Dynamic 3D Modeling

About Final Project

Reading, Implementing & Presenting
a Recent Paper in DIP or 3DV

How to Present a Paper / Work?
an Example.

[Submitted on 11 Nov 2024]

Edify 3D: Scalable High-Quality 3D Asset Generation

NVIDIA: Maciej Bala, Yin Cui, Yifan Ding, Yunhao Ge, Zekun Hao, Jon Hasselgren, Jacob Munkberg, Stella Shi, Fangyin Wei, Donglai Xiang, Jiashu Xu, Xiaohui Zeng, Qinsheng Z

We introduce Edify 3D, an advanced solution designed for high-quality 3D asset generation. Our meth then used to reconstruct the shape, texture, and PBR materials of the object. Our method can genera

Comments: Project website: this https URL
Subjects: **Computer Vision and Pattern Recognition (cs.CV)**; Artificial Intelligence (cs.AI); Graphics (cs.GR)
Cite as: arXiv:2411.07135 **[cs.CV]**
(or arXiv:2411.07135v1 **[cs.CV]** for this version)
https://doi.org/10.48550/arXiv.2411.07135

Submission history

From: Chen-Hsuan Lin [view email]
[v1] Mon, 11 Nov 2024 17:07:43 UTC (19,223 KB)

One, two, three

NVIDIA

Do What

Title
& / or
Abstract
& / or
Teaser

Figure 1: Edify 3D is a model designed for high-quality 3D asset generation. With input text prompts and/or a reference image, our model can generate a wide range of detailed 3D assets, supporting applications such as video game design, extended reality, simulation, and more.

Motivation (Why do this?)

Abstract &
Introduction
& / or
Related work

1. Introduction

The creation of detailed digital 3D assets is essential for developing scenes, characters, and environments across various digital domains. This capability is invaluable to industries such as video game design, extended reality, film production, and simulation. For 3D content to be production-ready, it must meet industry standards, including precise mesh structures, high-resolution textures, and material maps. Consequently, producing such high-quality 3D content is often an exceedingly complex and time-intensive process. As demand for 3D digital experiences grows, the need for efficient, scalable solutions in 3D asset creation becomes increasingly crucial.

Recently, many research works have investigated into training AI models for 3D asset generation (Lin et al., 2023). A significant challenge, however, is the limited availability of 3D assets suitable for model training. Creating 3D content requires specialized skills and expertise, making such assets much scarcer than other visual media like images and videos. This scarcity raises a key research question of how to design scalable models to generate high-quality 3D assets from such data efficiently.

Motivation (Why do this?)

一个矛盾

xxx很重要， 但是当前方法解决不了，因为xxx挑战

Motivation (Why do this?)

Limited
3D Assets

大量手工操作
专业技能
没有足够多的高质量数据

Key Idea / Pipeline (How to solve this?)

