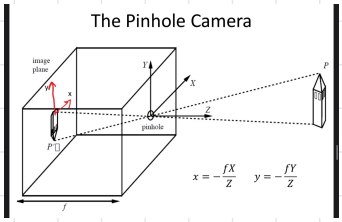
Image Formation

Image $I(x,y)$ measures how much light is captured at pixel (x,y)

① where does a point (X,Y,Z) in the world get imaged

② what's the brightness at the resulting point (x,y)



Projection model — projective transformation

① perspective projection: mapping points from [dept variation] 3D space to rays through proj center

- Parallel lines converge to a vanishing point

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} Ax \\ Ay \\ Az \end{bmatrix} + \lambda \begin{bmatrix} Dx \\ Dy \\ Dz \end{bmatrix} \quad \lambda \to \infty \Rightarrow x = \frac{fX}{Z} = \frac{Ax+\lambda Dx}{\lambda Dz} = \frac{Dx}{Dz}$$

Lines perpendicular to the camera optic axis will not converge

parallel to the image plane

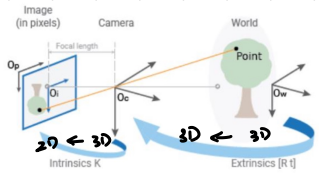
- Farther, smaller. $1/z$
- Tilted with respect to line of sight, smaller. cos Foreshortened

② Orthogonal projection: depth vary not much — affine transformation

Camera Calibration

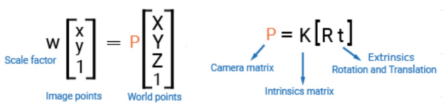
Pinhole camera to real cameras imaging 3D points

↓ silicon-based sensor convert light into electrical voltage {CCD ↑power sci / CMOS ↑noise life



[R t] Extrinsic param: location of the camera in the 3-D scene

K Intrinsic param: optical center and the focal length of the camera



$$\begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

$[c_x \quad c_y]$ — Optical center (the principal point), in pixels.

$(f_x \quad f_y)$ — Focal length in pixels.

$f_x = F/p_x$

$f_y = F/p_y$

F — Focal length in world units, typically expressed in millimeters.

$(p_x \quad p_y)$ — Size of the pixel in world units.

s — Skew coefficient, which is non-zero if the image axes are not perpendicular.

$s = f_x \tan\alpha$

① Tsai's method: $n \geq 6$ Control points 3D position + 2D coordinates in image

$(X_i\ Y_i\ Z_i)$ $i \geq 6$ and $(u_i\ v_i)$ $i \geq 6$ → Direct Linear Transform → $P = \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & \cdots & \cdots & \cdots \\ P_{31} & \cdots & \cdots & \cdots \end{bmatrix}_{3\times 4}$

$$\Rightarrow \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Q: $QP = 0$

$U, S, V = \text{svd}(Q)$

$P = V(:, 12)$

$K, RT = qr(P)$

Metric: reprojection error $= \| p^i - \pi(P_w^i, K, R, T) \|$

Refine by minimizing K, R, T, len distor $= \text{argmin}_{K,R,T,\text{len}} \| p^i - \pi(P_w^i, K, R, T) \|^2$

Levenberg-Marquardt

② Zhang's method: multi-views of a planar grid. $Z=0$ reduce complexity

2D-2D Homography

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

$QH = 0$ ≥ 4 non-colinear

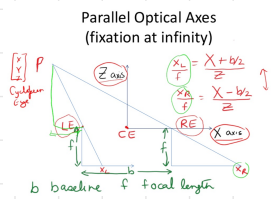
20-50 views for KRT $H = \text{SVD}$

After calibration, camera localization PnP

determine camera pose (R, T) relative to World Frame.

3 (P3P) +1 for disambiguation.

Triangulation



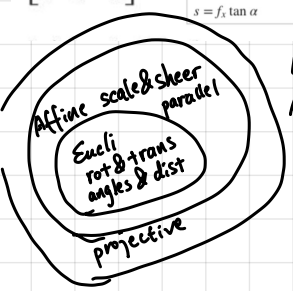
$$\text{disparity} = x_L - x_R = \frac{bf}{Z}$$

$$Z = \frac{bf}{x_L - x_R}$$

Multi-view geometry ⇒ reconstruct internet images

{X camera X 3D shape / ✓ N correspondences ⇒ solve for camera & depth of pts

reflection $\begin{bmatrix} \cos 2\theta & \sin 2\theta \\ \sin 2\theta & -\cos 2\theta \end{bmatrix}$



Eucli: $\varphi(a) = Aa + t$, $A^TA = I$

Affine: $\varphi(a) = Aa + t$, $\det(A) \neq 0$

Homo:

Projective line: $\begin{bmatrix} x \\ 1 \end{bmatrix}, \begin{bmatrix} 2x \\ 2 \end{bmatrix}, \begin{bmatrix} 6.3x \\ 6.3 \end{bmatrix} \ldots$ finite. $\begin{bmatrix} X \\ 0 \end{bmatrix}$ infinite

Projective plane: $\begin{bmatrix} \lambda x \\ \lambda y \\ \lambda \end{bmatrix}$ $\begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$

$a_1x_1 + a_2x_2 + a_3x_3 = 0$

$\begin{bmatrix} x \\ y \end{bmatrix} \leftrightarrow \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$

point on line: $\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 0$

intersect: $\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} x \\ x \\ 0 \end{bmatrix}$ → parallel

Affine: $\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

Perspective proj: $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z/f \end{bmatrix}$

Problem: Explain the concepts of vanishing points and the vanishing line for a plane in 3D under perspective projection.

Solution: **Plane Equation:** Consider a plane in 3D given by

$$N_xX + N_yY + N_zZ = d,$$

where $\mathbf{N} = (N_x, N_y, N_z)$ is the unit normal, and (X, Y, Z) are world coordinates on the plane. **Lines on the Plane:** A line in the plane is parametrized as

$$\mathbf{X}(t) = \mathbf{X}_0 + t\,\mathbf{D},$$

where $\mathbf{X}_0$ is a point on the plane, and $\mathbf{D}$ is the direction vector, satisfying $\mathbf{N} \cdot \mathbf{D} = 0$. **Projection and Vanishing Point:** Under perspective projection, a vanishing point arises as

$$\lim_{t\to\infty} (\text{image of } \mathbf{X}_0 + t\,\mathbf{D}),$$

corresponding to a point at infinity in direction $\mathbf{D}$ in homogeneous coordinates. **Vanishing Line Condition:** All directions $\mathbf{D}$ in the plane satisfy $\mathbf{N} \cdot \mathbf{D} = 0$. Their vanishing points form the vanishing line, expressed in homogeneous coordinates as

$$N_xx + N_yy + N_z = 0.$$

Conclusion: Every line on the plane has a vanishing point on this line, defining the vanishing line associated with the plane.

