# Improving Diffusion Model Sampling Speed, Generalization of Diffusion on Riemannian Manifolds, and Applications to Biometrics

Luke Chen

B.S. UC Berkeley Mathematics + Computer Science

Masters Student | KAIST Mathematical Sciences | Professor Wooseok Ha Lab (WHA Lab)

# Problem Setting

- $t = 0$ (data) $< \cdots < t = T$ (noise)
- **Forward SDE** (Variance Exploding): $dX_t = \sqrt{2t} dW_t$
- **Reverse SDE**: $dX_t = -2t \nabla_{x_t} log(p_t(x_t))dt + \sqrt{2t} dW_t$
- **Probability Flow ODE**: $dX_t = -t \nabla_{x_t} log(p_t(x_t))dt$
- **Kolmogorov Backward PDE**: $\frac{\partial p}{\partial t} + t \frac{\partial^2 p}{\partial x^2} = 0$, $p(x, T) = G(x)$ where $G = N(0, t^2 I)$
- **Feynman-Kac Formula**: $p(x, t) = \mathbf{E}[G(X_T)|X_t = x]$

# Feymann-Kac Formula

- Theorem. Consider the stochastic differential equation

$$dX(s) = \beta(s, X(s))ds + \gamma(s, X(s))dW(s).$$

Let $h(y)$ be a Boreal-measurable function. Fix $T > 0$, and let $t \in [0, T]$ be given. Define the function

$$g(t, x) = E^{t,x}h(X(T)).$$

Then $g(t, x)$ satisfies the following PDE of parabolic type:

$$g_t(t, x) + \beta(t, x)g_x(t, x) + \frac{1}{2}\gamma^2(t, x)g_{xx}(t, x) = 0,$$

with the terminal condition:

$$g(T, x) = h(x), \quad \text{for all} \quad x.$$

# Setting (continued)

- By solving $p(x,t)$ given by the Feynman-Kac formula through evaluating the expectation directly we obtain $\nabla_x logp(x,t) = -\frac{x}{(T^2-t^2)}$
- Now the **PF-ODE** becomes $\mathbf{dX_t} = \frac{\mathbf{tX_t}}{\mathbf{T^2-t^2}}\mathbf{dt}$
- We can solve analytically with a given terminal condition (noise), and try to use this to produce data samples directly without a trained model
- Analytical sol is $x(t) = \frac{C}{\sqrt{T^2-t^2}}$ where $C$ is constant
- However, this does not work as we can't sample $X_T$ due to blow up at $t = T$ and there is no information about the data distribution provided in the generation process...

# Setting (continued)

- We may try using $X_t = \frac{c}{\sqrt{T^2 - t^2 + \epsilon}}$
- Then construct couplings $(X_0, \frac{\sqrt{T^2+\epsilon}X_0}{\sqrt{T^2-t^2+\epsilon}})$
- Now sampling $X_T$ is just like sampling $t$ very close to $T$ where the degree is controlled by $\epsilon$
- We can parametrize $f_\theta(X_t, t)$ and try to minimize $|X_0 - f_\theta(\frac{\sqrt{T^2+\epsilon}X_0}{\sqrt{T^2-t^2+\epsilon}}), t)|_2^2$ over the couplings, optionally choosing $0 < t \leq T$ randomly or just restricted to $t = T$
- In a similar spirit to Consistency Models, another option is to minimize $|f_\theta(\frac{\sqrt{T^2+\epsilon}X_0}{\sqrt{T^2-t_{n+1}^2+\epsilon}}), t_{n+1}) - f_\theta(\frac{\sqrt{T^2+\epsilon}X_0}{\sqrt{T^2-t_n^2+\epsilon}}), t_n)|_2^2$