Introduction

&

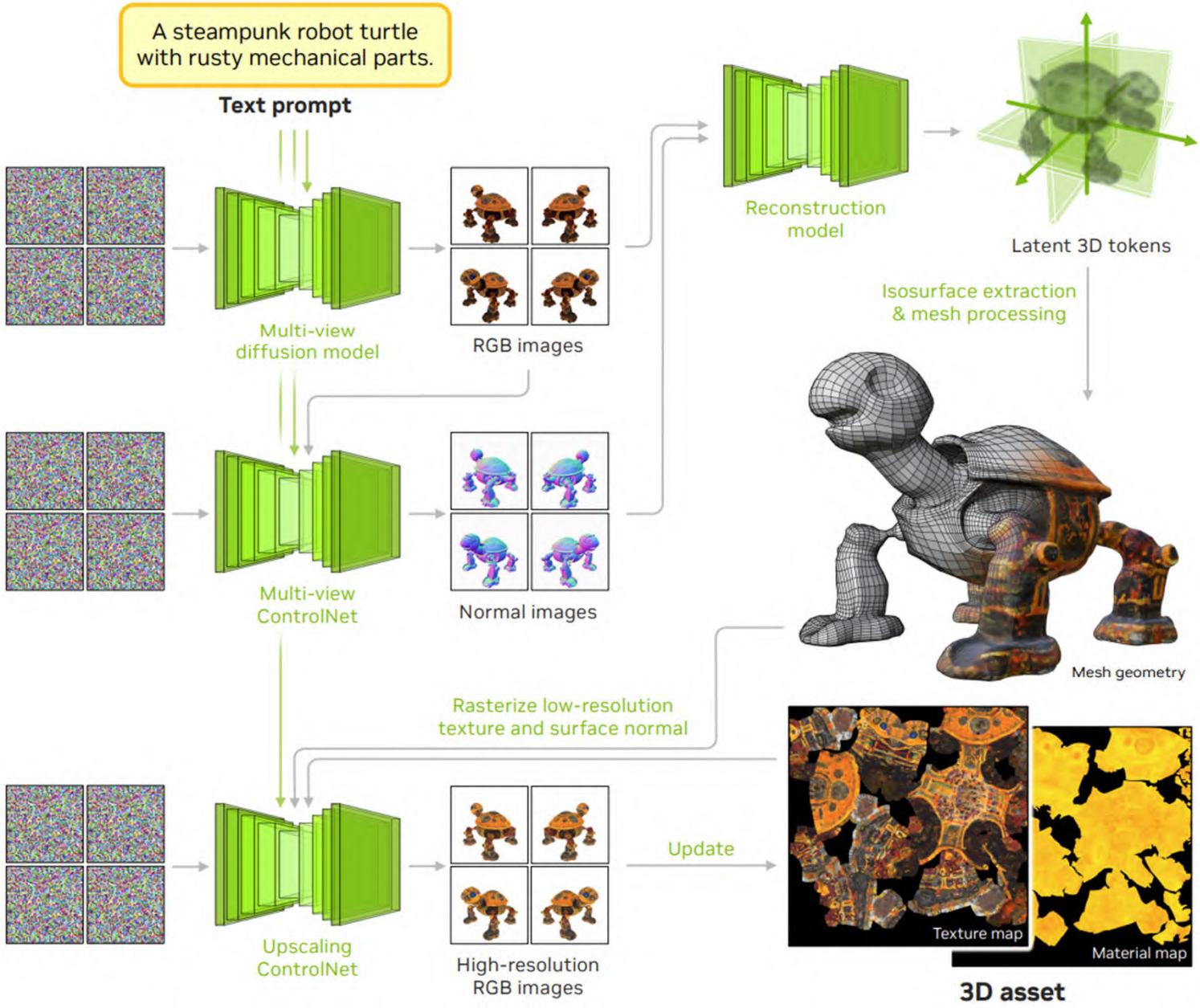
Pipeline Figure

Key Idea / Pipeline (How to solve this?)