Algorithm 4 Class-Conditional Sampling

input: one-hot vector c, classifier guidance scale γ
$\mathbf{x}_t = \mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for t from 0 to 1, step size $\frac{1}{T}$ **do**
  $u_{uncond} = u_\theta(\mathbf{x}_t, t, 0)$
  $u_{cond} = u_\theta(\mathbf{x}_t, t, c)$
  $u = u_{uncond} + \gamma(u_{cond} - u_{uncond})$
  $\mathbf{x}_t = \mathbf{x}_t + \frac{1}{T}u$
end for
return $\mathbf{x}_t$

Problem: Use least squares to find the best estimate of the Euclidean planar transformation (translation and rotation) E that minimizes

$$\sum_{j=1}^{4} \|E\mathbf{u}_j - \mathbf{v}_j\|^2,$$

with

$$\mathbf{u}_j = \{(-3,0),(1,1),(1,0),(1,-1)\} \quad \text{and} \quad \mathbf{v}_j = \{(0,3),(1,0),(0,0),(-1,0)\}.$$

Solution: A standard approach separates the estimation of rotation and translation:**Compute the centroids:**

$$\bar{\mathbf{u}} = \frac{1}{4}\sum_{j=1}^{4}\mathbf{u}_j = (0,0), \quad \bar{\mathbf{v}} = \frac{1}{4}\sum_{j=1}^{4}\mathbf{v}_j = \left(0, \frac{3}{4}\right).$$

Center the points: Define

$$\mathbf{u}'_j = \mathbf{u}_j - \bar{\mathbf{u}}, \quad \mathbf{v}'_j = \mathbf{v}_j - \bar{\mathbf{v}}.$$

Compute the cross-covariance matrix:

$$H = \sum_{j=1}^{4}\mathbf{u}'_j(\mathbf{v}'_j)^T = \begin{pmatrix} 0 & -9 \\ 2 & 0 \end{pmatrix}.$$

Obtain the rotation: Using the SVD of $H = U\Sigma V^T$, the optimal rotation is

$$R = VU^T = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix},$$

a 90° rotation. **Compute the translation:**

$$t = \bar{\mathbf{v}} - R\bar{\mathbf{u}} = \left(0, \frac{3}{4}\right).$$

Thus, the best Euclidean transformation is

$$E(\mathbf{u}) = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}\mathbf{u} + \begin{pmatrix} 0 \\ 0.75 \end{pmatrix}.$$

① Epipolar geometry : $\vec{x}, \vec{x}', \vec{t}$ coplanar $x' \cdot (t \times x) = 0$

② Get camera : $x'_{pixel} \underbrace{K'^{-1,T} \hat{T}^{E} R K^{-1}}_{F} x_{pixel} = 0$

$F \xrightarrow{K'^T F K' = E} E \xrightarrow{SVD} T, R$

③ Triangulate for depth $\begin{cases} x_{pixel} = KX \\ x_{pixel} = K'X' = K'(RX+T) \end{cases}$

Linear Least Square x best ∵ noise

skew-symmetric orthogonal

$$[x' y' z'] \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = 0$$

3D→2D

E R

$x'^T_{pixel} K'^{-1}_{T} E K^{-1} x_{pixel} = 0$

depth estim → merge depthmap → MVS → dense reconst → mesh

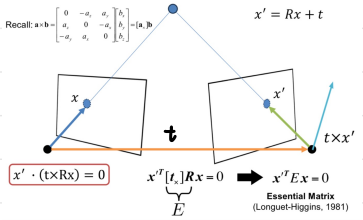
bundle adj → sparse reconst

epipolar line in 2nd image given x

$l' = Fx \quad l = F^T x'$

⇒ reduce search space

Equation of plane



Epipole estimate the motion

① in the image, forward & backward

② outside the image, sideways

③ image center, pure translation, no rot

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \frac{1}{z} \begin{bmatrix} -1 & 0 & x \\ 0 & -1 & y \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} + \begin{bmatrix} xy & -(1+x^2) & y \\ 1+y^2 & -xy & -x \end{bmatrix} \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix}$$

Assume that the camera moves with translational velocity $\mathbf{t} = (t_x, t_y, t_z)$ and angular velocity $\omega = (\omega_x, \omega_y, \omega_z)$. Eq.(3.1) is used to characterize the movement of $\mathbf{X}$,

$$\dot{\mathbf{X}} = -\mathbf{t} - \omega \wedge \mathbf{X}, \tag{3.1}$$

which can be written out in coordinates as Eq.(3.2):

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = - \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} - \begin{bmatrix} \omega_y Z - \omega_z Y \\ \omega_z X - \omega_x Z \\ \omega_x Y - \omega_y X \end{bmatrix}. \tag{3.2}$$

$\dot{x} = \frac{\dot{X}Z - X\dot{Z}}{Z^2}$ $\quad y = \frac{\dot{Y}Z - Y\dot{Z}}{Z^2}$ $\quad$ as $x = \frac{X}{Z}$ $\quad y = \frac{Y}{Z}$

$$= \frac{[-t_x - w_y Z + w_z Y]Z - [-t_z - w_x Y + w_y X]x}{Z^2}$$

$$= \frac{1}{z}(-t_x + x t_z) + xy w_x - (1+x^2) w_y + y w_z .$$

Light field

Amount of light flowing in every direction $L(x, y, z, \theta, \varphi) \Rightarrow (C, \sigma)$

power : $L(p_1, \overrightarrow{p_1 p_2})\, dA \cos\theta \, \frac{dA_2 \cos\theta_2}{r^2}$ $\quad S = \theta r$

Lambertian model : easy version, color x change based on view direction

$L = \rho \lambda (n \cdot s)$ $\begin{cases} \rho = 0 & \text{absort} \\ \rho = 1 & \text{reflect} \end{cases}$

cones / rods are pixels in eye

response : weak light at the peak sensitivity ⇔ strong light at lower sensitivity

$= \int L(\lambda) R(\lambda) d\lambda$ (power, sensitivity)

brightness across the edge changes a lot.

reflectance / illumination / surface geo (depth)

⇒ Bidirectional Reflectance Distribution Function.

$$f(w_i, w_r) = \frac{dL_r(w_r)}{dE_i(w_i)} = \frac{dL_r(w_r)}{L_i(w_i)\cos\theta_i dw_i}$$

radiance o / irradiance i

good design of visual systems ⇒ hierarchical & feedforward

Problem: Verify Rodrigues' formula by considering the powers of the skew-symmetric matrix associated with the cross product with a vector.

