188cv notes

global latex defines

examples

$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \left\{ \begin{bmatrix} a & & \\ & \ddots & \\ & & b \end{bmatrix} \right\} \begin{cases} case1 & 1 \\ case2 & 2 \end{cases}$

lecture 8 homography (2/1)

homography

$$\vec{x}' = H\vec{x}$$

```
# a == x
# b == x'
H = homography(a, b)
```

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

$$h_{33} = 1$$

Inverse warp

$$H^{-1} \begin{pmatrix} 600 \\ 100 \\ 1 \end{pmatrix} = \begin{pmatrix} 741.86 \\ 50.30 \\ 0.98 \end{pmatrix} \overset{\text{divide by 3rd element}}{=} \begin{pmatrix} 757 \\ 51.30 \\ 1 \end{pmatrix}$$

planar homography is a 3×3 matrix

scale factor doesn't affect the image pixel

$$P' = \alpha H P$$

$$P' = \alpha V$$

$$\frac{1}{\alpha}\left(\alpha\begin{pmatrix} a \\ b \\ 1 \end{pmatrix}\right)$$

forward warping is preferable to inverse warping

because of fewer gaps

2d transformations

Name	Matrix	# dof
translation	$[I \mid t]_{2\times 3}$	2
rigid (euclidean)	$[R \mid t]_{2\times 3}$	3
similarity	$[sR \mid t]_{2\times 3}$	4
affine	$[A]_{2\times 3}$	6
projective	$[\tilde{H}]_{3\times 3}$	8

when can we use homographies?

remove z depth because picture maps onto plane

$$\begin{bmatrix} \cdot & \cdot & 0 & \cdot \\ \cdot & \cdot & 0 & \cdot \\ \cdot & \cdot & 0 & \cdot \\ \cdot & \cdot & 0 & \cdot \end{bmatrix}_{4\times 4} \begin{bmatrix} x \\ y \\ z = 0 \\ 1 \end{bmatrix}$$

3 steps to apply a homography

1. convert to homogeneous coordinates
   - $p = \begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
2. multiply by homography matrix
   - $P' = HP$
3. convert $P' \Rightarrow p'$

- $P' = \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \Rightarrow p' = \begin{bmatrix} x'/w' \\ y'/w' \end{bmatrix}$

how to get H

homogeneous DLT

- get H from correspondences

create correspondences

- **Given**

$$\{p_i, p_i'\} \forall i \in \{1 \ldots 4\}$$

- find the best estimate of H s.t.

$$P' = H \cdot P$$

how many correspondences do we need?

H has 8 dof so there are 4 points and each point has 2 coordinates i.e. 4*2 = 8

how to determine H

1. $P' = HP$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \alpha \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- can make α anything

2. multiply out

$$\begin{aligned} x' &= \alpha(h_1 x + h_2 y + h_3) \\ y' &= \alpha(h_4 x + h_5 y + h_6) \\ 1 &= \alpha(h_7 x + h_8 y + 1) \end{aligned}$$

3. divide out α

$$\begin{aligned} x'(h_7 x + h_8 y + 1) &= h_1 x + h_2 y + h_3 \\ y'(h_7 x + h_8 y + 1) &= h_4 x + h_5 y + h_6 \end{aligned}$$

4. subtract over

$$
x'(h_7x + h_8y + 1) - (h_1x + h_2y + h_3) = 0
$$
$$
y'(h_7x + h_8y + 1) - (h_4x + h_5y + h_6) = 0
$$

5. $$
\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & x' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}
$$

homogeneous linear system

$$
A = \begin{matrix} \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & x' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix} \\ \vdots \end{matrix}
$$

$$
h = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix}
$$

$$
Ah = 0
$$

singular value decomposition [SVD]

> links: wiki , mit

This is a method to compute the eigenvalues of A

A is deconposed into $U\Sigma V$

- U is ortho-normal
- Σ is diagonal
- V is ortho-normal $\rightarrow$ unit norm constraint

$$
\begin{aligned}
A_{n\times m} &= U_{n\times n}\Sigma_{n\times m}V^T_{m\times m} \\
&= \sum_{i=1}^{9}\sigma_i u_i v_i^T
\end{aligned}
$$

AA^T and A^TA (grammian matrix)

when you have Ah and take derivative you get A^TA in optimization problems

$$||Ah||_{\geq}^2 = h^TA^TAh$$

We are going to solve this with an optimization problem

> aside
>
> $$||v||_p = \left(\sum_{i=1}^{\#points}|v_i|^p\right)^{\frac{1}{p}}$$

$$Ah = 0$$

$$\hat{h} = \ \text{argmin}\ ||Ah||_2^2 \text{ s.t. } ||h||_2^2 = 1$$

where $||h||_2^2 = 1$ needs to be equal to some constant not `= 0` so the solution doesn't tend towards 0

> aside
>
> $$||Ah||_2^2 = (Ah)^T(Ah)$$

$$L(\cdot) = h^TA^TAh + \lambda(h^Th - 1)$$

we want

$$\frac{\partial L}{\partial h} = 0$$

> aside
>
> $$h^Th = ||h||_2^2$$
>
> $$h^Th = 1$$
>
> $$\lambda[1 - h^Th] = 0$$
>
> we have this lambda as a penalty because we want to have a low lagrangian error

> aside
>
> $$\frac{\partial}{\partial h}h^Th = 2h$$

$$\frac{\partial L}{\partial h} = [h^T A^T A h + \lambda(h^T h - 1)] = 2A^T A h - 2\lambda h = 0$$
$$\therefore A^T A h = \lambda h$$

Choose eigenvector corresponding to smallest eigenvalue because we are minimizing

$$||Ah||_2^2 = ||h^T A^T A h||_2^2 = ||h^T \lambda h||_2^2$$

choose smallest λ

This is given by the SVD because column of V corresponding to the last entry of Σ gives us our solution i.e. the eigenvector corresponding to the smallest eigenvalue

The singular values are the eigenvalues of $A^T A$ might be a factor of something squared

solve for h using homogeneous DLT [Direct linear transformation]

Goal: given $\{p_i, p_i'\}$, solve H s.t. $P' = HP$

1. for each correspondence create $A_i \in \mathbb{R}^{2\times 9}$
2. stack for multiple correspondences $A \in \mathbb{R}^{2N\times 9}$ where N is the number of correspondences
3. Compute SVD of $A = U\Sigma V^T$
4. Store $h = v_i$ where i is last column of v (or v_i im not 100%)
5. Reshape $h \in \mathbb{R}^{9\times 1}$ to $H \in \mathbb{R}^{3\times 3}$

what are we missing

1. we seem to have correspondences
2. a method to get homography matrix from correspondences
3. and an easy way to compute homography

but correspondences seem to be **inacurate**

automating correspondence pipeline

1. feature point detection
   - detect corners, blobs, ... using sift

2. feature point description
    - use sift again because it is both a **detector** and **descriptor**
3. feature correspondence

RANSAC [random sample consensus]

Algorithm:

1. sample (randomly) the number of points required to fit the model
2. solve for model parameters using samples
3. score by the fraction of inliers within a preset threshold of the model
4. **repeat 1-3** until the best model is found with high confidence

choosing RANSAC parameters

> skipped over in lecture

- number of samples `N`
    - chose `N` such that with probability `p` at least one random sample is free from outliers (e.g. `p=0.99`)(outlier ratio: `e`)
- number of sampled points `s`
    - minimum number needed to fit the model
- distance threshold `δ`
    - choose `δ` so that a good point with noise is likely (e.g. `prob=0.95`) within threshold
    - zero-mean gaussian noise with std. dev. $\sigma : t^2 = 2.84\sigma^2$

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

	proportion of outliers e						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

RANSAC loop

> estimating homography using ransac **Ransac Loop**

1. get 4 point correspondences
2. compute `H` using homogeneous DLT
3. Count inliers
4. Keep `H` if # inliers > threshold
5. **repeat 1-3** if not
6. recompute `H` by pruning outliers

other reading

- Szeliski textbook `Section 6.1`
- Hartley and Zisserman, "Multiple view geometry," cambridge university press 2003 `sections 2 and 4`
- invitation to 3-d vision

lecture 9 8-point algorithm (2/3)

