Lecture 28: Recent topics in generative modeling

CS 5670: Introduction to Computer Vision

Slides on flow matching from Angjoo Kanazawa (with many from Yaron Lipman and Steve Seitz)

Today

- 3D generation (continuing from last time)
- Flow matching
- Video generators as vision problem solvers


Menu
Views
Capture
3D Gaussians
Camera Point view
All interactive sessions are recorded at 1080p with an A6000
Tractor
Metrics
93.11 (10.74 ms)
Playroom

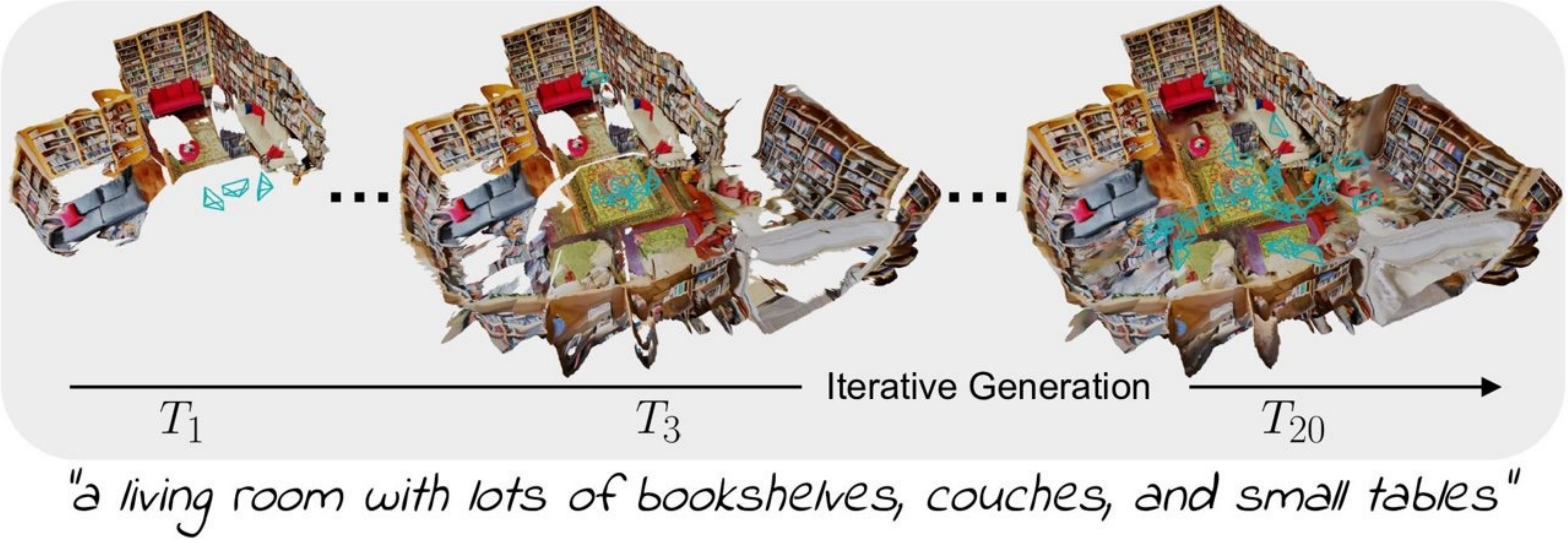

3D Gaussians
Use Old
Hierarchy
Hierarchy Rendering
Aggressive
Show SfM
1.000
Scaling Modifier
0
Limit low
10000000
Limit high
Use ZFar

Text2Room: Extracting Textured 3D Meshes from 2D Text-to-Image Models

Lukas Höllein[1*], Ang Cao[2*], Andrew Owens[2], Justin Johnson[2], Matthias Nießner[1]

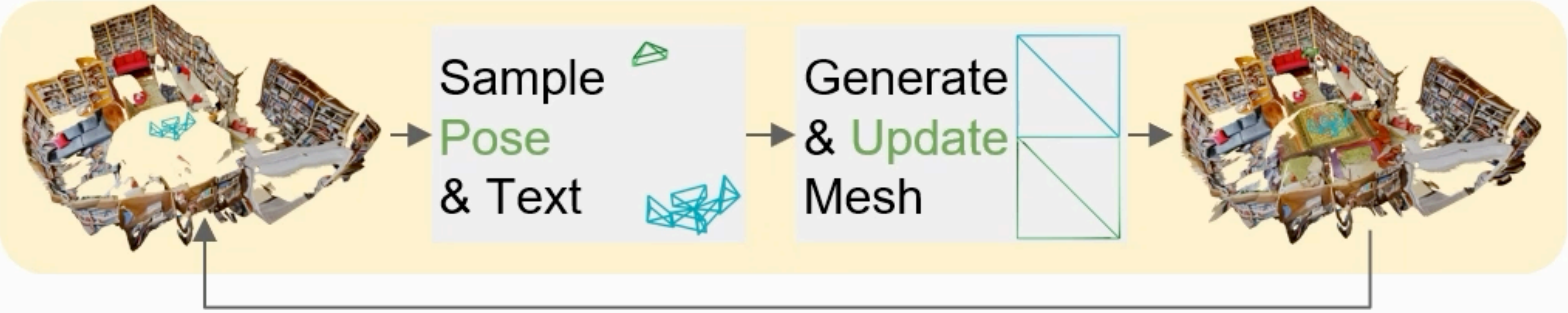
[1]Technical University of Munich, [2]University of Michigan

*joint first authorship



"a living room with a lit furnace, couch, and cozy curtains, bright lamps that make the room look well-lit"

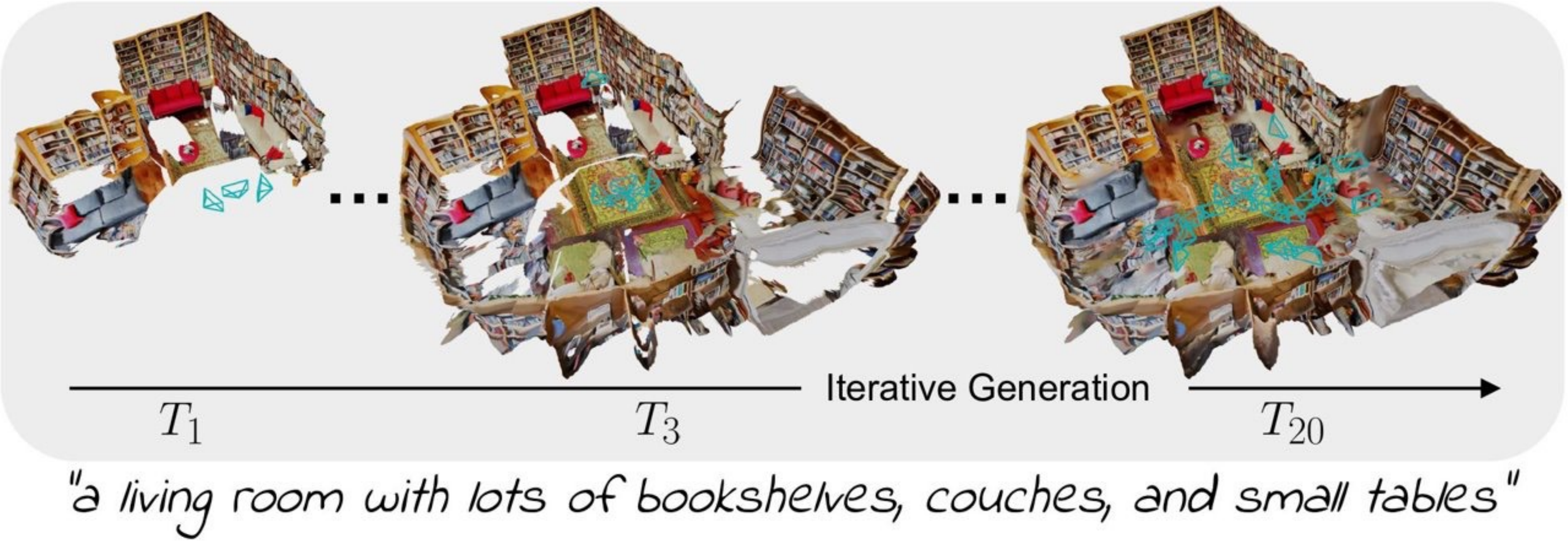
Scene Generation Stage

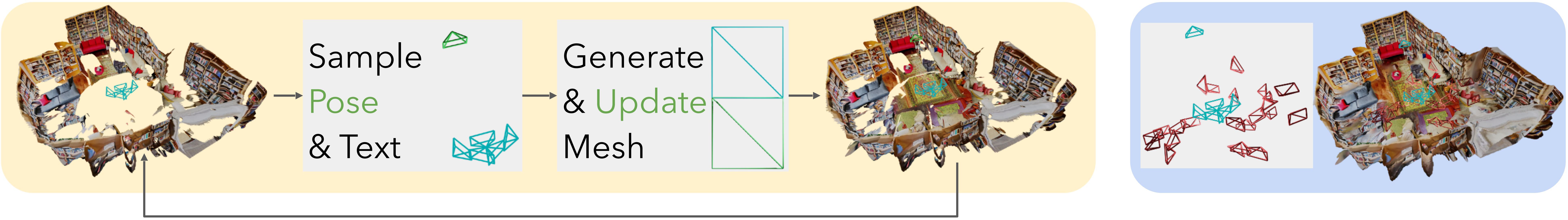


iterative scene generation

generated images

Two-Stage Scene Generation





Scene Generation Stage

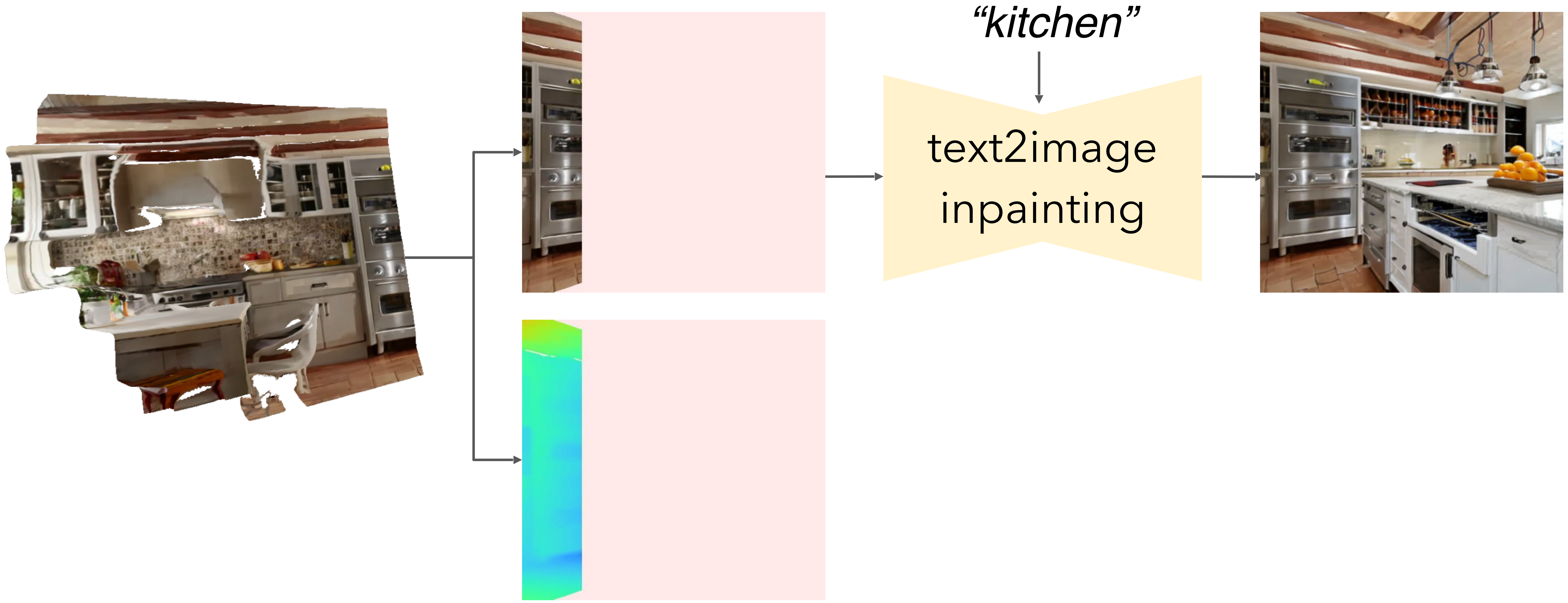
Completion Stage

Completion Stage

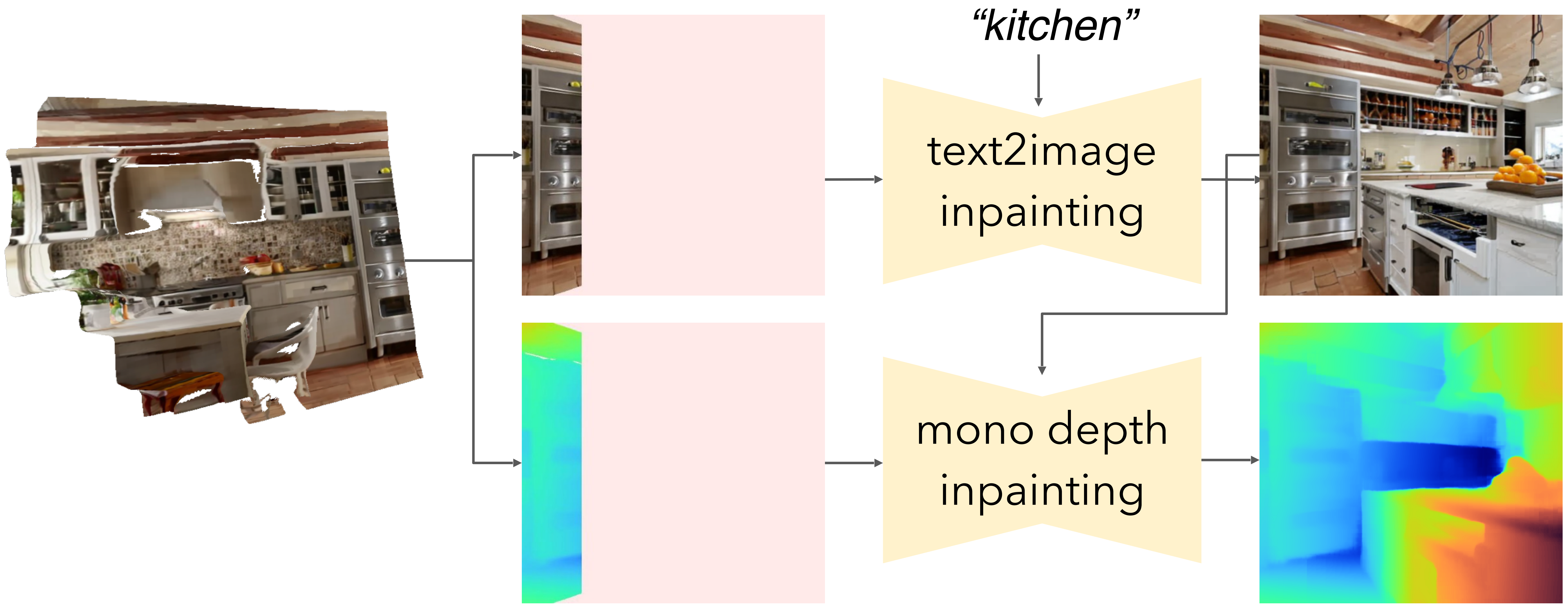
after first stage: mesh contains holes

completion fills-in holes

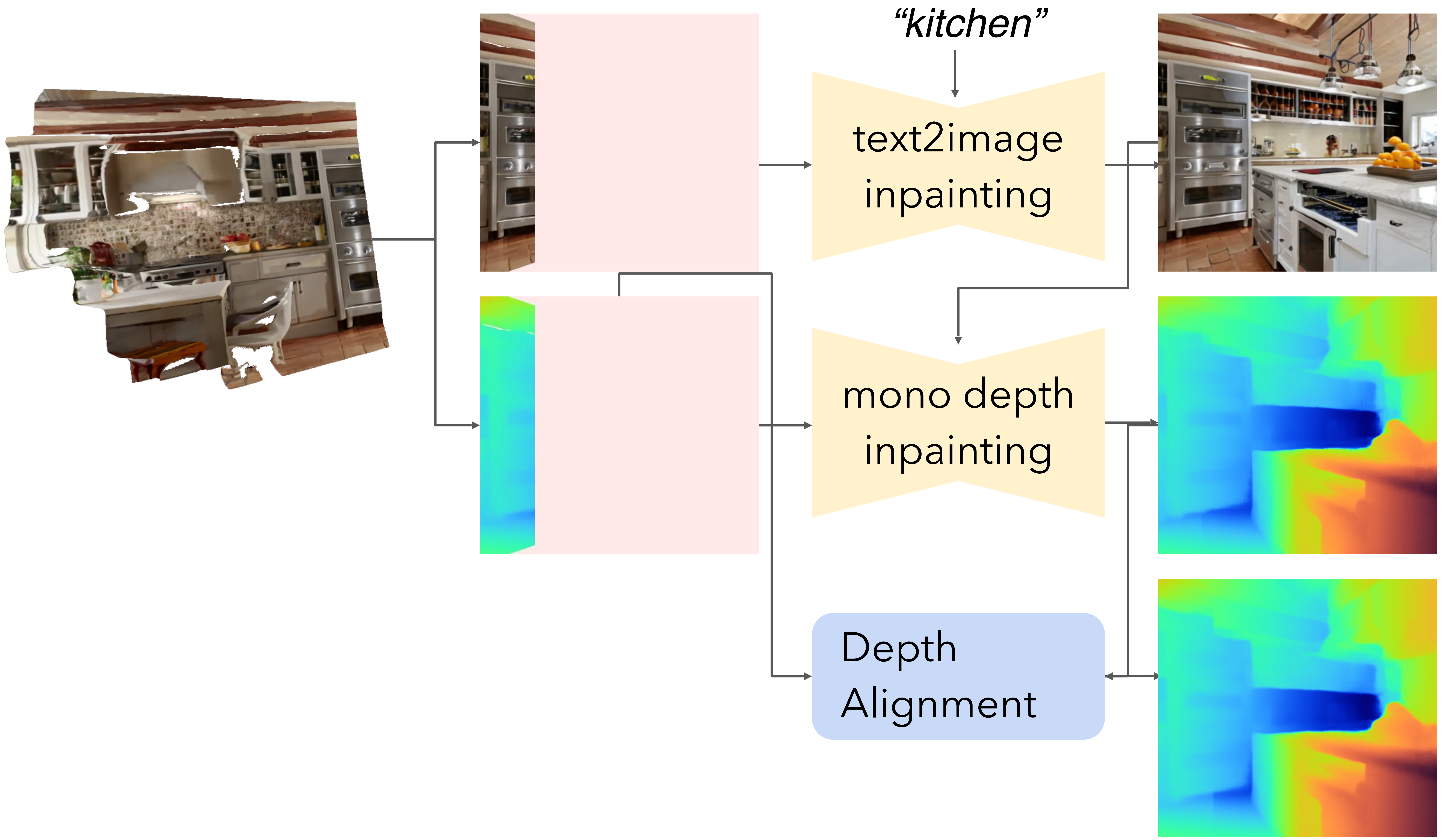
Iterative Scene Generation



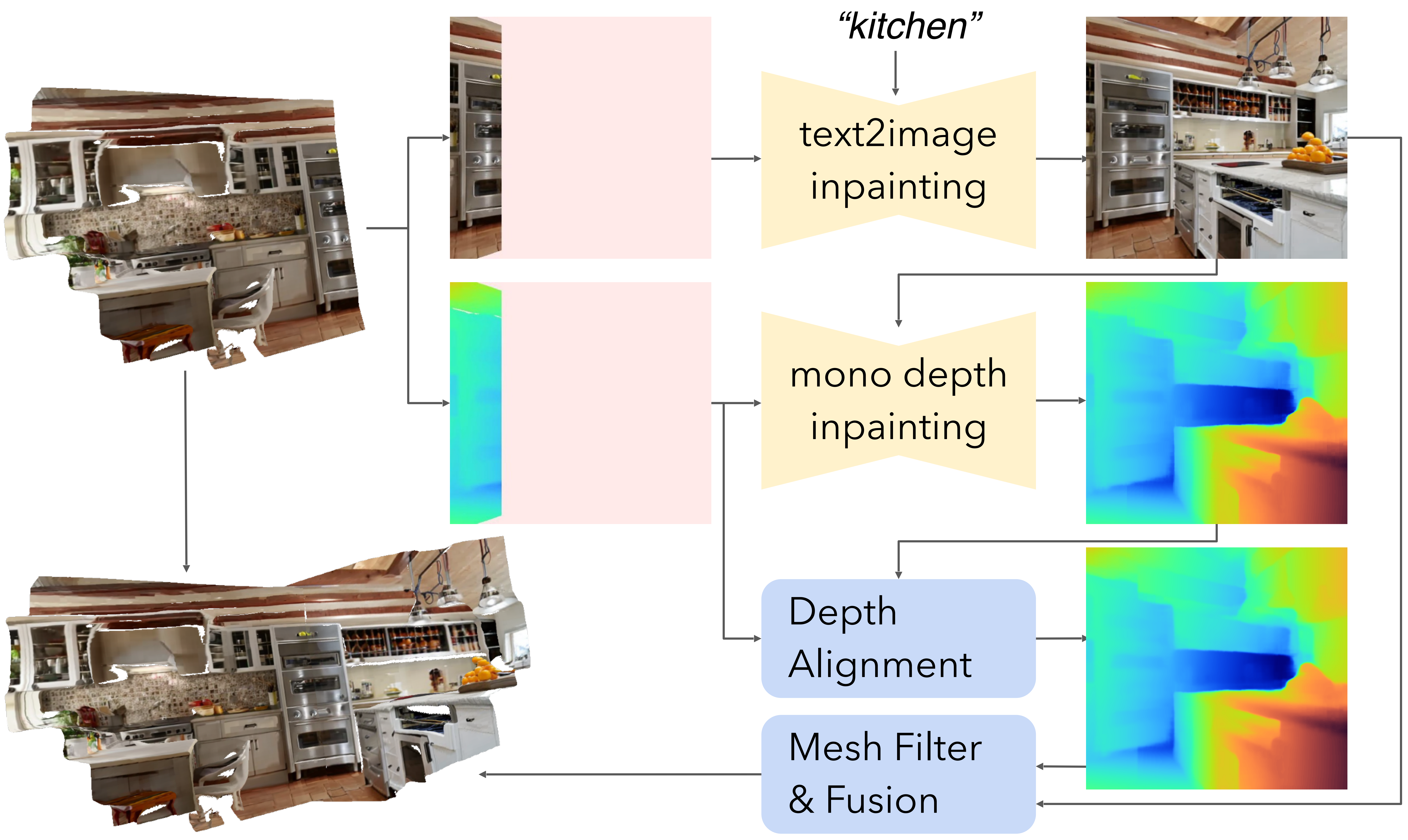
Iterative Scene Generation



Iterative Scene Generation



Iterative Scene Generation



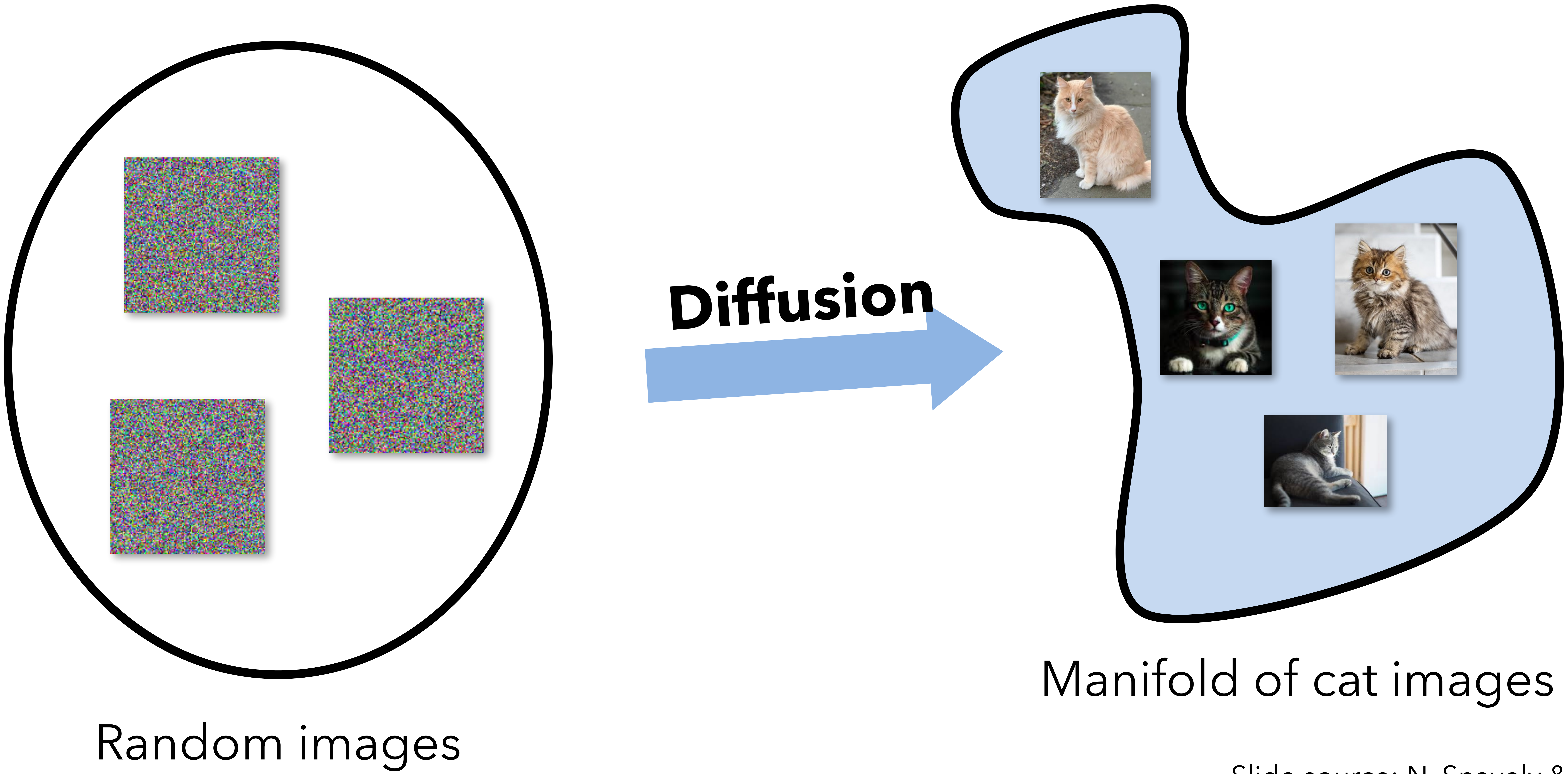
a living room with a lit furnace, couch, and cozy curtains, bright lamps that make the room look well-lit

Editorial Style Photo, Coastal Bathroom, Clawfoot Tub, Seashell, Wicker, Mosaic Tile, Blue and White