# Experiments

- Solving analytically with initial condition $X_0$ gives $X_t = \frac{\sqrt{1+T^2}}{\sqrt{1+T^2-t^2}} X_0$
- Hence we can construct couplings $(X_0^{(i)}, \frac{\sqrt{1+T^2}}{\sqrt{1+T^2-t^2}} X_0^{(i)})$ for some chosen $0 < t \leq T$
- We first verify that $t = T$ produces noise-like samples

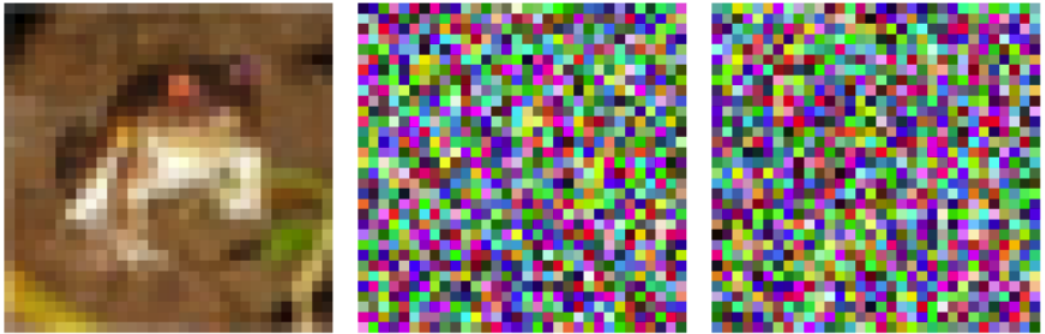


Figure: Noise generated by analytical trajectory (middle) and by adding explicit Gaussian noise defined by VE-SDE (right) from same image sample (left)

# Experiments (continued)

- Now we train a neural net model to match these couplings
- (1) Try with randomly selected intermediate $t$ values
- (2) Try only with $t = T$

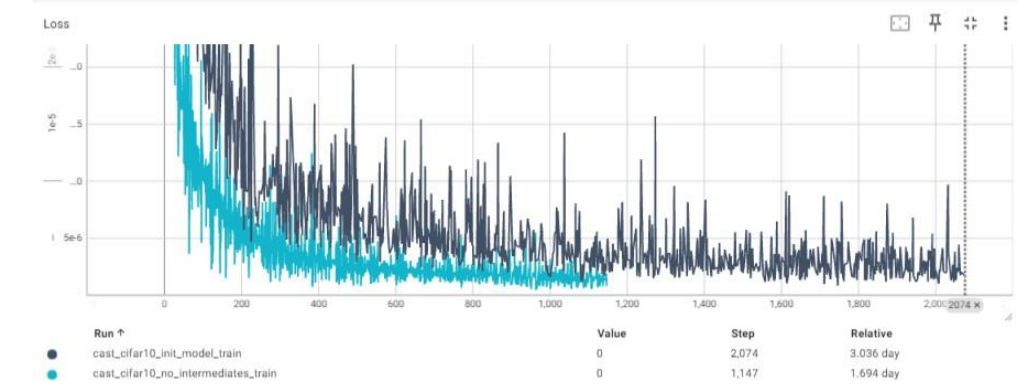



Figure: Loss progression of (1) in black and (2) in blue

# Experiments (continued)

- (1) Results are poor when model uses randomly sampled noise as input for generation
- (2) Results are solid (FID-wise) when model uses 'noise' samples created from images via the analytical trajectory (it performs near-perfect reconstruction, i.e. overfitting to dataset)