Solution: Let $\mathbf{s} = (s_1, s_2, s_3)$ be a unit vector and define its associated skew-symmetric matrix

$$\hat{\mathbf{s}} = \begin{pmatrix} 0 & -s_3 & s_2 \\ s_3 & 0 & -s_1 \\ -s_2 & s_1 & 0 \end{pmatrix}.$$

One can show that

$$\hat{\mathbf{s}}^2 = \mathbf{s}\mathbf{s}^T - \mathbf{I}, \quad \text{and} \quad \hat{\mathbf{s}}^3 = -\hat{\mathbf{s}}.$$

Expanding the matrix exponential in a Taylor series,

$$e^{\phi\hat{\mathbf{s}}} = \mathbf{I} + \phi\hat{\mathbf{s}} + \frac{\phi^2}{2!}\hat{\mathbf{s}}^2 + \frac{\phi^3}{3!}\hat{\mathbf{s}}^3 + \cdots,$$

and grouping even and odd terms using the identities above, one obtains

$$e^{\phi\hat{\mathbf{s}}} = \mathbf{I} + \sin\phi\,\hat{\mathbf{s}} + (1 - \cos\phi)\,\hat{\mathbf{s}}^2.$$

This is Rodrigues' formula for constructing the rotation matrix.

Filtering { Gaussian Filter smooth x presene edge / convolve an image
Anisotropic diffusion edge✓ ∧ comp expensive / High dif in uni + low dif on edge
Bilateral Filter edge✓ ∧ comp exp for ↑ kernel / ↑ weights to similar pixels
Non-Local Means keep details ∧ slow / avg similar patches across the image

⇒ data driven kernel + Full field of view

Algorithm 1 Training

1: **repeat**
2: $\quad \mathbf{x}_1 \sim$ clean image from training set
3: $\quad t \sim \text{Uniform}([0, 1])$
4: $\quad \mathbf{x}_0 \sim \mathcal{N}(0, I)$
5: $\quad \mathbf{x}_t = (1-t)\mathbf{x}_0 + t\mathbf{x}_1$
6: $\quad$ Take gradient descent step on $\nabla_\theta \|(\mathbf{x}_1 - \mathbf{x}_0) - u_\theta(\mathbf{x}_t, t)\|^2$
7: **until** happy

Algorithm 2 Sampling

1: **input**: T timesteps
2: $\mathbf{x}_t = \mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
3: **for** t from 0 to 1, step size $\frac{1}{T}$ **do**
4: $\quad \mathbf{x}_t = \mathbf{x}_t + \frac{1}{T} u_\theta(\mathbf{x}_t, t)$
5: **end for**
6: **return** $\mathbf{x}_t$

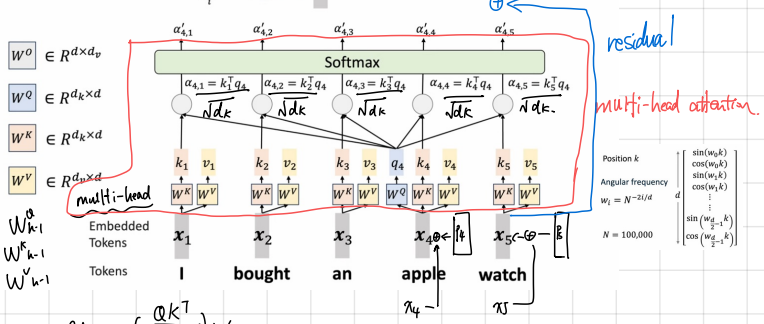
Algorithm 3 Class-Conditional Training

1: **repeat**
2: $\quad \mathbf{x}_1, c \sim$ clean image and label from training set
3: $\quad$ Make c into a one-hot vector
4: $\quad$ with probability p_{uncond} set c to zero-vector.
5: $\quad t \sim \text{Uniform}([0, 1])$
6: $\quad \mathbf{x}_0 \sim \mathcal{N}(0, I)$
7: $\quad \mathbf{x}_t = (1-t)\mathbf{x}_0 + t\mathbf{x}_1$
8: $\quad$ Take gradient descent step on $\nabla_\theta \|(\mathbf{x}_1 - \mathbf{x}_0) - u_\theta(\mathbf{x}_t, t, c)\|^2$
9: **until** happy

```
def homography_solve(u, v):
    N = u.shape[1]
    A = []
    b = []

    for i in range(N):
        x, y = u[0, i], u[1, i]
        x_p, y_p = v[0, i], v[1, i]

        A.append([x, y, 1, 0, 0, 0, -x_p*x, -x_p*y])
        b.append(x_p)
        A.append([0, 0, 0, x, y, 1, -y_p*x, -y_p*y])
        b.append(y_p)

    A = np.array(A, dtype=np.float64)
    b = np.array(b, dtype=np.float64)

    # Solve
    ATA = A.T @ A
    ATb = A.T @ b
    x_ = np.linalg.inv(ATA) @ ATb
    H = np.array([
        [x_[0], x_[1], x_[2]],
        [x_[3], x_[4], x_[5]],
        [x_[6], x_[7], 1.0   ]
    ], dtype=np.float64)

    return H
```

Updated feature $x'_4 = W^O(a'_{4,1} v_1 + a'_{4,2} v_2 + a'_{4,3} v_3 + a'_{4,4} v_4 + a'_{4,5} v_5)$

$= \sum_i a'_{4,1} W^O W^V x_i$



$\text{softmax}\left(\frac{QK^T}{\sqrt{dk}}\right)V$ ViT → MAE, DINO

Variational Autoencoders

- The Encoder maps x to z. $f(x) = z$
- But need to be able to sample from z! like p(z)
- So we need the encoder to learn p(z|x)
- But how to get p(z|x)....? $p(z|x) = \frac{p(x|z)p(z)}{p(x)}$
- What's wrong?
- So learn q(z|x) and make it close to p(z) $DL(q(z|x)||p(z))$
- Through ELBO (will do later):

$$\max \log p(x) >= \mathbb{E}_{q(z|x)} \log p(x|z) - DL(q(z|x)||p(z))$$

- i.e. max p(x|z) while regularize q(z|x) to be close to p(z)

Another perspective

Objective: $\mathbb{E}_{z\sim q(z|x)} \log p(x_i|z) - DL(q(z|x_i)||p(z))$

$D_{KL}(P \,\|\, Q) = \sum_{x \in X} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$.