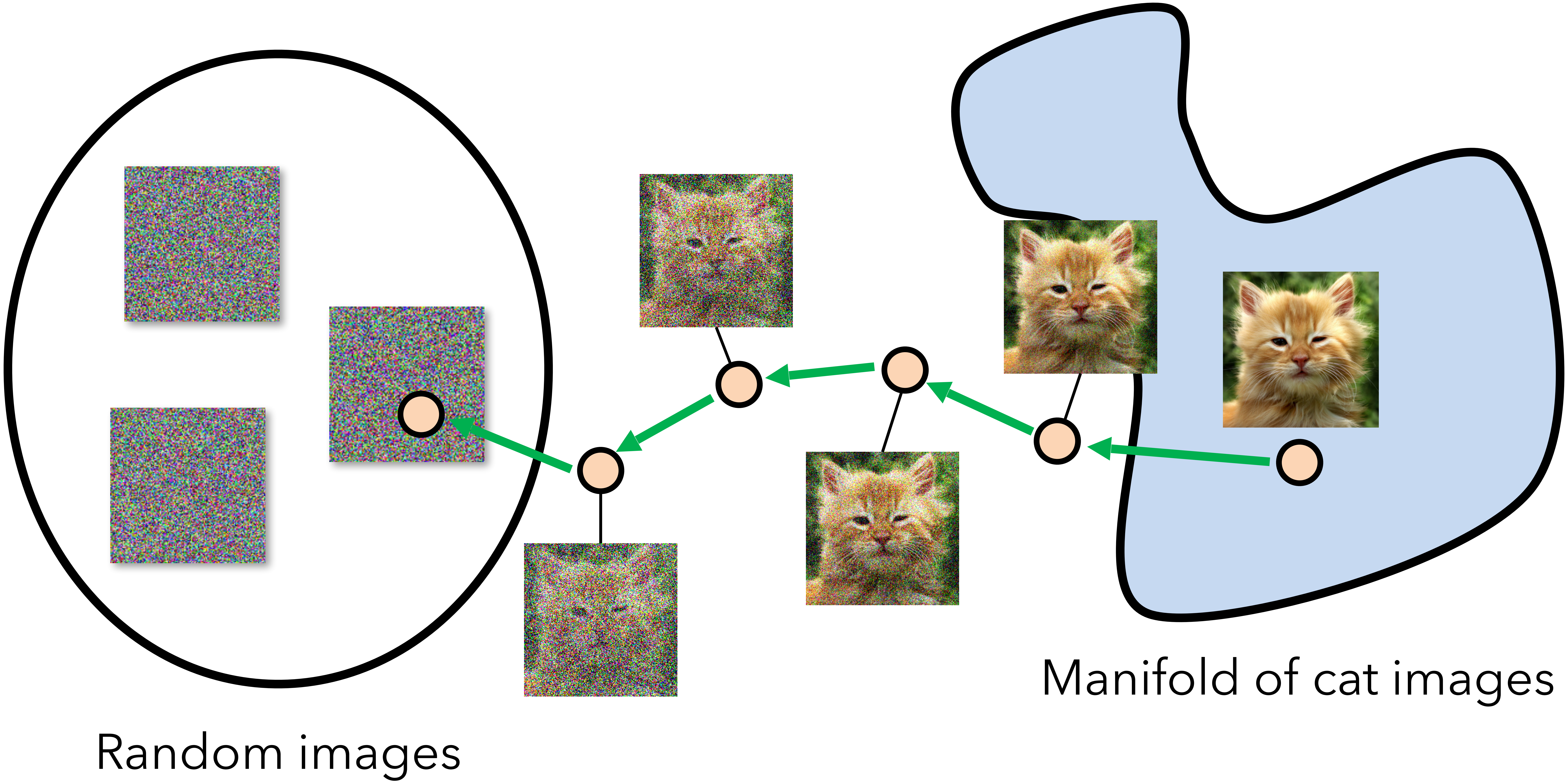
Today

- 3D generation (continuing from last time)
- **Flow matching**
- Video generators as vision problem solvers

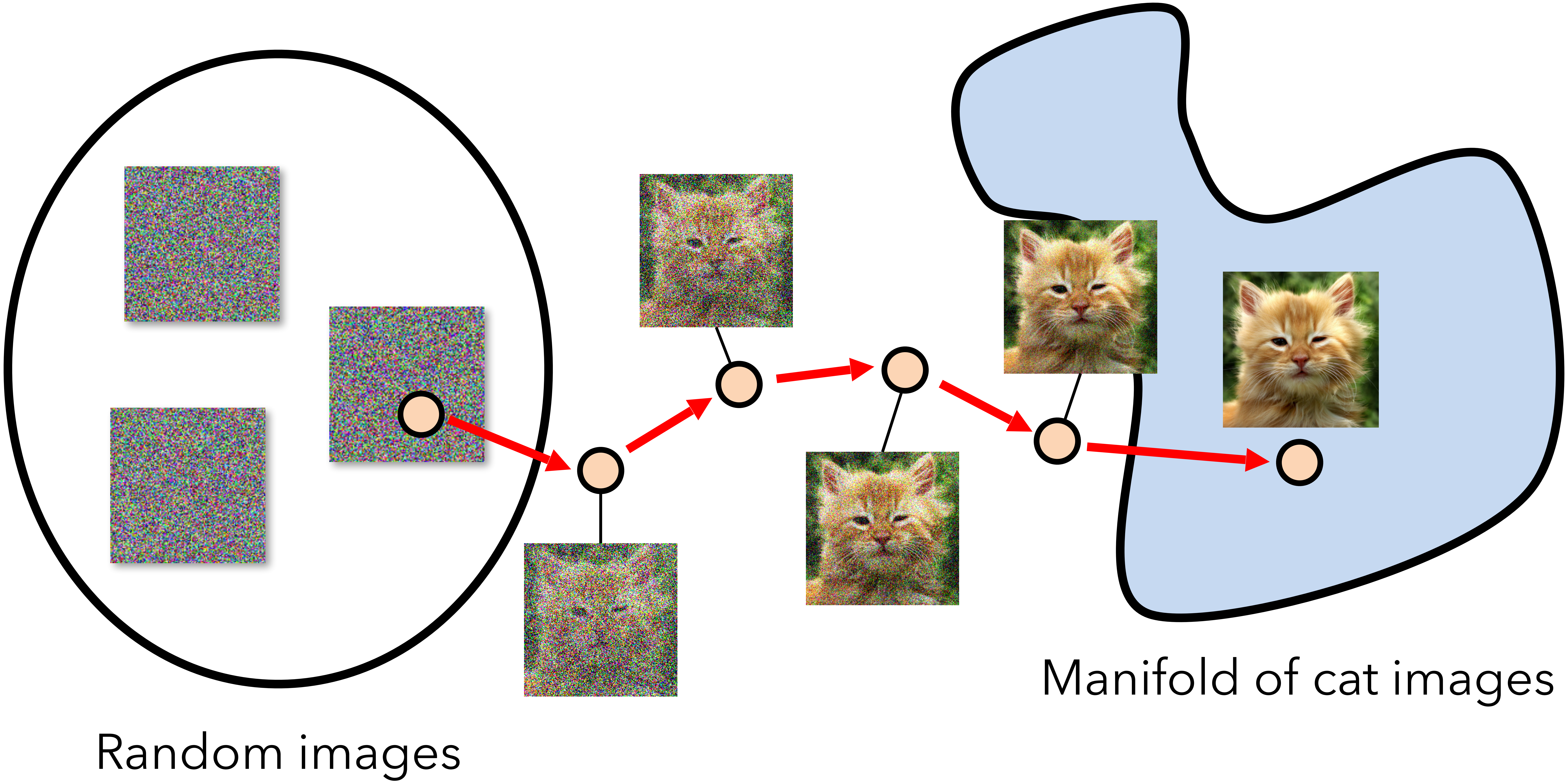
Recall: diffusion models



Slide source: N. Snavely & S. Seitz



Slide source: N. Snavely & S. Seitz



Slide source: N. Snavely & S. Seitz

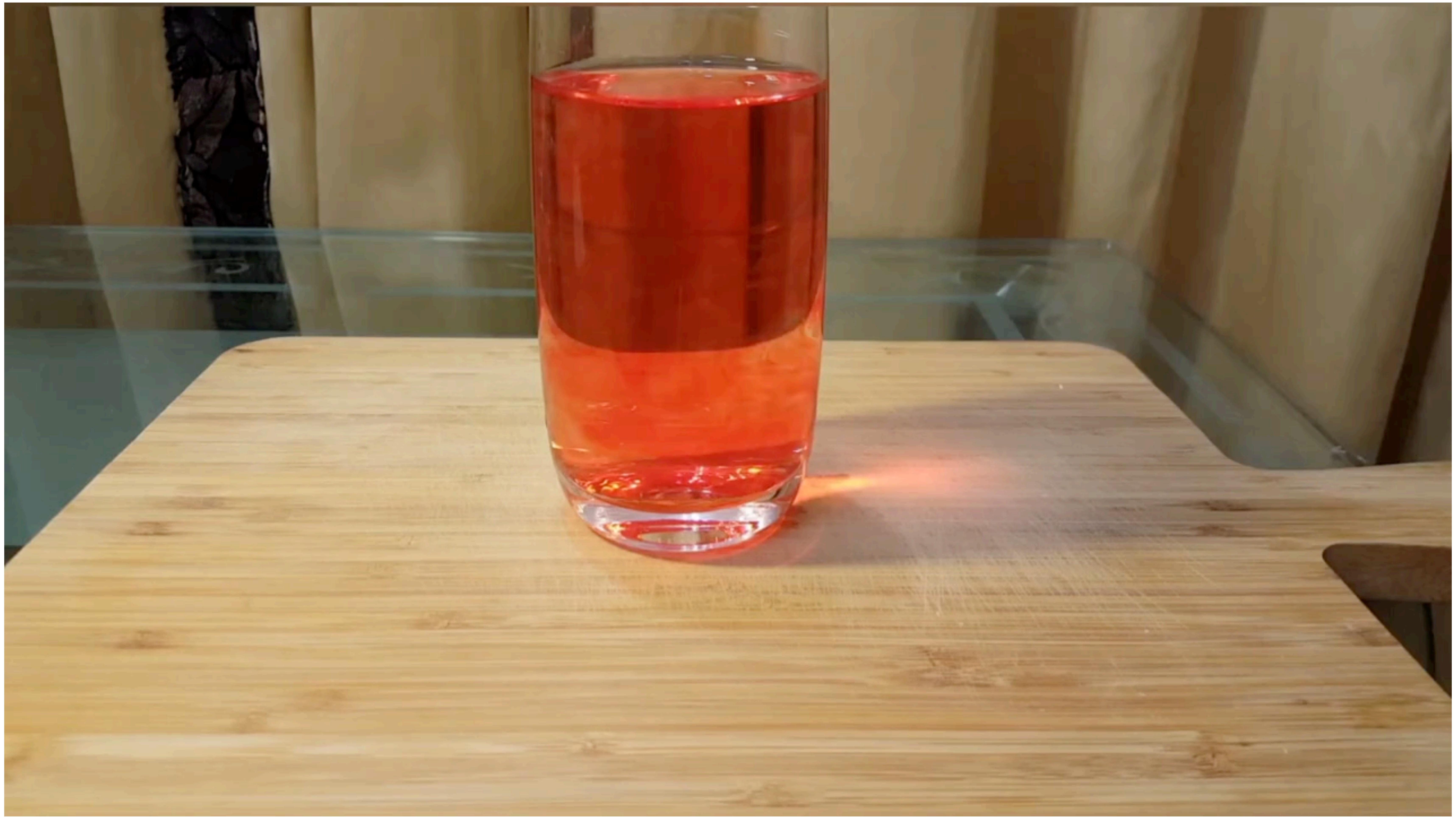
Diffusion: Physics Interpretation

Heat Diffusion

Source: Angjoo Kanazawa

Diffusion: Physics Interpretation

Reversing the process



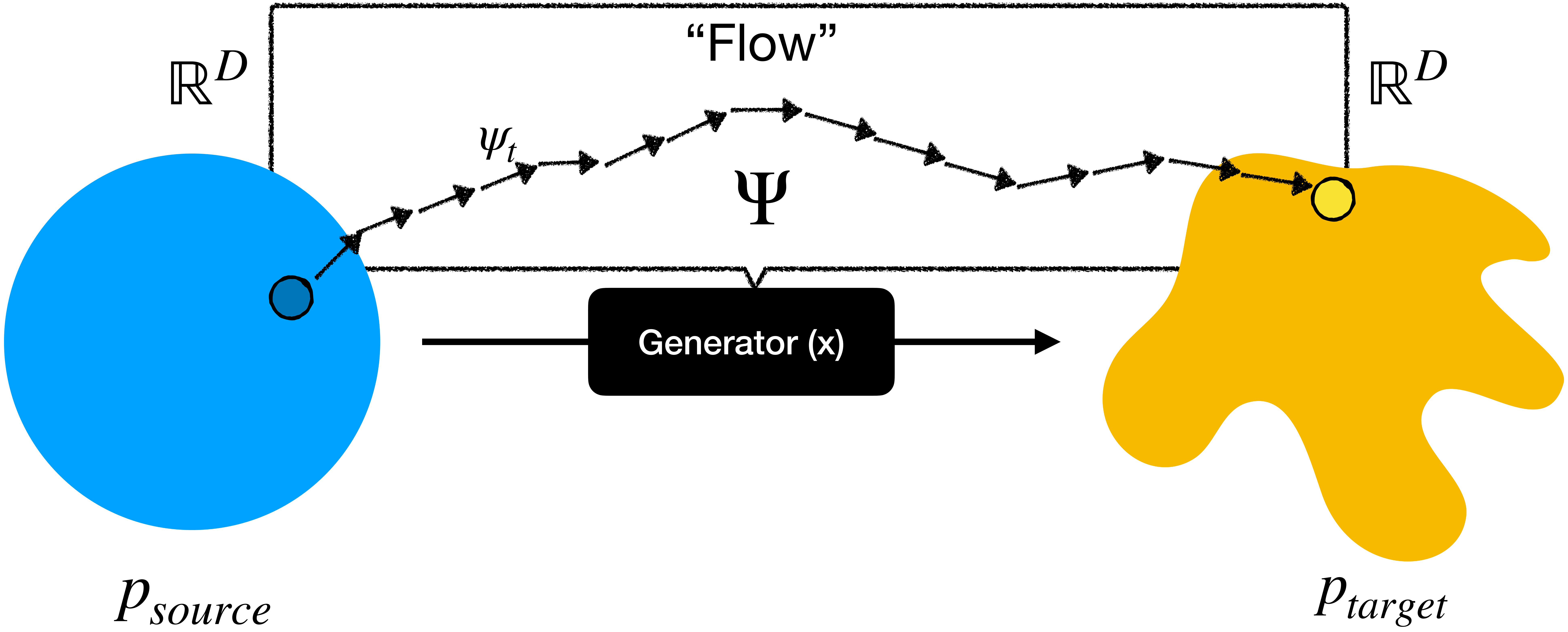
Source: Angjoo Kanazawa

Idea in Song et al. Score-Based Generative Modeling through Stochastic Differential Equations 2021

Flow matching

- Another perspective on generative modeling
- Widely used in state-of-art image and video generators
- Very similar to diffusion models
  - In fact, many popular formulations are equivalent [Gao et al., "Diffusion Meets Flow Matching", 2024].
- Simpler (in some ways)

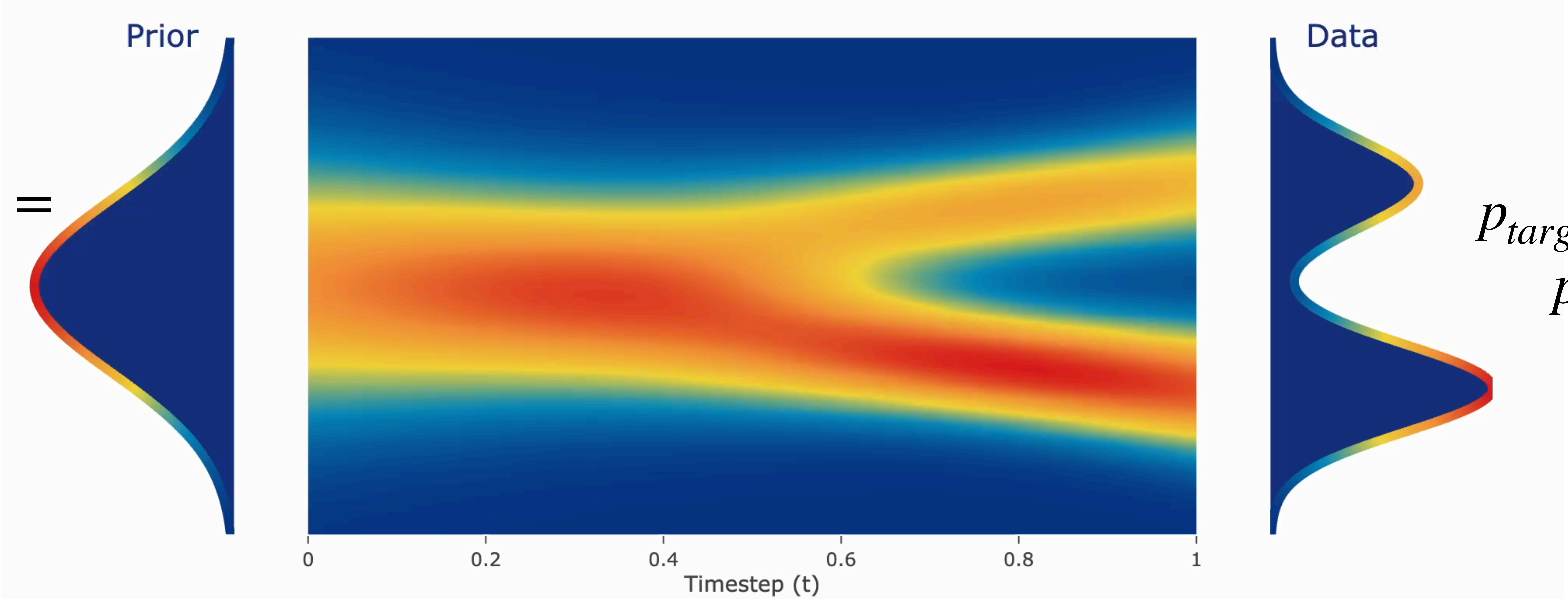
Flow matching models



Adapted from: Angjoo Kanazawa

What is Flow?

$p_{source} = p_0$

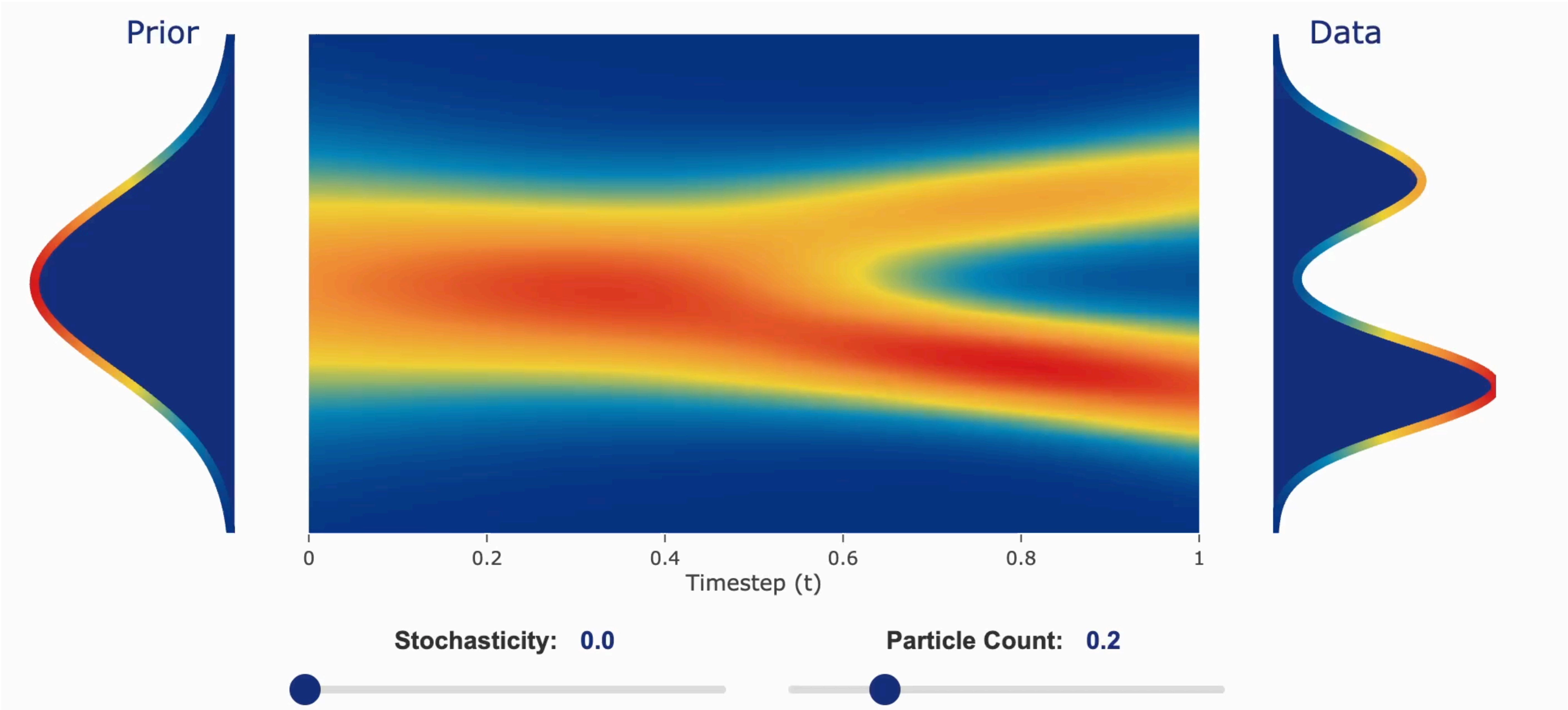


$p_{target} = p_1$

- Sample at $t = 0$ and end at $t = 1$. Yes, this this the reverse of diffusion's notation! :(
- It is a **velocity field.**
- It's like a river with some currents, every point defines how fast you move (velocity)
- You ride this river to go from one distribution to next

Adapted from: Angjoo Kanazawa

Riding the river = Integration



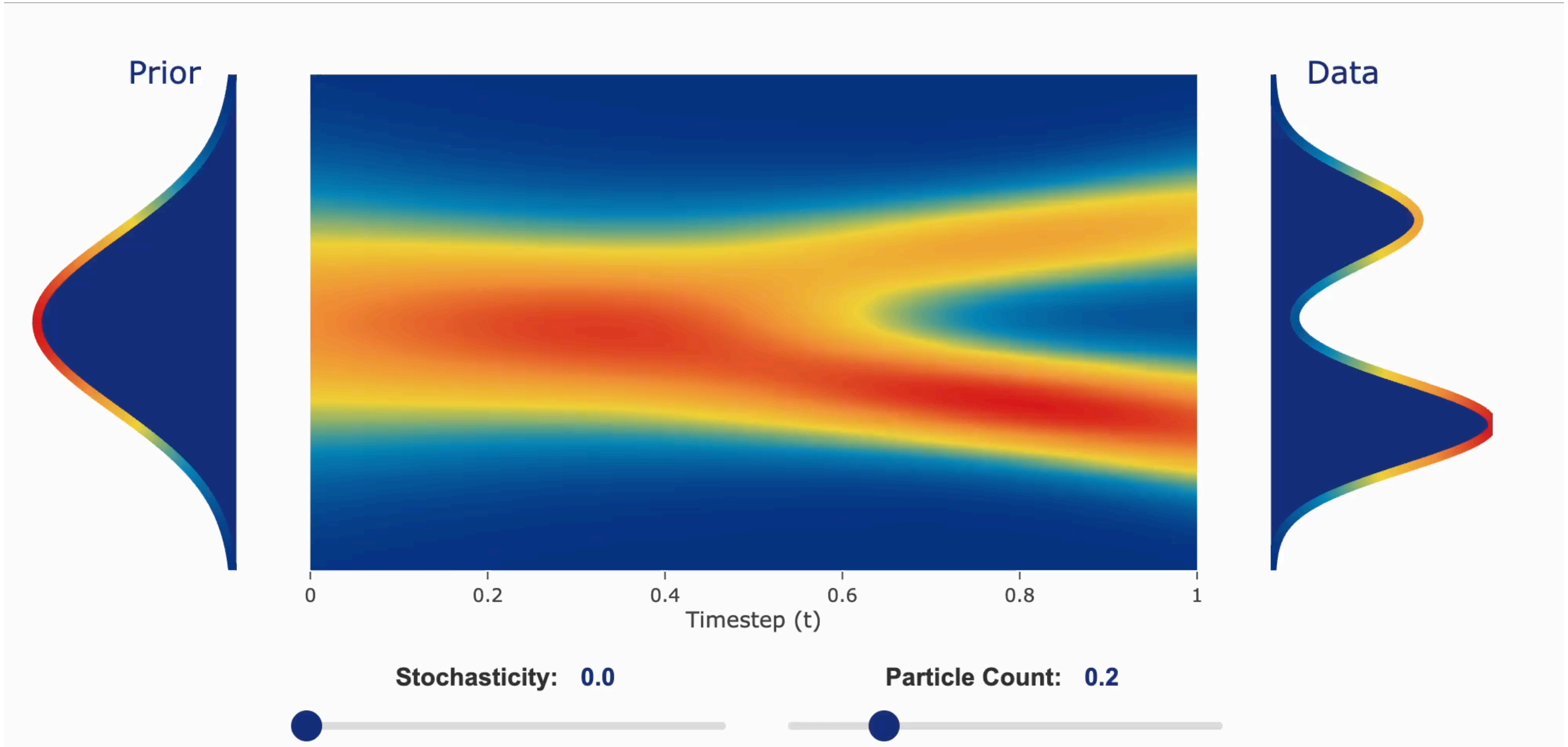
Simplest "Euler Integration":

$$x_{t+\Delta t} = x_t + v_\theta(x_t, t)\Delta t$$

- Riding this river means you add little bits of velocity defined at each location
- This is called "Integration": solving the Ordinary Differential Equation (ODE) with initial state x_0, through some differential parametrized by a network: $\frac{dx}{dt} = v_\theta(x, t)$
- You can add stochasticity when riding it, then it becomes a Stochastic Differential Equation (SDE)

Source: Angjoo Kanazawa

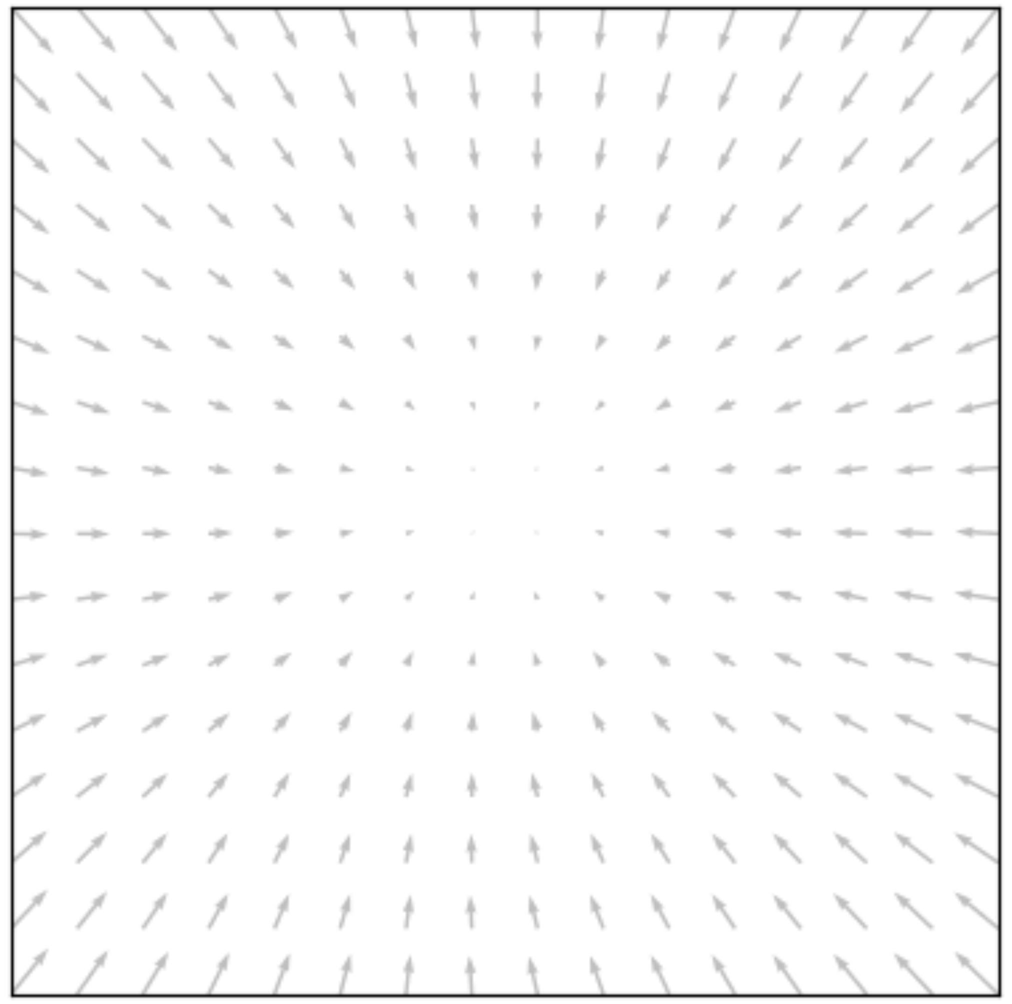
How can we learn this flow?