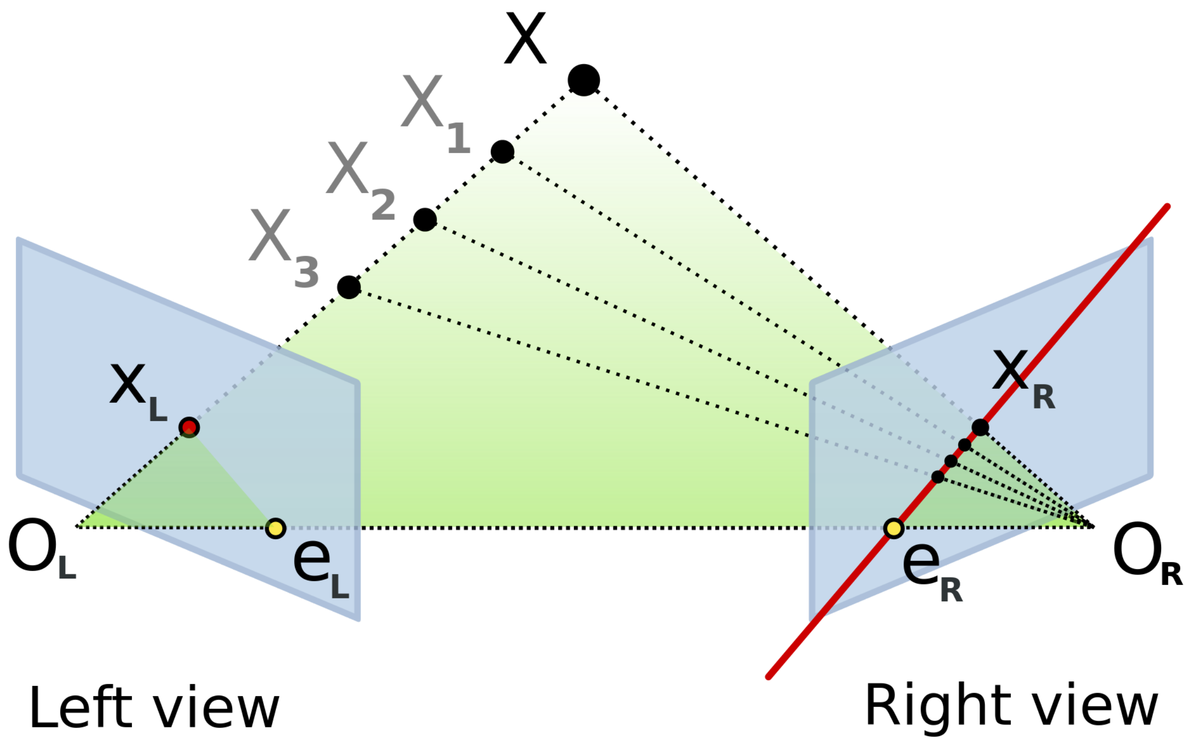
going from 2D to 3D

assuming perfect correspondence* among ≥ 2 images `2D` "reconstruct" `3D` scene we have:

- a point in the scene P
- 2 points projected onto image plane on points x_1^i and x_2^i for $i = 1 \ldots N$
- $x_1^i \in I_1$ and $x_2^i \in I_2$
- $T \in \mathbb{R}^3$ translation (Baseline) between focal point of each camera
- $R \in \mathbb{R}^3$, $R^T R = I$ (orthogonal), $det(R) = +1$

> **orthogonal matrixes** are matrixes where
>
> - the axes are unit norm and oriented with axis x,y,z
> - if you take the transpose you get the inverse i.e. $A^T = A^{-1}$
> - matrices can have 2 determinites ± 1
>
> $det(R) = -1$ would be the reflection

8-point algorithm



epipolar geometry

$$\text{Given } (x_1^i, x_2^i) \text{ for } i \in \{1 \dots N\}$$

Find

1. $$T \in SE(3)$$
2. $$R \in SE(3)$$
3. $$P^i \equiv X^i \equiv \begin{bmatrix} X^i \\ Y^i \\ Z^i \end{bmatrix} \in \mathbb{R}^3$$

> going to do a geometric construction then an algebraic construction

> **aside:**
>
> there exists at least one point $p \in \mathbb{E}^3$ where > $\mathbb{E}^3$ is a euclidean space
>
> Euclids axioms:
>
> 1. There exists at least one point
> 2. There exists at least one line
> 3. between two points there is one line

> We **represent** a point with coordinates $x \in \mathbb{R}^3$
>
> We have a **vector** which is defined as a difference between two points $\vec{v} = x_1 - x_2 \in \mathbb{R}^3$
>
> If p is in a **projected space** the
>
> $$p \in \mathbb{P}^3$$
>
> $$p = \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} = \bar{x}$$
>
> points and vectors in homogeneous coordinates are different as > shown below where we try to create a vector from two homogeneous > points
>
> $$\vec{v} = \bar{x}_1 - \bar{x}_2 = \begin{bmatrix} x_1 - x_2 \\ y_1 - y_2 \\ 0 \end{bmatrix}$$

- X_1 is the vector from the focal point of I_1 to P which passes through x_1
- X_2 is the vector from the focal point of I_2 to P which passes through x_2
- RX_2 is X_2 put into the reference frome of X_1 and T

We know that $\{X_1, RX_2, T\}$ are all in the same plane (**coplanar**). If they are not then they do not correspond.

How to prove they are coplanar?

If the cross product of two vectors

$$T \times RX_2$$

and the inner product with the third vector

$$\text{Triple Product} \rightarrow X_1^T(T \times RX_2) = 0$$

> T multiplies R so now it is non linear which makes it complicated

then the vectors are coplanar.

$$x_1, x_2, T \text{ are also coplanar}$$

Epipolar constraint:

$$\bar{x}_1^{i\,T} T \times R\bar{x}_2^i = 0 \forall i \in \{1 \dots N\} \text{ now find R and T}$$

Epipolar Plane is the plane made from T, x_1, x_2.

Epipolar Lines $\vec{e_1}, \vec{e_2}$ are the lines where the image plane I_1, I_2 intersects with the **Epipolar Plane**

Point where the image plane of one camera hits the image plane of the other camera is called the **Epipoles**

$$\bar{x}_2^{i\ T} T \times R\bar{x}_1^i = 0 \forall i \in \{1 \dots N\}$$

> cross product
>
> $$u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \quad v = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$
>
> **Fundamental theorem of Linear Algebra:**
>
> If something is linear then we can write it as a **multiplication** by a matrix.
>
> Fix u
>
> $$u \times v \overset{\text{linear in v}}{=} \begin{bmatrix} \hat{u} \end{bmatrix} v = \begin{bmatrix} u_x \end{bmatrix} v$$
>
> $$\hat{u} = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix}$$
>
> $$\begin{aligned} u \times v &= \hat{u} v \\ &= -v \times u \end{aligned}$$

$$\bar{x}_2^{i\ T} T \times R\bar{x}_1^i = 0 \forall i \in \{1 \dots N\}$$

$$x_2^T \hat{T} R x_1 = 0$$

- $\hat{T}$ is skew semmetric
- R is orthogonal

$$[\hat{T}][R] = Q \in \mathbb{R}^3$$

Q is the **Essential matrix**

Given (x_1^i, x_2^i) $i = 1 \dots N$ correspondence

$$\bar{x}_2^{i\ T} Q \bar{x}_1^i = 0 \forall i$$

For some $Q \in \mathbb{R}^3$

$$Q = \hat{T} R$$

1. find Q
2. get T and R
3. from T and R we back project and get X_1, X_2

If we find Q then decompose $\hat{T}$ R.

- Q has 8 dof (normalization)
- R has 3 dof
- T has 3 dof
- (R, T) has $\cancel{6}$ 5 dof (5 point algorithms) because of the **Epipolar Constraint** being equal to 0

Rotation matrices objects in $\mathbb{R}^{3\times 3}$ s.t. $R^T R = I_{3\times 3}$

Generalization of a sphere where a sphere is a set of x s.t. $x^T x = 1$

$SO(3)$ is a special orthogonal group (LIE group)

$TSO(3)$ is the tangent bundle to the orthogonal group and

$$Q \in TSO(3)$$

Let the Rotation matrix be a ball in $\mathbb{R}^{3\times 3}$ and pick the rotation matrix equal to I_9. Imagine a tangent plane to the ball. Vectors tangent to this ball are skew symmetric

$$S = \{S \in \mathbb{R}^{3\times 3} \mid S^T = -S\}$$

where S is the tangent to the ball.