$D_{KL}(q_\phi(z|x_i)||p(z)) = \mathbb{E}_{z\sim q_\phi(z|x_i)}\left[\log \frac{q_\phi(z|x_i)}{p(z)}\right]$

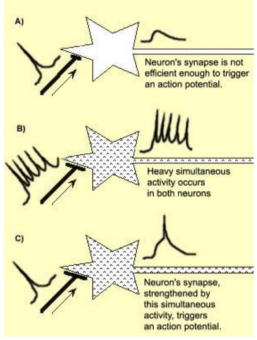
$= \mathbb{E}_{z\sim q_\phi(z|x_i)}[\log q_\phi(z|x_i)] - \mathbb{E}_{z\sim q_\phi(z|x_i)}[\log p(z)]$

Entropy! $-\mathcal{H}(q_\phi(z|x_i))$ $\quad \mathcal{H}(p) = -E_{x\sim p(x)}[\log p(x)] = -\int_x p(x) \log p(x) dx$

$-D_{KL}(q_\phi(z|x_i)||p(z)) = \mathbb{E}_{z\sim q_\phi(z|x_i)}[\log p(z)] + \mathcal{H}(q_\phi(z|x_i))$

Re-written Objective: $\mathbb{E}_{z\sim q(z|x)} \log p(x_i|z) + \log p(z) + \mathcal{H}(q(z|x_i))$

Early History



1943 McCulloch & Pitts Neuron

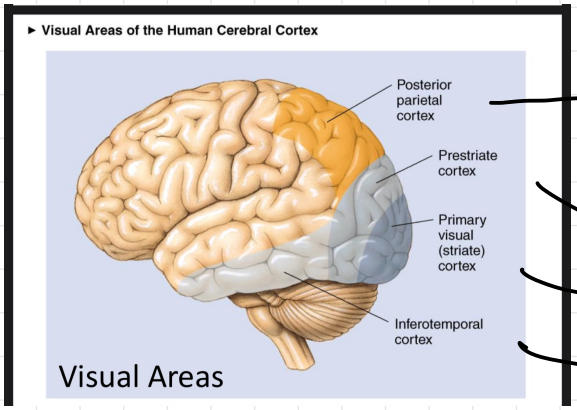
1949 D. Hebb: Synaptic Learning

1950 Turing's suggestion: produce a programme to simulate child

~1960 McCarthy, Minsky, Newell: Represent & reason in first-order logic

Rosenblatt, Widrow: Pattern Recognition, learning

Bellman, Kalman: Estimation & Control (Markov Decision ...)



→ Spatial awareness; Movement planning; Sensory info Integration

② V2 → detect complex visual features (edge, contour, texture)

① V1 → extract fundamental visual features for subsequent steps

→ visual object recognition

1962 Hubel & Wiesel: orientation sensitive neurons in V1

1980 Fukushima: Neural Network model for pattern recognition
∧ Lack the effectiveness of backpropagation ⇒ x scale
(unsupervised)
output abstract repres... i.e. learned feature activations

1989 Yann LeCun Backpropgation to train weights for handwritten digits

& with GPUs, ImageNet

2012 Krizhevsky, Sutskever & Hinton ++ CV obj detection benchmark

3R : Reorganization, Recognition, Reconstruction

6 lessons from baby for embodied intelligence.

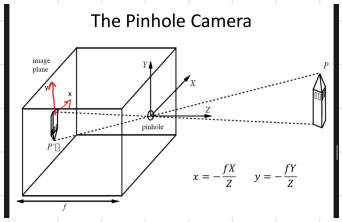
- multi-modal
- be incremental
- be physical
- Explore
- Be social
- language

Image Formation

Image $I(x,y)$ measures how much light is captured at pixel (x,y)

① where does a point (X,Y,Z) in the world get imaged

② what's the brightness at the resulting point (x,y)



Projection model

projective transformation

① perspective projection: mapping points from [depth variation] 3D space to rays through proj center

- Parallel lines converge to a vanishing point

$$\begin{bmatrix}x\\y\\z\end{bmatrix} = \begin{bmatrix}A_x\\A_y\\A_z\end{bmatrix} + \lambda\begin{bmatrix}D_x\\D_y\\D_z\end{bmatrix} \quad \lambda\to\infty \Rightarrow x = \frac{fX}{Z} = \frac{A_x+\lambda D_x}{\lambda D_z} = \frac{D_x}{D_z}$$

Lines perpendicular to the camera optic axis will not converge

parallel to the image plane

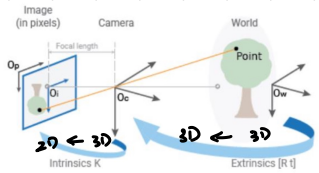
- Farther, smaller. $1/z$
- Tilted with respect to line of sight, smaller. cos Foreshortened

② Orthogonal projection: depth vary not much affine transformation

Camera Calibration

Pinhole camera to real cameras imaging 3D points

↓ silicon-based sensor convert light into electrical voltage {CCD ↑power sci; CMOS ↑noise life



[Rt] Extrinsic param: location of the camera in the 3-D scene

K Intrinsic param: optical center and the focal length of the camera

$$w\begin{bmatrix}x\\y\\1\end{bmatrix} = P\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix} \qquad P = K[R\,t]$$

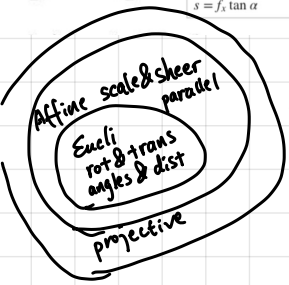
Scale factor · Image points · World points · Camera matrix · Intrinsics matrix · Extrinsics · Rotation and Translation

$[c_x \;\; c_y]$ – Optical center (the principal point), in pixels.