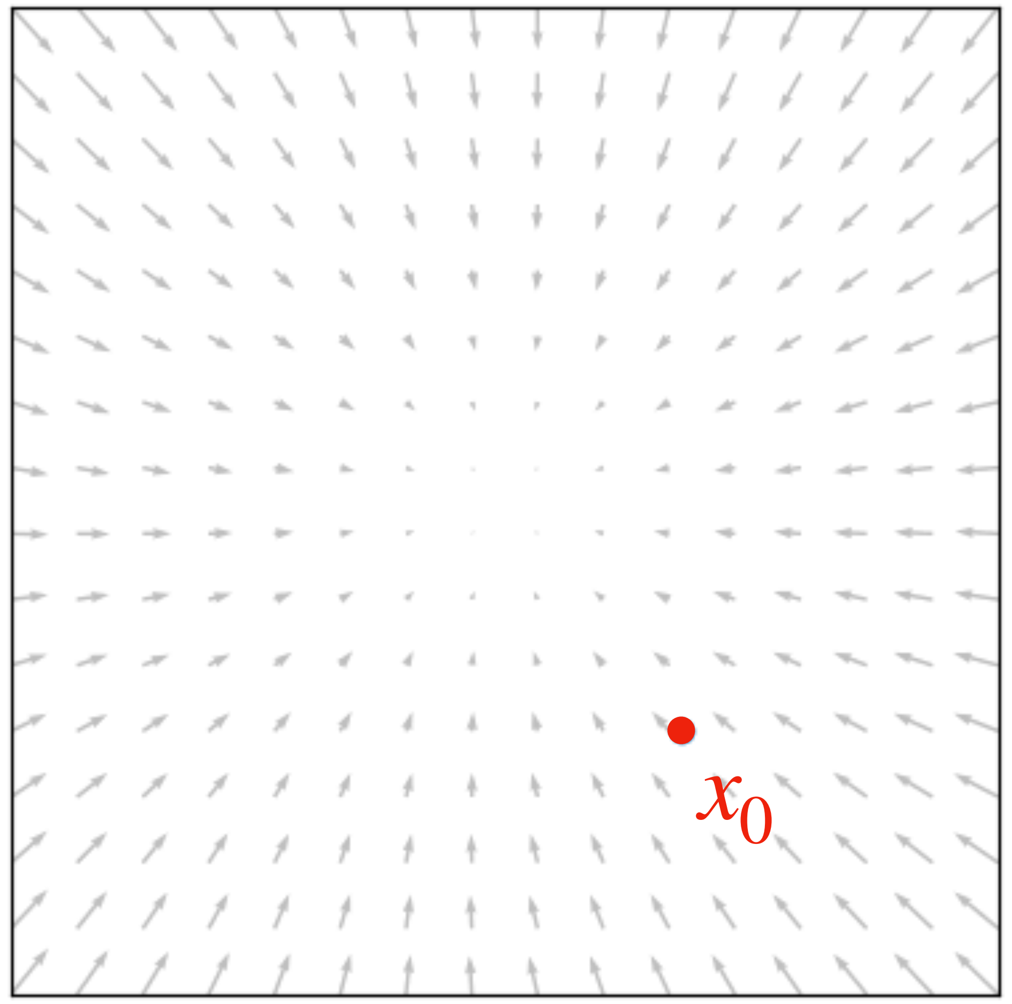
Source: Angjoo Kanazawa

In 2D

$$v_t(x)$$

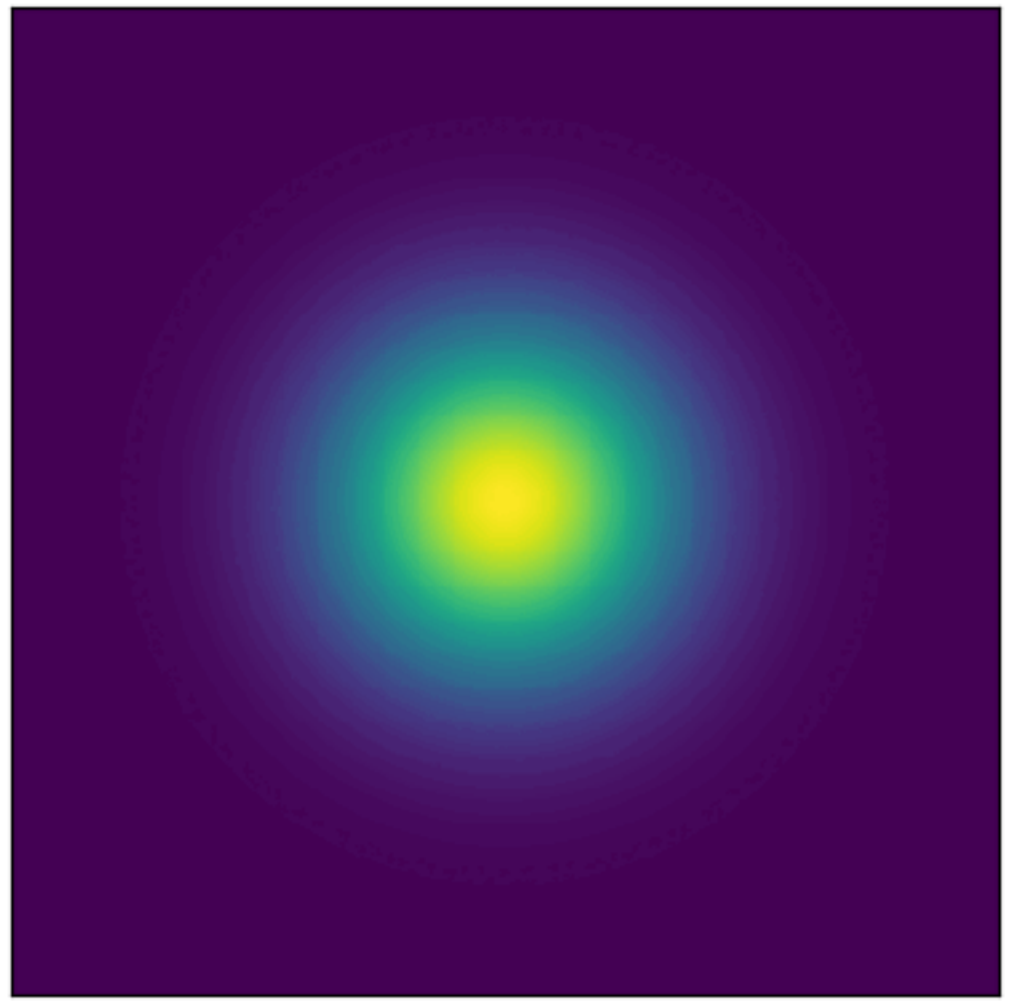


$$x_t = \Psi_t(x_0)$$



$$x_0 \sim p$$

$$x_t \sim p_t$$



Flow ODE

$$\dot{x}_t = v_t(x_t)$$

Slides from Yaron Lipman and Angjoo Kanazawa

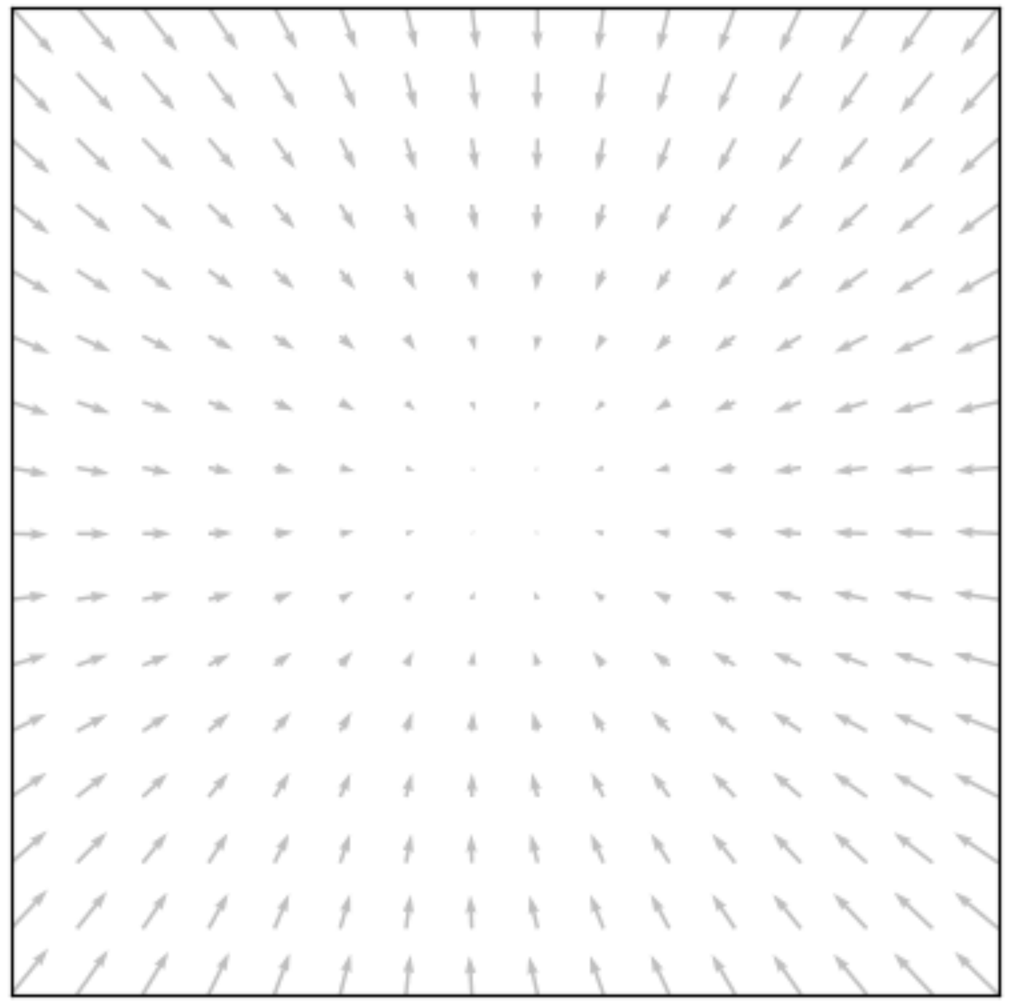
[Chen et al. 2018]

How can we train this?

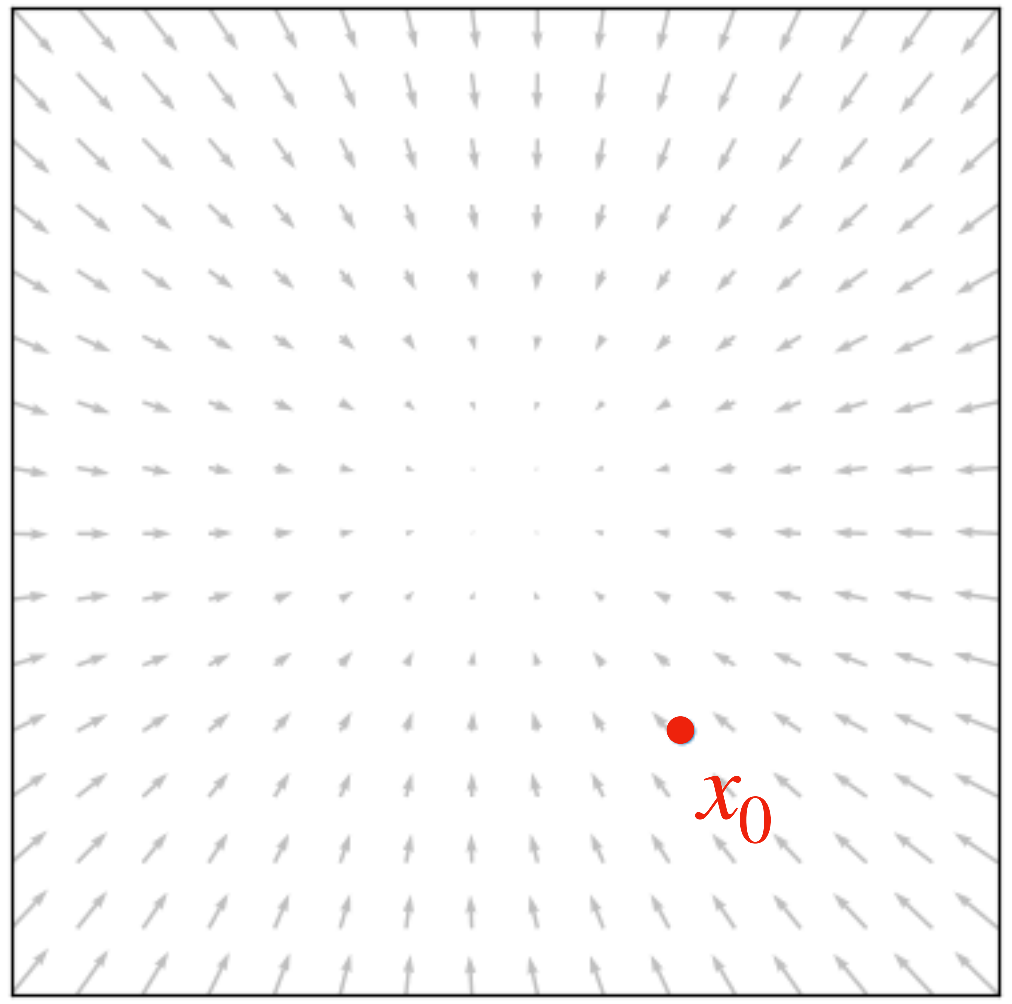
$x_0 \sim p$

Samples of p_1

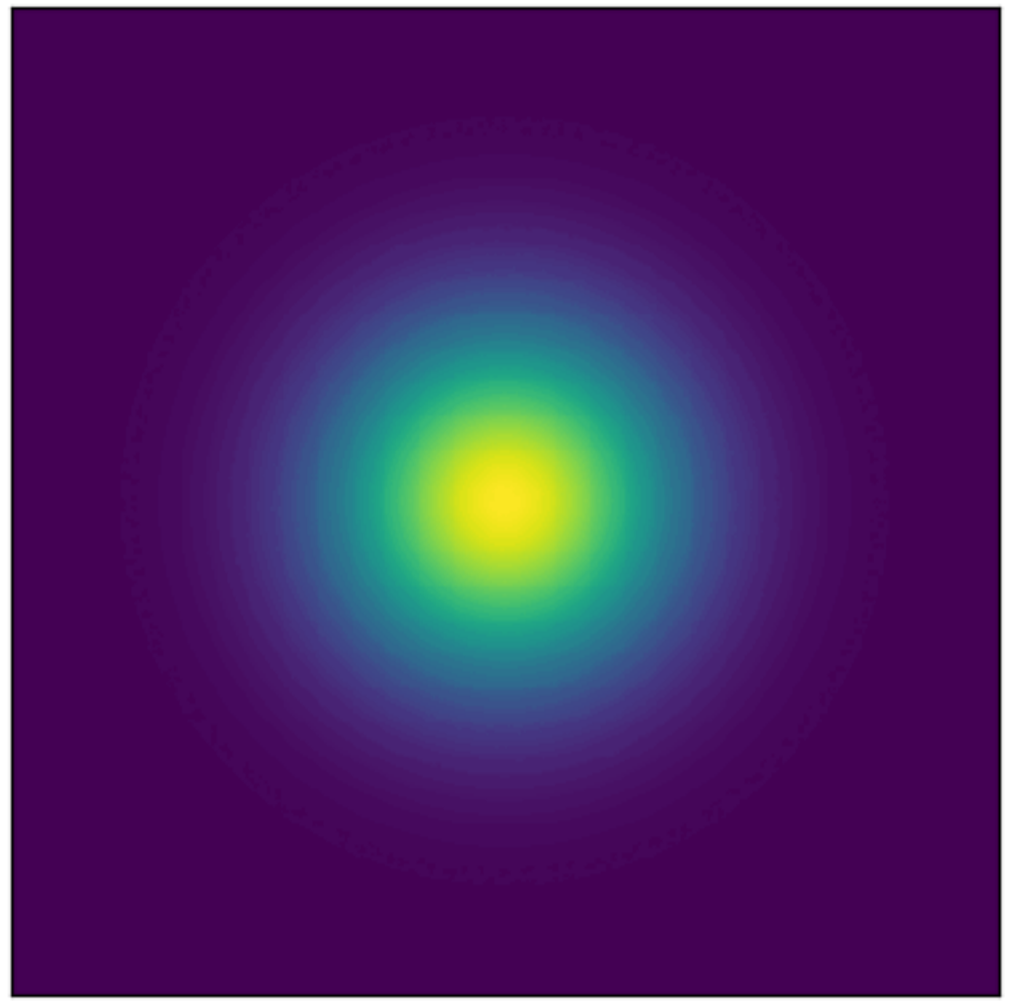
$v_t(x)$



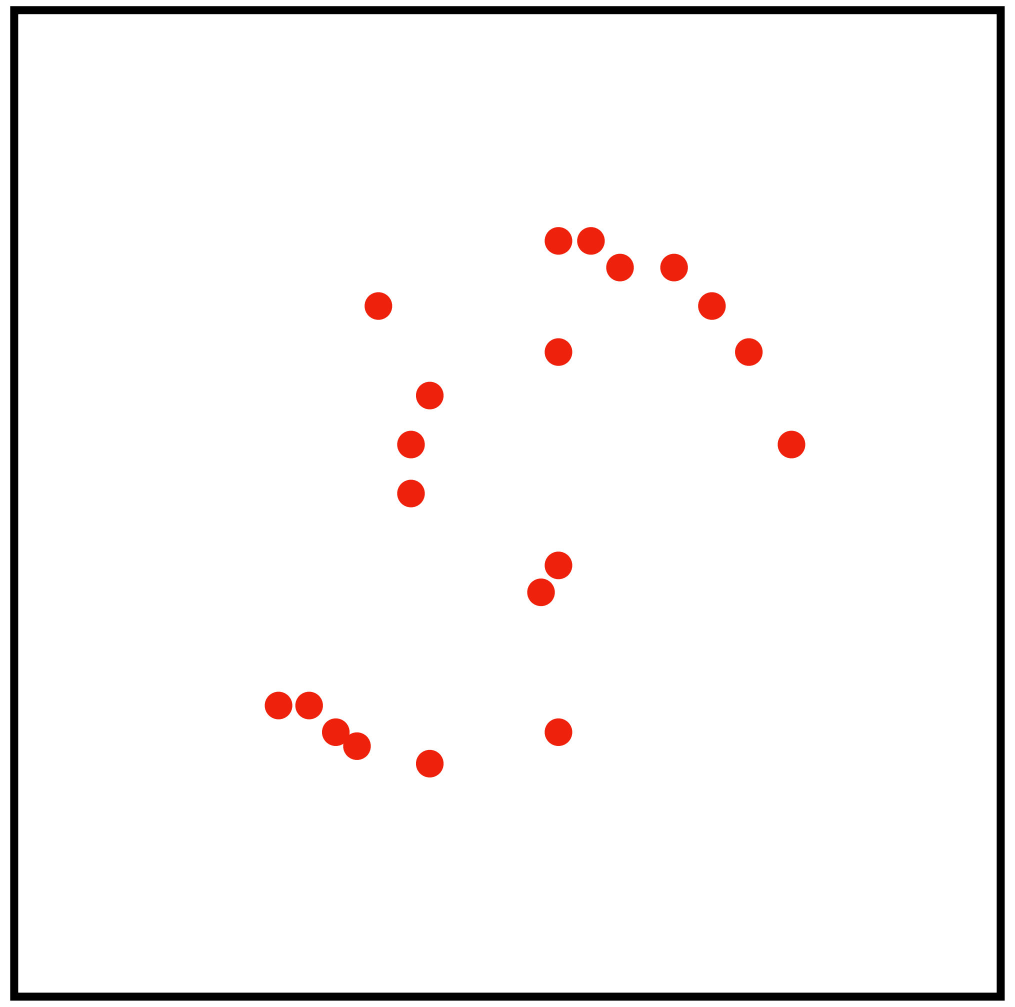
$x_t = \Psi_t(x_0)$



$x_t \sim p_t$



$q(x)$



Flow ODE

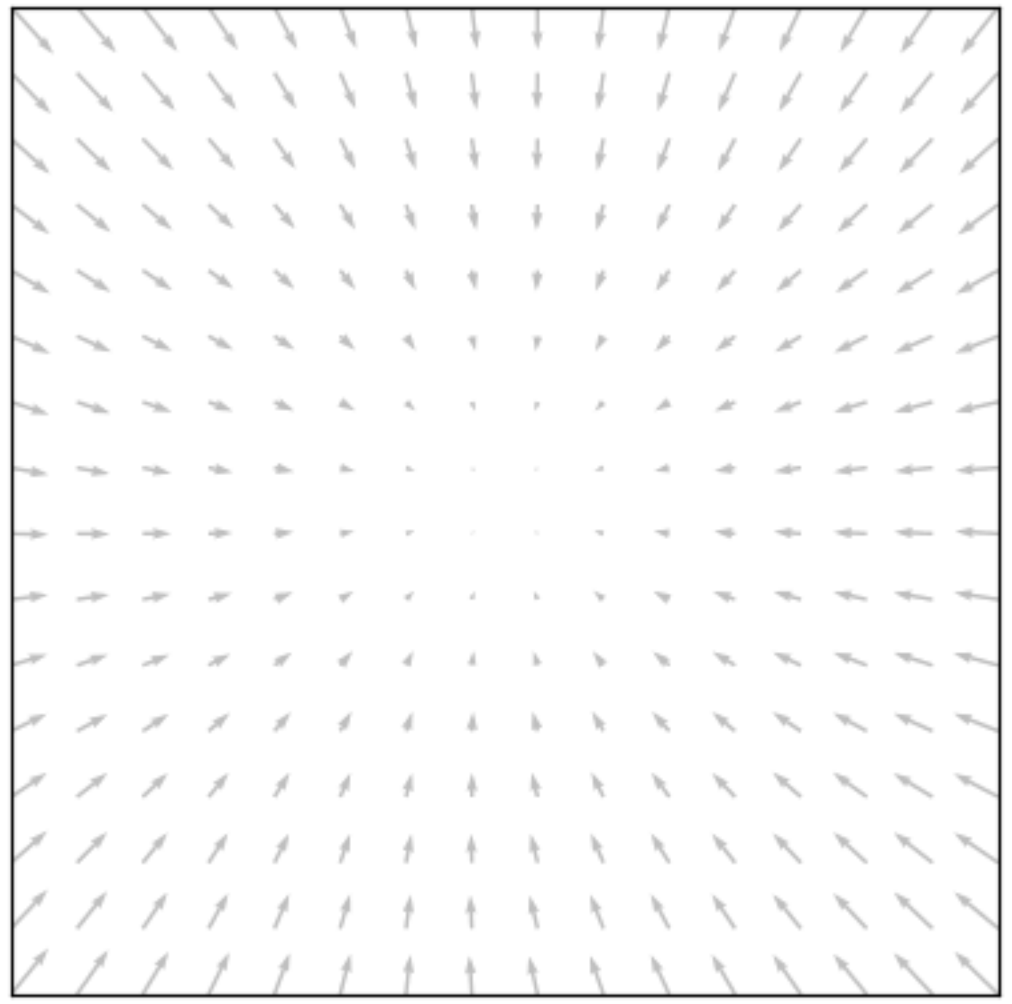
$$\dot{x}_t = v_t(x_t)$$

Slides from Yaron Lipman and Angjoo Kanazawa

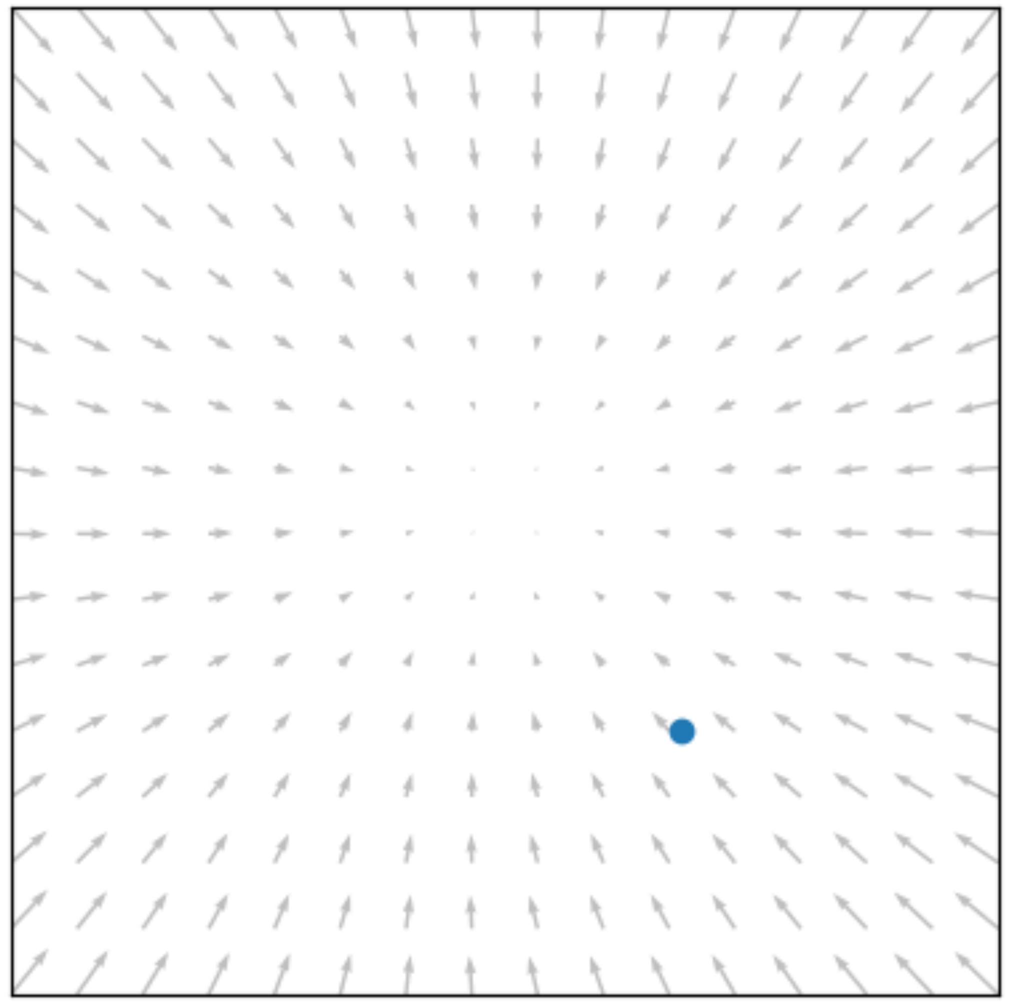
[Chen et al. 2018]

Caveat: Continuity equation

$v_t(x)$



$x_t = \Psi_t(x_0)$



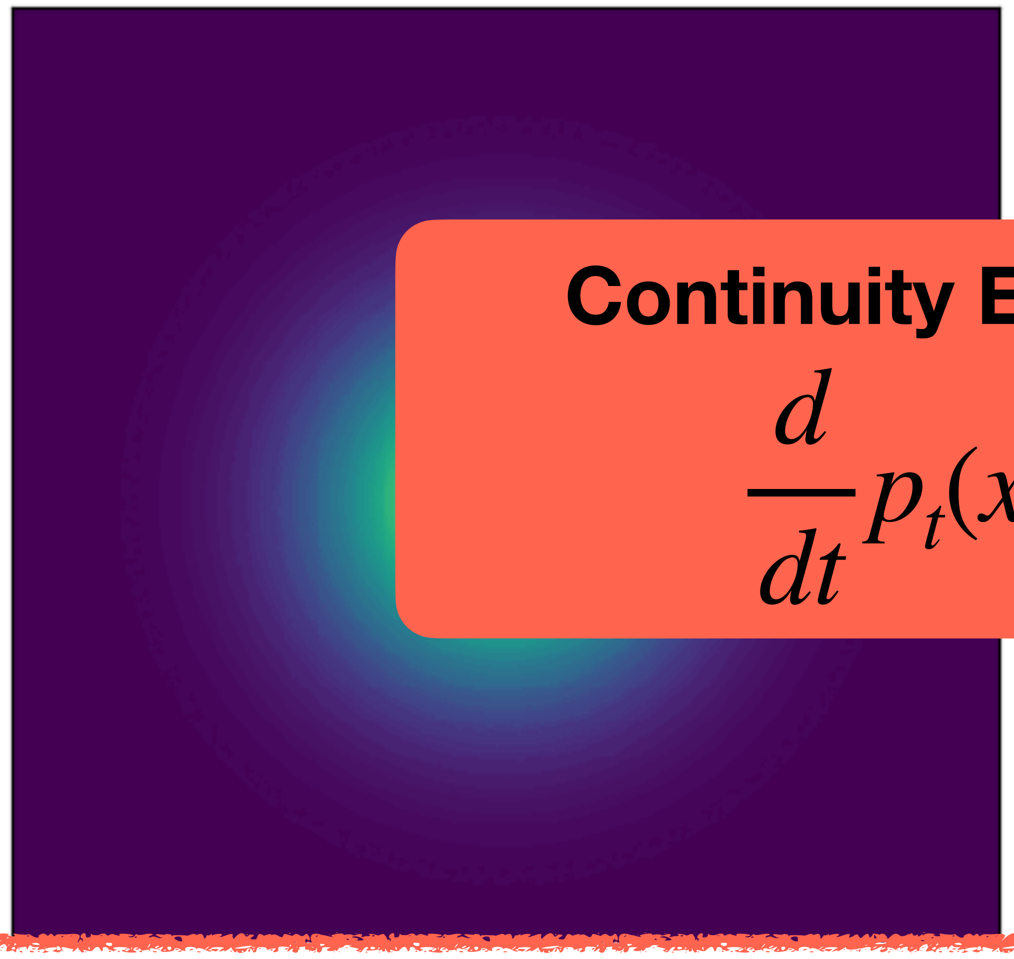
Flow ODE

$$\dot{x}_t = v_t(x_t)$$

$x_0 \sim p$

$\Downarrow$

$x_t \sim p_t$



Continuity Equation PDE (*fixed x*)

$$\frac{d}{dt} p_t(x) = -\operatorname{div}(p_t v_t)$$

where $\operatorname{div} v(x) = \sum_{i=1}^{d} \frac{\partial}{\partial x_i} v_i(x)$