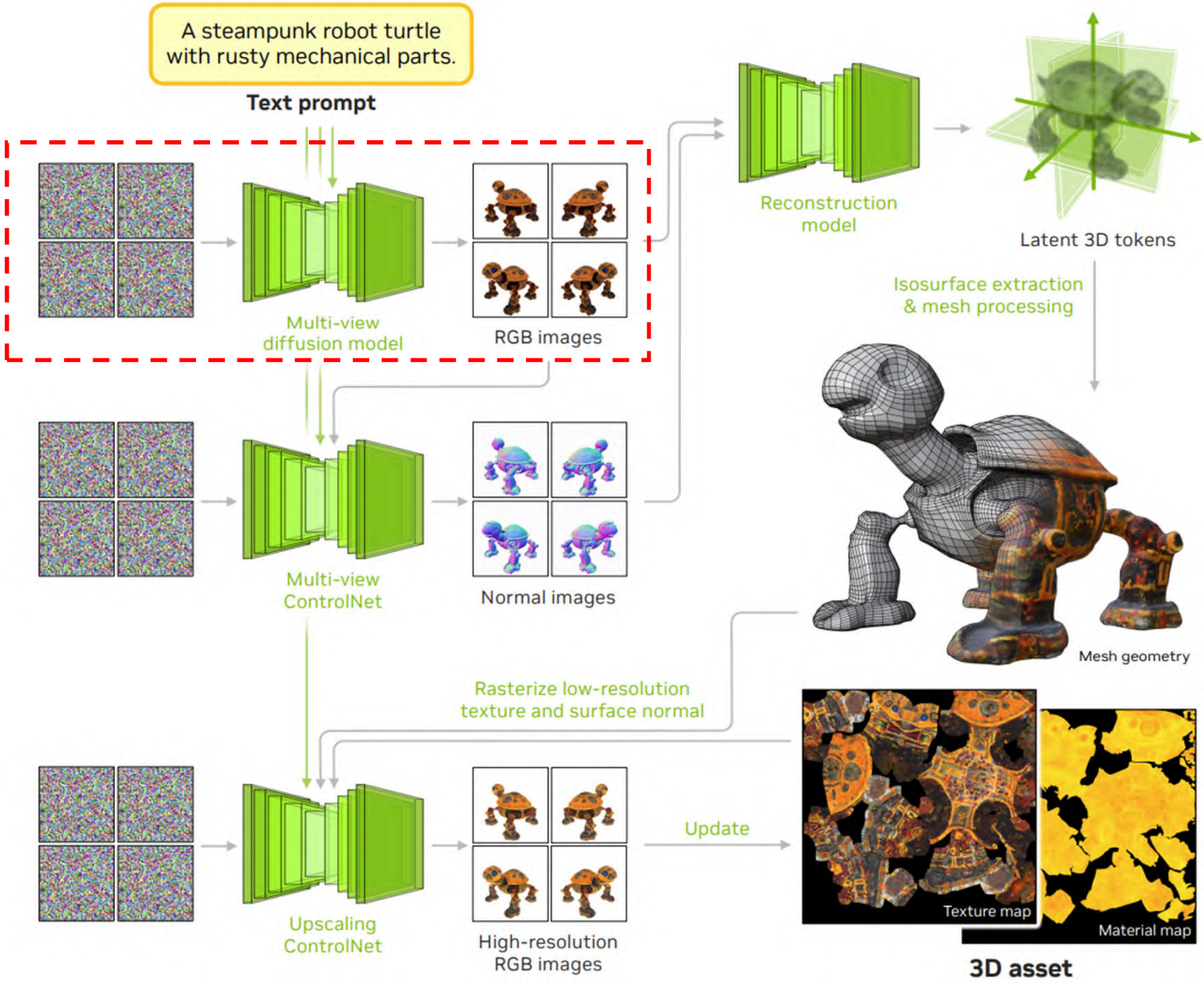
Model design. The core technology of Edify 3D relies on two types of neural networks: diffusion models (Ho et al., 2020; Song and Ermon, 2019) and Transformers (Vaswani et al., 2017). Both architectures have demonstrated great scalability and success in improving generation quality as more training data becomes available. Following Li et al. (2024), we train the following models:

- **Multi-view diffusion models**. We train multiple diffusion models to synthesize the RGB appearance and surface normals of an object from multiple viewpoints (Shi et al., 2023). The input can be a text prompt, a reference image, or both.
- **Reconstruction model**. Using the synthesized multi-view RGB and surface normal images, a reconstruction model predicts the geometry, texture, and materials of the 3D shape. We employ a Transformer-based model (Hong et al., 2023) to predict a neural representation of the 3D object as latent tokens, followed by isosurface extraction and mesh processing.