$(f_x \;\; f_y)$ – Focal length in pixels.
$f_x = F/p_x$
$f_y = F/p_y$
F – Focal length in world units, typically expressed in millimeters.
$(p_x \;\; p_y)$ – Size of the pixel in world units.

s – Skew coefficient, which is non-zero if the image axes are not perpendicular.
$s = f_x \tan\alpha$

$$\begin{bmatrix}f_x & s & c_x\\0 & f_y & c_y\\0 & 0 & 1\end{bmatrix}$$



Eucli: $\varphi(a) = Aa + t$, $A^TA = I$

Affine: $\varphi(a) = Aa + t$, $\det(A) \neq 0$

Homo:

Projective line: $\begin{bmatrix}x\\1\end{bmatrix}, \begin{bmatrix}2x\\2\end{bmatrix}, \begin{bmatrix}1.3x\\6.2\end{bmatrix}$ (finite) .. $\begin{bmatrix}x\\0\end{bmatrix}$ (infinite)

Projective plane: $\begin{bmatrix}\lambda x\\\lambda y\\\lambda\end{bmatrix}$ $\begin{bmatrix}x\\y\\0\end{bmatrix}$

$a_1x_1 + a_2x_2 + a_3x_3 = 0$ $\quad \begin{bmatrix}x\\y\end{bmatrix} \leftrightarrow \begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}$

point on line: $\begin{bmatrix}a_1\\a_2\\a_3\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix} = 0$

intersect: $\begin{bmatrix}a_1\\a_2\\a_3\end{bmatrix} \times \begin{bmatrix}b_1\\b_2\\b_3\end{bmatrix} = \begin{bmatrix}x\\x\\0\end{bmatrix}$ → parallel

Affine $\begin{bmatrix}x'\\y'\\w'\end{bmatrix} = \begin{bmatrix}a_{11} & a_{12} & t_x\\a_{21} & a_{22} & t_y\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}x\\y\\1\end{bmatrix}$

Perspective proj $\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1/f&0\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix} = \begin{bmatrix}x\\y\\z/f\end{bmatrix}$

Calibration method.

① Tsai's method: $n \geq 6$ control points 3D position + 2D coordinates in image

$(X_i\ Y_i\ Z_i)$ $i\geq 6$ and $(u_i\ v_i)$ $i\geq 6$ → Direct Linear Transform → $P = \begin{bmatrix}P_{11} & P_{12} & P_{13} & P_{14}\\P_{21} & \cdots & \cdots & \cdots\\P_{31} & \cdots & \cdots & \cdots\end{bmatrix}_{3\times4}$

$\Rightarrow \lambda\begin{bmatrix}u\\v\\1\end{bmatrix} = P\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}$

Q: $QP = 0$ → $U,S,V = \text{svd}(Q)$ → $P = V(:,12)$ → $K, RT = qr(P)$

Metric: reprojection error $= \|p^i - \pi(P^i_w, K, R, T)\|$

Refine by minimizing K, R, T, len distor $= \text{argmin}_{K,R,T,\text{len}} \|p^i - \pi(P^i_w, K, R, T)\|^2$

Levenberg-Marquardt

② Zhang's method: multi-views of a planar grid. $Z=0$ reduce complexity

2D-2D Homography

$$\lambda\begin{bmatrix}u\\v\\1\end{bmatrix} = \begin{bmatrix}h_{11} & h_{12} & h_{13}\\h_{21} & h_{22} & h_{23}\\h_{31} & h_{32} & h_{33}\end{bmatrix}\begin{bmatrix}X\\Y\\1\end{bmatrix} \qquad QH = 0 \quad \geq 4 \text{ non-colinear}$$

20-50 views for KRT $\qquad H = \text{SVD}$

After calibration, camera localization PnP

determine camera pose (R,T) relative to World Frame.

3 (P3P) +1 for disambiguation.

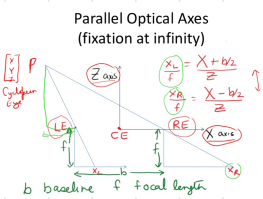
Calibrated: DLT or PnP $\qquad$ Uncalibrated: DLT

$(X_w^i\ Y_w^i\ Z_w^i)$ $i\geq4$ for non-collinear → DLT/PnP → R, T

$(u^i\ v^i)$ → DLT ($i\geq6$ for non-coplanar) → K, R, T

R, T and K, R, T → non-linear optimization → ↓ reprojection error.

Triangulation



$\text{disparity} = x_L - x_R = \frac{bf}{z}$

$z = \frac{bf}{x_L - x_R}$

Multi-view geometry ⇒ reconstruct internet images

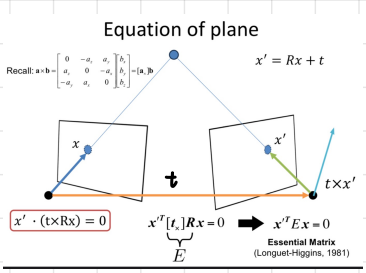
{ ✗ camera ✗ 3D shape / ✓ N correspondences } ⇒ solve for camera & depth of pts

rot $\begin{bmatrix}\cos\theta & -\sin\theta\\\sin\theta & \cos\theta\end{bmatrix}$ reflection $\begin{bmatrix}\cos2\theta & \sin2\theta\\\sin2\theta & -\cos2\theta\end{bmatrix}$

Multi-view geometry

① Epipolar geo: { two camera centers; two image projections of any 3D point } => on the same plane.

② get camera from points using epipolar geo



$\vec{x}' \cdot (t \times \vec{x}') = 0$

$\vec{x}' \cdot (t \times (Rx + t)) = 0$

$\vec{x}' (t \times Rx + t \times t) = 0$

$\vec{x}' (t \times Rx) = 0$

$\vec{x}'^T \underbrace{\hat{T} R}_{E} x = 0$

skew-symmetric, orthogonal

$$\begin{bmatrix} x' & y' & z' \end{bmatrix} \underbrace{\begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}}_{E} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = 0$$

$$\begin{bmatrix} x' & y' & z' \end{bmatrix} \underbrace{\begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix}}_{\text{SVD}} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = 0$$

$E \to T, R$ ✓

Epipole: estimate the motion
① in the image, forward & backward
② outside the image, sideways
③ image center, pure translation, no rot

∧ we don't have $\vec{x}, \vec{x}'$ in 3D
we have corresponding 2D points
$\vec{x}, \vec{x}'$ is in the camera coord
i.e. $x_{pixel} = K\vec{x} = K[R|T] P_w$

$\vec{x} = K^{-1} x_{pixel}$ $\vec{x}' = K'^{-1} x'_{pixel}$

$\Rightarrow x'^T_{pixel} \underbrace{K'^{-T} E K^{-1}}_{F} x_{pixel} = 0$

$F \xrightarrow{K'^T F K'} E \xrightarrow{SVD} T, R$

epipolar line in 2nd image given x

$l' = Fx$ $l = F^T x'$

=> reduce search space

Practice:

Images → feature extraction → feature matching (corresponding points) → estimate camera poses → triangulation 3D points → bundle adjustment → sparse reconstruction → MVS { depth estimation; merge depth map to 3D point cloud } → dense reconstruction

① Epipolar geometry: $\vec{x}, \vec{x}', \vec{t}$ coplanar $\vec{x}' \cdot (t \times x) = 0$

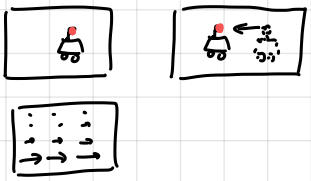
② Get camera: $x'_{pixel} \underbrace{K'^{-T} \hat{T} R}_{F} K^{-1} x_{pixel} = 0$ ($\hat{T} R = E$)

