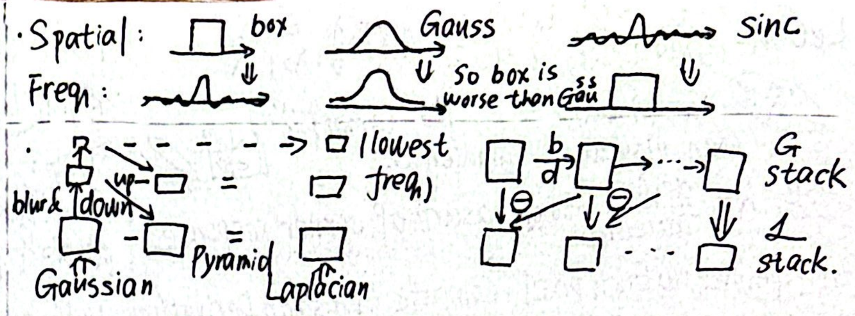
· 平行投影: parallel projection: 所有投影线平行
正交: Orthographic: 不仅平行, 且垂直于投影平面
If COP far away & zoom in (focal length ↑), getting close to parallel perspective.
· Scaled Orthographic or 'Weak Perspective': 先正交, 后 normal scaling. · Spherical Projection: close to eye. And it doesn't depend on focal length: Blurry
· Aperture too large & too small (less light & Diffraction)
· $\frac{1}{d_o}+\frac{1}{d_i}=\frac{1}{f}$ · Aperture ↓, Depth of Field ↑, need more Exposure; F-num = $\frac{\text{focal length}}{\text{aperture diameter}} \propto$ DOF
· FOV: Field of view. f ↑, FOV ↓
· Lens Flaws: ① Chromatic Aberation 色差: 不同λ不同折射率 (edge处明显) ② Center & Fringe处折射率不同. Pincushion: 放大: $c<f$; Barrel: $>$

· Metamer: 同色异谱 Cone 捕 color 亮度, produce one number
· Bayer Grid: 1 B 1 R 2 G · Pupil: Aperture Retina: Film
· RGB Cube Space: Gray: 对角线; Hue: 绕 Gray Diag 角度; Saturation: 到 Gray Diag 距离 | Trichromatic: people with 3 kinds of cones

---

· Lambertian Reflectance Model (cloth, concreate)
$I = I_{source} \cdot \rho \cdot (\underline{N} \cdot \underline{L})$ $(\cos\theta)$ $\in[0,1]$
· More ~~White Balance~~ contrast: ① Power-law: $s = c \cdot r^{\gamma}$
② Negative: $s=1-r$ ③ Contrast Stretch:
④ Log: $S = c \cdot \log(1+r)$ ⑤ Image Histogram
$S = 255 \times CDF(r)$, $r\in[0,255]$, $CDF(\cdot)\in[0,1]$
· Nyquist Theorem: $f_{sampling} > 2 f_{max}$
· To handle wiggly image, can try to get rid of high freq.

· Conv: $G[i,j] = \sum_{u=-k}^{k}\sum_{v=-k}^{k} H[u,v]F[i-u,j-v]$, $G=H*F$
$a*b=b*a$, $a*(b*c)=(a*b)*c$, $a*(b+c)=a*b+a*c$
$\alpha a*b = \alpha(a*b)$, unit impulse e: $a*e=a$; $\sqrt{2}\sigma$
· Conv twice with Gaussian with width σ ⇔ once with
· Image Pyramid: Gaussian Blur, and downsample.
Filter size should (sigma) double for each ½ reduction.
(Filter size 与扣除高频 range 呈正比)
· Partial 结果看起来坚 ⇒ operator 横向作用 In CS180
$\partial(h*f)/\partial x = (\partial h/\partial x)*f$ | Kernel sum: 1 | Conv Flip !!!
· Image - Smoothed = Detail: $f + \alpha\cdot(f - f*g)$
· Def: Unit impulse - Gaussian = Laplacian of Gaussian
FT: $F(w) = \int_{-\infty}^{+\infty} f(x) e^{-iwx}dx$, $f(x) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} F(w) e^{iwx} dw$
$F[g*h] = F[g]F[h]$, $F^{-1}[gh] = F^{-1}[g]*F^{-1}[h]$