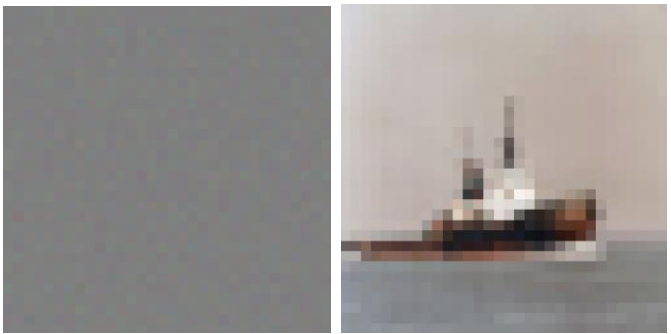


Figure: Image generated by (1) on the left and (2) on the right

# Next Steps

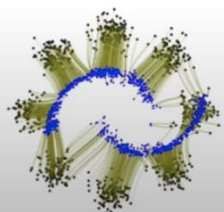
- (1) Try training with an additional 1M+ images generated by EDM, same as how CAF trained, to alleviate overfitting
- (2) Try using a different loss function (currently loss is MSE between image and network output) or add sub-losses such as GAN
- (3) Try using a different loss formulation?
- Additionally, could change model architecture (currently using CAF's velocity network architecture - although it is highly unlikely changing this will yield significant improvements..)

# Conditional Flow Matching / Rectified Flows (Euclidean Space)

## Introduction to Conditional Flow Matching (Lipman et al., 2023; Tong et al., 2023)

- Conditional Flow Matching (**CFM**) is **a simulation-free framework** designed to find a pushforward map that transforms a source distribution $p_0$ into a target distribution $p_1$.
- As in Continuous Normalizing Flows, the pushforward is characterized as the flow map $\psi_1$, generated by a time-dependent vector field $u_t$, where $\frac{d\psi_t(x))}{dt} = u_t(\psi_t(x)$.
- In CFM to learn vector field $u_t$ based on data, it is sufficient to learn on **pairs of samples** related by paths.
- CFM typically uses Euclidean geometry and **straight-line interpolants**,

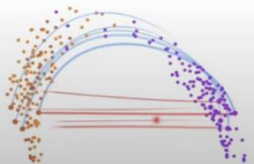
$$x_t = tx_1 + (1 - t)x_0.$$

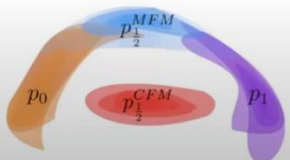


# Manifold Hypothesis

## Motivation

- To ensure a valid reconstruction of the trajectory, the support of $p_t$ should remain within manifold $\mathcal{M}$.
- Operating on $\mathcal{M}$ is challenging as it requires different coordinate system and may incur overfitting.
- The manifold hypothesis posits that $\mathcal{M}$ concentrates around data points → **interpolants $x_t$ should remain close to data points $\mathcal{D}$.**

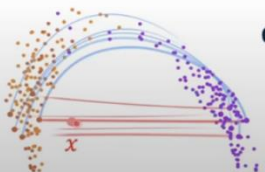




# Metric Flow Matching

## The Foundation of MFM: Data-Dependent Metric

- We still operate in the **ambient space,** but we **change the geometry** to assign higher costs to regions away from the data, achieved by learning a suitable Riemannian metric.
- A **Data-Dependent Metric** $\mathbf{G}(x; \mathcal{D})$ is a map that assigns a symmetric positive definite matrix to each point in the ambient space parametrized by the dataset $\mathcal{D}$.
- We restrict metrics to diagonal matrices.



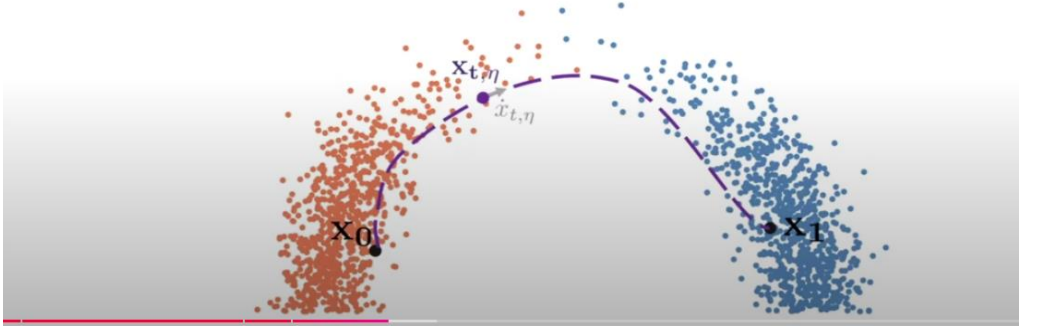

$$\mathbf{G}(x; \mathcal{D}) \in \{\mathrm{diag}(v) : v \in \mathbb{R}^d_{>0}\}$$