- **orthogonal group** is the **ball**
- **translation** part that lives on the **tangent plane**
- **tangent bundle** in differential geometry

1. pretend $Q = \mathbb{R}^{3\times 3}$
2. solve $\bar{x}_2^{i\,T} Q \bar{x}_1^i = 0 \forall i = 1 \dots N$ for Q
    1. this is linear in Q
3.

$$[\chi]_{N\times 9} \begin{bmatrix} q_{11} \\ q_{12} \\ q_{13} \\ \vdots \\ q_{33} \end{bmatrix} = \begin{bmatrix} \vdots \\ 0 \\ \vdots \end{bmatrix}$$

$$\chi = \begin{bmatrix} x_2 x_1 & y_2 x_1 & x_1 & \dots \\ & \vdots & & \end{bmatrix}_{N\times 9} = x_2 \otimes x_1 \rightarrow \text{kronecker product}$$

1. **Given** correspondences $(x_1^i, x_2^i) \rightarrow$ construct $\chi \in \mathbb{R}^{N\times 9} \rightarrow$ get essential matrix Q
    i. Find $\vec{q} \in Q^v$ s.t. $\chi \vec{q} = 0$
    ii. $\vec{q} \in nullspace(\chi)$
    iii. $$\chi \overset{\text{SVD}}{=} U\Sigma V^T$$ $$\Sigma = \begin{bmatrix} \sigma_1^2 & & \\ & \ddots & \\ & & \sigma_9^2 \end{bmatrix}$$ $$\sigma_9^2 = 0 \text{ must be 0 or else } \chi \text{ has an empty nullspace}$$
    iv. 9th column of V, V_9 is the eigenvector
    v. rearrange V_9 into $3x3$ to get Q
2. Given essential matrixes Q get R and $\hat{T}$

$$Q = U_{3\times 3_Q} \Sigma_{3\times 3_Q} V_{3\times 3_Q}^T = U_Q \begin{bmatrix} 1 & & \\ & 1 & \\ & & 0 \end{bmatrix} V_Q^T$$
$$R = U \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} V^T$$
$$\hat{T} = U \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Sigma U^T \rightarrow T$$

1. From previous lectures:

$$\bar{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$
$$X = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} z = \begin{bmatrix} x/z \\ y/z \\ z/z \end{bmatrix} z$$
$$X_2 = ...RX_1 + T$$
$$z_2 \bar{x}_2 = R\bar{x}_1 z_1 + T$$
$$\begin{bmatrix} -R\bar{x}_1 & \bar{x}_2 & 1 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ 1 \end{bmatrix} = T$$

algebraic derivation

$$X^i = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \in \mathbb{R}^3$$

$$X = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$x = \begin{bmatrix} X/Z \\ Y/Z \end{bmatrix}$$

$$\bar{x} = X \cdot \frac{1}{z} = \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix}$$

$$X = \bar{x} \cdot z$$

$$\bar{x}^i = \lambda^i X^i$$

$$X_2^i = RX_1^i + T$$

$$\frac{\bar{x}_2}{z_2} = R\frac{\bar{x}_1}{z_1} + T$$

$$\frac{1}{z_2}\bar{x}_2 = R\bar{x}_1\frac{1}{z_1} + T$$

- 3 equations because 3 vectors
  - use one of the vectors to solve for $\frac{1}{z}$
  - substitute into the other 2
  - then you got rid of z_1, z_2
- how to get rid of T
  - multiply through by $\hat{T}$

$$\hat{T}\frac{\bar{x}_2}{z_2} = \hat{T}R\bar{x}_1\frac{1}{z_1} + \hat{T}T$$

$$\hat{T}T = 0$$

$$\hat{T}\frac{\bar{x}_2}{z_2} = \hat{T}R\bar{x}_1\frac{1}{z_1}$$

- multiply through by x_2^T because $\bar{x_2}^T\hat{T}$ is orthogonal to T and x_2
- inner product with x_2 you get the volume of a solid with 0 volume

$$\bar{x_2}^T\hat{T}\frac{\bar{x}_2}{z_2} = \bar{x_2}^T\hat{T}R\bar{x}_1\frac{1}{z_1}$$

$$\bar{x_2}^T\hat{T}\frac{\bar{x}_2}{z_2} = 0$$

$$0 = \bar{x_2}^T \hat{T} R \bar{x}_1 \frac{1}{z_1}$$

- it is equal to 0 so we can get rid of $\frac{1}{z_1}$

$$0 = \bar{x_2}^T \hat{T} R \bar{x}_1$$

lecture 10 3d reconstruction in practice (2/8)

Parts of lecture

1. 3D reconstruction in practice
2. ∞ in CS

review of last week

List of the assumptions we made in last lecture

1. **Assumed** perfect correspondence: $(x_1^i, x_2^i) \forall i = 1 \dots N$ back projected to $P \equiv X = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$

$$X_2^T (T \times (RX_1)) = 0$$

$$T \times A \equiv \hat{T} A$$

$$\hat{T} = \begin{bmatrix} 0 & -T_3 & T_2 \\ T_3 & 0 & -T_1 \\ -T_2 & T_1 & 0 \end{bmatrix} \rightarrow \text{skew symmetric}$$

$$X_2^T (\hat{T}(RX_1)) = 0$$

$$X_2^T \hat{T} R X_1 = 0$$

$$X_2^T Q X_1 = 0 \text{ where } Q \text{ is the essential matrix}$$

This equation is homogeneous `aka = 0` so we can multiply by whatever scale factor so pick

$$\frac{1}{z_2} X_2^T Q X_1 \frac{1}{z_1} = 0$$

$$\frac{1}{z_2} X_2^T = \bar{x}_2^T$$

$$X_1 \frac{1}{z_1} = \bar{x}_1$$

$$\bar{x}_2^T Q \bar{x}_1 = 0$$

2. We solved for a **generic matrix** $Q \in \mathbb{R}^{3\times3}$ which has `9 dof` but **in reality** $Q \in TSO(3)$ which has `5 dof`
3. **Assumed** the camera was calibrated so that when we take the perspective projection all we do is divide by the 3rd component Z and we got a vector measured in meters, milimeters, inches, etc.

But our reference frame has the origin at the optical center of the camera but we do not know where that is. In theory the `z` is a distance of `1` but in practice we do not know what it is.

We break it down into `3` consecutive SVD's, which is a solution of a linear system of equations.

> **aside**
>
> Even though divide and conquer is a normal practice in engineering
>
> In general in situations when you have uncertainty such as `sensory measurements` this is a **very bad** idea
>
> - data processing equality
> - pitfall of premature decisions
>
> deep learning solves these problems **end to end** from **data** to **decision** by forgoing breaking the problem down into pieces .. we want to understand the structure of the problem even solving end to end becomes easier and more interpretable

4. Bundle Adjustment

solving the 8 point algorithm ignoring that $Q \in TSO(3)$

$$(x_1^i, x_2^i) \forall i = 1 \dots N$$

$$\bar{x}_2^{i\,T} Q \bar{x}_1^i = 0$$

$$Q \in \mathbb{R}^{3\times3} \text{ forget that } Q = \hat{T} R$$

Now we differentiate between these Q's

$$\text{rename to } F \in \mathbb{R}^{3\times 3} \text{ the Fundamental Matrix}$$

$$\text{and rename } Q = \hat{T}R \text{ the Essential Matrix}$$

$$\bar{x}_2^{i\,T} F \bar{x}_1^i = 0$$

F^V is F strung out as a vector

and where V_9 is the 9th vector of V

$$\begin{bmatrix} x_2^1 \otimes x_1^1 \\ x_2^2 \otimes x_1^2 \\ \vdots \\ x_2^N \otimes x_1^N \end{bmatrix}_{N\times 9} \begin{bmatrix} F^V \end{bmatrix}_{9\times 1} = \chi \begin{bmatrix} F^V \end{bmatrix}_{9\times 1} = 0$$

$$\chi = U\Sigma V^T$$

$$\Sigma = \begin{bmatrix} \sigma_1^2 & & \\ & \ddots & \\ & & \sigma_9^2 = 0 \end{bmatrix} \rightarrow V_9 = F^V \rightarrow \text{ rearrange into 3 x 3}$$

We want to find the `F` which is closest to `Q` in $\mathbb{R}^9$

$$Q \in TSO(3) \iff Q = U_q \Sigma_q V_q^T$$

$$\Sigma_q = \begin{bmatrix} \sigma & & \\ & \sigma & \\ & & 0 \end{bmatrix}$$

How do we find $F = U_f \Sigma_f V_f^T$ which is closest to Q?

We can use the **Frobeneous norm**. Is the sum of the singular values.

Let $\hat{Q}$ be the closest F to Q

$$\hat{Q} = U_f \begin{bmatrix} \frac{\sigma_1+\sigma_2}{2} & & \\ & \frac{\sigma_1+\sigma_2}{2} & \\ & & 0 \end{bmatrix} V_f^T$$

This gives us the closest essential matrix.