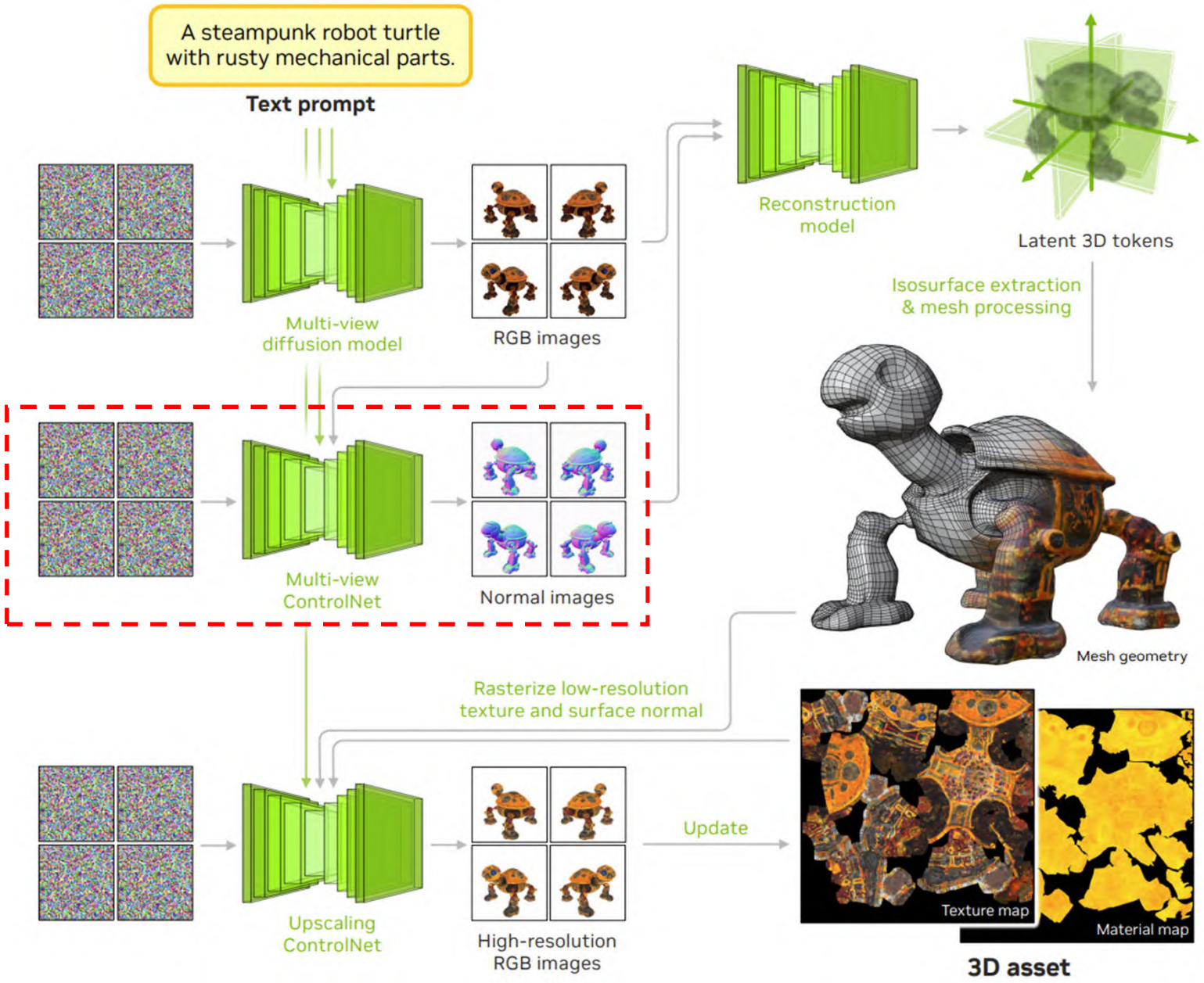
The final output of Edify 3D is a 3D asset that includes the mesh geometry, texture map, and material map. Fig. 2 illustrates the overall pipeline of Edify 3D.