# Metric Flow Matching

**Algorithm 2** Pseudocode for METRIC FLOW MATCHING

**Require:** coupling $q$, initialized network $v_{t,\theta}$, **trained** network $\varphi_{t,\eta}$, data-dependent metric $g(\cdot)$

**while** Training **do**
  Sample $(x_0, x_1) \sim q$ and $t \sim \mathcal{U}(0,1)$
  $x_{t,\eta} \leftarrow (1-t)x_0 + tx_1 + t(1-t)\varphi_{t,\eta}(x_0, x_1)$
  $\dot{x}_{t,\eta} \leftarrow x_1 - x_0 + t(1-t)\dot{\varphi}_{t,\eta}(x_0, x_1) + (1-2t)\varphi_{t,\eta}(x_0, x_1)$
  $\ell(\theta) \leftarrow \|v_{t,\theta}(x_{t,\eta}) - \dot{x}_{t,\eta}\|^2_{g(x_{t,\eta})}$
  Update $\theta$ using gradient $\nabla_\theta \ell(\theta)$
**return** vector field $v_{t,\theta}$



# Future Research Direction

- Treat Probability Density of Iris Image Sample (or Face Image Sample) as a Riemannian Manifold
- Apply the Metric Flow Matching algorithm with potential modification

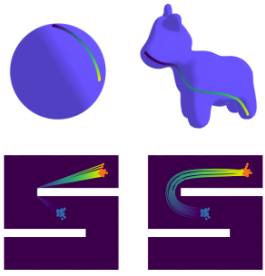



Figure 1: Our approach makes use of user-specified *premetrics* on general manifolds to define flows. On select simple manifolds, the geodesic can be computed exactly and leads to a *simulation-free* algorithm. On general manifolds where the geodesic is not only computationally expensive but can lead to degeneracy (*e.g.*, along boundaries), we propose the use of *spectral* distances (*e.g.*, biharmonic), which can be computed efficiently contingent on a one-time processing cost.

# Riemannian Flow Matching (Ricky T.Q. Chen, Yaron Lipman)

Simple yet powerful methodology for learning continuous normalizing flows on a general Riemannian manifold M

Time Dependent Flow on M is defined by solving the following ODE over t ∈ [0, 1]

$$\frac{d}{dt}\psi_t(x) = u_t(\psi_t(x)), \qquad \psi_0(x) = x,$$

(Chen et al., 2018)

# Riemannian Flow Matching Algorithm (Typical ODE Solver = dopri15)

**Algorithm 1** Riemannian CFM

**Require:** base $p$, target $q$, scheduler $\kappa$
  Initialize parameters $\theta$ of $v_t$
  **while** not converged **do**
    sample time $t \sim \mathcal{U}(0,1)$
    sample training example $x_1 \sim q$
    sample noise $x_0 \sim p$
    **if** simple geometry **then**
      $x_t = \exp_{x_1}(\kappa(t)\log_{x_1}(x_0))$
    **else if** general geometry **then**
      $x_t = \texttt{solve\_ODE}([0,t], x_0, u_t(x|x_1))$
    **end if**
    $\ell(\theta) = \|v_t(x_t;\theta) - \dot{x}_t\|_g^2$
    $\theta = \texttt{optimizer\_step}(\ell(\theta))$
  **end while**

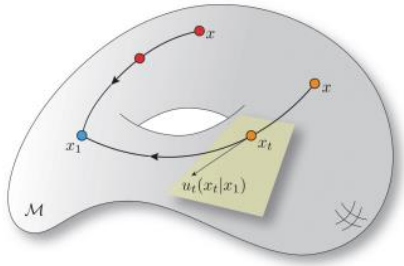



Figure 2: The conditional vector field $u_t(x|x_1)$ defined in equation 13 transports all points $x \neq x_1$ to $x_1$ at exactly $t = 1$.