- Need to conserve probability mass
- In the river, analogy you cannot add or remove water
- It has to come from somewhere and go somewhere

Slides from Yaron Lipman and Angjoo Kanazawa

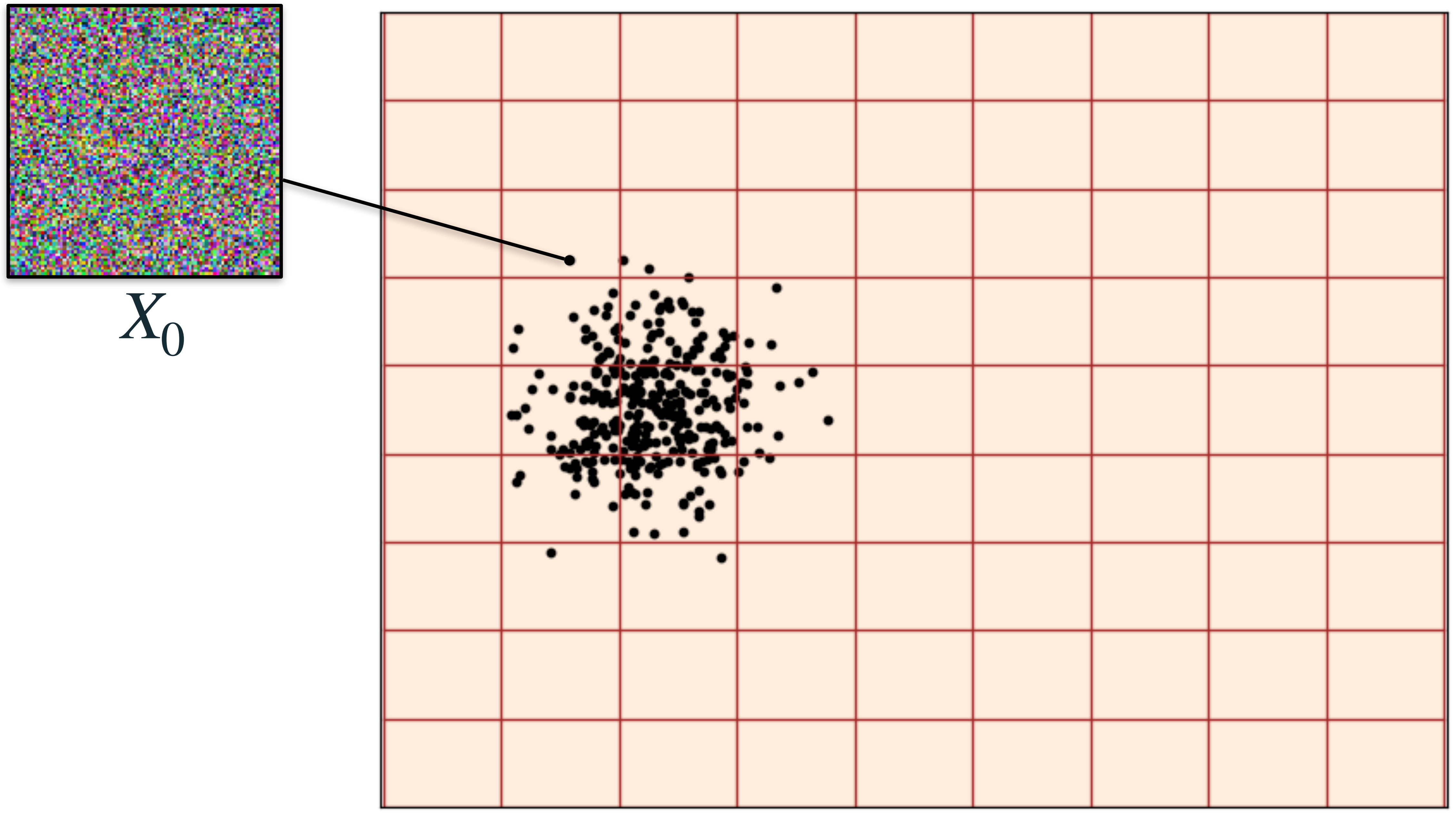
[Chen et al. 2018]

The flow can be though about learning a warping function

$$X_t = \psi_t(X_0)\ , \quad t \in [0,1]$$

Warping

Source $X_0 \sim p$



Slides from Yaron Lipman and Angjoo Kanazawa

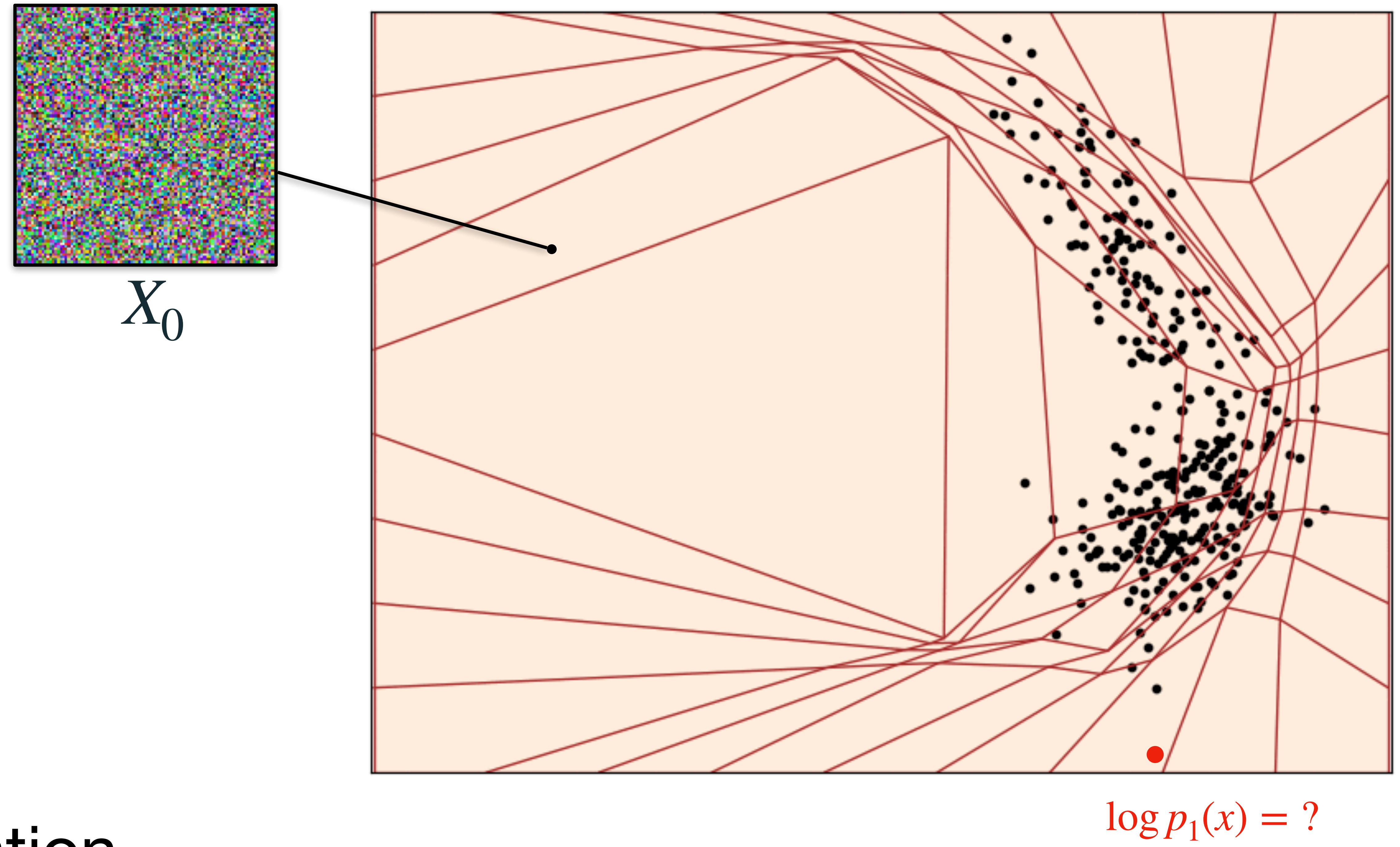
Early work trained flow with Maximum Likelihood

$$D_{KL}(q \,\|\, p_1) = -\mathbb{E}_{x\sim q} \log p_1(x) + c$$

$$X_t = \psi_t(X_0)\ , \quad t \in [0,1]$$

Warping

Source $X_0 \sim p$



- Chaining ψ_t needs to satisfy the continuity equation
- This requires ODE integration *during training* with invertible neural networks

Slide adapted from Yaron Lipman and Angjoo Kanazawa

Early flow learning methods

- Based on normalizing flows (not covered in this class)
- Tries to directly deal with this continuity equation constraint
- Very slow to train (need to integrate while training)
- Other constraints like invertibility of ψ_t
- Nice idea with promising results but limited capability + not practical to train

Source: Angjoo Kanazawa

Instead, model flow with velocity

$$\frac{\mathrm{d}}{\mathrm{d}t}\psi_t(x) = u_t(\psi_t(x))$$

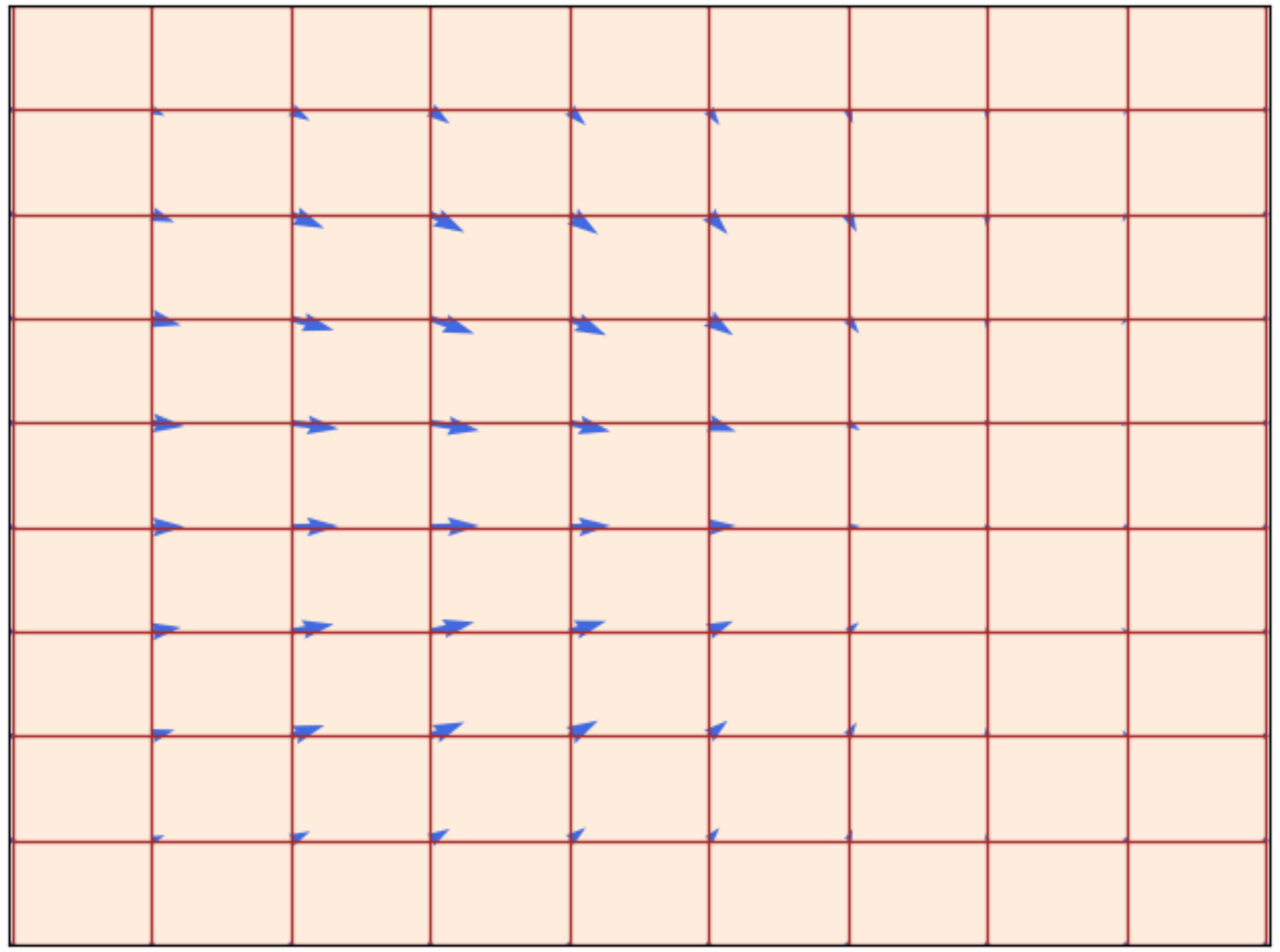
Flow

$$\psi_t(x)$$

Solve ODE ↑ ↓ **Differentiate**

$$u_t(x)$$

Velocity



Slides from Yaron Lipman

Source: Angjoo Kanazawa

Flow Matching [Lipman et al. '22]

- Directly learn the velocity field!
- Don't have to worry bout the continuity equation because velocity fields can't add / subtract mass. You only re-distribute
- Continuity equation satisfied by construction.
- Basically just learn velocity fields for each data sample (conditional velocity field), all will be fine.

Source: Angjoo Kanazawa

Diffusion vs. Flow matching

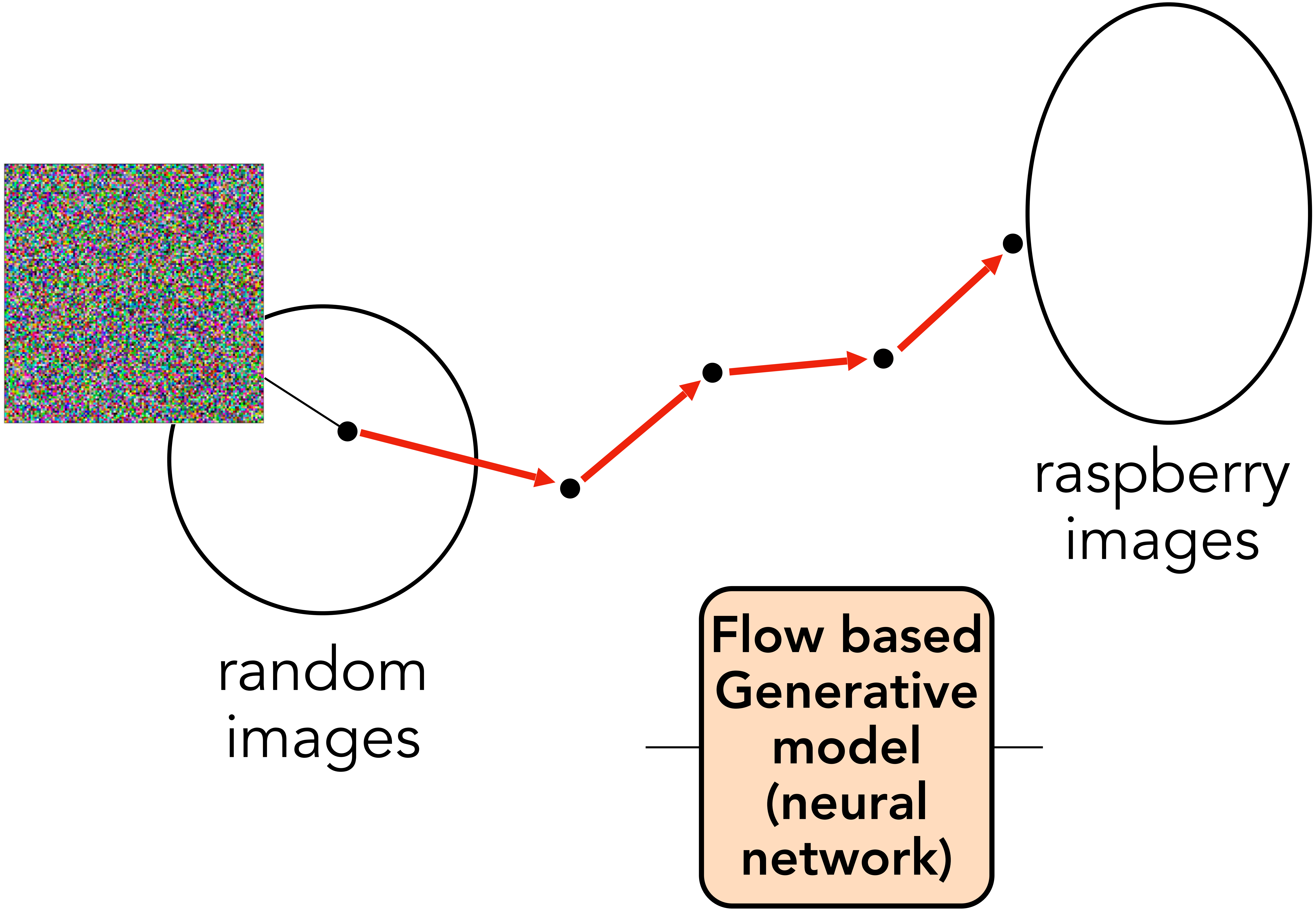
- They both end up learning flow, but diffusion poses the problem as learning a specific noising process and learning to denoise it.
- Diffusion: spread out like heat diffusion, learn how to undo it
- Flow matching: More general, just directly learn a velocity field from one to another

Source: Angjoo Kanazawa

How do you train the flow?

Adapted from Angjoo Kanazawa

Quick Recap:

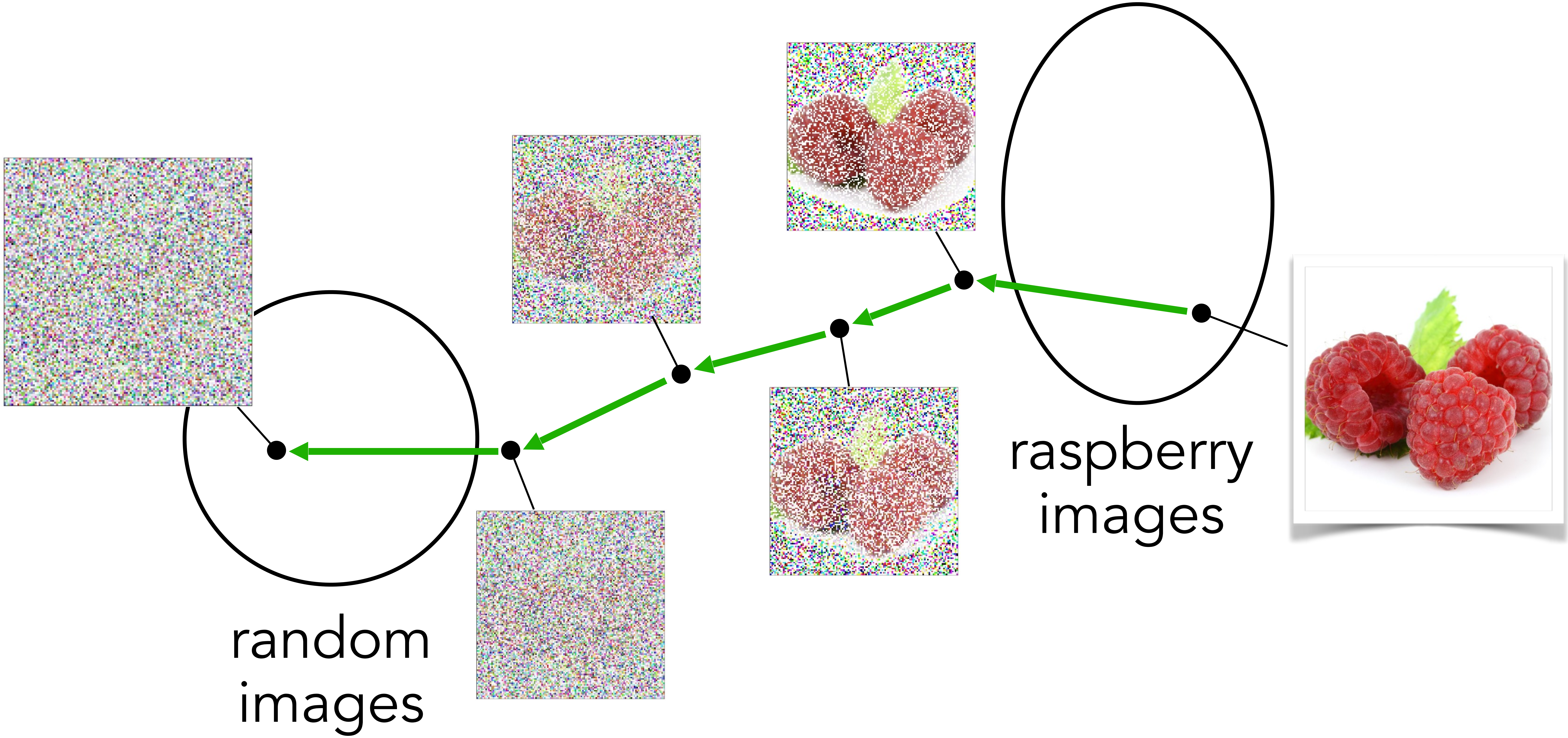


Source: Angjoo Kanazawa

slide from Steve Seitz's video

Training

1. Take real data and corrupt it

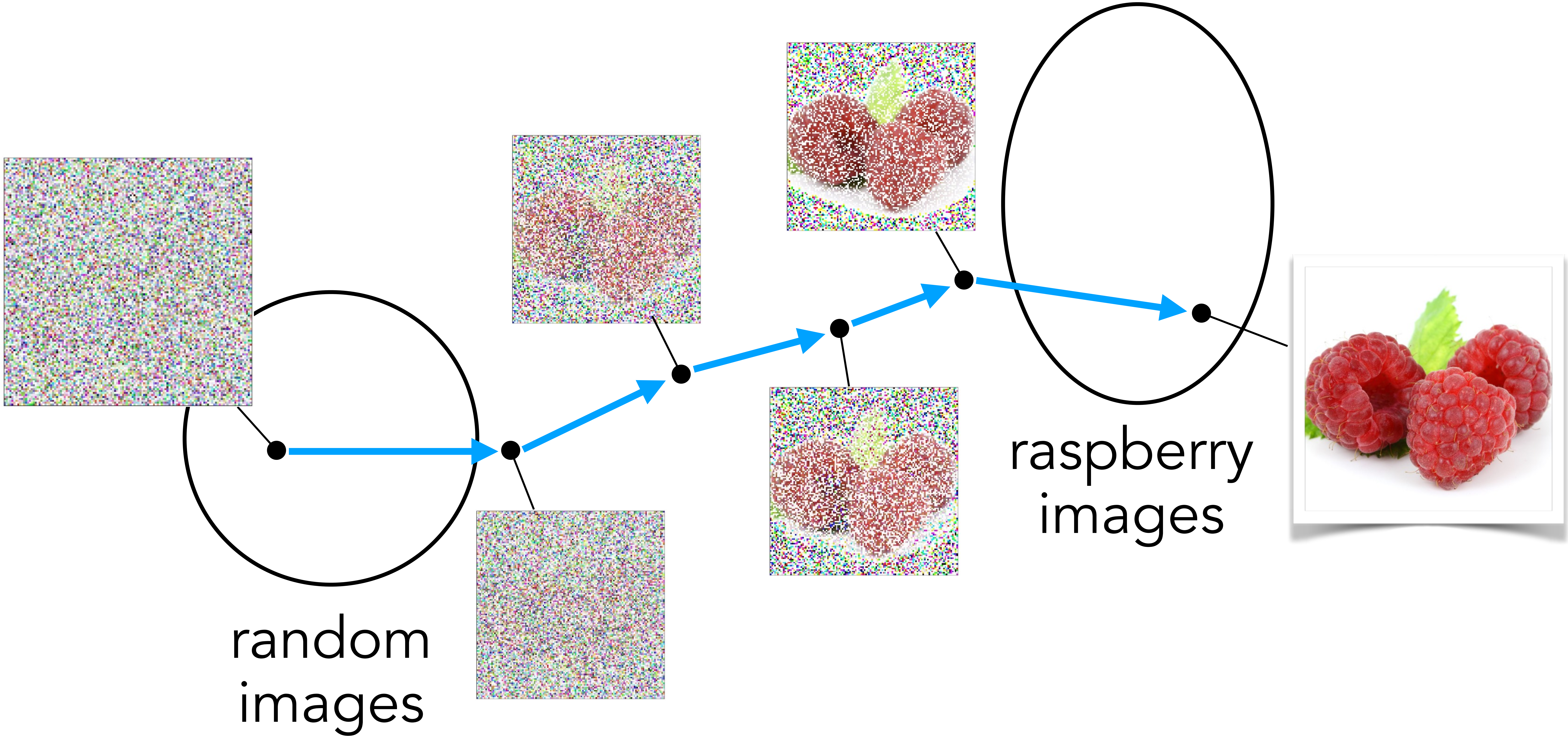


Source: Angjoo Kanazawa

Figure from Steve Seitz's [video]

Training

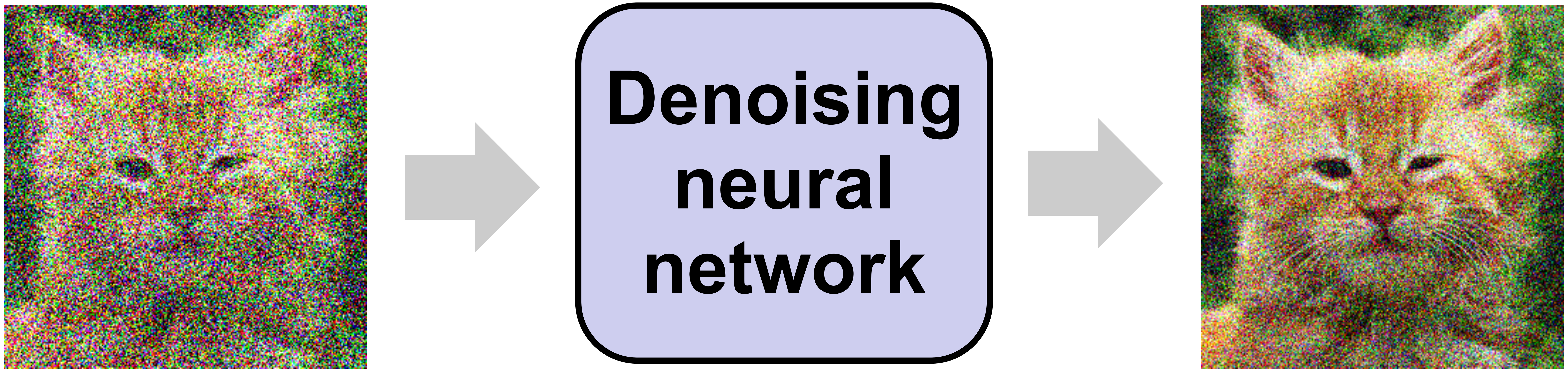
1. Take real data and corrupt it
2. Learn to undo the process!