1. find `F`
2. find the closest `Q` to that `F`

This is not the same `Q` which solved the matrix in the first place.

Assumptions

When can we find the solution to a linear system of equations $Ax = b$?

Has a solution when b is in the **range space / column space** of A

Here we have `b=0` aka the nullspace $Ax = 0$.

1. **Assumption:**

We must have a 0 singular value aka $\sigma_9^2 = 0$.

we assume that there is a nullspace of when in reality there isn't for

$$\chi F^V = 0$$

If $Ax \neq 0 \nexists x \mid Ax = 0$

Then we say $Ax = n$ where $n = \arg\min_x ||Ax||$

σ_9^2 will be the smallest σ_i^2 but it will not be 0.

It will be the same solution but not an exact solution.

2. **Assumption:**

There is a **unique solution**.

When does $Ax = b$ have a unique solution?

When the determinant is non zero i.e. $\det A \neq 0$

3. **Assumption:**

correspondences are in **general position** i.e. means the data doesn't line up in a lower dimension space.

what have we fixed

- We fixed that F is not an essential matrix
- We fixed that χ might not have a null space
  - what happens when χ not only has a null space but has $\text{nullspace}(\chi) > 1$

we said there might be multiple solutions

$$\hat{Q} = U_f \begin{bmatrix} 1 & & \\ & 1 & \\ & & 0 \end{bmatrix} V_f^T$$

$$R = U_f \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} V_f^T$$

$$T = \ldots$$

We have 4 solutions called *Twisted pairs*

If we flip the camera 180°.

We assume that the points are **positive depth**.

Say we are just looking at coordinates of points. If we flip the camera 180° then the depth of one is `-` while the depth of the other camera is `+`. i.e. 4 possible solutions.

noise around the points

x_1^i, x_2^i have noise called n_1, n_2

Which we will model as **Gaussian noise**.

Another problem is that you give an **outlier** which is worse.

Robust statistics for outliers

- regression problems
  - unknown and you try to find it
  - We want to find $T, R, X_1, and X_2$
- decision problem
  - which data are inliers and which are outliers

We use **RANSAC**

1. model parameters (hypothesis)
2. inliers and outliers

We need a function given 2 points tests whether they correspond.

$${\bar{x}_2^i}^T F \bar{x}_1^i = 0$$

$$\bar{x}_2^{i\,T} F \bar{x}_1^i = n^i \rightarrow \text{ small}$$

If you get an outlier

$$\bar{x}_2^{i\,T} F \bar{x}_1^i = m^i \gg 0$$

In 2d we can have an error written like

$$y^i = mx^i + b + n^i$$

$$y^i - mx^i - b = n^i$$

We write

$$\bar{x}_2^{i\,T} F \bar{x}_1^i = n^i$$

- We start with a random sample of points.
- Find F
- count who is in and who is out
- new random collection of points
- Find F
- count who is in and who is out
- loop
- find the hypothesis gathers the most support (number of votes)
- take those inliers and calculate F

End to End

Our **approximations** $\hat{Q}, \hat{T}, \hat{R}, \hat{X}_1^i, \hat{X}_2^i,$

$$X_2 = RX_1 + T$$

$$x_2^i = \frac{X_2}{z_2} + n_2^i$$

$$x_1^i = \frac{X_1}{z_1} + n_1^i$$

Starting from

$$\hat{Q}, \hat{T}, \hat{R}, \hat{X}_1^i, \hat{X}_2^i,$$

find

$$Q, T, R, X_1^i, X_2^i,$$

that minimize

$$\sum ||x_1^i|| + ||x_2^i||$$

Plug in

$$(x_2^i z_2 + n_2^i z_2) = R(x_1^i z_1 + n_1^i z_1) + T$$

Simplify

$$x_2^i z_2 - R x_1^i z_1 - T = R n_1^i z_1 - n_2^i z_2$$

We want to estimate $x_1^i, x_2^i, R, T, z_1, z_2$

Rewrite as

$$x_2^i z_2 - R x_1^i z_1 - T = n \rightarrow \text{ projection error}$$

$$x_2 = \frac{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} (R x_1 + T)}{\begin{bmatrix} 0 & 0 & 1 \end{bmatrix} (R x_1 + T)} + n$$

$$= \prod (R x_1 + T) + n$$

$$\widetilde{\sum_{i=1}} || x_2^i - \prod (R x_1^i z_1^i + T) || \text{ s.t. } i = 1 \ldots N$$

N can be large

$$\underset{w}{\arg\min} f(x^i, w)$$

$$w_0 = \{\hat{R}, \hat{T}, \hat{X}_i\}$$

Gradient Descent

$$w_{t+1} = w_z + \alpha \sum_T \nabla_w f(x^i, w_z)$$

Learning rate α is up to you. This is **optimization**.

This is called **bundle adjustment**

projection [central perspective projection]

$$x = \prod (X) = \frac{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} X}{\begin{bmatrix} 0 & 0 & 1 \end{bmatrix} X} = \begin{bmatrix} X \\ Y \end{bmatrix} \frac{1}{z}$$

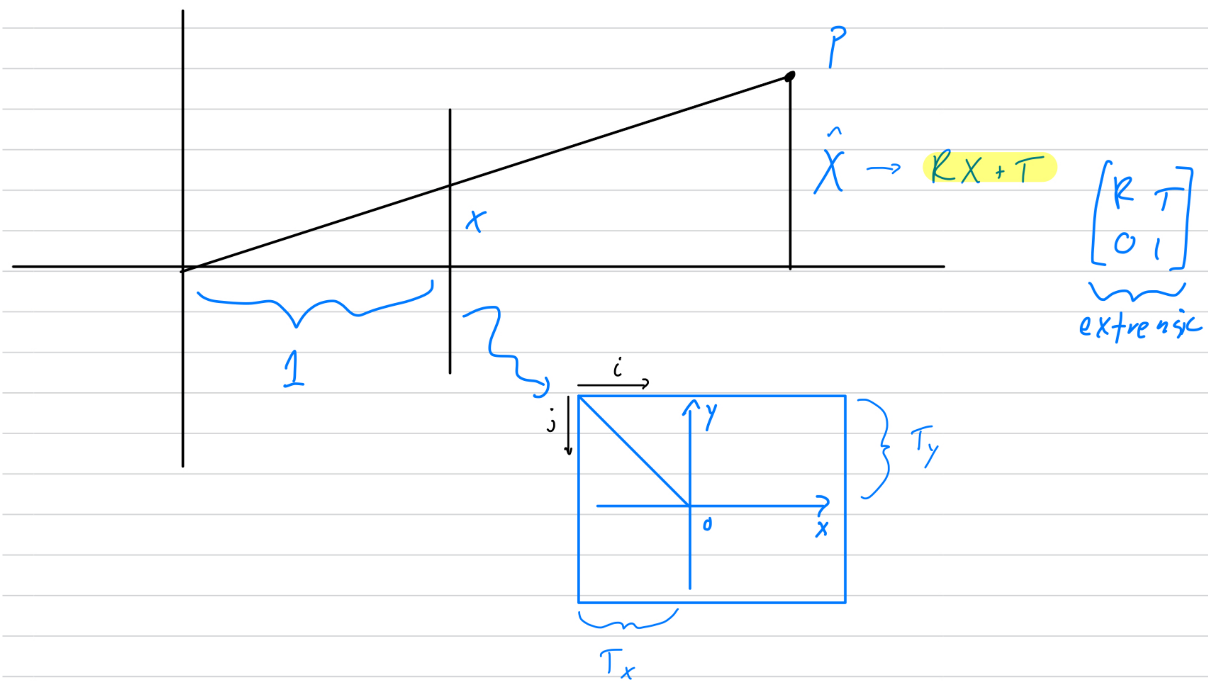
$$\begin{bmatrix} i \\ j \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & \theta & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Focal length is s_x, s_y it is not really the focal length. It is the change of units from meters to the size of pixel in x and y measured in units of focal length.