(Chen et al., 2018)

# Riemannian Flow Matching on Earth & Climate Science Datasets

Table 2: Test NLL on Earth and climate science datasets. Standard deviation estimated over 5 runs.

| Dataset size (train + val + test) | Volcano 827 | Earthquake 6120 | Flood 4875 | Fire 12809 |
|---|---|---|---|---|
| *CNF-based* | | | | |
| Riemannian CNF (Mathieu & Nickel, 2020) | -6.05±0.61 | 0.14±0.23 | 1.11±0.19 | -0.80±0.54 |
| Moser Flow (Rozen et al., 2021) | -4.21±0.17 | -0.16±0.06 | 0.57±0.10 | -1.28±0.05 |
| CNF Matching (Ben-Hamu et al., 2022) | -2.38±0.17 | -0.38±0.01 | **0.25**±0.02 | -1.40±0.02 |
| Riemannian Score-Based (De Bortoli et al., 2022) | -4.92±0.25 | -0.19±0.07 | 0.48±0.17 | -1.33±0.06 |
| *ELBO-based* | | | | |
| Riemannian Diffusion Model (Huang et al., 2022) | -6.61±0.96 | **-0.40**±0.05 | 0.43±0.07 | -1.38±0.05 |
| *Ours* | | | | |
| Riemannian Flow Matching w/ Geodesic | **-7.93**±1.67 | -0.28±0.08 | 0.42±0.05 | **-1.86**±0.11 |

(Chen et al., 2018)

# Riemannian Flow Matching Experimental Results

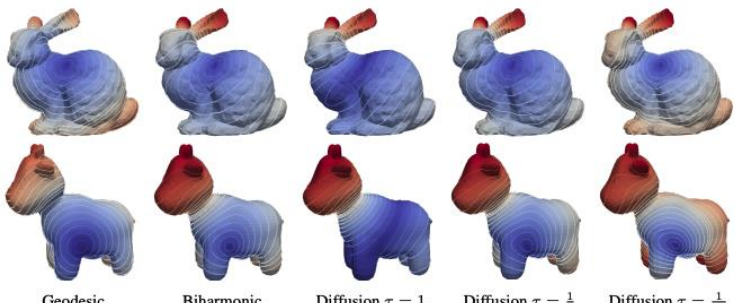



Figure 3: Contour plots of geodesic and spectral distances (to a source point) on general manifolds. Geodesics are expensive to compute online and are globally non-smooth. The biharmonic distance behaves smoothly while the diffusion distance requires careful tuning of the hyperparameter $\tau$.

(Chen et al., 2018)