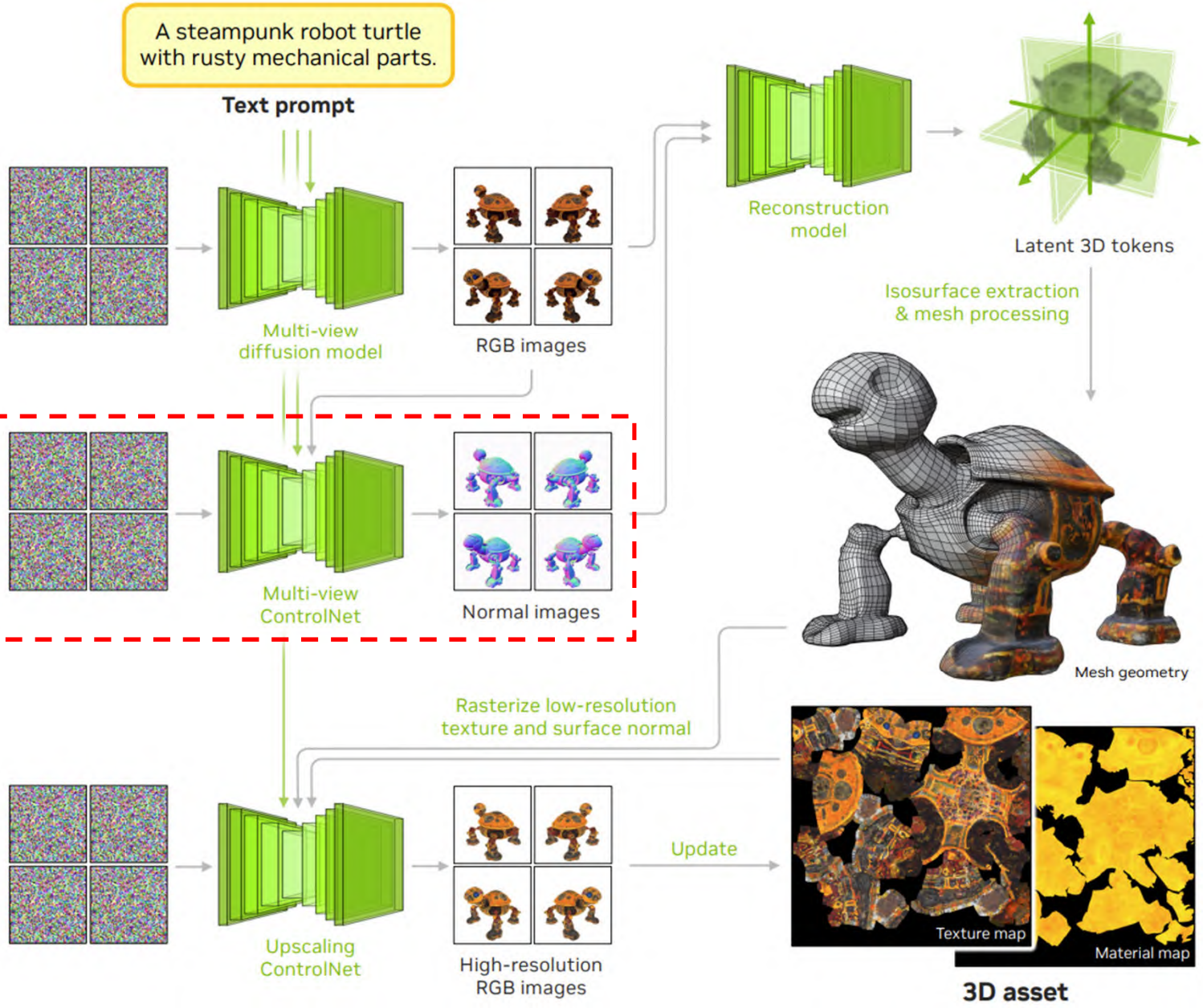

A steampunk robot turtle with rusty mechanical parts.
Text prompt
Multi-view diffusion model
RGB images
Reconstruction model
Latent 3D tokens
Isosurface extraction & mesh processing
Multi-view ControlNet
Normal images
Mesh geometry
Rasterize low-resolution texture and surface normal
Upscaling ControlNet
High-resolution RGB images
Update
Texture map
Material map
3D asset