K is the **intrinsic calibration matrix**.

$$\hat{X} = RX + T = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}_{4\times 4}$$

This is called the **extrinsic calibration matrix**. From the world reference frame where we measure our "checkerboard" to the optical center of the camera. It is a change of coordinates in 3D.



notes on infinity (not on midterm/final)

Euclid postulates

1. there exists a point
2. there exists a line
3. for two points there is a line
4. for two lines there is **at most** point [euclidean]

How can we put coordinates on infinity?

$$l = \{(x, y) \mid y = ax + b\} \rightarrow \text{ explicit}$$

$$l = \{(x, y, 1) = \bar{x} \mid l^T \bar{x}\} \rightarrow \text{ implicit form}$$

$$l_1 || l_2, a_1 = a_2, b_1 \neq b_2$$

$$l_1 = (x, y) \mid l_1^T \bar{x} = 0$$

$$l_2 = (x, y) \mid l_2^T \bar{x} = 0$$

$$y = a_1 x + b_1 = a_2 x + b_2$$

$$(a_1 - a_2)x + (b_1 - b_2) = 0$$

if they are parallel

$$a_1 - a_2 = 0 \text{ and } b_1 - b_2 \neq 0$$

this fails so put a flag `z = [0,1]`

$$(a_1 - a_2)x + z(b_1 - b_2) = 0$$

$$if a_1 \neq a_2 : z = 1$$

$$if a_1 = a_2 : z = 0$$

$$l_1 = \{\bar{x} \mid l_1^T \bar{x} = 0\}$$

$$\begin{bmatrix} l_1 & l_2 & l_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = 0 \mid z = 0, 1 \in \mathbb{R}$$

Points are the same as lines

$$l^T \bar{x} = 0$$

$$\bar{x} l^T = 0$$

This happens because we brought infinity into the picture.

$$\frac{l^t}{||l||} \frac{\bar{x}}{||x||} = 0$$

lines are spheres too.

5. for two lines there is **at one** point [projective]

we must realize that for any point P

$$P \equiv \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\bar{P} \equiv \lambda \begin{bmatrix} x \\ y \\ z \end{bmatrix} \forall \lambda$$

Space of lines in euclidean.

$$P^2 = \frac{\mathbb{R}^3 - \{0\}}{\mathbb{R}}$$

question

How to compute the rank with SVD?

Find how many values of $\Sigma > 0$

Generalized cross validation can give u this threshold.

lecture 11 light/specular reflection (2/10)

how to formalize brightness?

1. lighting
2. material
3. geometry

all used to paint a pixel

mathematically describe brightness

1. $\vec{x}$ is the location
2. $\vec{w}$ is the angle
3. t is the time
4. λ is the wavelength

$$L(\vec{x}, \vec{w}, t, \lambda)$$

Far field or directional approximation

$$L(\vec{x}, \vec{w}, t, \lambda) \rightarrow L(\vec{w}, t, \lambda)$$

Remove color and assume light not changing in time and also apply the **far field approximation**

$$L(\vec{x}, \vec{w}) \to L(\vec{w})$$

1. I is the image
2. G is geometry

$$\frac{\partial I(x)}{\partial G} \Rightarrow 0$$

mirror reflections

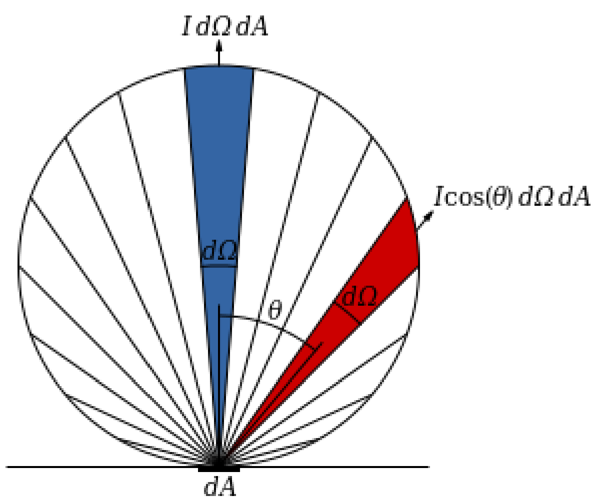
$$f(w_{in}, w_{out}) = \begin{cases} 1, w_i n = w_o ut \\ 0, else \end{cases}$$

specular reflection

- $\vec{n}$ is the normal to the surface

Bidirectional Reflectance Distribution Function (BRDF)

1. Light
   i. $L(\vec{w}) \to L_{src}(\theta_i, \phi_i)$
2. Matter
   i. BRDF $f(w_{in}, w_{out})$; $f(\theta_i, \phi_i, \theta_r, \phi_r)$
   ii. "lambertian surface": $f(w_{in}, w_{out}) = \frac{1}{\pi}$ // could be any constant
3. Geometry
   i. $\vec{n}$

> Foreshortening (lambertian case)
>
> 
>
>
> $$\vec{l}^T \vec{n} = |\vec{l}||\vec{n}| \cos\theta$$
>
> where

> - l is the incident light
> - n is the surface normal

why does white out happen?

Surface is a double integral over all incident angles.

$$\Omega = \text{ all incident angles}$$

$$L^{surface}(\theta_r, \phi_r) = \iint_\Omega L^{source}(\theta_i, \phi_i)\cos(\theta_i)\sin(\phi_i) f(\theta_i, \phi_i, \theta_r, \phi_r) d\theta_i d\phi_i$$

1. lighting contrabution
   i. $L^{source}(\theta_i, \phi_i)\cos(\theta_i)\sin(\phi_i)$
2. material contrabution
   i. $f(\theta_i, \phi_i, \theta_r, \phi_r)$

If we **assume** Lambertian Surface with `Albedo =` $\frac{1}{\pi}$

$$f(\theta_i, \phi_i, \theta_r, \phi_r) = \frac{1}{\pi}$$

assume sky radiance is constant

$$L^{source}(\theta_i, \phi_i) = C$$

Plug it back in

$$L^{surface}(\theta_r, \phi_r) = C$$

directional lighting

assume that over the observed region all source of incoming flux is from one direction

- s is some scaling factor

$$L(x, w, t, \lambda) \to L(x, t, \lambda) = s(t, \lambda)\delta(w = w_0(t))$$

$$L(x, w) \to L(w) \to s\delta(w = w_0) = \begin{cases} L(w_0) = s \\ L(w) = 0 \end{cases}$$

$$L(\theta_i, \phi_i) \equiv \vec{l} = \begin{bmatrix} l_x \\ l_y \\ l_z \end{bmatrix}$$

1. light direction

   i. $\vec{l} = \frac{l}{||l||_2}$
2. light strength
   i. $||l||_2$

"n-dot-i" shading

brdf image

$$L^{out}(\hat{w}) = \int_{\Omega_{in}} f(\hat{w}_{in}, \hat{w}_{out}) L^{in}(\hat{w}_{in}) cos\theta_{in} d\hat{w}_{in}$$

- a is some constant
- n is the surface normal
- l is the light source

$$I = a\hat{n}^T\vec{l}$$

ideal point light source

1. s is strength of light source
2. $||x - x_0||$ is distance from source to point

$$L(x, w) = \frac{s}{||x - x_0||^2} \delta\left(w = \frac{x - x_0}{||x - x_0||}\right)$$

lecture 12 groups/orbits/canonizability/covariant detectors (2/22)

groups

group is a *set* G with an operation "$\circ$" (often called the product) on the elements of G that:

1. is **closed**
   i. if $g_1, g_2 \in G$ then also $g_1 \circ g_2 \in G$
2. is **associative**
   i. $(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3) \forall g_1, g_2, g_3 \in G$
3. has a **unit element** e

   i. $e \circ g = g \circ e = g \forall g \in G$
4. is **invertible**
   i. $g \circ g^{-1} = g^{-1} \circ g = e \mid \forall g \in G, \exists g^{-1} \in G$

Groups also act on a space. every 2x2 matrix (that is invertible) transforms everything on a plane transforms everything on the plane, (x, y) to $a(x, y)$

$\vec{x} \in \mathbb{R}^2$ then $g(x)$ or $g\vec{x}$ or $x \circ g$ is a transformaiton of $\vec{x}$

- translation
- rotation
  - isometry
  - rigid motion $SE(N)$
- scaling
  - similarity
- affine
- projective
- diffeomorphism
- contrast

groups "act"

Apply transformation to a matrix e.g. transforms all elements by a ϕ is **a group**

contrast transformation is where you change the brightness of all pixels in an image but if one pixel a is darker than another pixel b then after the transform a will still be darker than b. The order of the pixels will not change. **this is for global transformations**

if you perform a **local** transformation then some pixels will change while others will not.

orbits, equivalence classes, base/quotient

Nuisance transformation is a transfomation that changes the image but not the object of interest.

what remains constant even though all pixel values change?