collapse (L): Pyramid: $I_i = L_i + \text{Upsample}(I_{i+1})$; Stack: 相加
· Template match: $h[m,n] = \frac{\sum_{k,l}(g[k,l]-\bar{g})(f[m+k,n+l]-\bar{f}_{m,n})}{\left[\sum_{k,l}(g[k,l]-\bar{g})^2\sum_{k,l}(f[m+k,n+l]-\bar{f}_{m,n})^2\right]^{0.5}}$
Slowest, invariant to local average intensity & contrast
· "Median" Filter to eliminate salt & peper ⇒ non-linear (NL), time consuming, but very good. (More robust)

$\begin{bmatrix}x'\\y'\end{bmatrix} = \begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix}$ 左旋θ度; $\begin{bmatrix}x'\\y'\end{bmatrix} = \begin{bmatrix}1 & sh_x\\ sh_y & 1\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix}$
· Linear Transformation: Scale. Rotate. Shear. Mirror/Relflection ⇒ can use 2x2 matrix to represent
· Affine: Linear and translation. $\begin{bmatrix}a & b & c\\ d & e & f\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}x\\y\\w\end{bmatrix} = \begin{bmatrix}x'\\y'\\w\end{bmatrix}$
parallel lines 仍平行, 2D中仍两个 triangle 可互 affine 过去
· Projective transformation: Affine and Projective warp
平行线不必平行, ratios are not preserved (Affine: are)
· DOF: Affine: 6 Projective: 8 Loss of detail
· Bilinear Interpolation: Inverse instead of Splatting
$c(x,y) = ed\,c(x_1,y_1) + gf\,c(x_2,y_2) + bc\,c(x_3,y_3) + ahc(x_4,y_4)$

· Mosaic: Plenoptic Function (全光函数) 描述了一个特定观察点在任意时间和方向上能看到的任意波长光的强度
$P(\theta, \phi, \lambda, t, V_x, V_y, V_z)$.
· As long as COP don't move, can 合成 any camera view who shares the same COP. But can loss 深度信息 (Geometry). Depth info need Multi-view (parallax).
· Perspective distortion: "scale": line parallel to image
foreshortening: line or surface not parallel to image
$A = \begin{bmatrix}x & y & 1 & 0 & 0 & 0 & -x\cdot x_p & -y\cdot x_p\\ 0 & 0 & 0 & x & y & 1 & -x\cdot y_p & -y\cdot y_p\end{bmatrix}$ $h = \begin{bmatrix}h_{11}\\h_{12}\\h_{13}\\ \vdots \\h_{32}\end{bmatrix}$ $b = \begin{bmatrix}x_p\\y_p\end{bmatrix}$
$(x,y,1)^T \xrightarrow{H} (x_p, y_p, 1)$ | ≥ 4 pairs, $\min_h \|Ah-b\|_2^2$
$h^* = (A^TA)^{-1}A^Tb$
· Harris Corner: $E(u,v) = \sum_{(x,y)\in W}[I(x+u,y+v) - I(x,y)]^2$
u,v 小: $E(u,v) = \sum_{(x,y)\in W}[u\ v]\begin{bmatrix}I_x^2 & I_xI_y\\ I_xI_y & I_y^2\end{bmatrix}\begin{bmatrix}u\\v\end{bmatrix}$

Let $M = \sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} = R^{-1}\begin{bmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{bmatrix} R$

$a=(\lambda_{max})^{-1/2}$ $b=(\lambda_{min})^{-1/2}$

Can visualize gradients.