Source: Angjoo Kanazawa

Figure from Steve Seitz's video

Denoising with a neural network



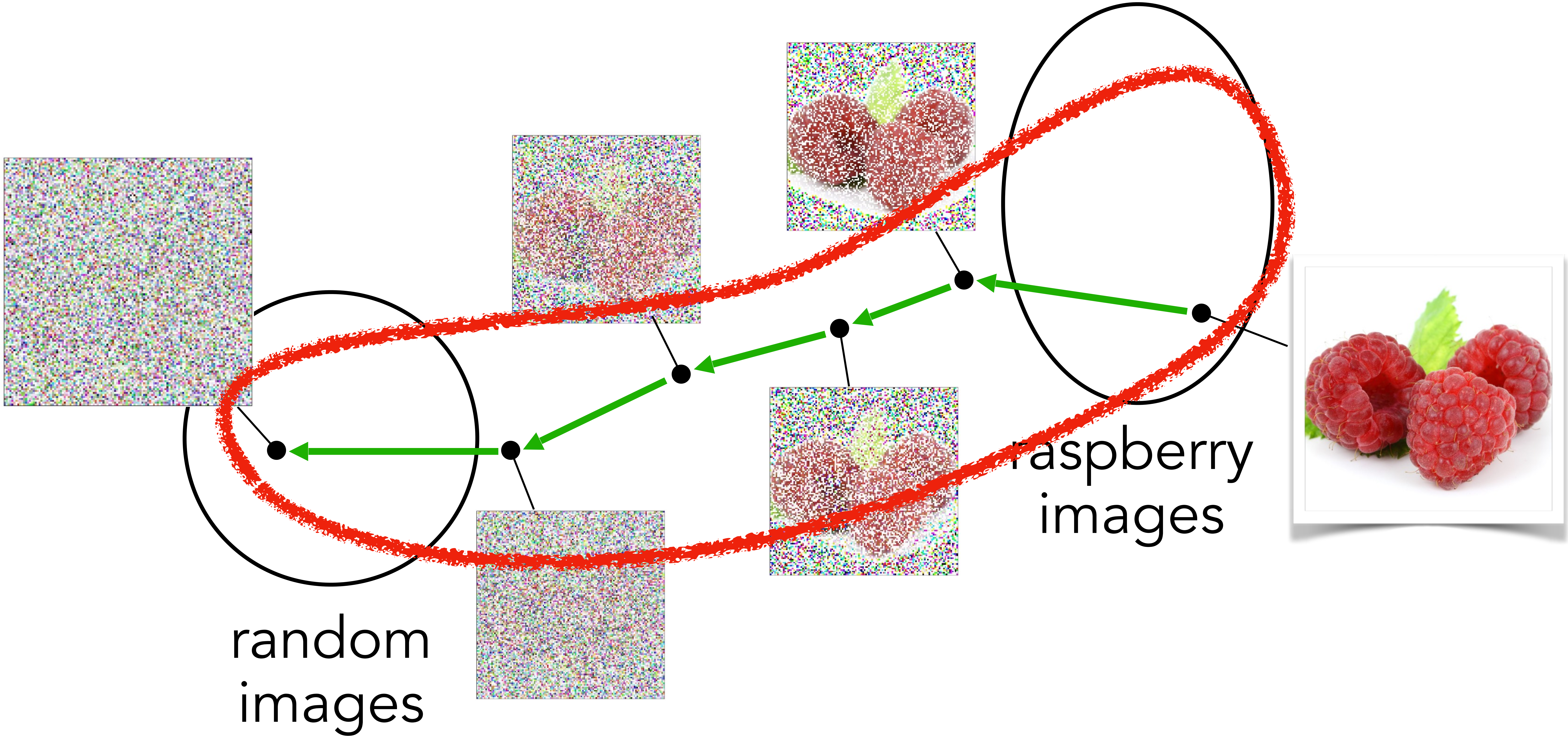
Just like in diffusion, can be a U-Net or transformer (DiT)

Adapted from Angjoo Kanazawa

Slide source: Steve Seitz

How do we pick the intermediate path?

How to generate the path?



Adapted from Angjoo Kanazawa

Figure from Steve Seitz's [video]

What is the path?

- How to add noise? What kind of noise? What schedule do we use?
- This is complicated in the diffusion literature (lots of math).
- Can we keep it simple?

Because it has to follow physical diffusion process. Every time step some gaussian has to be added. But in the end you want it to be a N(0,1) etc..

Flow Matching [Lipman et al. 2022]

Flow matching says that you can (essentially) add noise however you like!

Adapted from Angjoo Kanazawa

How do we construct x_t?

TLDR: Sample noise, add it, then reconstruct the data

- Flow matching is quite general! For now, just consider time-dependent weighted combinations.
- Flow matching says we can choose any weighted combination, as long we start from a sample in the source (e.g. gaussian) and end with a sample in the target distribution (image).

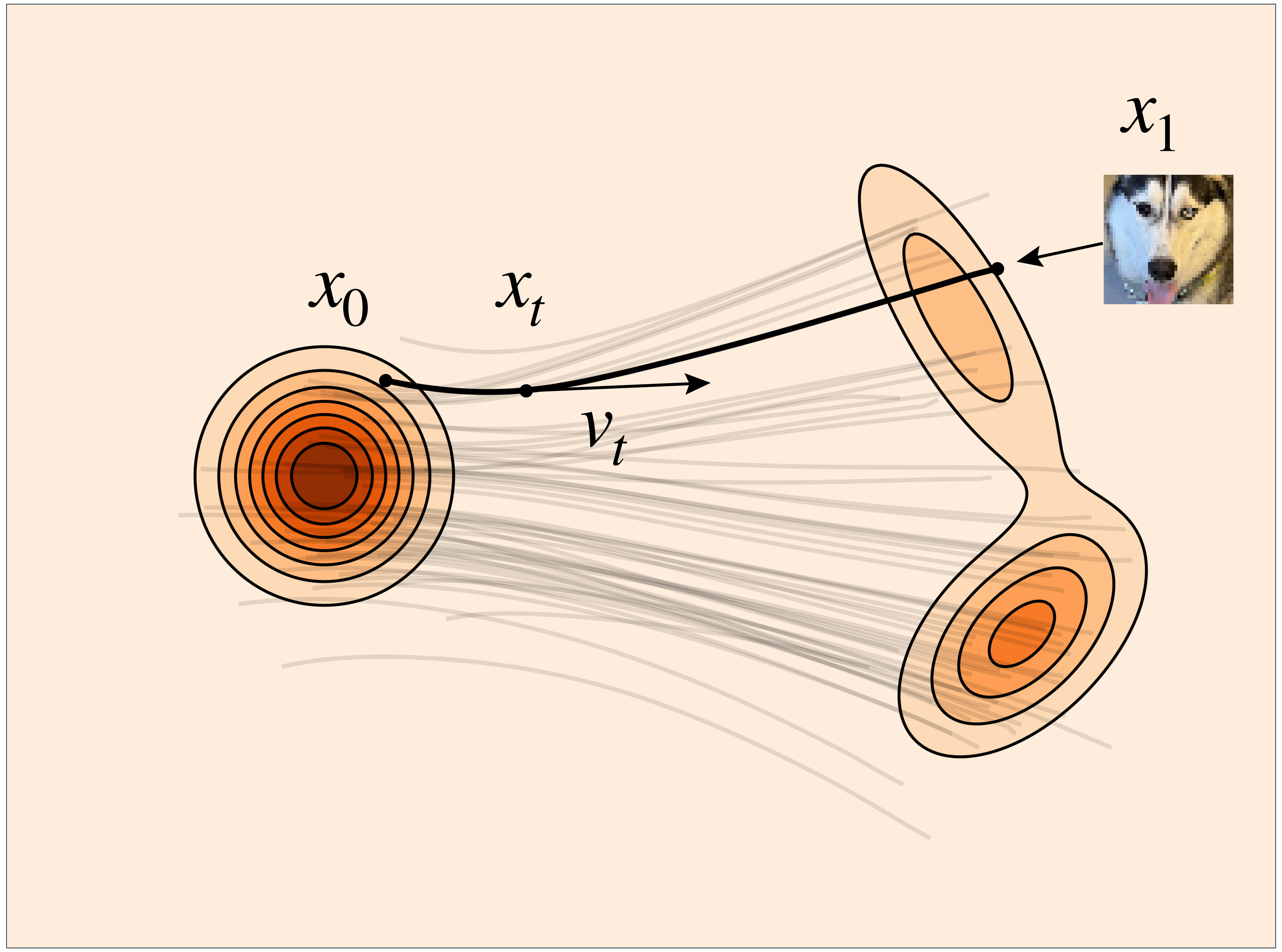
$$x_t = \alpha_t x_0 + \sigma_t x_1$$

$$x_0 \sim p_0(x) \qquad x_1 \sim p_1(x)$$

Adapted from Angjoo Kanazawa

Flow training

- For each image x_1
  - Sample some noise x_0
  - Combine them however you want to get x_t
  - Now learn to predict the velocity at x_t
    - What is the velocity? Depends on how you got x_t



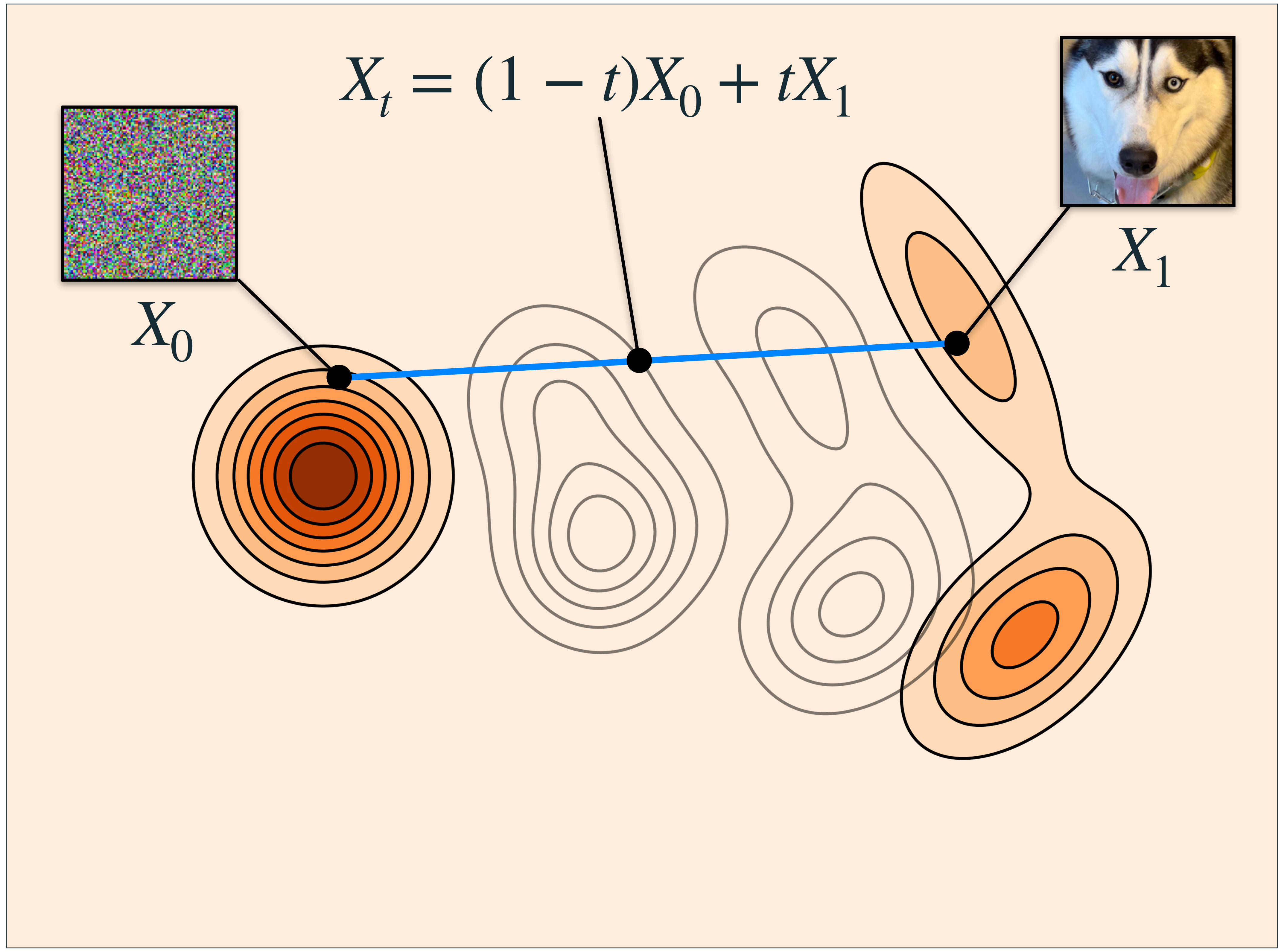
Adapted from Angjoo Kanazawa

A Very Simple Way

Linear interpolation!

$$x_t = \alpha_t x_0 + \sigma_t x_1$$

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



Source: Angjoo Kanazawa

What is the velocity supervision?

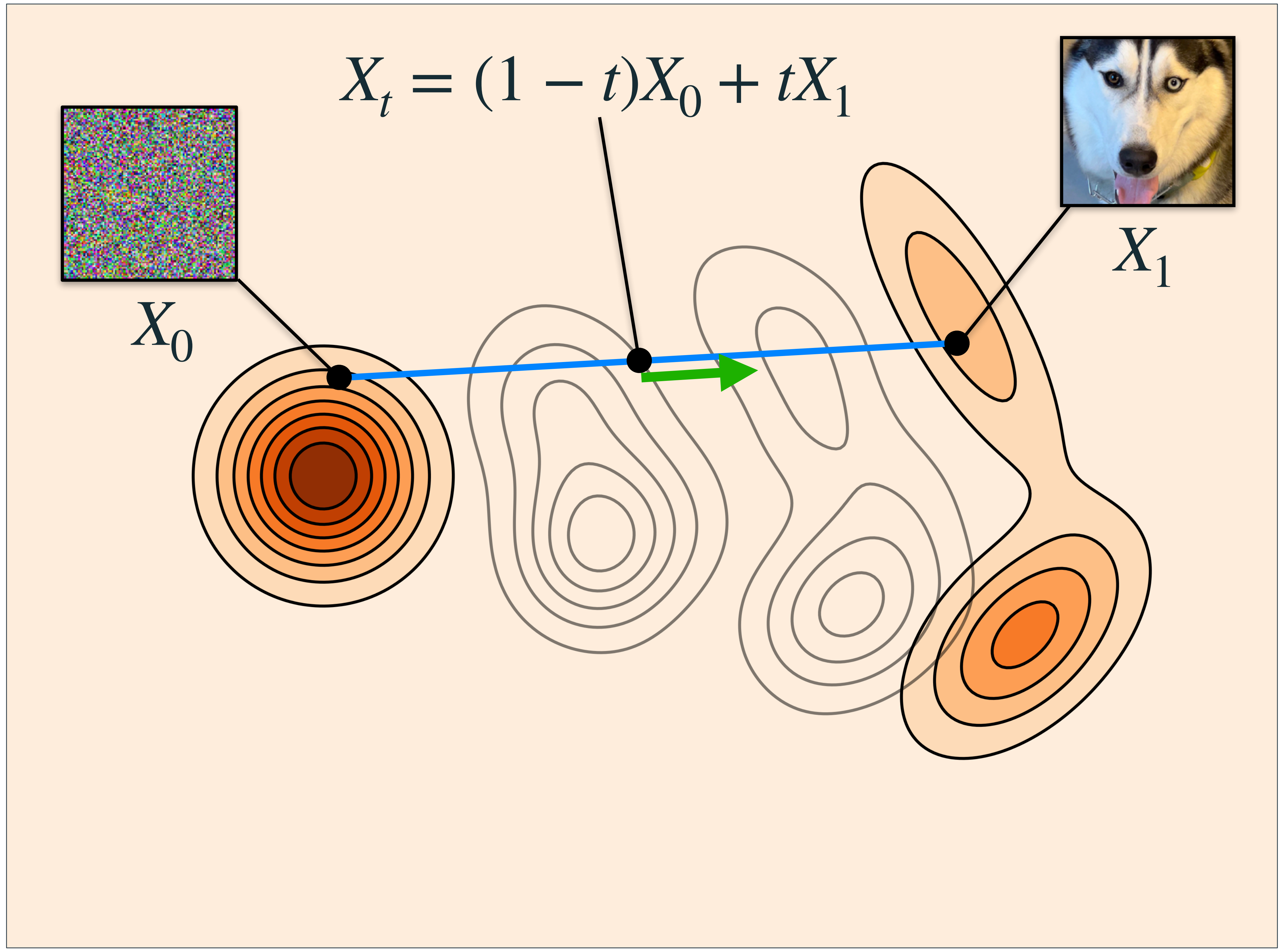
$x_t = \alpha_t x_0 + \sigma_t x_1$

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

$$\frac{dx_t}{dt} = -x_0 + x_1$$

$$= x_1 - x_0$$

$$\mathbb{E}_{t,X_0,X_1}\left\| u_t^\theta(X_t) - (X_1 - X_0) \right\|^2$$



*Conditioned on a single sample

Source: Angjoo Kanazawa

Inside a Training Loop

Flow Matching

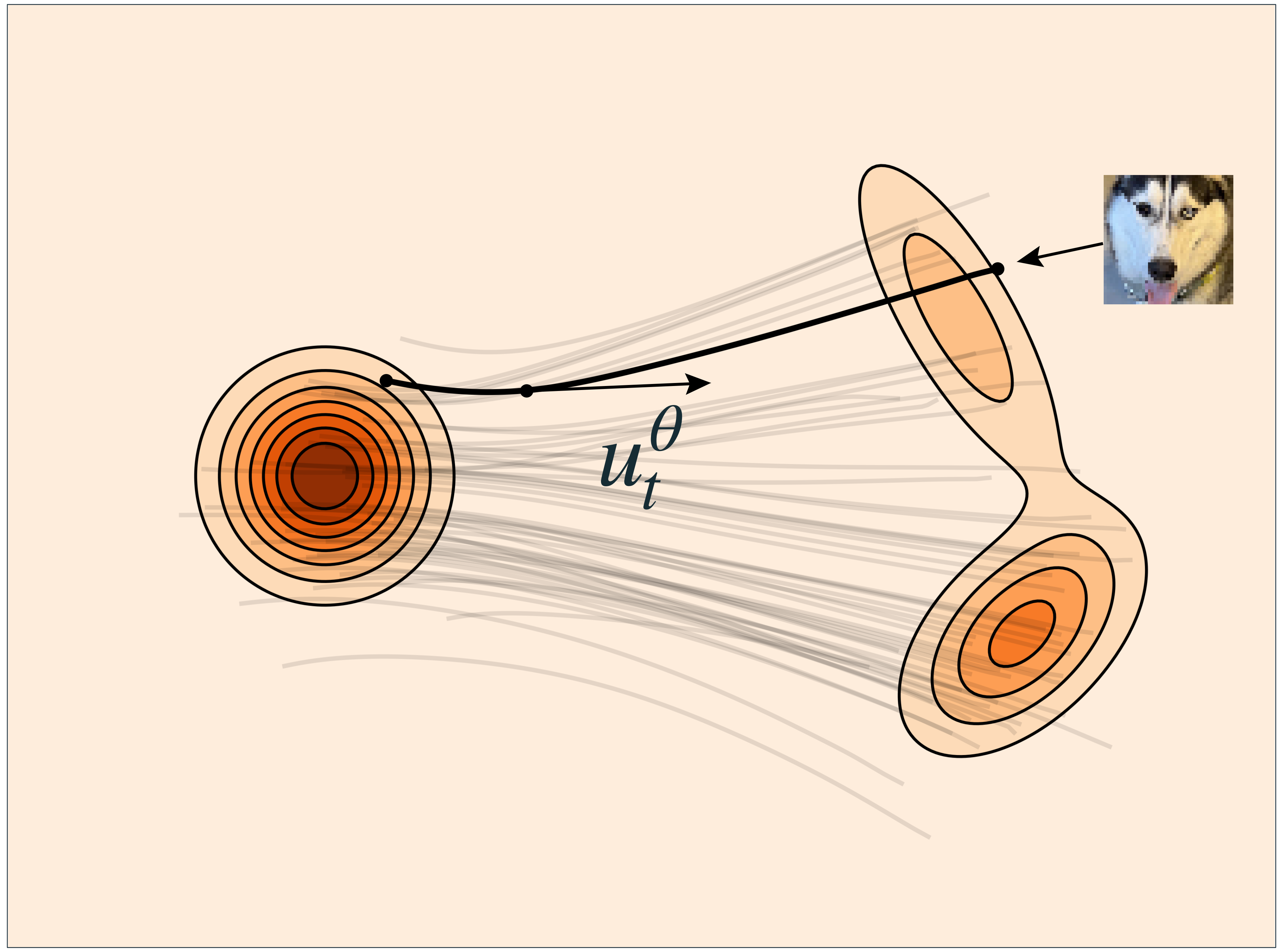
```
x = next(dataset)
t = torch.rand(1) # Sample timestep (0,1)
noise = torch.randn_like(x) # Sample noise
x_t = (1-t) * noise + (t) * x # Get noisy x_t

flow_pred = model(x_t, t) # Predict noise in x_t
flow_gt = x - noise # ground truth flow (w/ linear sched)
loss = F.mse_loss(flow_pred, flow_gt) # Update model
loss.backward()
optimizer.step()
```

Source: Angjoo Kanazawa

Test-time sampling

- Just take a small step in the velocity
- Use any ODE Solver, i.e. integration you like, e.g., Euler integration:

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



Sample
from $X_0 \sim p$

Adapted from: Angjoo Kanazawa

Inside Sampling Loop

```
velocity = model(x_t, t) # Predict noise in x_t
x_t = x_t + dt * velocity # Step in velocity
```

Source: Angjoo Kanazawa

Training: Model parameterization

- You can make your network output undo the noise in many different ways, predicting x_1, v, noise, or flow

$$v_t = \alpha_t x_1 - \sigma_t x_0 \qquad u_t = x_1 - x_0 = \epsilon - x_0$$
$$u_t = x_t - x_0$$

- These are all equivalent because of the linear relationship with x_t. You can derive all of these as long as you know one of them

$$x_t = \alpha_t x_0 + \sigma_t x_1$$

- For example

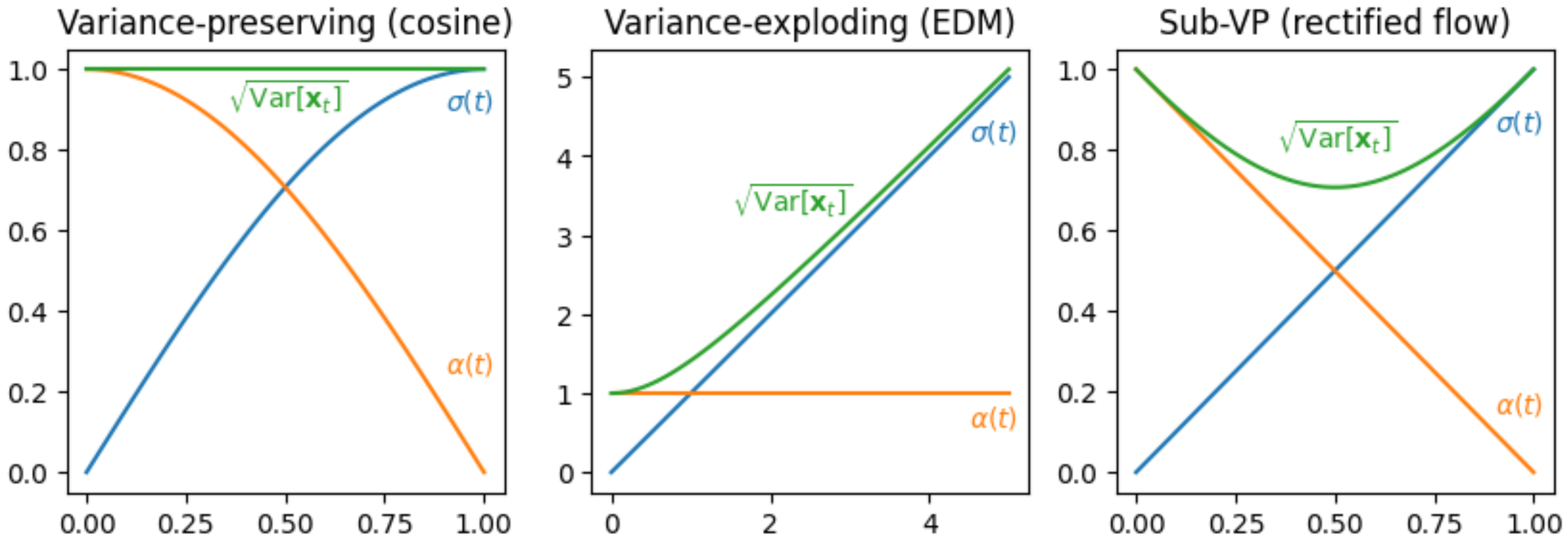
$$\mathbb{E}[(\hat{\mathbf{x}}_0 - \mathbf{x}_0)^2] = \mathbb{E}\left[\left(\frac{\mathbf{x}_t - \sigma(t)\hat{\varepsilon}}{\alpha(t)} - \frac{\mathbf{x}_t - \sigma(t)\varepsilon}{\alpha(t)}\right)^2\right] = \mathbb{E}\left[\frac{\sigma(t)^2}{\alpha(t)^2}\left(\hat{\varepsilon} - \varepsilon\right)^2\right].$$

Slides from Yaron Lipman

Source: Angjoo Kanazawa

Other options lead to prior works

- Other choices: $$x_t = \alpha_t x_0 + \sigma_t x_1$$
  - Preserve variance (VP-ODE) - DDPM ("standard" diffusion formulation you've seen already)
  - Exploding variance (VE-ODE) - Score Matching/DDIM
  - Linear interpolation (Flow Matching we've discussed so far, also Rectified Flow)



Adapted from: Angjoo Kanazawa

Training: Flow Matching vs. Diffusion

Algorithm 1: Flow Matching training.

Input : dataset q, noise p
Initialize v^θ
while *not converged* **do**
- $t \sim \mathcal{U}([0,1])$ ▷ sample time
- $x_1 \sim q(x_1)$ ▷ sample data
- $x_0 \sim p(x_0)$ ▷ sample noise
- $x_t = \Psi_t(x_0|x_1)$ ▷ conditional flow
- Gradient step with $\nabla_\theta \|v_t^\theta(x_t) - \dot{x}_t\|^2$

Output: v^θ

Algorithm 2: Diffusion training.