# Riemannian Flow Matching Results: Biharmonic Distance

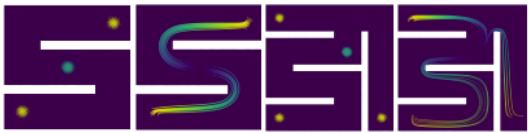
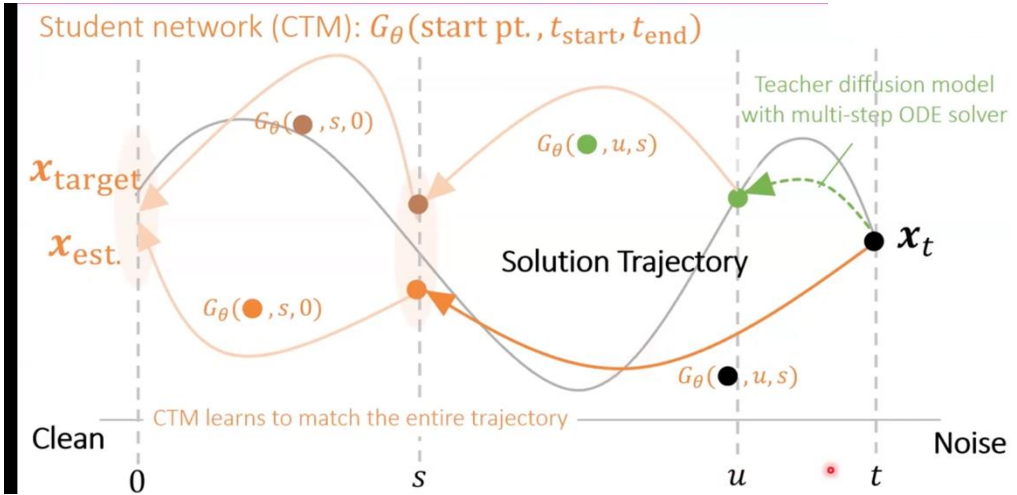



Figure 6: (*a, c*) Source (*cyan*) and target (*yellow*) distributions on a manifold with non-trivial boundaries. (*b, d*) Sample trajectories from a CNF model trained through RCFM with the Biharmonic distance.

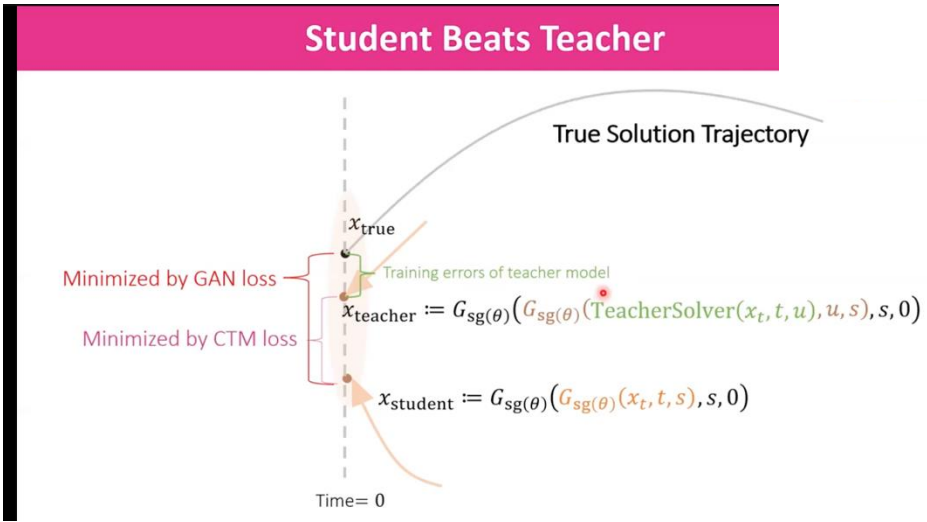
(Chen et al., 2018)

# Consistency Trajectory Models

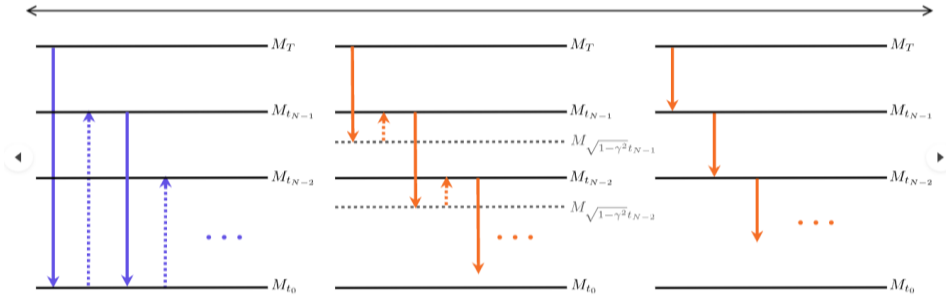
1 Step SOTA Image Generation Quality in Euclidean Space



# Consistency Trajectory Models (Continued)



# Consistency Trajectory Models: Gamma (Hybrid Sampling)




Overview of CTM's γ-sampling.