Use $R = \frac{\det M}{\text{Trace}M}$ to: measure of corner response

∴ Pipeline: ① Each pixel: Compute Gaussian Derivative
② Each pixel: Frame window and compute M
③ Compute R ④ Threshold R ⑤ Find local maxima

For window size: choose one of the best corner (R)

Harris: R invariant to rotation, partial invariant to intensity change (scale & shift)

· ANMS: For point set P, $p_i \in P$, compute r_i:
$r_i = \min_{p_j\in P,\, R_j > cR_i} \|p_j - p_i\|_2$, then sort top-k points

· Match: □ → θ orientation
40x40 patch, blur, downsample to 5x5
normalize: $z = \frac{x-\mu}{\sigma}$ ⇒ Descriptor

Then two points' Descripter's L_2 distance: Smaller, the better.

· Lowe: $\frac{\text{1-NN dist}}{\text{2-NN dist}} < \varepsilon$

· Ransac: Loop: ① Select four pairs (Random) ② Compute H, (p_i', Hp_i) ③ Find inliers: dist < ε ④ Keep largest set of inliers ⑤ Recompute H with this !!

· World obj $\xrightarrow{\text{extrinsic}}$ camera $\xrightarrow{\text{intrinsic}}$ image

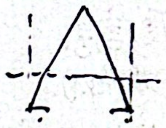
· m_x m_y: pixel/mm, $u_i = m_x f\frac{x_c}{z_c} + o_x = f_x \frac{x_c}{z_c} + o_x$

(o_x, o_y): Image Sensor origin 左上. 但 Image plane origin 中间. $v_i = m_y f\frac{y_c}{z_c}+o_y = f_y\frac{y_c}{z_c}+o_y$

Intrinsic matrix: $\begin{bmatrix}u\\v\\1\end{bmatrix} = \begin{bmatrix} f_x & s & o_x & 0\\ 0 & f_y & o_y & 0\\ 0&0&1&0\end{bmatrix}$ $\in R^{3\times4}$ or $\in R^{3\times3}$, Ex: $\begin{bmatrix}R_{3\times3} & t_{3\times1}\\ 0_{1\times3} & 1\end{bmatrix} \in R^{4\times4}$ $R^{3\times4} = [R|t]$.

DOF: In: 4+1 (skew) Ex: 6

· Camera Calibrated: In & Extrinsic Matrix known.



$\frac{B-(u_l-u_r)}{z-f} = \frac{B}{z}$, $z = \frac{fB}{\underbrace{u_l-u_r}_{\text{Disparity}}}$

· Parallax: Disparity caused by eyes

· Dense Depth Corresponding search: For each epipolar (对极线): For each pixel/window in left image:
· Compare with every pixel/window on same epipolar in right
· Pick position with minimum cost (SSD, correlation).

· Window size: small: lack sufficient intensity variation
big: contain pixels with different depth !!

补充:

· Pixel: □ → (a,b) but center is actually (a+0.5, b+0.5)

· Rigid: $\begin{bmatrix}\cos & -\sin & t_x\\ \sin & \cos & t_y\\ 0&0&1\end{bmatrix}$ Similarity: $\begin{bmatrix}s\cdot\cos & -s\cdot\sin & t_x\\ s\cdot\sin & s\cdot\cos & t_y\\ 0&0&1\end{bmatrix}$

DOF: 3个 4个

· 被观察物体是一个平面，则无论camera 何地何角度拍出来的两个视图之间都可用一个3x3 Homography 关联

```
def pyramid(A, B, base=15).
  h, w = A.shape. If min(h,w) < base return align(A, B, (0,0), delta
dim = (w//2, h//2), A_s, B_s = resize(A, dim), resize(B, dim)
Sub_result = Pyramid(A_s, B_s)
  return align(A, B, origin = 2× sub_result, delta=1)
```