Input : dataset q, noise p
Initialize s^θ
while *not converged* **do**
- $t \sim \mathcal{U}([0,1])$ ▷ sample time
- $x_1 \sim q(x_1)$ ▷ sample data
- $x_t = p_t(x_t|x_1)$ ▷ sample conditional prob
- Gradient step with $\nabla_\theta \|s_t^\theta(x_t) - \nabla_{x_t} \log p_t(x_t|x_1)\|^2$

Output: v^θ

$p_t(x_t | x_1)$ general
$p(x_0)$ is general

$p_t(x_t | x_1)$ closed-form from of SDE $dx_t = f_t dt + g_t dw$
- *Variance Exploding*: $p_t(x | x_1) = \mathcal{N}(x | x_1, \sigma_{1-t}^2 I)$
- *Variance Preserving*: $p_t(x | x_1) = \mathcal{N}(x | \alpha_{1-t} x_1, (1 - \alpha_{1-t}^2) I)$

$$\alpha_t = e^{-\frac{1}{2}T(t)}$$

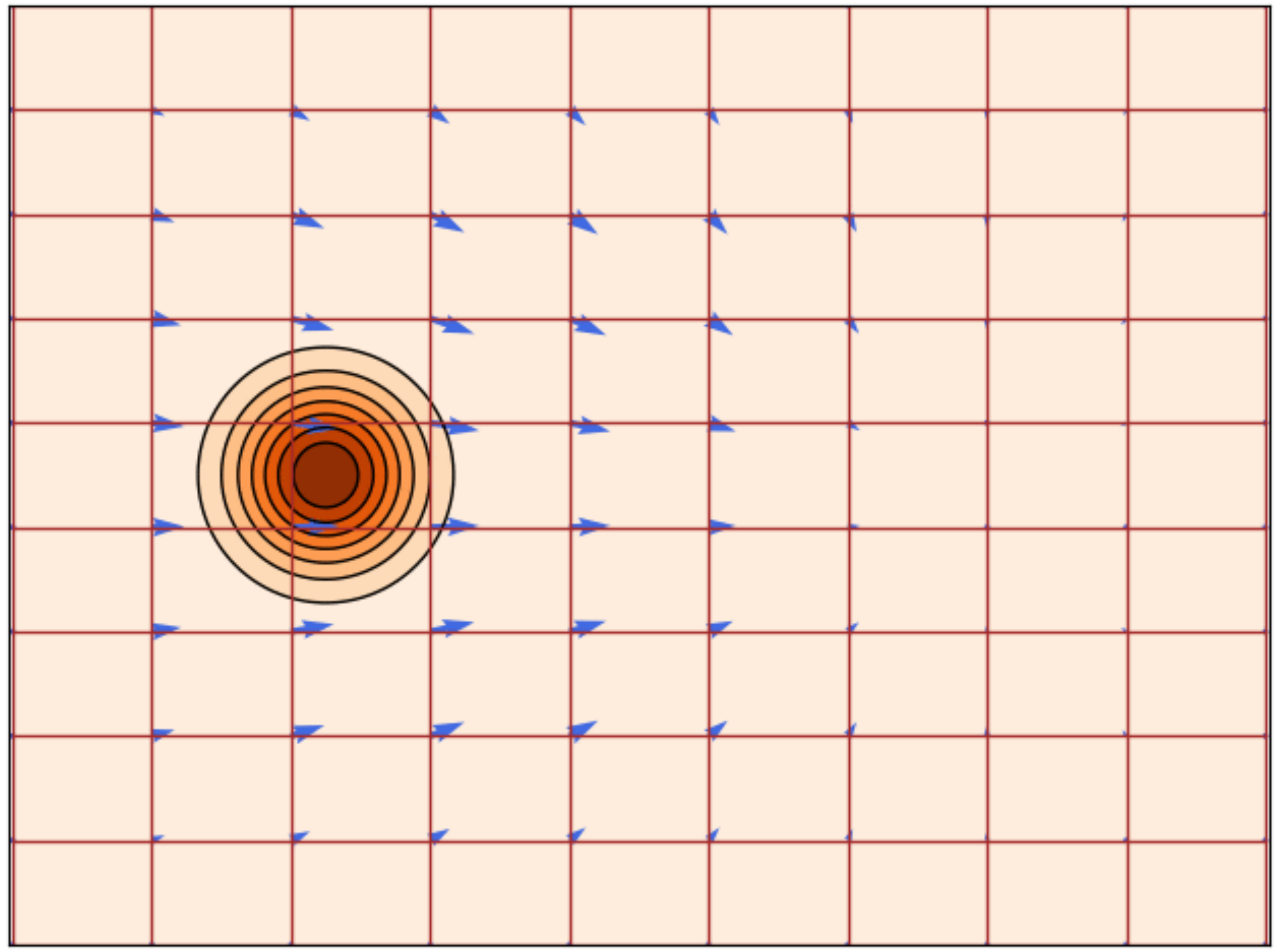
$p(x_0)$ is Gaussian
$p_0(\cdot | x_1) \approx p$

Source: Angjoo Kanazawa

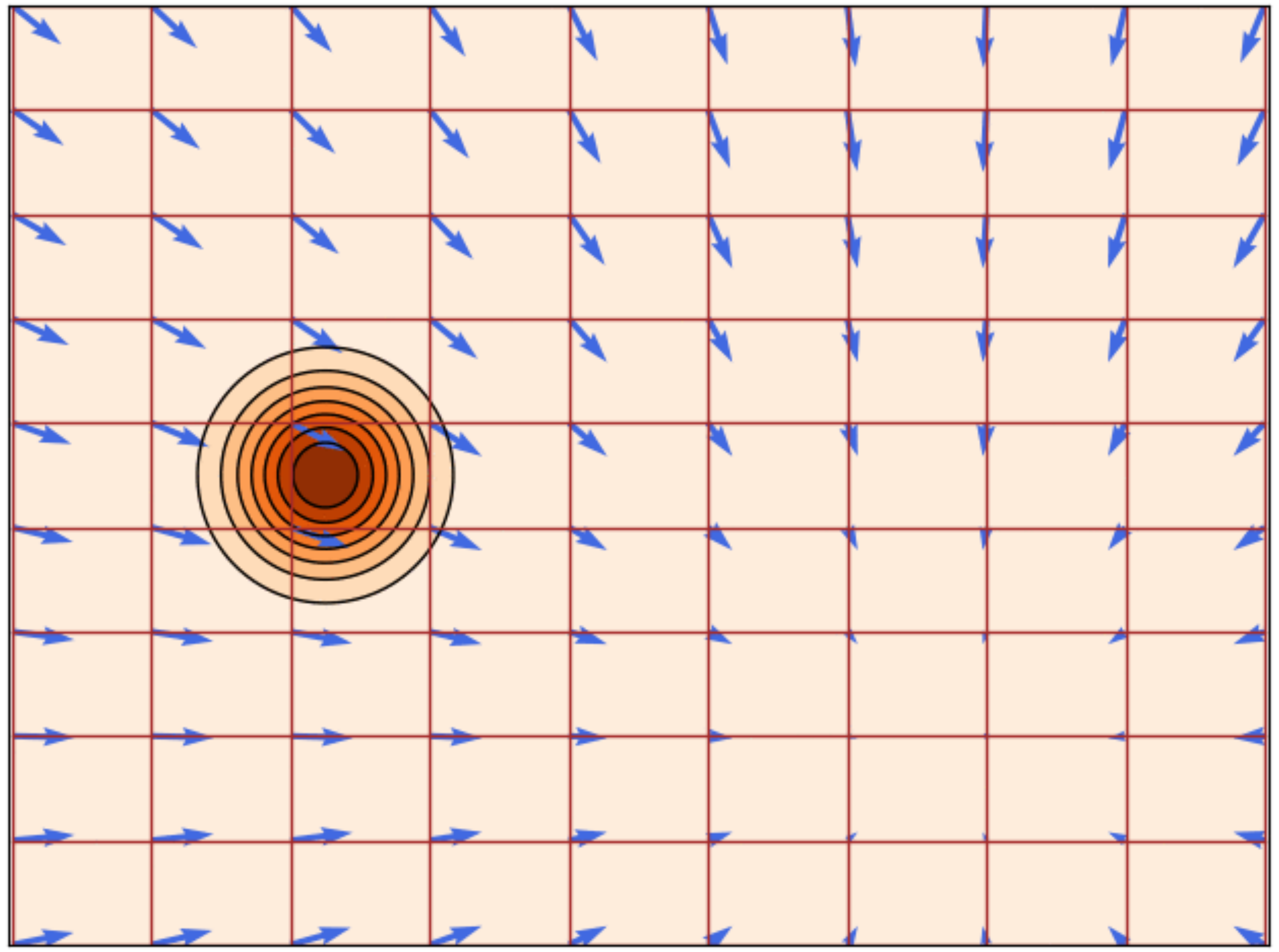
Why does this work?

- What we want is the flow (velocity field) that takes samples from p_0 to p_1 when integrated (called Marginal Flow)

- But what we did was to train flow for each sample (Conditional Flow)



Marginal Flow (what we want)



Conditional Flow (what we trained)

Source: Angjoo Kanazawa

It turns out that the gradient is the same!

- **Flow Matching loss:**

$$\mathscr{L}_{\text{FM}}(\theta) = \mathbb{E}_{t,X_t} \left\| u_t^\theta(X_t) - u_t(X_t) \right\|^2$$

We can't do this because we don't know what ground truth flow is.

- **Conditional Flow Matching loss**:

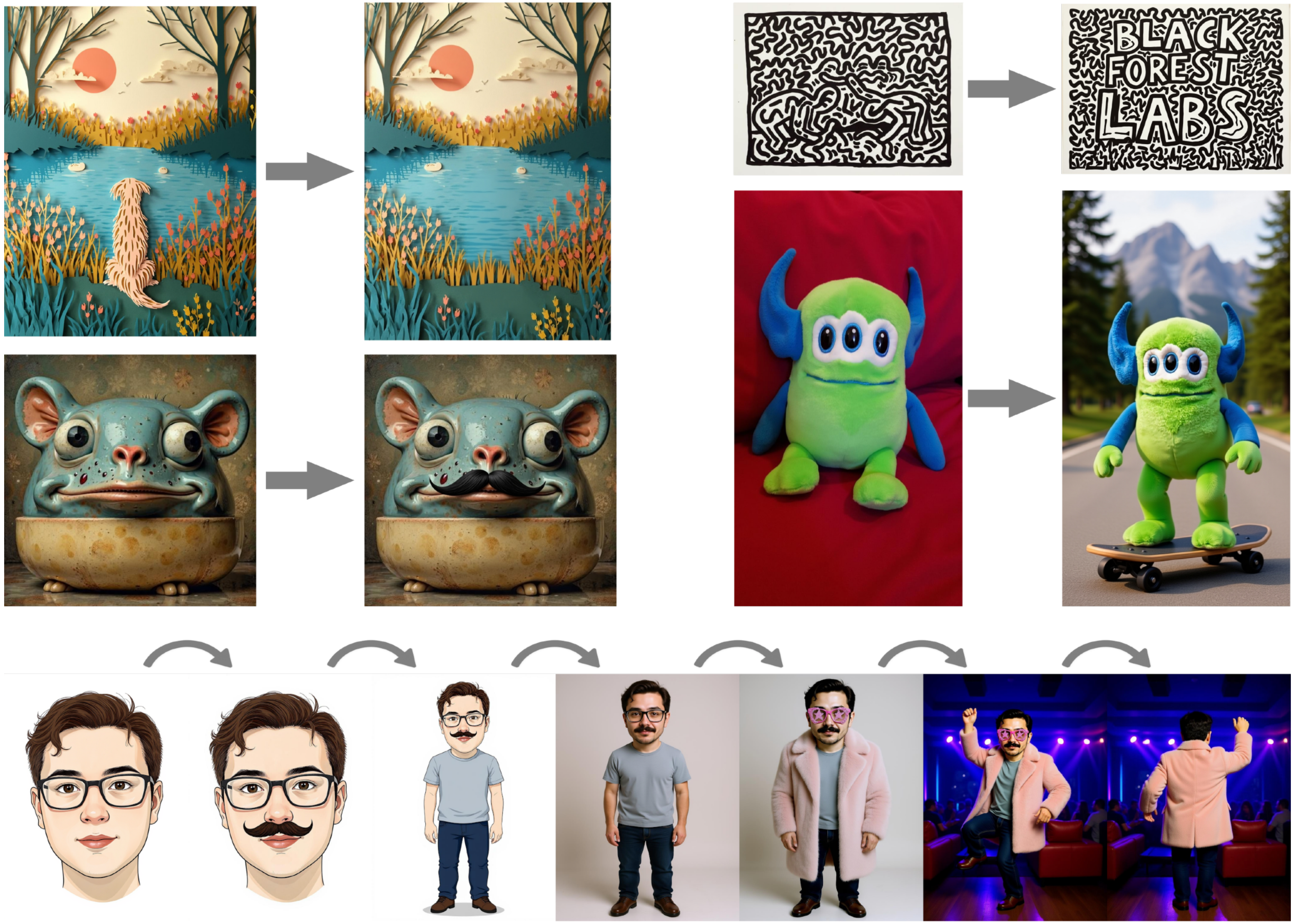
$$\mathscr{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,X_1,X_t} \left\| u_t^\theta(X_t) - u_t(X_t \mid X_1) \right\|^2$$

Theorem: Gradient of the losses are equivalent:

$$\nabla_\theta \mathscr{L}_{\text{FM}}(\theta) = \nabla_\theta \mathscr{L}_{\text{CFM}}(\theta)$$

Source: Angjoo Kanazawa

Flow matching in state-of-art generators

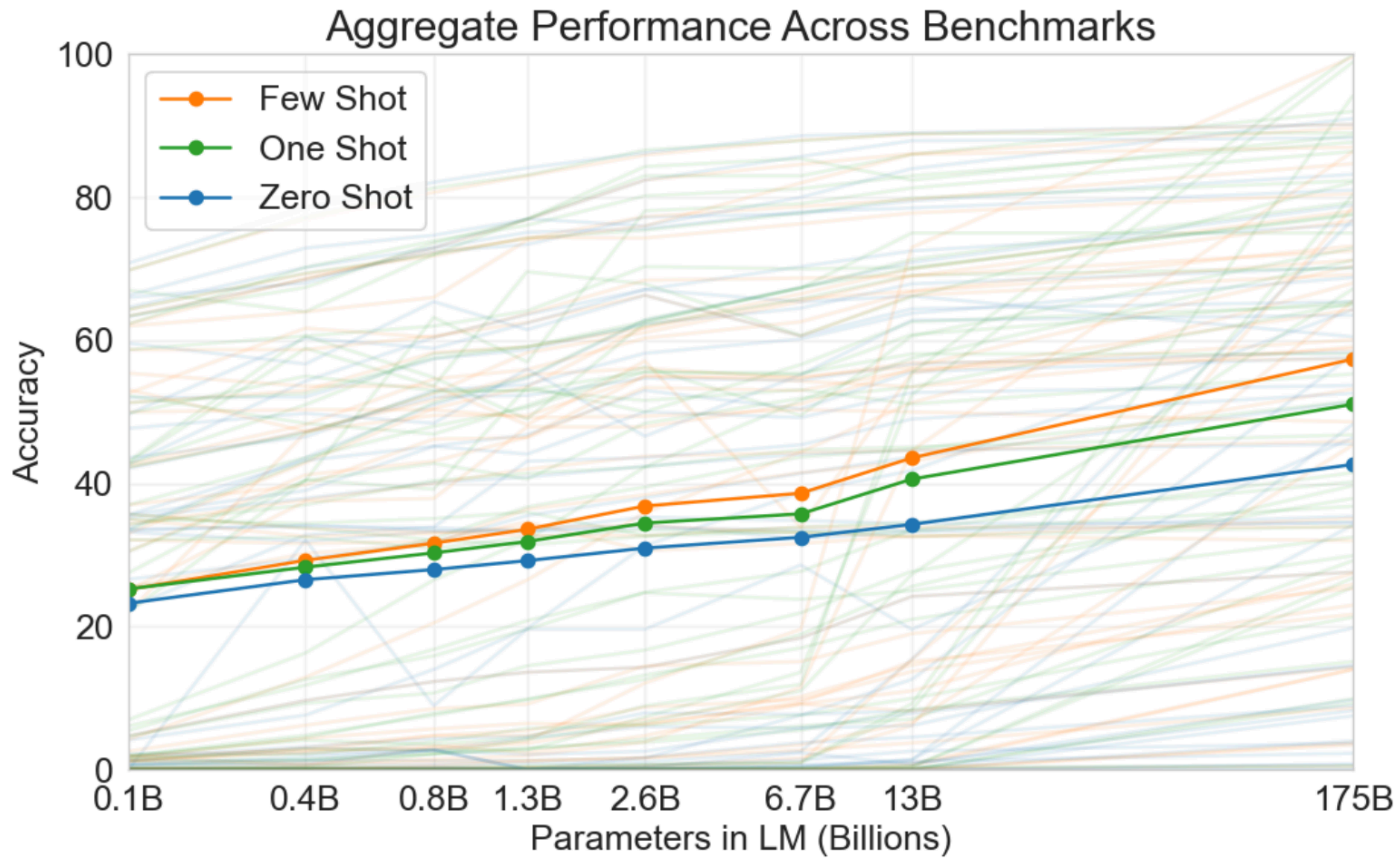


Source: FLUX.1-Kontext

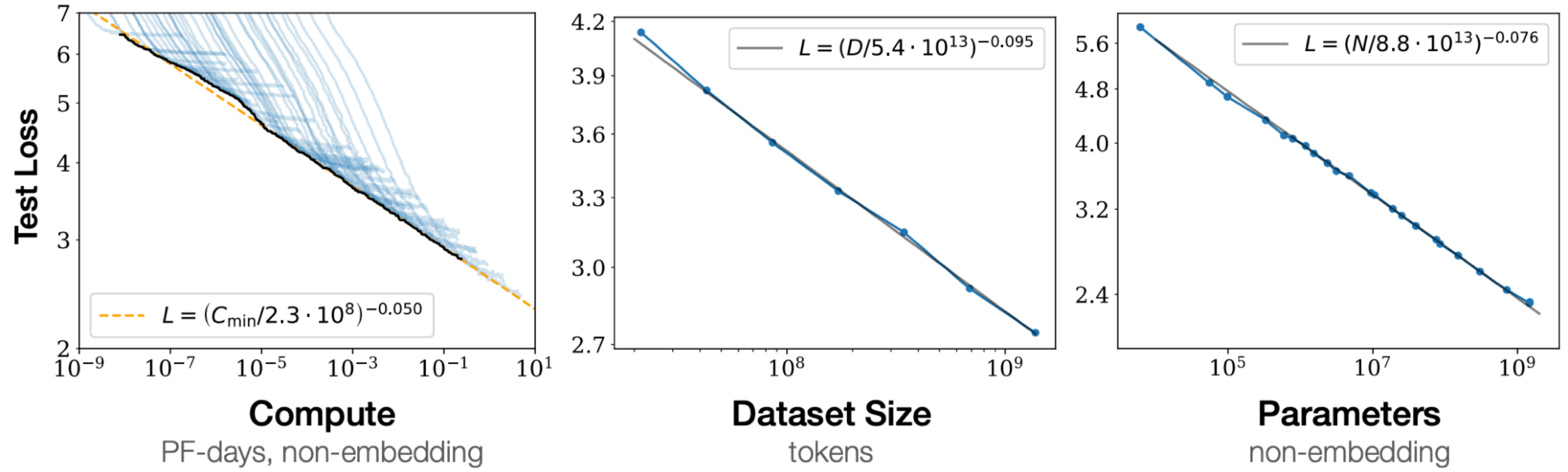
Today

- 3D generation (continuing from last time)
- Flow matching
- **Video generators as vision problem solvers**

Scaling language models

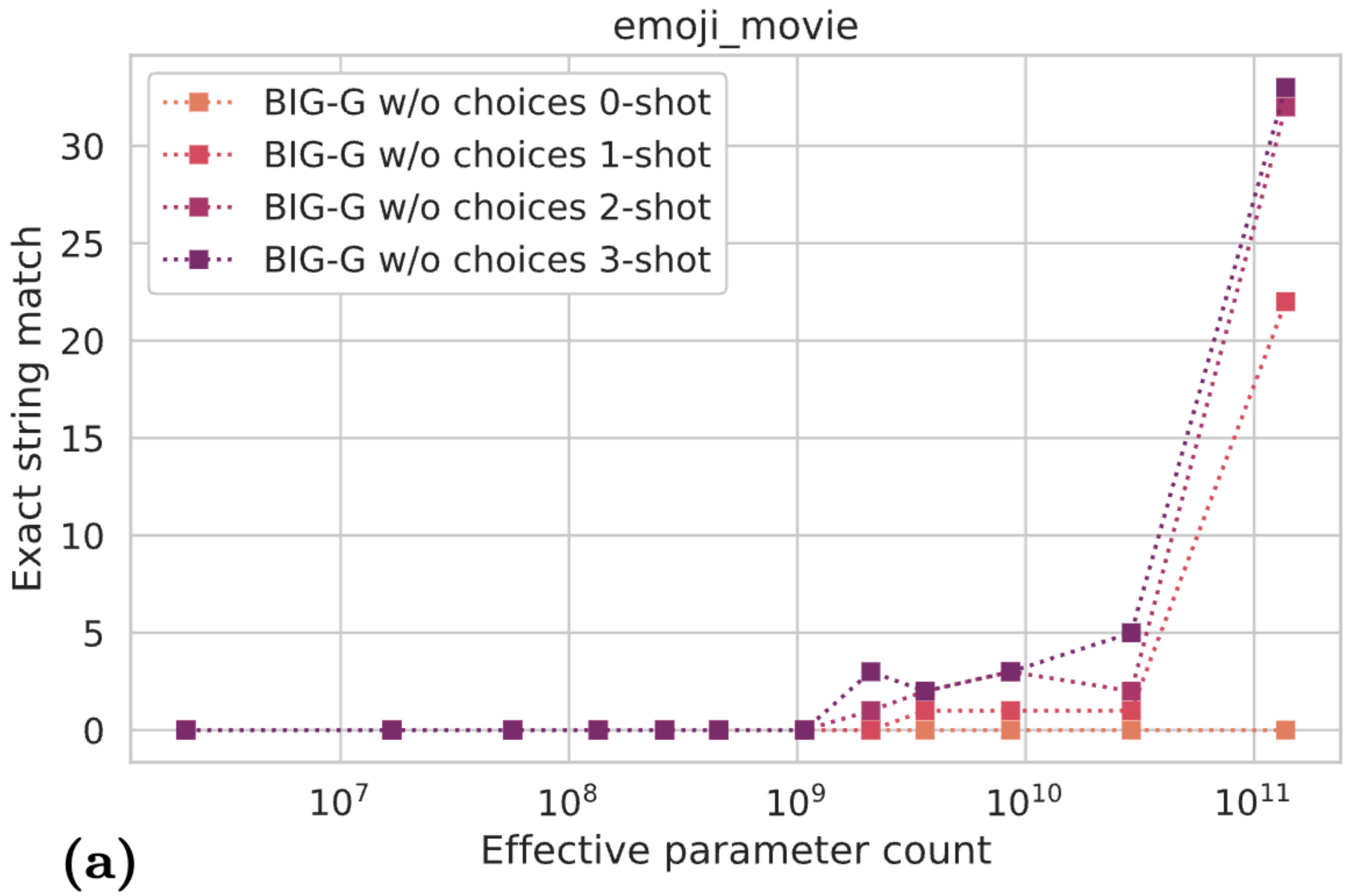


[Brown et al., “Language Models are Few-Shot Learners”, 2020]

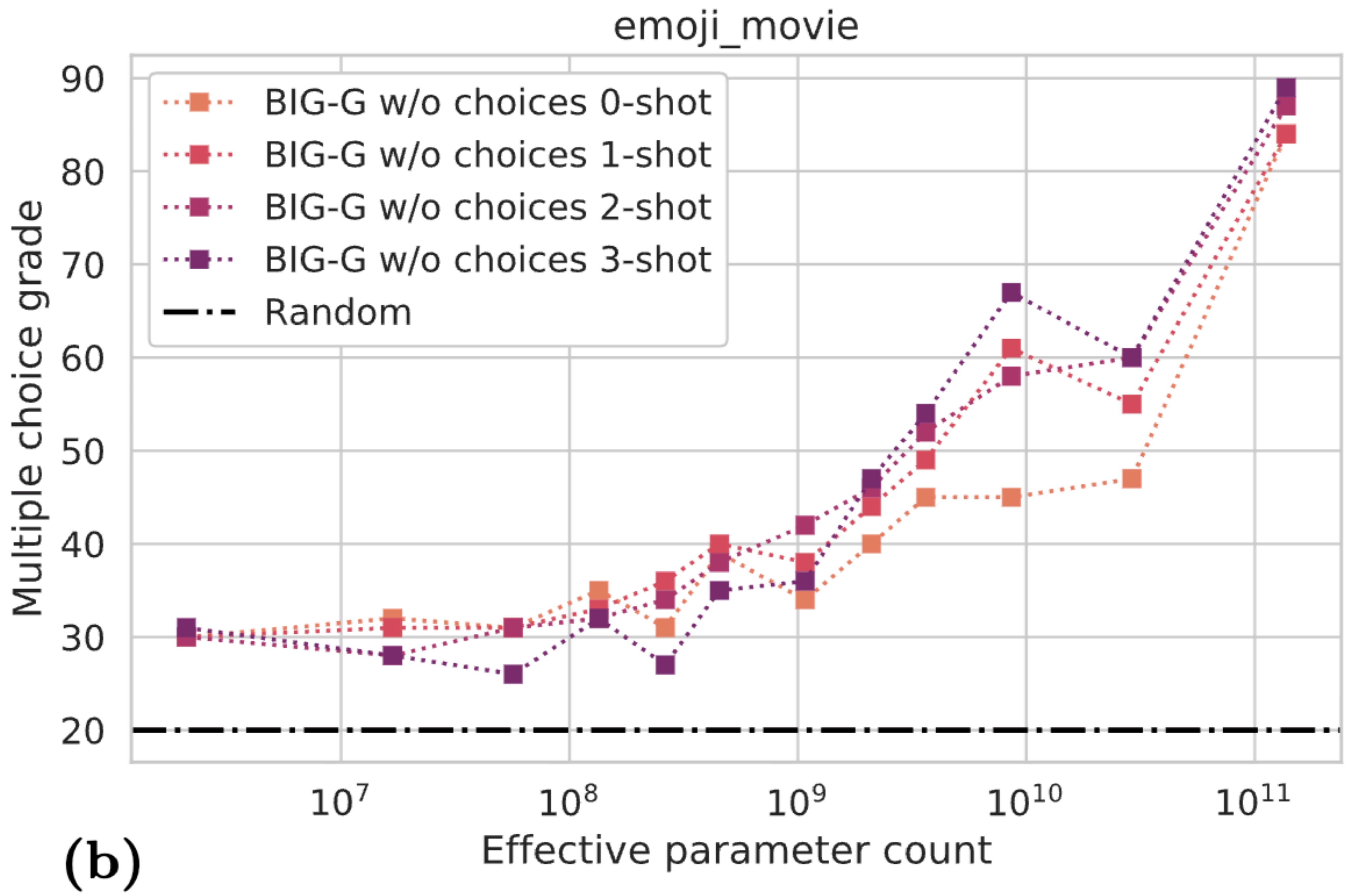


[Kaplan et al., “Scaling Laws for Neural Language Models”, 2020]

Scaling language models



(a)



(b)

```
Q: What movie does this emoji describe? 👧🐟🐠🐡

2m:   i'm a fan of the same name, but i'm not sure if it's a good idea
16m:  the movie is a movie about a man who is a man who is a man …
53m:  the emoji movie 🐟🐠🐡
125m: it's a movie about a girl who is a little girl
244m: the emoji movie
422m: the emoji movie
1b:   the emoji movie
2b:   the emoji movie
4b:   the emoji for a baby with a fish in its mouth
8b:   the emoji movie
27b:  the emoji is a fish
128b: finding nemo
```

(c)

[Srivastava et al., "Beyond the Imitation Game", 2022]

Will this happen to vision too?

Does tracking “emerge” in video diffusion models?

Videos generated by Veo 3 + point tracks from CoTracker [Karaev, et al., 2023]



Ayush Shrivastava* Sanyam Mehta* Daniel Geng

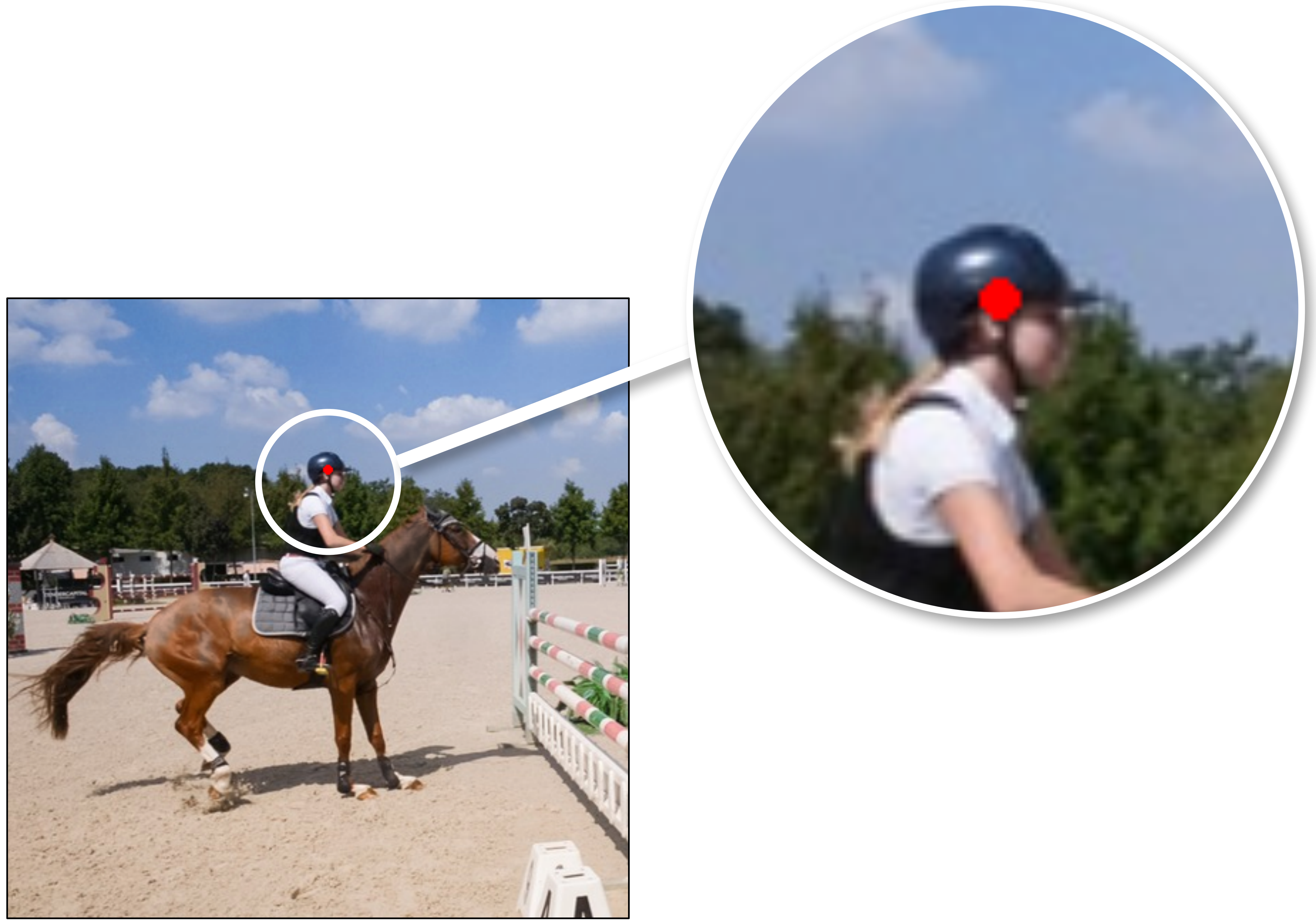
Point Prompting: Counterfactual Tracking with Video Diffusion Models, arXiv 2025.