$F \xrightarrow{K'^T F K'} E \xrightarrow{SVD} T, R$

③ Triangulate for depth: Linear Least Square x best ∵ noise { $x_{pixel} = KX$; $x_{pixel} = K'X' = K'(RX + T)$ }

Dynamic Perspective

Moving camera → scene depth



$(\Delta x, \Delta y)$

optical flow $\begin{cases} u = \frac{\Delta x}{\Delta t} \\ v = \frac{\Delta y}{\Delta t} \end{cases}$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \frac{1}{Z}\begin{bmatrix} -1 & 0 & x \\ 0 & -1 & y \end{bmatrix}\begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} + \begin{bmatrix} xy & -(1+x^2) & y \\ 1+y^2 & -xy & -x \end{bmatrix}\begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix}$$

Assume that the camera moves with translational velocity $\mathbf{t} = (t_x, t_y, t_z)$ and angular velocity $\omega = (\omega_x, \omega_y, \omega_z)$. Eq.(3.1) is used to characterize the movement of $\mathbf{X}$,

$$\dot{\mathbf{X}} = -\mathbf{t} - \omega \wedge \mathbf{X}, \tag{3.1}$$

which can be written out in coordinates as Eq.(3.2):

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = -\begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} - \begin{bmatrix} \omega_y Z - \omega_z Y \\ \omega_z X - \omega_x Z \\ \omega_x Y - \omega_y X \end{bmatrix}. \tag{3.2}$$

$$\dot{x} = \frac{\dot{X}Z - X\dot{Z}}{Z^2} \qquad \dot{y} = \frac{\dot{Y}Z - Y\dot{Z}}{Z^2} \qquad \text{as } x = \frac{X}{Z} \quad y = \frac{Y}{Z}$$

$$= \frac{\{-t_x - w_y Z + w_z Y\}Z - \{-t_z - w_x Y + w_y X\}X}{Z^2}$$

$$= \frac{1}{Z}(-t_x + x t_z) + xy w_x - (1+x^2) w_y + y w_z .$$

Radiometry of Image Formation

Image is an array of brightness values. (3 arrays for RGB Images)

Pinhole camera models where a point is projected

Brightness at a pixel (x, y)? ↗ pixel size.

⇒ Irradiance W/m^2 response at a pixel ⇒ $E \propto A \Delta t$

E (all direction) ↙ light intensity ↘ exposure time.

Radience $L = \frac{E}{dA \cos\theta \, d\Omega}$ (in a specific direction)

the power of light travels

Light field

Amount of light flowing in every direction $L(x, y, z, \theta, \varphi) \Rightarrow (C, \sigma)$

power : $L(p_1, \overrightarrow{p_1 p_2}) \, dA \cos\theta \, \frac{dA_2 \cos\theta_2}{r^2}$ $\quad s = \theta r$

Lambertian model: easy version, color x change based on view direction

$L = \rho \lambda (n \cdot s)$ $\begin{cases} \rho = 0 & \text{absort} \\ \rho = 1 & \text{reflect} \end{cases}$ ⇒ Bidirectional Reflectance Distribution Function.

$$f(w_i, w_r) = \frac{dL_r(w_r)}{dE_i(w_i)} \text{ (radience)} = \frac{dL_r(w_r)}{L_i(w_i)\cos\theta_i \, dw_i} \text{ (irradiance)}$$

Color Vision

cones / rods are pixels in eye

response: weak light at the peak sensitivity
<=> strong light at lower sensitivity

$= \int L(\lambda) R(\lambda) d\lambda$
power sensitivity

brightness across the edge changes a lot.
reflectance / illumination / surface geo (depth)

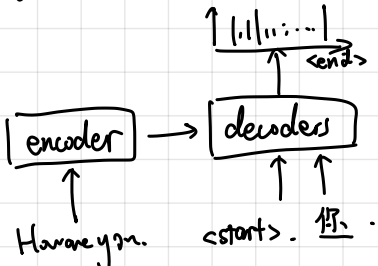
Human Vision

weighting function [-1 0 1] => receptive field
good design of visual systems => hierarchical & feedforward

Sequence Model.

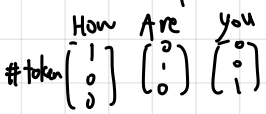
Filtering { Gaussian Filter smooth x presene edge / convolve an image
Anisotropic diffusion edge✓ ∧ comp expensive / High dif in uni + low dif on edge
Bilateral Filter edge✓ ∧ comp exp for ↑ kernel / ↑ weights to similar pixels
Non-Local Means keep details ∧ slow / avg similar patches across the image

=> data driven kernel + Full field of view

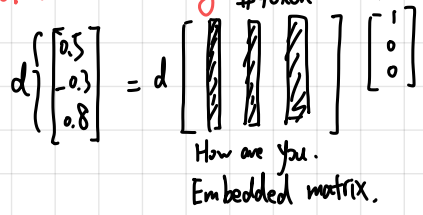


① Tokenization: seprate ["How", "Are", "you"]