$NCC(a,b) = dot\left(\frac{a-mean(a)}{\|a-mean(a)\|_2}, \frac{b-mean(b)}{\|b-mean(b)\|_2}\right)$.

```
def Conv(img, kernel): H,W = img.shape.
padded = np.pad(image, (kernel[0]//2, kernel[1]//2)).
output = np.zero_like(image)
kernel = np.flipud(np.fliplr(kernel))
for i in range(H)  for j in range(W):
  window = padded[i, i+kernel.shape[0], j: j+kernel.shape[1]]
  output[i,j] = np.sum(window * kernel)
```

$L_k = L_k^A * m_k + L_k^B * (1-m_k)$, m_k 随 k 是 Gaussian Stack

```
homo = np.stack(coord[0], coord[1], np.ones(coord.shape[0]))
source = homo @ H_inv , source = (source/source[:,2])[:, :2]
sour_x = source[:,0], sour_y = source[:,1]
 x0 = np.floor(sour_x).astype(int).  y0 ...
 x1 = x0+1 , y1 = y0+1 , wa = (x1 - sour_x)(y1 - sour_y)
 Ia = Im[x0, y0] , interpol = wa*Ia + ...
im_warped[coord[0], coord[1]] = interpol ...
```

```
Feather: mask = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) > 0
mask = mask.astype(uint8) × 255
dist = cv2.distanceTransform(mask, cv2.DIST_L2).
weights = dist/dist.max().
```

```
anms: for i in range(num_corners):
  mask = h_scores > 0.9 * h_scores[i]
  L2 = np.sum((coord[mask] - coord[i])**2, axis=1).
  radius[i] = np.min(L2). return np.argsort(radius)[::-1][:nip]
```

```
Feather mix: numerator, denominator = np.zeros((out_H, out_W, 3))
for i in range(num_images): H_final = T @ homo_list[i]
warped_image = BilinearWarp(images[i], H_final, (out_H, out_W
weight_mask = creatFeather(warped_image).
numerator += warped_image * weight_mask[:, :, np.newaxis]
denominator += weight_mask[:, :, np.newaxis]
Panorama = numerator / denominator
```

Proj4 Review Pseudocode

- Camera Calibration

```
aruco_detector = (), ob_li, img_li,
tag_3d_corner = np.array([[0,0,0],
[tagsize,0,0],[t,t,0],[0,t,0]]).
for each grayscale(img):
    corner, id = aruco_detector(img).
    if target_id in id:
        pts = corner[target_id].squeeze(axis=0)
        img_li.append(pts.reshape(-1,1,2))
        ob_li.append(tag_3d_c.reshape(-1,1,3))
-, K, dist, -, - = cv2.calibrateCamera
(ob_li, img_li, image_size).
```

- Pose estimation

```
for each grayscale(img):
    corner, id = aruco_detector.detect(img)
    tag_2d = corner[target_id][0]
    -, rvec, tvec = cv2.solvePnP
    (tag_3d_corner, tag_2d, K, dist)
    w2c = [R t; 0001]  R为cv2.Rodrigue(rvec)
    c2w = inverse(w2c)
    c2w_li.append(c2w)
    undistort.append(undistort(img, K, dist))
```

```
-PE(x) = {x, sin(2^0 πx), cos(2^0 πx) ...}
enc_li = [input]
-for i from 0 to L-1:
    enc_li.append(sin(2^i·pi·input))
    enc_li.append(cos(2^i·pi·input))
result = Concat(enc_li)
```

- Pixel2Ray: (K, c2w, uv)

```
uv_homo = [uv.x, uv.y, 1]
c_direction = inv(K) @ uv_homo
R = c2w[:3,:3]
o (translation) = c2w[:3,3]
w_direction = R @ c_direction
d = normalize(w_direction)
return o, d
```