We are looking for this **invariance**.

looking for an **INVARIANT** which is any property of the data that remains constant as you perform **nuisance transformations** to the data

what is a **nuisance** depends on the task

example shape space

- triangles coordinates given as 3 points
  - ways to represent triangles
    - $\vec{x}_1 = (x_1 \in \mathbb{R}^2, y_1 \in \mathbb{R}^2)$ ...
    - $T \in \mathbb{R}^{2\times 3}$ is 3 points each 2 elements
    - $T \in R^6$ string out the 3 points
- **problems** with these representations of triangles
  - if you change ordering then the triangles T are different
  - if you change your reference frame then the triangles change
  - triangles exist in $\mathbb{R}^2$ but they can be represented as a string of 3 points so they are in $\mathbb{R}^6$ because of 6 dof
- triangle modulo the axis of action of translation
  - these are **orbits**
  - **translation**
    - $\mathbb{R}^6/\mathbb{R}^2$
    - $\mathbb{R}^2$ comes from the fact that for any one triangle if we only act upon it with translation we only have `2 dof`
  - translation & **rotation**
    - $\mathbb{R}^6/SE(2) = \mathbb{R}^3$
  - similarity transformation -- translation & rotation & **scale**
    - $\mathbb{R}^6/(SE(2) * \mathbb{R}) = \mathbb{R}^2$
    - two triangles are **similar** if you can slide one ont top of another after stretching it and they align without changing the angles
  - **affine** transformation -- **multiply by matrix** and **add offset**
    - $\mathbb{R}^6/A(2) = 0$
    - all triangles are the same under affine transformations

if you want to find similar triangles you must search over the **entire orbit**

- **given** a point on an orbit $T \in R^6$

- we can generate the entrie orbit by just acting on the group
- is there a point on the orbit that is **special**?
    - suppose we can find that special point for every other **orbit**
- take all orbits and map them to a smaller dimensional space
    - **Base** of the orbit space
- then you can just compare coordinates
- pick a point of the triangle and make that be the origin
    - covariant detector
        - **detector** is a function that gives you an element of a group and if you act on the image with that group that element also acts in the same way
    - now if you want to compare triangles you only need to compare 2 points
- pick another point and make it $(1, 0)$
    - now triangles are represented by **1 point** and this definition will fill out $\mathbb{R}^2$
- set of isoceles triangles are a "zero measure set" in the set of triangles meaning that there is a zero percent chance that any triangle will be isoceles

infinite dimension space, finite dimension group

We can apply these transformations to images

if we want to make an image **invariant to transformation**

we must make a function that maps a part of the image to the origin

if you translate the image then the reference frame translates with the image.

pick the brightest point in the image

- translation covariant detector

image in this reference frome

- translation invarient descriptor

want it to be invariant to group transformations

stable locally but fragile with respect to singular perturbations

this is why we don't find one feature point we take many

covariant detectors

- I image
- g group

transversality, critical loci

This takes an image and returns an element of that group as a function of only that image.

$$\psi(I, g) = 0 \Rightarrow g = g(I)$$

$$\det\left(\frac{\partial\phi}{\partial g}\right) \neq 0$$

canonizability

covariant detector is a function that given an **image** and a **group**

and given a particular image of a particular group returns a function

$$\psi : I \times G \Rightarrow \mathbb{R}^{dim(G)}; (I, g) \mapsto \psi(I, g)$$

- the zero-level set $\psi(I, g) = 0$ (implicit function) uniquely determines $\hat{g} = \hat{g} = \hat{g}I$
    - determines an isolated point
- if $\psi(I, \hat{g}) = 0$ then $\psi(I \circ g, \hat{g} \circ g) = 0 \quad \forall g \in G$
    - so that if you transform that image with that group that element gets transformed by the same element of the group

canonizable an image region is canonizable if it admits at least one covariant detector

once you have a covariant detector then you have a covariant descriptor b/c the image in the reference $\hat{g}$ (determined by the covariant detector) is invariant

canonized descriptor

$$\phi(I) \doteq I \circ \hat{g}^{-1}(I) \mid \psi(I, \hat{g}(I)) = 0$$

examples

- harris: bad (non-commutative)
- LoG: good (linear)
- HoG: (hessian of gaussian) (SIFT) better (monge-ampere)
    - under wiener's illumination model
    - it smothes the image

    - picks the brightest pixel
    - does this at different scale w.r.t. to different gaussian curves
    - does this locally
    - what about rotation
        - look at direction of gradient
        - pick the maximum
        - you orient yourself
    - what about contrast
        - look at direction of gradient
        - look at histograms
    - invariant to translation, rotation, scale, and local contrast transformations
- TST: ???
- moments of the superpixel tree ???

office example

how to extract the office that is invariant

function that doesnt change as illumination and visibility changes then

visibility is difficult becaause things are occluded

if you give a view of something where details are occluded then you cannot generate the entire **orbit** of the object

this is why we look at local descriptors

representation

$$\vec{x} \rightarrow z \rightarrow \vec{y}$$

function of the data that's useful for a task

- $\vec{x}$ is now the image
    - **data** images
- $\vec{y}$ is the **task variables**
    - old
        - $\vec{y} = \begin{cases} x \in \mathbb{R}^{3 \times N} \\ g(t) \in SE(3) \end{cases}$
        - $g(t)$ tragectory of camera as we move around the room

    - $g(t) \to (R, T)$
  - new
    - $\vec{y} = \{1, \ldots, k\}$
- **representation** z

take data and do things to it is **data torturing**

data processing inequality [DPI]

never create information by torturing the data

I is information

$$I(\vec{x}; \vec{y}) \geq I(z; \vec{y}) \forall z = z(\vec{x})$$

information measures uncerainty

uncertainty measures the volume of a distribution

lecture 12 video Representations of data

classification

- $x \in X$
  - an image
- $y \in Y = \{1, \ldots, k\}$
  - element of a class/label associated with int from 1 to k
- $\bar{y} \in \{0, 1\}^k$
  - one hot vector
- $f : X \mapsto Y$
  - a classifier
  - input space X to output space Y
- $x \mapsto f(x) = \hat{y}$
  - $f(x) = \hat{y}$ a vector

how to learn

Given a dataset $\mathscr{D}$

- collection of inputs x_i and outputs y_i

$$\mathscr{D}_N = \{x_i, y_i\}_{i=1}^N \rightarrow \text{training set / ground truth}$$

Find a function that maps X to Y we will find a subset of these functions w

$$f_w : X \mapsto Y$$

$$x_i \mapsto \hat{y}_i = f(x_i) = \hat{y}(x_i) = y_i$$

such that

$$\{x_j, ?\}_{j=1}^{M=\text{inf}} \rightarrow \text{ test set}$$

$$f(x_j) = \hat{y}_j \rightarrow \text{ estimate}$$

how do we find parameters w of f

- feed input data x_i
- returns estimated labels $\hat{y}_i$
- we want that to be the same as a true label y_i

$$f_w(x_i) = \hat{y}_i = y_i \forall i = 1 \dots N$$

to do this we can define a penalty or a loss

loss function l

takes estimated label and true label

$$l(\hat{y}, y) = \begin{cases} 0 & \hat{y} = y \\ 1 & \hat{y} \neq y \end{cases} \rightarrow \text{ symmetric 0,1 loss}$$

conditional risk R

- true label $k \in (1 \dots K)$
- loss of calling $\hat{y}$
- probability that the actual label is k $P(k \mid x)$
- we do not know this probability $P(\hat{y} \mid x)$
    - called the posterior

$$
\begin{aligned}
R(\hat{y} \mid x) &= \sum_{k=1}^{K} l(\hat{y} \mid k) P(k \mid x) \\
&= \sum_{k \neq \hat{y}} P(k \mid x) \rightarrow \text{ the sum over all mistakes made} \\
&= 1 - P(\hat{y} \mid x) \rightarrow \text{ posterior}
\end{aligned}
$$

Want to find $\hat{y}$ wich is the **maximizer of the posterior**

$$\hat{y} = \arg\max_k P(k \mid x)$$

pick

$$\hat{y} = k$$

if

$$P(k \mid x) > P(j \mid x) \forall j \neq k$$

aka the **Bayssian discriminant** (posterior)

Basian Decision Rule pick the class that maximizes the posterior probability. Guaranteed to minimize the conditional risk.

average risk

- parameters w of the function f
- the dataset $\mathscr{D}$

$$
\begin{aligned}
R(w_j, \mathscr{D}_N) &= \sum_{i=1}^{N} R(\hat{y}_i \mid x_i) P(x_i) \\
&= \sum_{i=1}^{N} \sum_{k=1}^{K} l(f_w(x_i), k) P(k \mid x_i) P(x_i) \\
\sum_{k=1}^{K} l(f_w(x_i), k) P(k \mid x_i) &\overset{\text{minimized}}{\rightarrow} \text{ when } f_w(x_i) = \arg\max_k P(k \mid x) \\
P(x_i) &\rightarrow \text{ is positive}
\end{aligned}
$$