② One-hot. ∧ x capture relation/semantic meaning



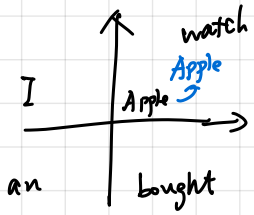
② Token embedding

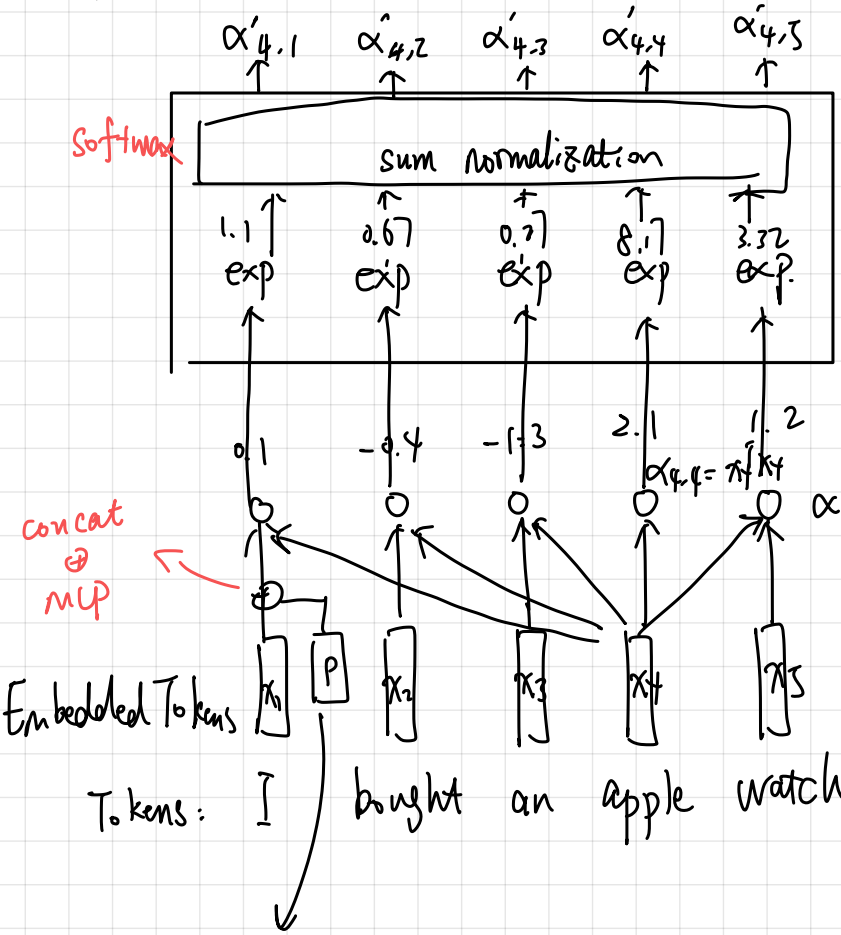


But just embedded matrix is not enough,

Apple should rely on context to solve ambiguity

$\Rightarrow x_4' = \alpha'_{4,1} x_1 + \alpha'_{4,2} x_2 + \cdots \alpha'_{4,5} x_5$





$\alpha_{4,5} = x_4^T x_5$ dot product to caculate similarity

↑ concep relevance is asymmetrical

delicious apple ✓ ✗

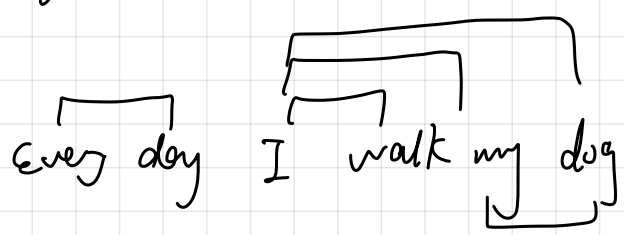
The positional embedding derive intuition from binary encoding

Position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
2^3	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	→ low oscillation
2^2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	
2^1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	
2^0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	→ high oscillation

ViT

Relative positional encoding

I walk my dog every day <=>



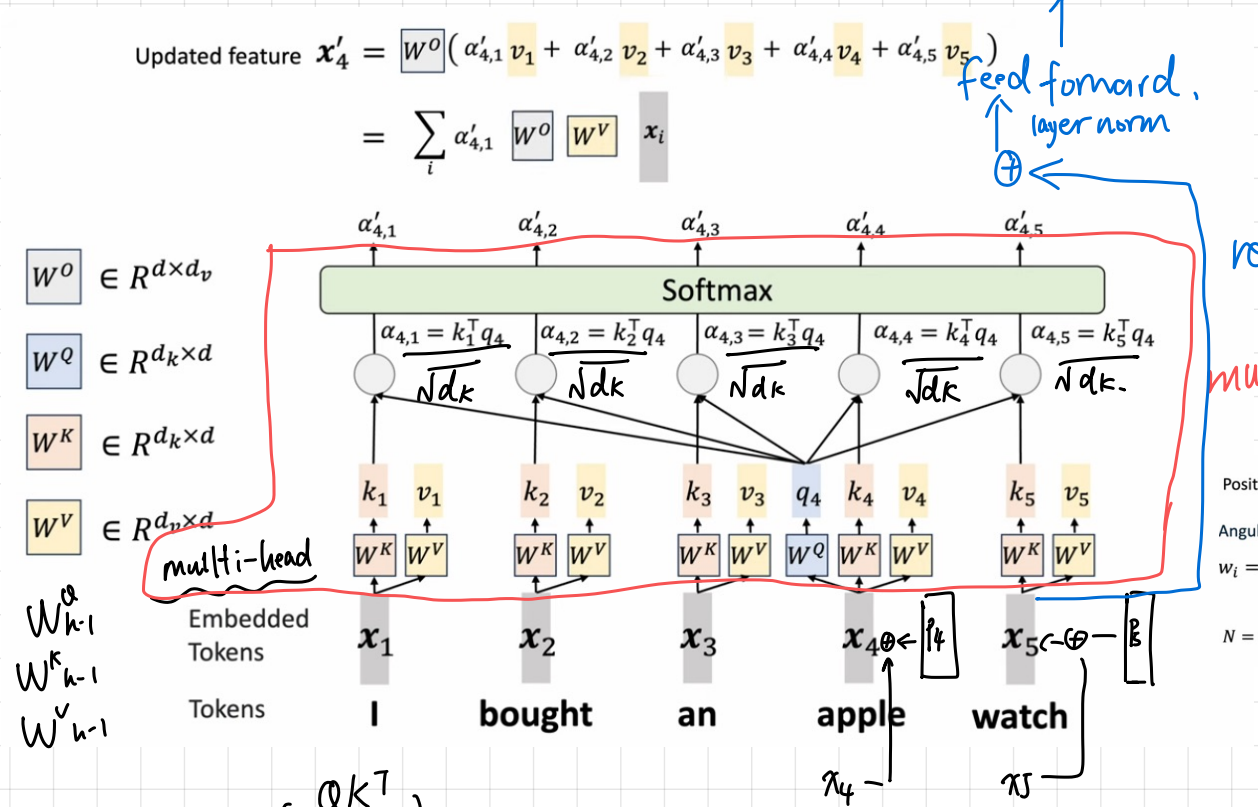
=> Rope

For pixels ?