$r_d = \frac{X_w - r_o}{\|X_w - r_o\|_2}$

- Volumn Rendering (sigma, rgb, step_size)

```
alphas = 1.0 - exp(-sigma*step_size)
trans = torch.cumprod(torch.cat([
torch.ones(B,1,1), 1-alphas[:, :-1]])
rendered = torch.sum(alphas*Trans
*rgbs, dim=1), return rendered.
```

- Epipolar Geometry + Calibration:

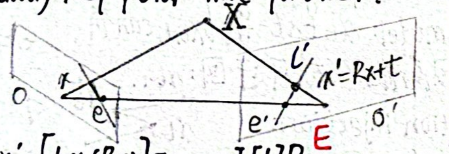
Baseline: line connecting two camera centers

Epipolar plane: Plane containing baseline

Epipoles = Intersections of baseline with image planes = projs of other camera center

So epipolar line has to go through epipole!

·If image planes parallel to camera center baseline ⇒ epipoles infinitely far away, epipolar line parallel!



$x' \cdot [t \times (Rx)] = 0$, $x'^T \underbrace{[t]_\times R}_{E} x = 0$

Recall: $a \times b = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = [a]_\times b$

So Ex is the epipolar line associated with x ($l' = Ex$)   E: DOF: 5 ($l = E^T x'$)

·If uncalibrated, $\hat{x}$ is image coord, so $x = K^{-1}\hat{x}$, $x' = K'^{-1}\hat{x}'$ DOF: 7

$\Rightarrow \hat{x}'^T F \hat{x} = 0$, $F = (K'^{-1})^T E (K^{-1})$

·Even if you do everything right, you will still be off because of noise which is called Reprojection Error.

·If we don't know the camera, we know points correspondence $(\hat{x}', \hat{x})$: $[u'\ v'\ 1]\, F \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0$ $\hat{x}'^T F \hat{x}$

We need eight or more matches!

·Essential Matrix E can be 分解!

$E = [t]_\times R$ 知道,是可知t&R, 因$[t]_\times$矩阵形式特殊!

·How to calibrate camera?

① Take a picture of an obj with known 3D geometry

② Identify correspondence.

③ $\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \underbrace{\begin{bmatrix} m_{11} & \cdots & m_{14} \\ & & \vdots \\ m_{31} & \cdots & m_{34} \end{bmatrix}}_{M} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$

$\Leftrightarrow \begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 & -u_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 & -v_1 \\ & & & & & \vdots & & & & & & \end{bmatrix}$

$\cdot \begin{bmatrix} m_{11} \\ \vdots \\ m_{34} \end{bmatrix} = \vec{0} \Rightarrow$ need six or more matches!

④ M factor back to K[R t] 因K有特殊形式 ⇒ QR分解

如进,3D object 常用一个plane 则依Zhang's Method, Recover 3x3 and decompose: $\begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix}$

·How to pose estimation?

① Calibrate (K)   ② Undistort

③ Perspective-n-Point:

$\min_{R,T} \sum \|x_i - K[R\ T]X\|^2$

- Structure from Motion

·Feature detection with SIFT. And then match features between each pair of images. Refine matching using RANSAC to estimate fundamental matrix between each pair

最终得到一个点在各图片中的位置

·Then: $g(X,R,T) = \sum_{i=1}^{m}\sum_{j=1}^{n} \underbrace{w_{ij}}_{\text{point i visible in j?}} \cdot \left\| P(x_i, R_j, t_j) - \begin{bmatrix} u_{ij} \\ v_{ij} \end{bmatrix} \right\|^2$

Minimizing this is called bundle adjustment!

·Incremental SfM:

① Pick a strong initial pair of images

② Initialize model with 2-frame SfM

③ While there are connected img remaining:

a. Pick img which sees most existing 3D points

b. Estimate the pose of that camera

c. Triangulate any new points.

d. Run bundle adjustment.

·附 triangulate: 知 K, x, x', K', R, T.