## Effect of γ-Sampling

CTM enables long ``jumps'' along the solution trajectory — allows our new sampling method, γ-sampling.

# Future Research Direction #2

- Treat the probability density of Iris Images or Face Images as a Riemannian Manifold
- Apply Riemannian Flow Matching (Generally SOTA for Riemannian Generative Models for Stanford Bunny, Spot the Cow, Earth/Climate Science datasets) with a slight modification to improve the suitability for iris images
- Potential Problem: Training Time / GPU Resource Tradeoff

- Possible Improvement: Can the **Consistency Trajectory Model** be extended to **distill the probability flow diffusion ODE path on a Riemannian Manifold in 1 Step?**
  - Would be much faster than using the dopri15 numerical ODE solver, as in RFM
  - Test on common diffusion image datasets that are not iris images

# Ricci Curvature by Treating Images as Riemannian Manifolds

Figure 3: Ricci curvature with geometric weights. Notice the explicit detection of edges. This should be expected, since edge curvature is expected to be of significantly higher value than the value of curvature in homogenous areas.

**Definition 1.** *The Ricci curvature of $I$ along the edge $e_0$ is given by:*

$$\mathrm{Ric}(e_0) = w(e_0)\left[\left(\frac{w(e_0)}{w(c_1)} + \frac{w(e_0)}{w(c_2)}\right) - \left(\frac{\sqrt{w(e_0)w(e_1)}}{w(c_1)} + \frac{\sqrt{w(e_0)w(e_2)}}{w(c_2)}\right)\right].$$

All terms $w$ represent weight functions ([8]). These functions are supposed to reflect geometric entities such as length and area.

Figure 2: Ricci curvature obtained with combinatorial weights.

# Ricci Flow for Image Denoising Task

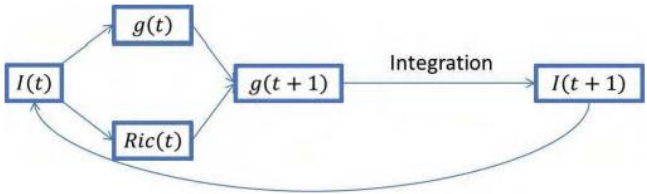



Figure 4: The Ricci flow diagram. Integration means that the newly iterated image is reconstructed from the evolved gradient field.

$$\frac{\partial \mathcal{G}}{\partial t} = -Ric(I)\,,$$

G is the image metric and Ric denotes the Ricci curvature

Figure 6: Ricci flow applied to Lenna image. **Top left:** Original image. **Top right:** Original image with 3.5 db noise. **Middle left:** Image after Beltrami flow. **Middle right:** Image after non-local means.**Bottom:** Image after 3 iterations of the Ricci flow. Note the fair reconstruction of the very noisy image.

# Future Research Direction #3

- Treat the Iris Image itself as a Riemannian Manifold
- Observe the Ricci Curvature for Edge Detection
- If it looks good, then we can test Ricci Flow for Denoising an Iris Image (Same as Generation)

# Potential Impact on Biometrics

Generating synthetic iris or face images can be used for…

1) “Anti-Spoofing”
2) Increased security of iris biometric or face biometric systems!

- Question: How important is sampling speed for generating synthetic iris or face images? (Typically number of NFEs matters for diffusion for image/video generation)

# Acknowledgements


- Big thanks to my collaborator Soohan Kim
- And to my Masters advisors Professor Wooseok Ha and Professor Youngjoon Hong