Because R is a sum of positive numbers it is minimized when each element is minimized

$$
\begin{aligned}
f_w(x) &= \arg\max_k P(k \mid x) \forall x \\
P(k \mid x) &\rightarrow \text{ we do not know this}
\end{aligned}
$$

Represent $\bar{f_w}(x)$ as a one hot vector i.e. a binary vector of `0/1` `dim k`

$$\arg\max_k \left[\bar{f_w}(x)\right]_k = \arg\max_k \left[\overline{\mathscr{D}(\bullet, x)}\right]_k$$

$$f_w(x) = \arg\max_k P(k \mid x) \in \{1, \ldots, k\}$$

$$\bar{f_w}(x) \in \cancel{\{0,1\}^k} \quad \mathbb{R}_+^k$$

$$\arg\max_k \left[\bar{f_w}(x)\right]_k = \arg\max_k \left[P(\bullet \mid x)\right]_k$$

Baysean Discriminant

- yield minimum error rate
- probability of all other classes but the true one

$$\bar{f}_w(x) \approx P(\bullet \mid x)$$

> ### KL Divergence
>
> If we can make this `0` then `P == Q` everywhere
>
> sampled uniformly from the set X $x_i \sim M(X)$
>
> $$\mathrm{K}L(P \| Q) = \sum_{i=1}^{N} P(x_i) \log \frac{P(x_i)}{Q(x_i)} \quad x_i \sim M(X)$$
>
> We do not have $P(x_i)$ but we have $x_i \sim P(x)$
>
> - Q is an approximation of P
>
> $$\sum_{x_i \sim P(x)} \log \frac{P(x_i)}{Q_w(x_i)}$$
>
> Want the closest Q to P
>
> $$\arg\min_w \mathrm{K}L(P \| Q_w) = \sum_{x_i \sim P(x)} \cancel{\log P(x_i)} - \log Q_w(x_i)$$
>
> $\log P(x_i)$ does not depend on w
>
> $$\arg\min_w H_{P,Q}(P \| Q_w) = \sum_{x_i \sim P(x)} - \log Q_w(x_i)$$

use this for

$-$

$$
\begin{aligned}
\bar{f}_w(x) &\approx P(\bullet \mid x) \\
f_w &= \arg\min_w \sum_x \mathrm{KL}(P(\bullet \mid x) \parallel \bar{f}_w(x)) \\
&= \arg\min_w H_{P,f_w} \\
&= \sum_{x_i \sim P(x)} \sum_{y_i \sim P(y|x)} -\log \bar{f}_w(x) \mid_{y_i}
\end{aligned}
$$

$\bar{f}_w$ is a one hot vector and we will fish the component y_i

reiterate

The discriminant

$$f_w(x) \in \{1, \ldots, k\} \to \text{hypothesis}$$

binary vector of k components one hot

$$\bar{f}_w(x) \in \{0,1\}^k \to \text{one-hot}$$

approximate the **posterior distribution Baysean optimal descriminant**

$$f_w(x) \mid_y \to P(y \mid x)$$

We can relax this to $\mathbb{R}^k$

$$f_w(x) \mid_y \in \mathbb{R}^k$$

$$y = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} k$$

inner product $\langle a, b \rangle$

- $f_w(x)$ is the descriminant
- $[f_w(x)]_y$ is $f_w(x)$ evaluated at the component y

$$\langle f_w(x), y \rangle = [f_w(x)]_y$$

lecture 13 recap of video (2/24)

- $x \in X$ input
- $\hat{y}$ is the predicted output

- $Y = \{1, \ldots, k\}$ real output which is unkown during testing but given during training

we torture the data to remove irrelevant data from our input

represent data by everything that matters for the task but nothing more

best label is the **posterior distribution**

$$P(y \mid x)$$

$$\underbrace{\begin{bmatrix} P(\bullet \mid x) \end{bmatrix}_k}_{f_w}$$

Posterior $\equiv$ **Baysean Discriminant** is the best you can do in terms of minimizing the expected error

Risk is the average loss

> Loss functions
>
> procedural recall curve P/R
>
> receiver operator characteristics ROC

at test time we want to pick

$$\hat{y} = \arg\max_k P(k \mid x)$$

when we are given $P_w(y \mid x)$ system

At training time we are given inputs and outputs so that the function it makes approximates the Posterior

Kullback-Leiber divergence is a measure of discrepancy

- 0 if same in distribution

Emperical cross entropy

> $P(x, y) = P(x)P(y \mid x) = \mathbb{E}$

$$\begin{aligned} H_{P,f_w} &= \mathbb{E} - \log f_w(y \mid x) \rightarrow \text{ Expected Loss} \\ &\approx \sum_{x_i \in \mathscr{D}} - \log f_w(y_i \mid x_i) \rightarrow \text{ Emperical Loss} \end{aligned}$$

> This is called **Motecarlo Integration**
>
> when you pick $x_i \in \mathscr{D}$ they are not uniformly selected at random. They are selected with a weight based on their frequency in the data set.
>
> $$\int f(x)dP(x) =$$
>
> $$\int f(x)p(x)dx$$
>
> $$\sum_{x_i \sim \mathscr{U}} f(x_i)p(x_i)$$
>
> where $\mathscr{U}$ means uniform
>
> but if you sum w.r.t. $x_i \sim P(x)$
>
> $$\sum_{x_i \sim \mathscr{P}(x)} f(x_i)\cancel{p(x_i)}$$
>
> the vairance of the estimation error is indepenednt of the dimension of x in theorey you can do it in a very high dimension. But drawing this from $P(x_i)$ in a high dimension is difficult.

The data set is given to you and you won't see it again. We want to make it small for future data not past data.

> emperical loss is an estimation of the expected loss because you can only average over a finite data set $x_i \in \mathscr{D}$

Question does $P(y \mid x)$ minimizes *Emperical Loss*?

> $P(y \mid x)$ minimizes **expected loss** (we cannot compute this)

If you pick a f which is equal to 1 every time $x_i = y_i$

$$f_w(y_i \mid x_i) = \delta(y_1 - \hat{y}(x_i))$$

But if you get data other than exactly what you got in `training` you will not be able to recognize it.

$$\forall \hat{y}(x_i) \mid \hat{y}(x_i) = y_i \rightarrow \text{this gives an over-fitted function}$$

how does the posterior relate to the empirical loss

$$H_{P,f_w} = \mathbb{E} - \log f_w(y \mid x) \rightarrow \text{ Expected Loss}$$

is minimized when

- $\hat{y}(x_i)$ is the expected output

$$f_w(y \mid x) = P(x)P(y \mid x)$$

$$\forall f_w(y_i \mid x_i) = \delta(y_i - \hat{y}(x_i)) \text{ given } \hat{y}(\bullet)$$

why choose KL divergence

IMRE CSISZAR - Why Least-Square & maximum entropy

Blanut - Arimoto

distances between dirstibutions is a non clear question

entropy is a measure of volume of a distribution

entropy is a measure of a distrubution (not vectors or numbers)

- X is a random variable
- H is entropy

$$H(X) = -\sum_{x \sim \mathscr{U}} p_x(x) \log p_x(x)$$

given a bunch of separated δs

- high Entropy
- low Variance

conditional entropy

how much uncertainty in the random variable X contains the random variable Y

$$I(X;Y), H(Y \mid X) - H(X)$$

$$H(Y)$$

> $$\int p(x)dx = 1$$
>
> this could happen when all mass is in one point
>
> or where the mass is spread

connect it back

invariance

classification

- posterior being the best possible fxn
- nuissance var in the data which we can remove without losing information
    - e.g. triangles cannonize them before using them
    - $I \to x$
        - X is our new data

there is residual variability still in the data

no function to remove occulusions

we want to learn that type of invariant

to learn we feed an example into a classifier

if we have a classifier that is a posterior distribution

posterior distribution has all the information in the data that matters for the task and nothing else

all the variability that doesnt matter for the task is removed **minimal sufficient statistic**

local regions don't change much

assume scene moves locally

most people just manually label data then learn a classifier

lecture 14 intro to Deep Learning (3/1)