则 $x = KX$, $x' = K'(RX+T)$

依然 $Ax=0$, 一组(x, x')给两个sample. X有3个unknown, 故2个(x, x')对应pair足够

·Issue: Output Sparse point cloud

·Volumentric Stereo: Assign RGB values to voxels in V photo-consistent with images.

·Space Carving: Initialize to a volumn V. Choose a voxel on the outside of volumn. Project to visible input images and carve if not photo-consistent. Repeat

- NeRF.   ·Three key components:

① Neural Volumetric 3D Scene Representation

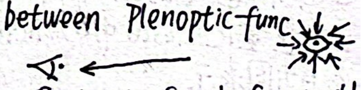
② Volumetric Rendering function.

③ Optimization by analysis-by-synthesis

·NeRF represents scenes as an implicit func $(x, y, z, \theta, \phi) \to F_\Omega(\text{MLP}) \to (r, g, b, \sigma)$.

·Big assumption! A ray never changes color

·Difference between Plenoptic func and nerf



·Analysis-by-Synthesis: Search for a world state from which you can explain many obs through synthesis

· Small diff in input, but output can be very diff:

$\Rightarrow PE(x) = [x, \sin(2^0\pi x), \cos(2^0\pi x) \ldots]$

· Render. Multi-plane Image

Alpha Blend: $I = C_a\alpha_a + C_b\alpha_b(1-\alpha_a)$

$\Rightarrow I = \sum_{i=1}^{D} C_i\alpha_i \prod_{j=1}^{i-1}(1-\alpha_j)$ 从远到近，逐层叠上

For ray $r(t) = o + td$

$c(\text{color}) = \sum_{i=1}^{n} T_i\alpha_i c_i$, where:

$T_i = \prod_{j=1}^{i-1}(1-\alpha_j)$ ← How much light leak through. Accounts for occlusion

$\alpha_i = 1 - \exp(-\sigma_i\delta_i)$ ← How much light by segment i

δ_i is the step size

— Texture · Julesz: Conjecture:

Texture can't be spontaneously discriminated if they have same first order and second order statistics of texton and differ only in third or higher order statistics.

· Texture Representation ① LM FilterBank for 图像特征提取

② 表示方法：各种类型直方图

a. Pixel: 统计灰度值，差 特征

b. Filter response: 捕捉纹理

c. Joint vs Marginal: 需要大量数据且分辨率低 vs. 需要特征独立

· Bag of Visual Words ① 从图像提取大量局部图像块(patches)

② 字典构建: clustering

③ 向量量化: cluster中心为视觉单词

④ 直方图表示

GAN: Reconstruction + fool fake img detector

(D): $\arg\max_D E_{x,y}[\log D(G(x)) + \log(1-D(y))]$

Conditional GAN: $D(x,y)$ 判断 (x,y) pair 是否真实

$\arg\min_G \max_D E_{x,y}[\log D(x,G(x)) + \log(1-D(x,y))]$ 匹配

不仅看输出img是否逼真 还看是否与input匹配

— Generative, flow matching and Diff

认为小部分有意义图片位于巨大图像空间中特定的流形(manifold)

· Parametric Texture Synthesis

① Convolve input & noise with filter bank

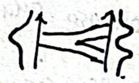
② Match per-channel histograms

指强制调整噪声Filter响应值使其直方图与目标纹理一致

③ Collapse Pyramid and Repeat.

· Diffusion 用NN预测噪声而后去噪

Flow Matching



· Flow, Velocity Field.

Riding the river: Integration.

希望学速度场 $u_\theta(x,t)$，通过在速度场中flow可将噪声分布变为目标图分布

· 不学位置映射 $\psi_t(x)$ 而学速度场 $u_t(x)$. (我往哪个方向走，速度多少).