Naive tokenization of a pixel for 224x224 image.

over 50k tokens => Too much

=> patches + positional embeddings



$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$

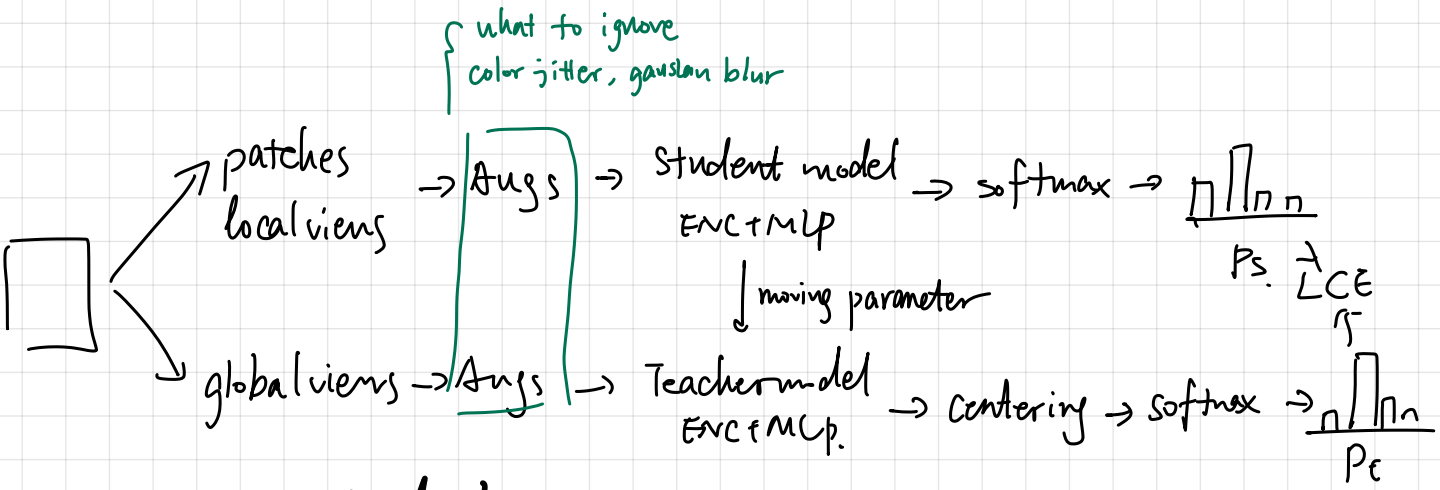
MAE.

$L = \sum (x_i - \quad)^2$

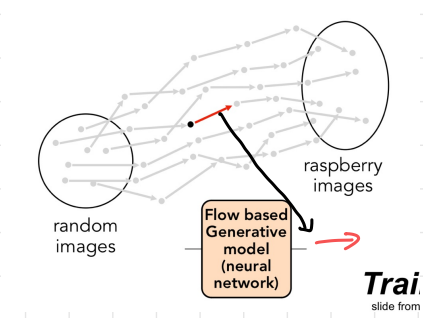
25% remaining patches
75% masked image patches
→ Encoder → compact representation → Decoder →

DINO Distillation with NO Labels

A student network learns from a teacher network



Diffusion model



estimate the probability distribution of images P(x)

① sample from a gaussian distribution $z \sim N(0,1)$

② project to images

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_1 \sim$ clean image from training set
3: $t \sim \text{Uniform}([0,1])$
4: $\mathbf{x}_0 \sim \mathcal{N}(0, I)$
5: $\mathbf{x}_t = (1-t)\mathbf{x}_0 + t\mathbf{x}_1$
6: Take gradient descent step on $\nabla_\theta \left\|(\mathbf{x}_1 - \mathbf{x}_0) - u_\theta(\mathbf{x}_t, t)\right\|^2$
7: **until** happy

Algorithm 2 Sampling

1: **input**: T timesteps
2: $\mathbf{x}_t = \mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
3: **for** t from 0 to 1, step size $\frac{1}{T}$ **do**
4: $\mathbf{x}_t = \mathbf{x}_t + \frac{1}{T} u_\theta(\mathbf{x}_t, t)$
5: **end for**
6: **return** $\mathbf{x}_t$

Algorithm 3 Class-Conditional Training

1: **repeat**
2: $\mathbf{x}_1, c \sim$ clean image and label from training set
3: Make c into a one-hot vector
4: with probability p_{uncond} set c to zero-vector.
5: $t \sim \text{Uniform}([0,1])$
6: $\mathbf{x}_0 \sim \mathcal{N}(0, I)$
7: $\mathbf{x}_t = (1-t)\mathbf{x}_0 + t\mathbf{x}_1$
8: Take gradient descent step on $\nabla_\theta \left\|(\mathbf{x}_1 - \mathbf{x}_0) - u_\theta(\mathbf{x}_t, t, c)\right\|^2$
9: **until** happy

Algorithm 4 Class-Conditional Sampling

1: **input**: one-hot vector c, classifier guidance scale γ
2: $\mathbf{x}_t = \mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
3: **for** t from 0 to 1, step size $\frac{1}{T}$ **do**
4: $u_{uncond} = u_\theta(\mathbf{x}_t, t, 0)$
5: $u_{cond} = u_\theta(\mathbf{x}_t, t, c)$
6: $u = u_{uncond} + \gamma(u_{cond} - u_{uncond})$
7: $\mathbf{x}_t = \mathbf{x}_t + \frac{1}{T} u$
8: **end for**
9: **return** $\mathbf{x}_t$

Objective: $\mathbb{E}_{z\sim q(z|x)} \log p(x_i|z) - DL(q(z|x_i)||p(z))$

Another perspective

$$D_{KL}(q_\phi(z|x_i)||p(z)) = \mathbb{E}_{z\sim q_\phi(z|x_i)}\left[\log \frac{q_\phi(z|x_i)}{p(z)}\right]$$

$$= \underbrace{\mathbb{E}_{z\sim q_\phi(z|x_i)}[\log q_\phi(z|x_i)]}_{\text{Entropy!}\ -\mathcal{H}(q_\phi(z|x_i))} - \mathbb{E}_{z\sim q_\phi(z|x_i)}[\log p(z)]$$

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right).$$

$$\mathcal{H}(p) = -E_{x\sim p(x)}[\log p(x)] = -\int_x p(x)\log p(x)dx$$

$$-D_{KL}(q_\phi(z|x_i)||p(z)) = \mathbb{E}_{z\sim q_\phi(z|x_i)}[\log p(z)] + \mathcal{H}(q_\phi(z|x_i))$$

Re-written Objective: $\mathbb{E}_{z\sim q(z|x)} \log p(x_i|z) + \log p(z) + \mathcal{H}(q(z|x_i))$

Variational Autoencoders

- The Encoder maps x to z. $f(x) = z$
- But need to be able to sample from z! like p(z)
- So we need the encoder to learn p(z|x)
- But how to get p(z|x)….? $p(z|x) = \frac{p(x|z)p(z)}{p(x)}$
- What's wrong?
- So learn q(z|x) and make it close to p(z) $DL(q(z|x)||p(z))$
- Through ELBO (will do later):

$$\max \log p(x) >= \mathbb{E}_{q(z|x)} \log p(x|z) - DL(q(z|x)||p(z))$$

- i.e. max p(x|z) while regularize q(z|x) to be close to p(z)