- data = image x
- dataset = $\mathscr{D} = \{(x_i, y_i)\}_{i=1}^N$
- task = classification $y \in \{1, \ldots, k\}$
- $\hat{y}$ one hot vec corr to prediction
- $\bar{y}$ one hot vector corr to true class

$$x \to f_w \to \hat{y}$$

$$x \mapsto f_w(x) = \hat{y} \approx y$$

$$l(y, \hat{y}) = \begin{cases} 0 & y = \hat{y} \\ 1 & else \end{cases} \to \text{0-1 loss}$$

$$= \|\hat{\bar{y}} - \bar{y}\|_2^2 \to \text{least-squares}$$

$$\hat{y} = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

$$??? \to \text{cross entropy loss}$$

Emperical b/c it's on the data you have experienced

$$\text{Emperical Risk} \quad \sum_{i=1}^{N} l(y_i, \hat{y}(x_i)) \underbrace{P(y_i \mid x_i) P(x_i)}_{(x_i, y_i) \sim P}$$

$$\sum_{(x_i, y_i) \sim \mathscr{D}} l(y_i, \hat{y}(x_i))$$

$$= \sum_{x_i, y_i} -\log P_w(y_i \mid x_i) \to \text{cross entropy loss}$$

$$\hat{w} = \arg\min_w \sum_{(x_i, y_i) \in \mathscr{D}} -\log P_w(y_i \mid x_i)$$

log of each probability

minimize # errors

max # correct output

$$\max \sum_i P(\hat{y}_i > y_i \mid x_i) \to \hat{y}_i = \arg\max_y P(y \mid x_i)$$

$$\hookrightarrow f_w(x) \in \mathbb{R}^k \mid f_w(x_i) \mid_{y_i} \text{ is maximized when } y_i = P(y \mid x_i)$$

$$\hat{w} = \arg\max_w \sum_{i=1}^{N} -\log P_w(y_i \mid x_i)$$

Define

- e_i is a basis vector i.e. 1 hop basis $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = e_1$

$$P_w(y \mid x) = \frac{e^{\langle f_w(x), \bar{y} \rangle}}{\sum_{e_{i=1}} e^{\langle f_w(x), e_i \rangle}} \to \text{Softmax}$$

$$\min \sum_i -\log P_w(y_i \mid x_i) = \sum_i \langle f'_w(x_i) \bar{y}_i \rangle \underbrace{- \log \sum_e e^{\langle f_w, e \rangle}}_{\text{log sum exp [LSE]}}$$

again

$$x \to f_w \to \hat{y} \in \mathbb{R}^k$$

$$f_w(x_i) = \bar{y}_i$$

$$P_w(y_i \mid x_i) = \frac{e^{-\langle f_w(x_i), \bar{y}_i \rangle}}{\sum_j e^{-\langle f_w(x_i), \bar{y}_j \rangle}}$$

Loss

$$\mathscr{L} = \sum_{i=1}^{N} -\log P_w(y_i \mid x_i) = H_{P, P_w}(y \mid x)$$

$$= \underbrace{\sum_{i=1}^{N} \langle fw(x_i), \bar{y}_i \rangle - \log \sum_{j=1}^{k} e^{\langle f_w(x_i), \bar{y}_j \rangle}}_{\text{Emperical cross-entropy}}$$

> Risk is average loss

Entropy (of rand var X) & cross entropy

$$\begin{aligned} H(x) &= H(f) \\ &= H(f_x(\cdot)) \\ &= H(p(x)) \\ &= -\sum_{x_i} p(x_i) \log p(x_i) \to \text{entropy} \\ H_{p,q}(x) &= -\sum_i q(x_i) \log p(x_i) \to \text{cross-entropy} \\ &= \sum_{x_i \sim q(x)} -\log p(x_i) \end{aligned}$$

why is cross-entropy called that

- take regular entropy
- calculate it by taking samples from a different distribution

$$H(X,Y) = \sum_i p(x_i, y_i) \log p(x_i, y_i)$$

conditional entropy

tells you what uncertainty is there on the variable y given that you know the variable x

- p is the true distribution which we do not know
- P_w is the distribution which we have drawn

$$\begin{aligned} H(Y \mid X) &= \sum_i \underbrace{p(x_i, y_i)}_{p(x_i)p(y_i|x_i)} \log p(y_i|x_i) \to \text{conditional entropy} \\ &= \sum_{x_i} \sum_{y_i} p(y_i \mid x_i) \log p(y_i \mid x_i) \\ &= \sum_{x_i \sim p(x)} \sum_{y_i \sim p(y_i|x_i)} \log p(y_i \mid x_i) \end{aligned}$$

Emperical cross entropy

$$H_{P,P_w}(Y \mid X) = \sum_{(x_i,y_i)\sim\mathscr{D}} -\log P_w(y_i \mid x_i)$$

Mutual information between X and Y (not used)

$$\begin{aligned} I(X,Y) &= H(Y \mid X) - H(X) \\ &= H(X \mid Y) - H(Y) \end{aligned}$$

Deep Learning

$$x \in X \to f_w \to z \to \text{activator function} \to y \in Y$$

1. loss function
   i. cross-entropy
   ii. least squares
   iii. 0-1 loss
   iv. ...
2. class of functions

$$f_w : X \to \mathbb{R}^k$$

$$x \mapsto f_w(x) = z \begin{cases} \langle w, x \rangle \\ w^T x \\ w^T x + w_0 \text{affine} \\ w^T \phi(x) \\ \vdots \end{cases}$$

3. optimization

linear

$$L(w) = \sum_{i=1}^{N} \|\bar{y}_i - w^T x_i\|_2^2$$

$$L(w) = \sum_{i=1}^{N} \|\bar{y}_i - w^T \underbrace{\phi(x_i)}_{\text{linear in parameters}} \|_2^2$$

incorporate bias $w_1 \bar{x} = w_1 x + b_1$

deep neural network

$$f_w(x) = \dots w_3 \delta(w_2 \delta(\underbrace{\underbrace{\delta(w_1 \bar{x})}_{\text{activation}=\langle w_1, x \rangle_t = z_1}}_{z_2})) \to \text{SGD}$$

SGD = stochastic gradient descent

$$f_0 = \delta(w_0 x)$$

$$\underbrace{f_N(\dots f_2(f_1(f_0(x))))}_{\text{composition}}$$

$$L(w_i, \mathscr{D}_N) = \sum_{(x_i, y_i) \in \mathscr{D}} - \log p(y_i \mid x_i) = L(w)$$

$$L(w_i, \mathscr{D}_N) = \sum_{i=1}^{N} - \log p(y_i \mid x_i)$$

Over parameterized $|w| \gg N$ `occam's razor example`

make your model simpler in another way

- reduce # parameters
- reduce it's information
    - by injecting noise

Gradient Descent

- w initial set of weights
- w_{t+1}
- w_t previous of the weights
- ∇ gradient

$$w_{t+1} = w_t - \alpha \nabla_w L(w)$$

$$w_{t+1} = w_t + \alpha \sum_{i=1}^{N} \nabla_w \log P_w(y_i \mid x_i)$$

- B mini batch (random subset of $\mathscr{D}$)
- B^c is the compliment
- n is noise

$$w_{t+1} = w_t + \alpha \underbrace{\sum_{(x_i,y_i)\in B}}_{\text{mini batch}} \nabla_w l + \cancel{\sum_{(x_i,y_i)\in B^c} \nabla_w l}$$

$$w_{t+1} = w_t + \alpha \nabla L_w(w) + n_t$$

$$w_{t+1} = w_t + \alpha \sum_{(x_i,y_i)\in B} \nabla_w l$$

`SGD`

we are computing the noisy gradient

SGD was designed to make gradient descent simplier when you have a lot of points

This noise was an undesriable side effect

Gradient Descent

$$\begin{cases} w_t & = w_t + \nabla_w L \\ & = w_t + \nabla_w L \mid_B + \nabla_w L \mid_{B^c} \end{cases}$$

SGD

$$w_t = w_t + \nabla_w L \mid_B$$

$$w_t = \underbrace{\underbrace{w_t + \nabla_w L \mid_B}_{\text{GD}} - \nabla_w L \mid_{B^c}}_{\text{SGD}}$$

lecture 15 Convolutional Neural Networks (3/3)

pass

---

sources:

- SVD
  - wiki
  - mit
- Direct linear transformation
- invitation to 3-d vision
  - For the constraint that the essential matrix has two equal singular values and a third singular value of 0, see the proof of Theorem 5.1 (page 82)
  - You may need to enforce this constraint if the correspondences you use for the eight-point algorithm are not perfect (i.e. they have some noise) as the SVD step for solving for the essential matrix does not enforce the constraint (in the case of perfect correspondences, the constraint will automatically be satisfied).
- lecture 8
  - Szeliski textbook `Section 6.1`
  - Hartley and Zisserman, "Multiple view geometry," cambridge university press 2003 `sections 2 and 4`
- BRDF
- specular reflection