· Construct x_t: $x_t = \alpha_t x_0 + \sigma_t x_1$

$x_0 \sim p_0(x)$ $x_1 \sim p_1(x)$, 则:

$\frac{dx_t}{dt} = x_1 - x_0$, 故loss: 向量 恒定速度

$\mathbb{E}_{t,x_0,x_1} \| u_t^\theta(x_t) - (x_1 - x_0) \|^2$

$x_{t+\Delta t} = x_t + \Delta t \cdot \frac{dx}{dt}\Big|_{x_0,t}$

· Algorithm: Flow Matching training

Input: dataset q_1, noise p, ini: v^θ.

while not converged do:

$t \sim \mathcal{U}([0,1])$.

$x_1 \sim q_1(x_1)$.

$x_0 \sim p(x_0)$.

$x_t = \Psi_t(x_0 | x_1)$.

GD with $\nabla_\theta \| v_t^\theta(x_t) - \frac{dx_t}{dt} \|^2$.

· Algorithm: Diffusion.

Input: dataset q_1, noise p, ini s^θ

while not converged do:

$t \sim \mathcal{U}[0,1]$.

$x_1 \sim q_1(x_1)$

$x_t = p_t(x_t | x_1)$

GD with: $\nabla_\theta \| s_t^\theta(x_t) - \nabla_{x_t}\log p_t(x_t|x_1) \|^2$

Diffusion Sampling:

· Flow model: $x_t + \Delta t \cdot \frac{dx}{dt}\big|_{x_0,t}$ can有问题因为离散化，故可迂回: x_{t-1}. $\hat{x}_t$ x_t $\hat{x}_{t+1}$

· Condition injection:

① Classifier Guidance

Use classification loss gradient as guidance vector. But not favored.

Middle images OOD & 联合训练难!

② CFG: Unet to predict noise.

but: $\epsilon_\phi(\cdot;\cdot)$ (image, condition).

At one stop, generate both:

$\epsilon_\phi(x_t; c)$ & $\epsilon_\phi(x_t; \emptyset)$.

Then $\epsilon_\phi(x_t; c) - \epsilon_\phi(x_t; \emptyset)$.

是guidance，让生成图片强烈推向guidance方向

Proj 5A. ① $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon$

② One step Denoise: $\hat{x}_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\,\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}}$

③. Iterative Denoising with speed up.

$x_{t'} = \frac{\sqrt{\bar{\alpha}_{t'}}\,\beta_t}{1-\bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t'})}{1-\bar{\alpha}_t} x_t + \sigma$.

且 $\alpha_t = \frac{\bar{\alpha}_t}{\bar{\alpha}_{t'}}$, $\beta_t = 1 - \alpha_t$. $\tilde{\beta}_t = \left(\frac{1-\bar{\alpha}_{t'}}{1-\bar{\alpha}_t}\right)\beta_t$ $\sqrt{\tilde{\beta}_t}$

$\sigma \sim \mathcal{N}(0,1)$.

④. CFG: $\epsilon = \epsilon_u + \gamma(\epsilon_c - \epsilon_u)$.

⑤. Visual Anagrams (错觉)

Up view: c_1 Down view: c_2

$\epsilon_1 = $ CFG of Unet(x_t, t, c_1).

$\epsilon_2 = $ flip(CFG of Unet(flip(x_t), t, c_2)). $\epsilon = (\epsilon_1 + \epsilon_2)/2$

⑥ 远看(lowfreq): c_1; 近看(highfreq) c_2

$\epsilon_1 = $ CFG of Unet$(x_t; t; c_1)$

$\epsilon_2 = $ CFG of Unet$(x_t; t; c_2)$.

$\epsilon = f_{\text{lowfreq}}(\epsilon_1) + f_{\text{highfreq}}(\epsilon_2)$.

为了获得更多density信息，应视角垂直以add info

长 宽

$1920 \times 1080 \Rightarrow (1, 1080, 1920)$. row col

DDPM: For t from T to T/N. step size $-T/N$: do:

$t' = t - T/N$, $z \sim \mathcal{N}(0,1)$ if $t > T/N$ else $z = 0$