Point prompting

Query point

First frame

Point prompting



First frame + ●

Point prompting

First frame + ●

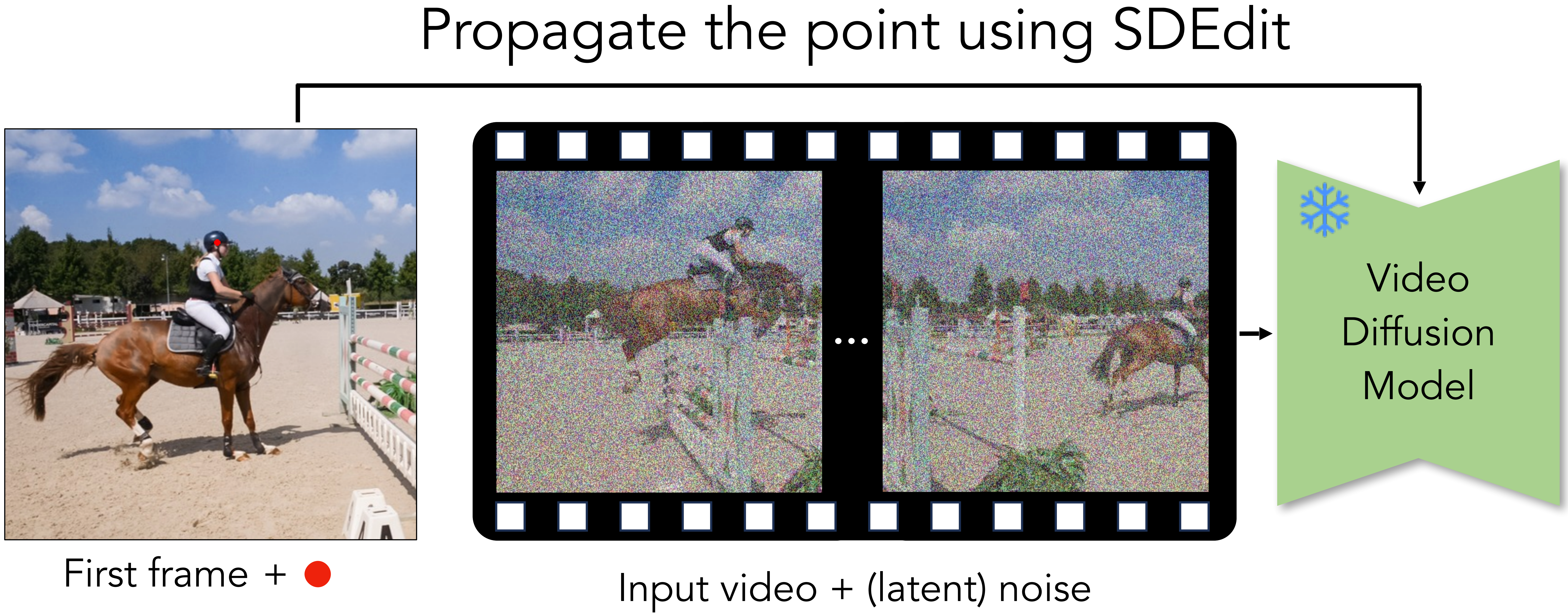
Input video

Point prompting

First frame + ●

Input video + (latent) noise

Point prompting

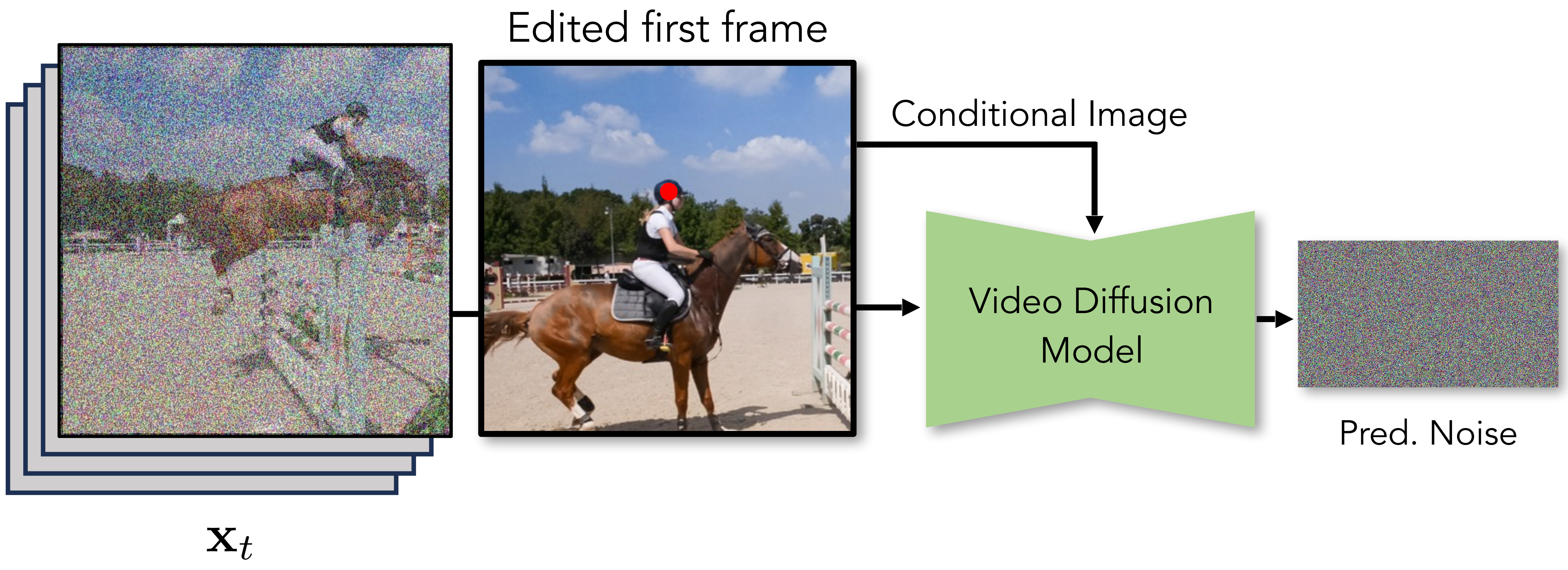


WAN 2.1,14B image-conditioned video diffusion model

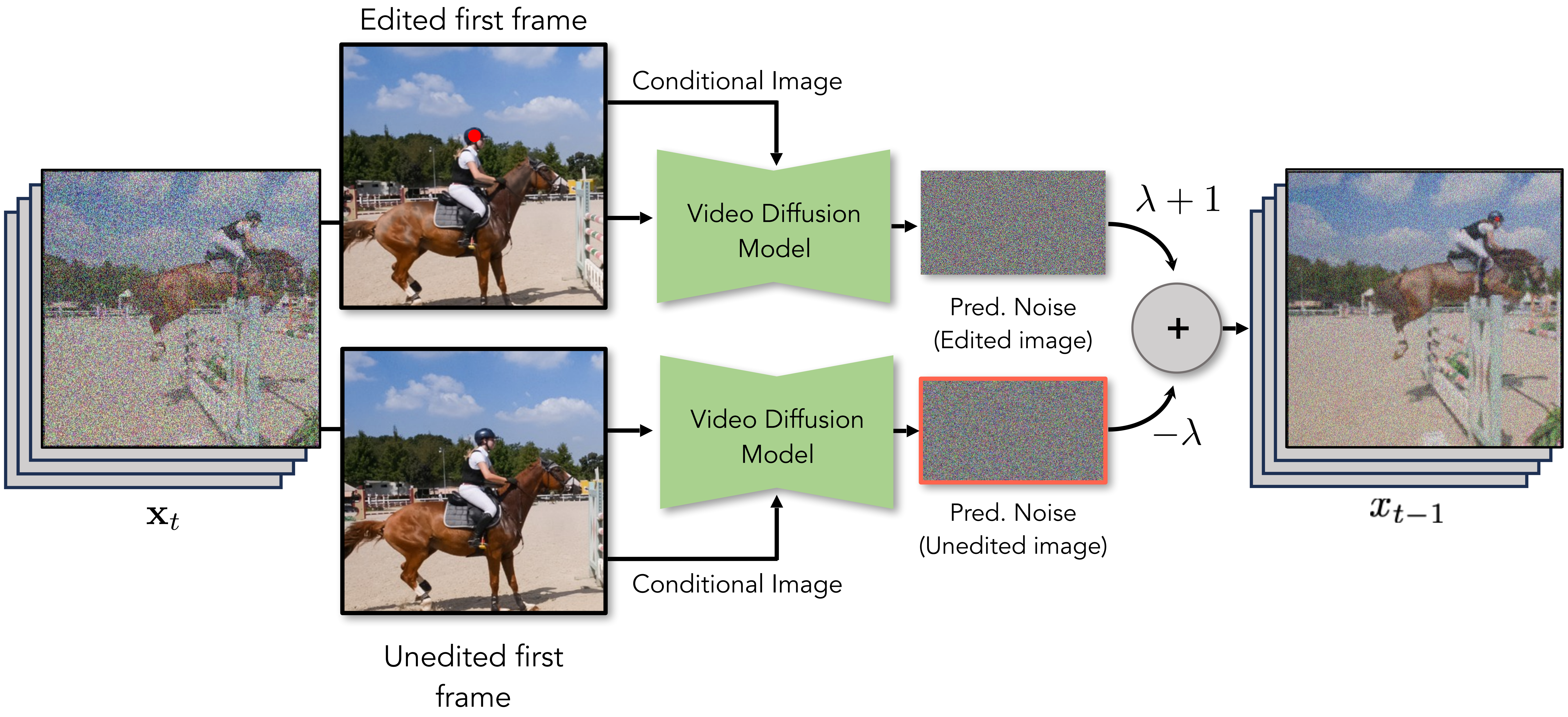
Point prompting

Generated video with propagated point

Enhancing the counterfactual signal

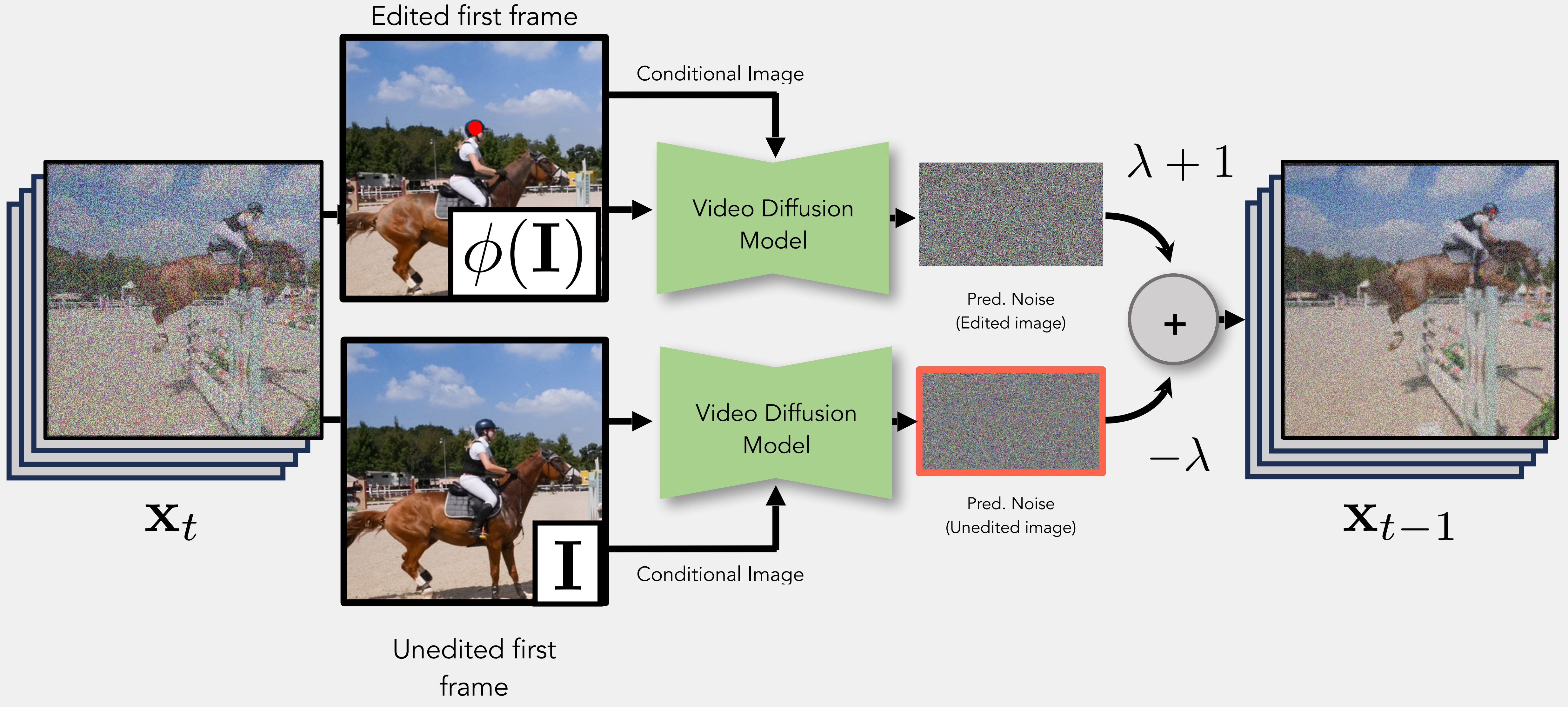


Enhancing the counterfactual signal



Negative prompting with CFG [Ho & Salimans, 2022]

Enhancing the counterfactual signal



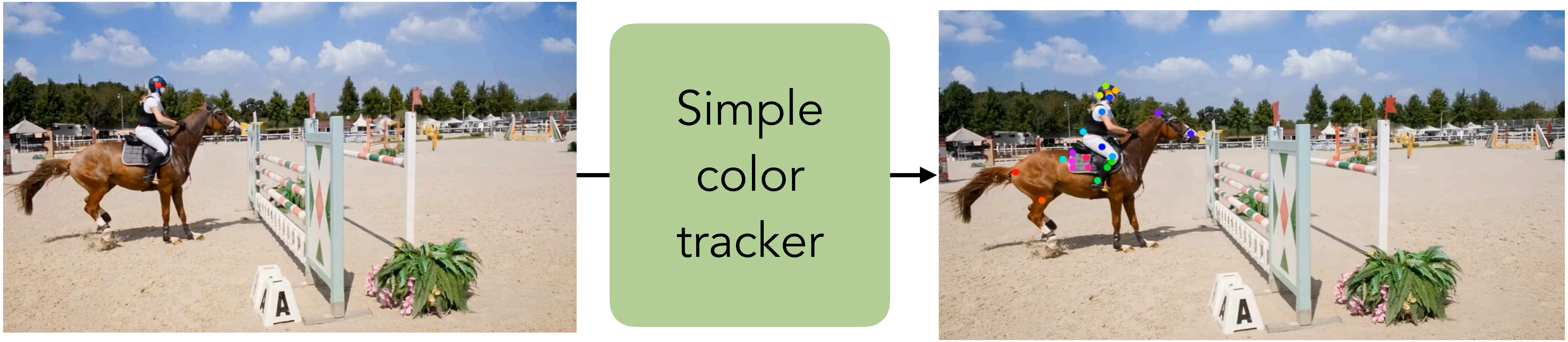
Probabilistic interpretation:

$$\nabla_{\mathbf{x}_t} \log \left(p(\mathbf{x}_t \mid \phi(\mathbf{I})) \left[\frac{p(\phi(\mathbf{I}) \mid \mathbf{x}_t)}{p(\mathbf{I} \mid \mathbf{x}_t)} \right]^{\lambda} \right)$$

See [Ho & Salimans, 2022]

+ color rebalancing

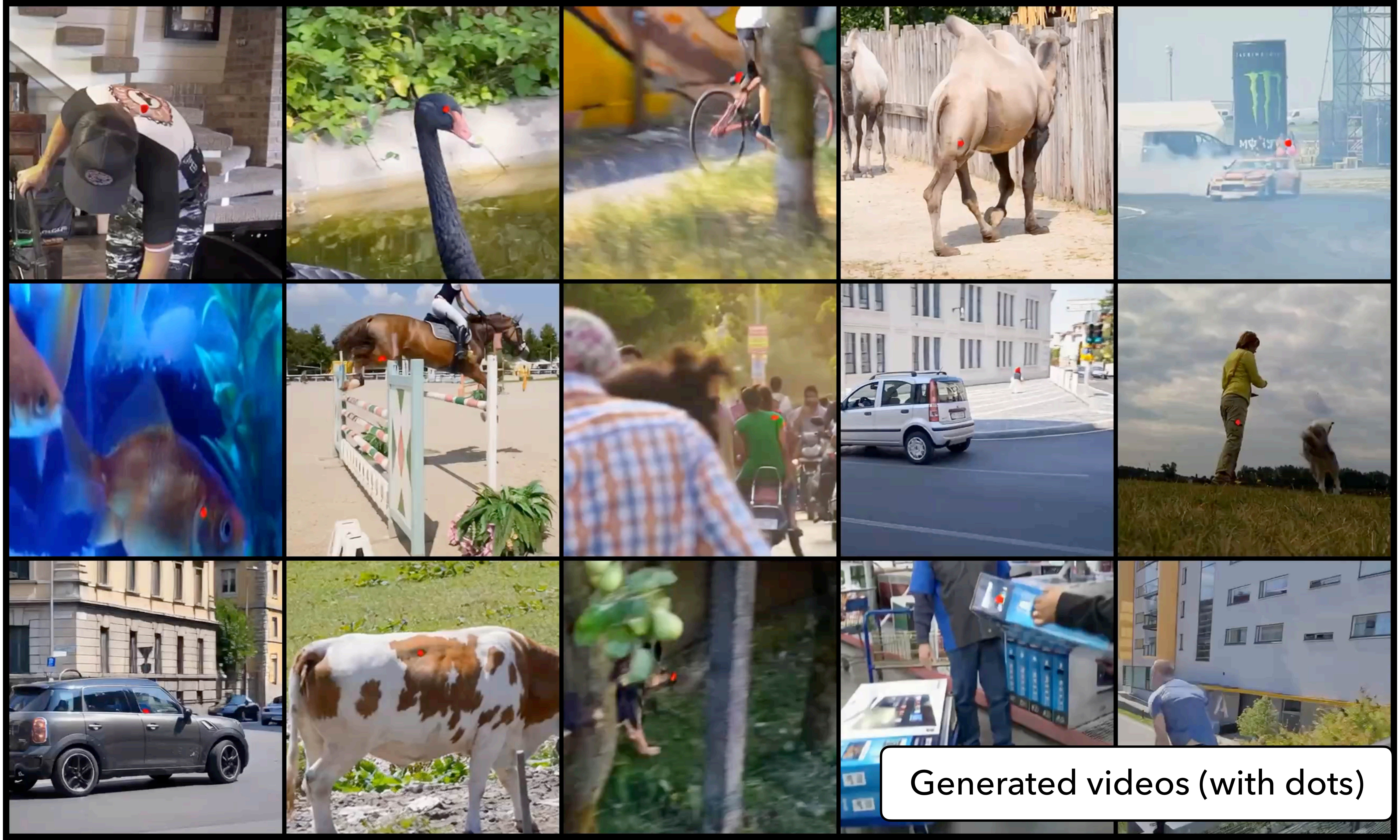
From dots to tracks



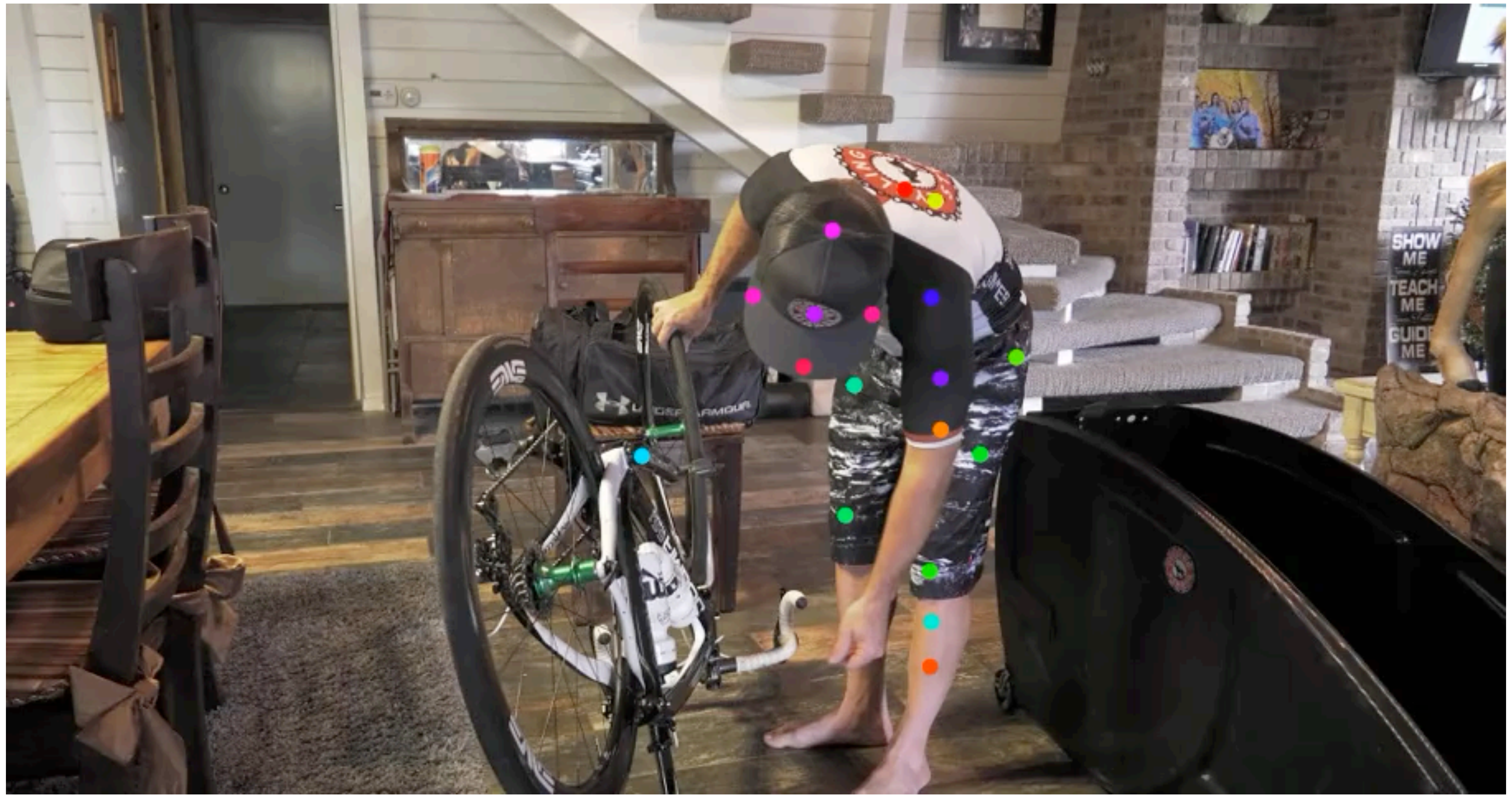
- Threshold colors, then do template matching.
- Interpolate position between frames when no dot is found.
- Generate one video per query (slow!).


Problem: artifacts in generated video

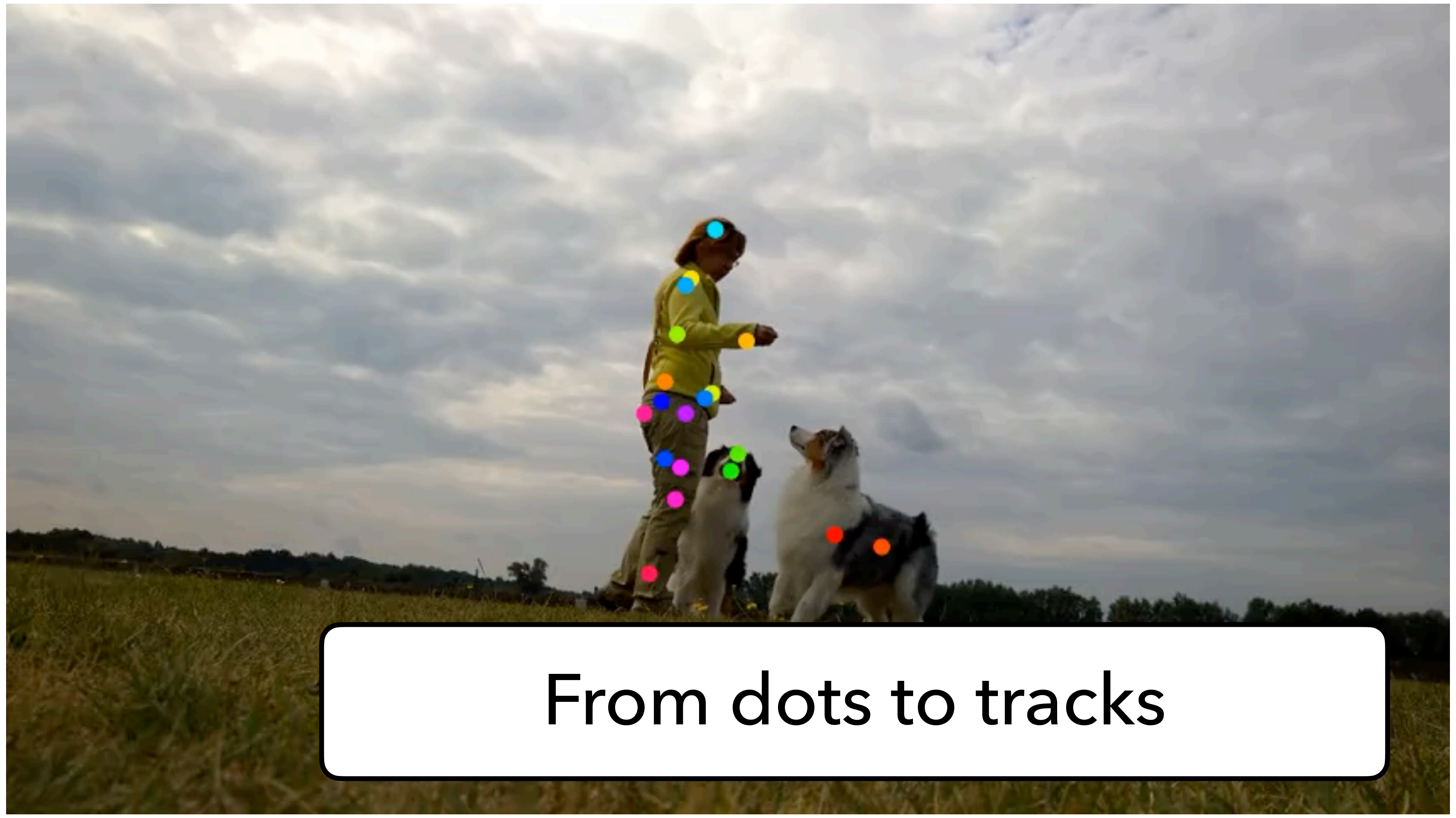
Coarse-to-fine: regenerate a small region by inpainting.



