

goal: find inliners

Q: why cannot use least-square method directly?
A: least-square will minimize total error of all points. But outliers have huge impact.

step 1. randomly sample the number of points required to fit the model
2. solve for model parameters using samples (use DLT)
3. Score by the fraction of inliners within a preset threshold of the model
4. Repeat 1~3 until we ~~for~~ find the best model ~~(with highest score)~~ (with largest number of inliers) ~~with high confidence~~

$$1-(1-\underbrace{\underbrace{\underbrace{(1-e)}_{\text{inlier}}{}^{s}}_{\text{2 inliers}}}_{\text{at least 1 outlier}})^{N} = p$$

all fail

at least 1 success

ex: 0.5 e: probability that a point is outlier

2 s: number of points in a sample to fit model

N: number of samples

99% p: desired probability that we get a good sample. (confidence)

$$\Rightarrow N = \frac{\log(1-p)}{\log(1-(1-e)^s)}$$

or Early Termination: $T = (1-e) \cdot$ total number of data points

when a model can cover T inliers, you can stop.

② recompute H use all inliers ~~from~~ RANSAC (use DLT) one more detail : $\boxed{\text{Blending}}$

method 1 : simple average $p = (p_1 + p_2)/2$

2 : feathering $p = (w_1 \cdot p_1 + w_2 \cdot p_2)/(w_1 + w_2)$, w_i is distance from ~~[illegible]~~ border in original img.

~~X~~ 3 : Pyramid blending

Pyramid Blending

save every scale colorful img, big size

- Gaussian pyramid:
  1. Gaussian filtering
  2. subsample

  ~~Rep~~ Repeat ~~~~

Problem: cannot reconstruct back?

note: resolution ratio whole pyramid $= \frac{4}{3}$ original

save every scale residual img, small size

- Laplacian pyramid:
  1. Gaussian filtering
  2. compute difference (residual)
  3. subsample

"difference of Gaussian approximate Laplacian"

Reconstruct: 1. upsample
3. sum with residual

$\boxed{\text{Blending with Laplacian pyramid}}$

img I^A : I_1^A I_2^A ~

img I^B : I_1^B I_2^B ~

mask m : m_0 m_1 ~~

residual : l_1^A l_2^A ~

residual : l_1^B l_2^B ~

blend in every scale

$l_k = m_k l_k^A + (1 - m_k) l_k^B$

then, use l_k to Reconstruct (upsample, add l_k)