Generated videos (with dots)







From dots to tracks

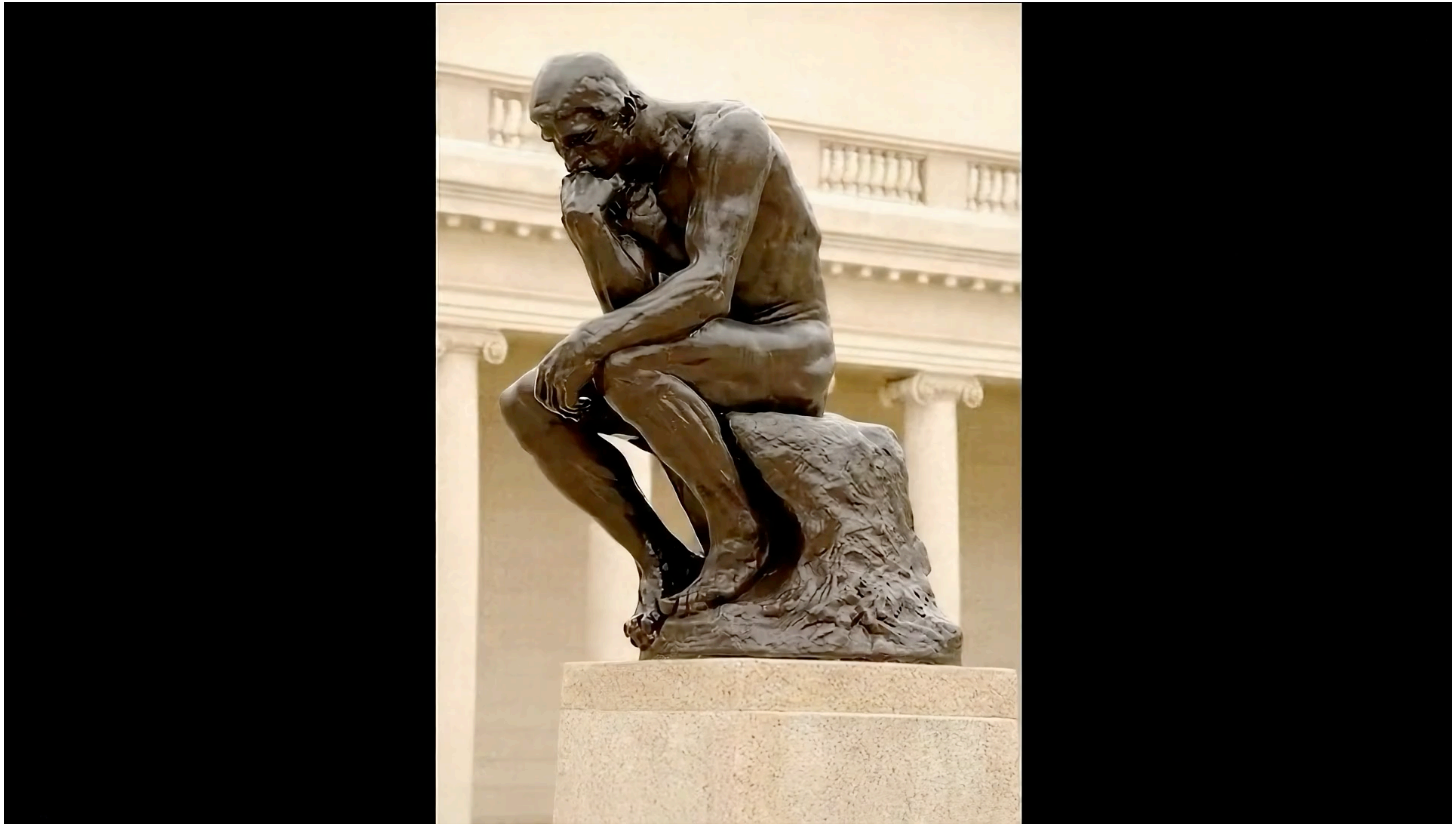
Failure case: misinterpreting prompt

Failure case: losing point

Failure case: symmetry

What else can video models do?

Novel view synthesis



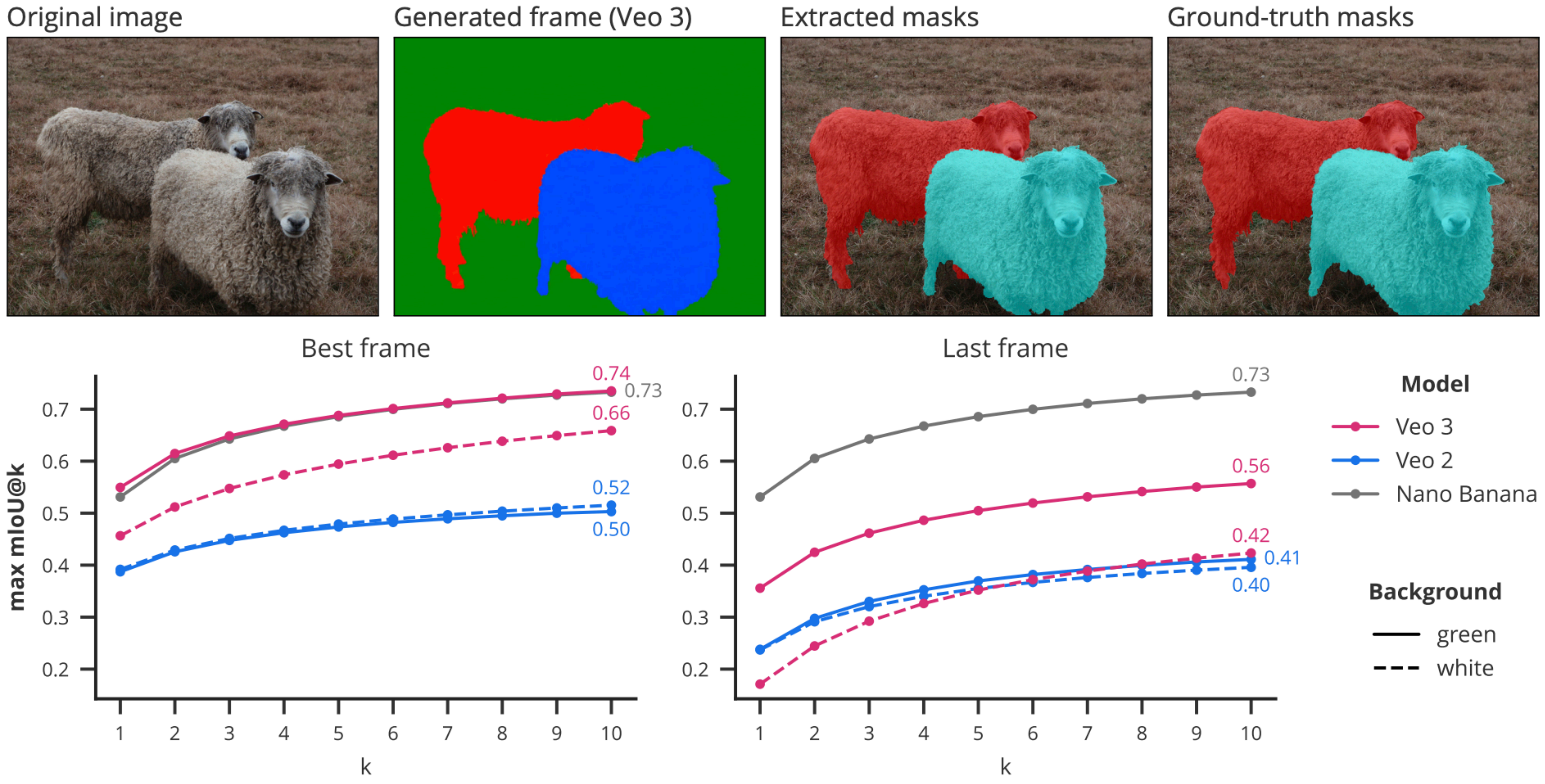
From [Wiedemer et al., “Video models are zero-shot learners and reasoners”, 2025]

Object segmentation

"Create an animation of instance segmentation being performed on this photograph: each distinct entity is overlaid in a different flat color.
Scene:
• The animation starts from the provided, unaltered photograph.
• The scene in the photograph is static and doesn't move.
• First, the background fades to {white, green}.
• Then, the first entity is covered by a flat color, perfectly preserving its silhouette.
• Then the second entity, too, is covered by a different flat color, perfectly preserving its silhouette.
• One by one, each entity is covered by a different flat color.
• Finally, all entities are covered with different colors.
Camera:
• Static shot without camera movement.
• No pan.
• No rotation.
• No zoom.
• No glitches or artifacts."

From [Wiedemer et al., "Video models are zero-shot learners and reasoners", 2025]

Object segmentation



Not extremely accurate…

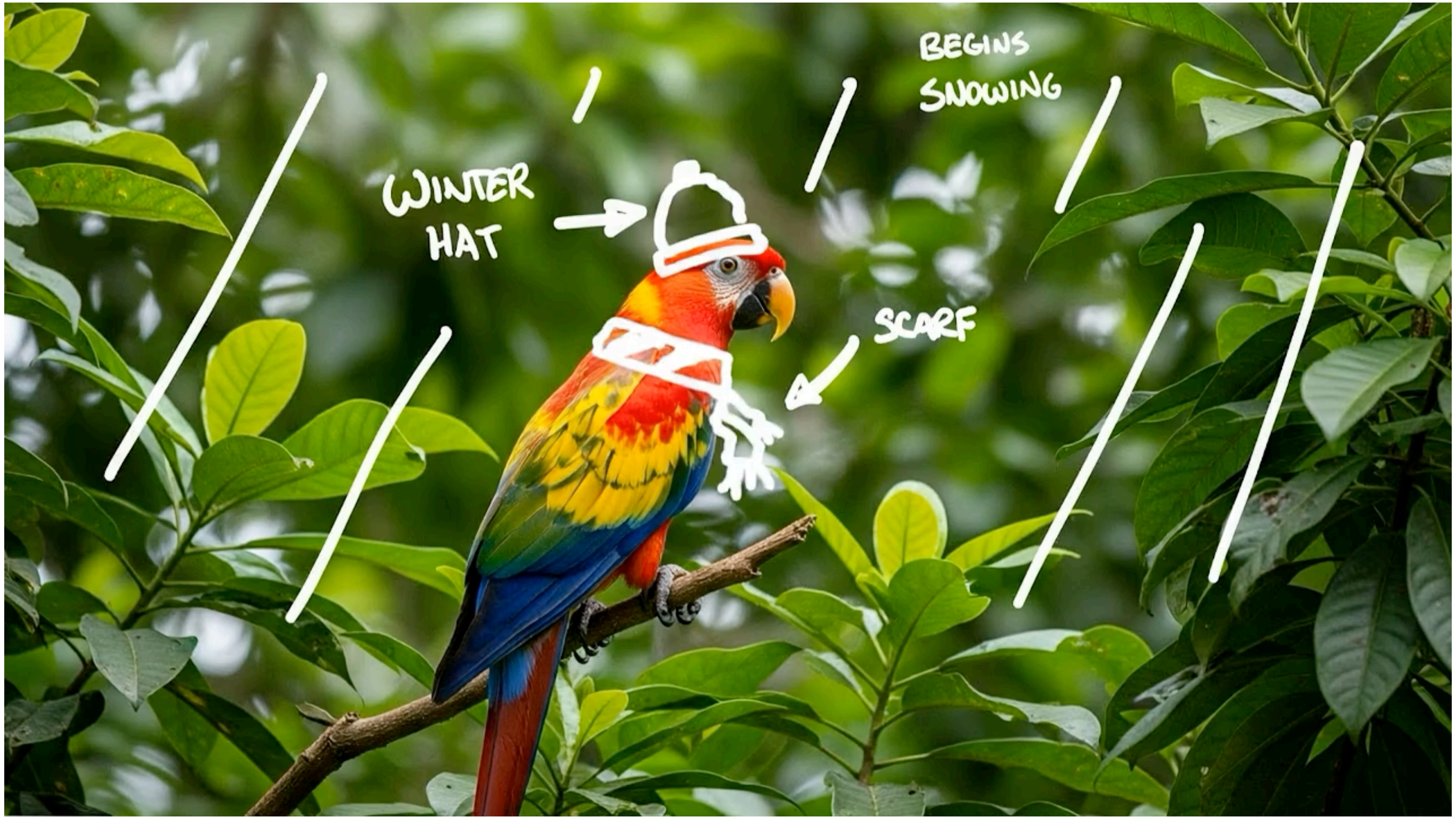
Figure 4 | **Class-agnostic instance segmentation** on a subset of 50 easy images (1-3 large objects) from LVIS [61]. Prompt: *"[...] each distinct entity is overlaid in a different flat color [...] the background fades to {white, green} [...]"* Details & full prompt: Sec. B.2.

From [Wiedemer et al., "Video models are zero-shot learners and reasoners", 2025]

Colorization

From [Wiedemer et al., “Video models are zero-shot learners and reasoners”, 2025]

"Doodle-based" editing



From [Wiedemer et al., "Video models are zero-shot learners and reasoners", 2025]

Visual Jenga

"A hand quickly removes each of the items in this image, one at a time."

From [Wiedemer et al., “Video models are zero-shot learners and reasoners”, 2025]
Task from [Bhattad, et al. “Visual Jenga: Discovering Object Dependencies via Counterfactual Inpainting”, 2025]

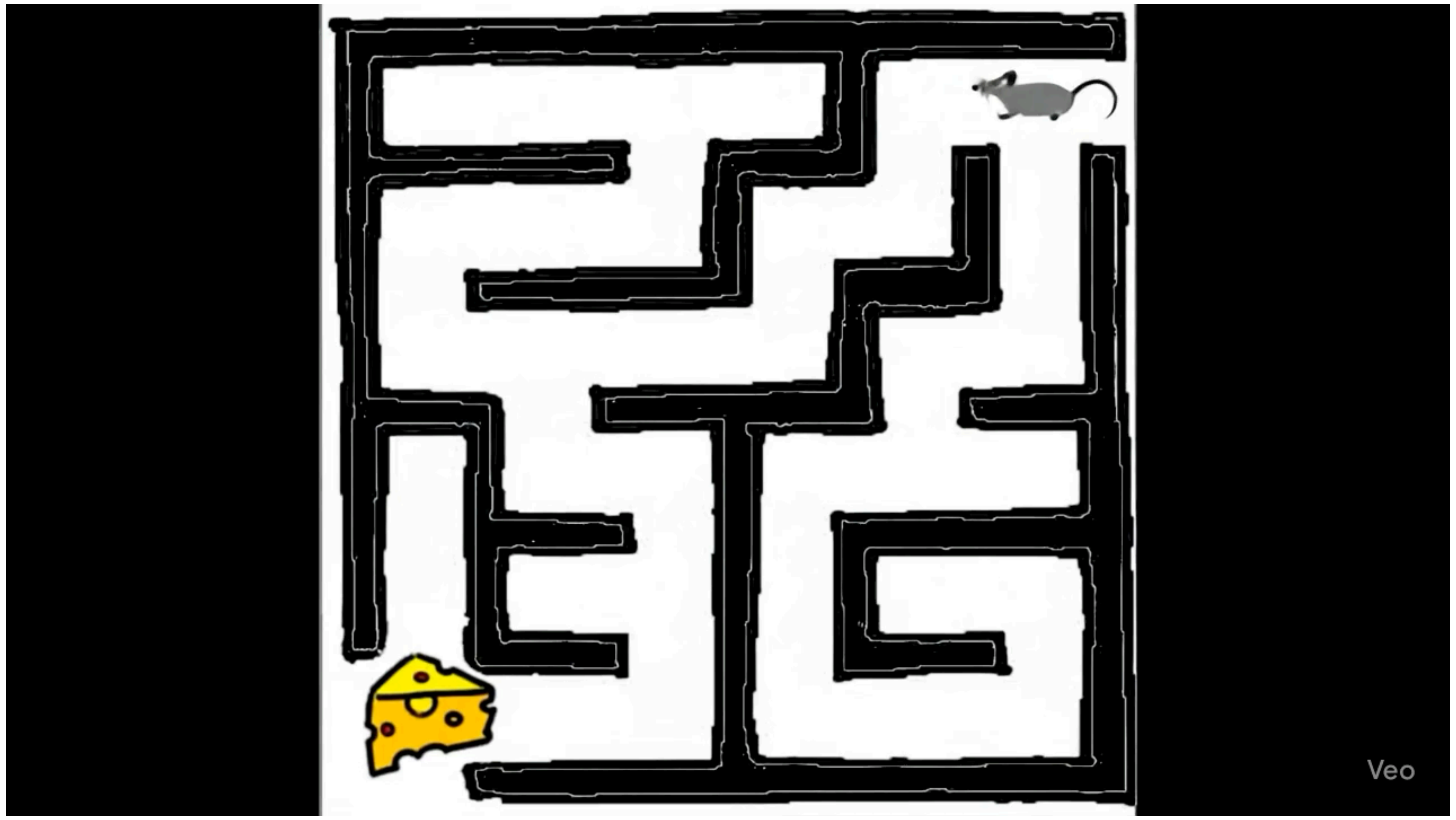
Intuitive physics

From [Wiedemer et al., "Video models are zero-shot learners and reasoners", 2025]

Intuitive physics

From [Wiedemer et al., “Video models are zero-shot learners and reasoners”, 2025]

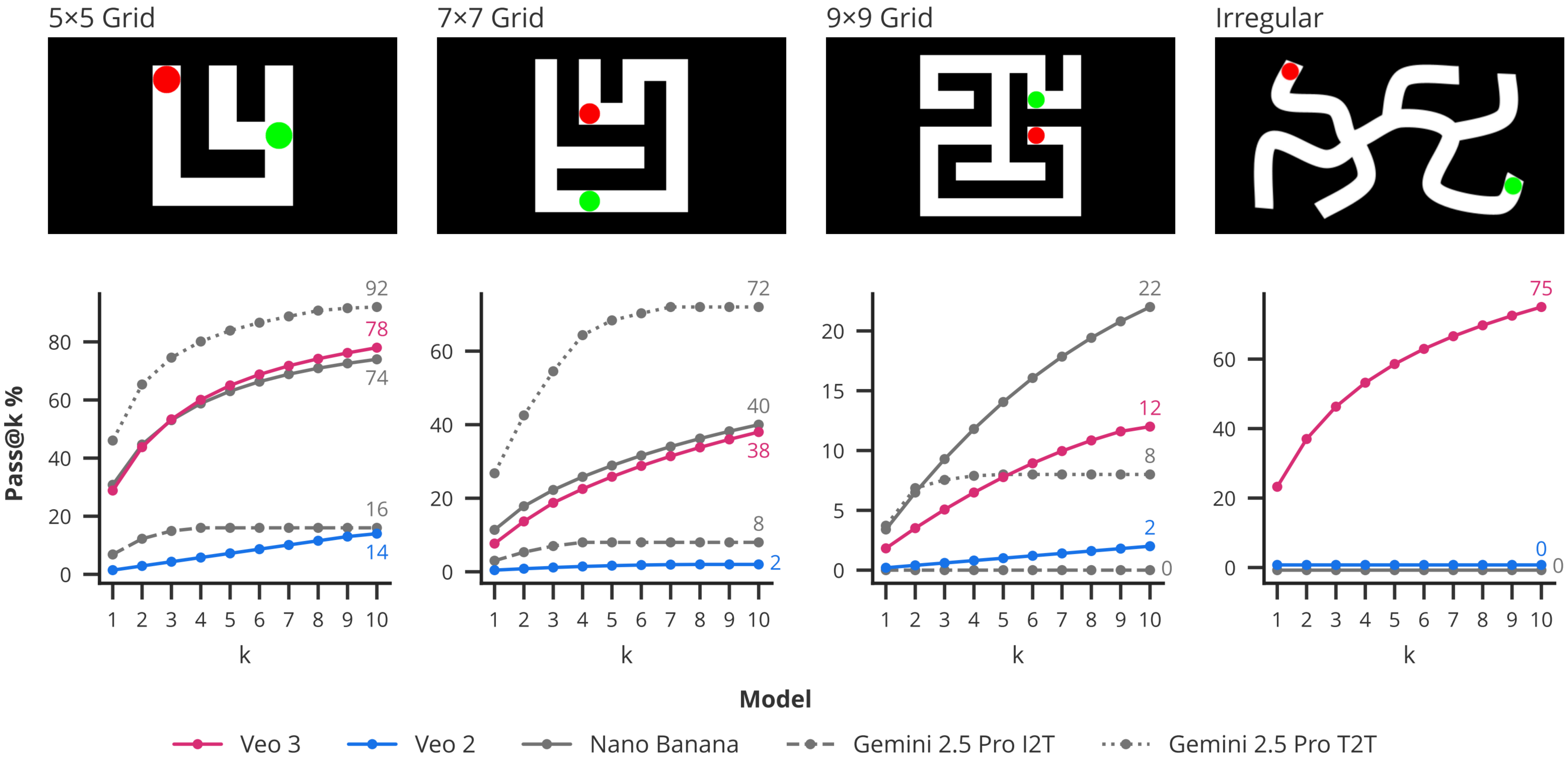
Solving mazes



"Without crossing any black boundary, the grey mouse from the corner skillfully navigates the maze by walking around until it finds the yellow cheese."

From [Wiedemer et al., "Video models are zero-shot learners and reasoners", 2025]

Accuracy with different models



Perceiving visual illusions

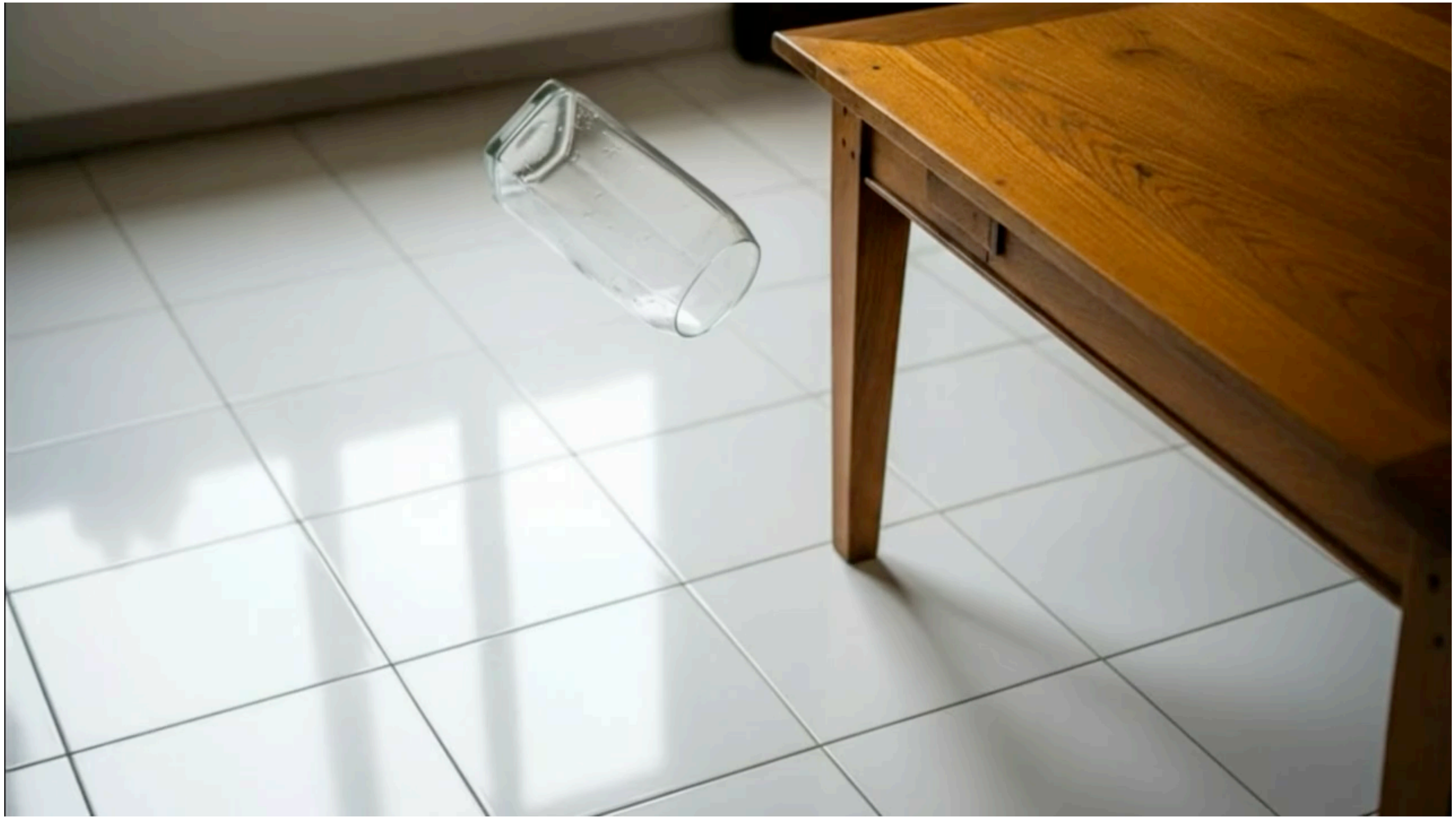
  

From [Wiedemer et al., "Video models are zero-shot learners and reasoners", 2025]

Failure cases

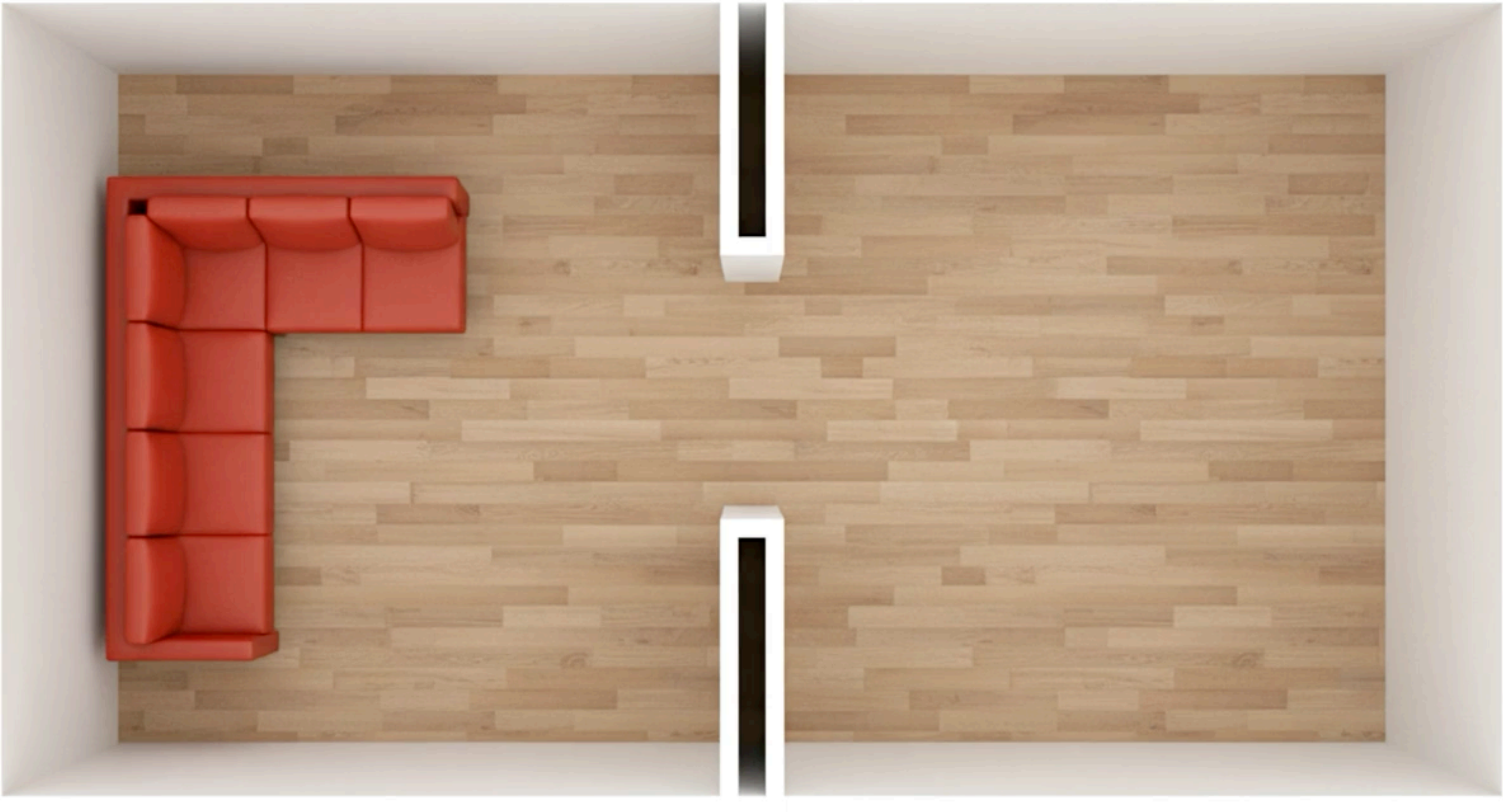
From [Wiedemer et al., “Video models are zero-shot learners and reasoners”, 2025]

Failure cases



From [Wiedemer et al., “Video models are zero-shot learners and reasoners”, 2025]

Failure cases



From [Wiedemer et al., “Video models are zero-shot learners and reasoners”, 2025]

Open-ended discussion

- Compare the challenges of training and using generative models in vision vs. NLP.
- Will large vision models solve everything?
- What (if any) the possible obstacles in the way?
- What else (if anything) do you think is needed?

Next class: review for final exam