一个用于生成多视角图片的Diffusion Model



一个用于生成对应Normal Map的ControlNet

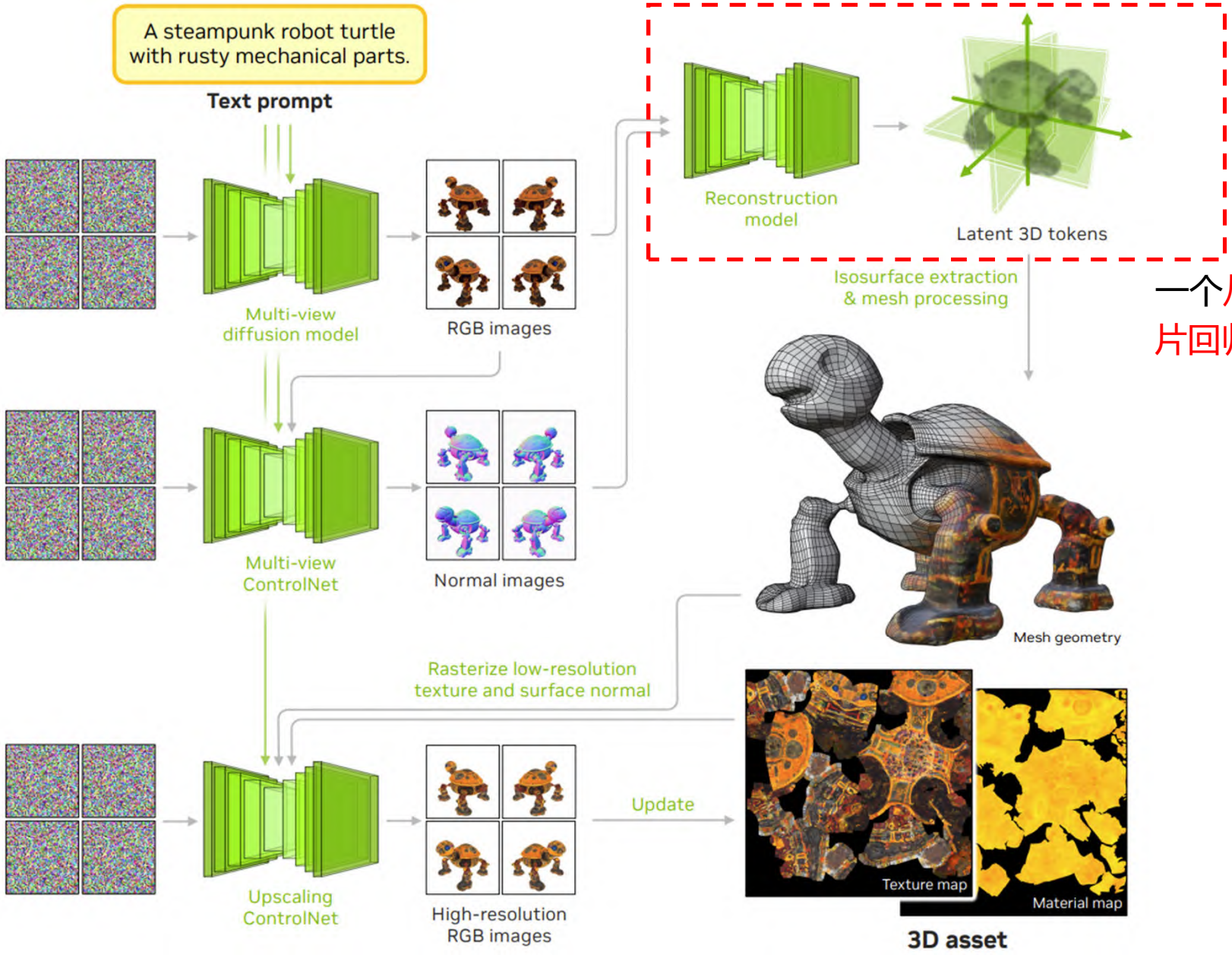




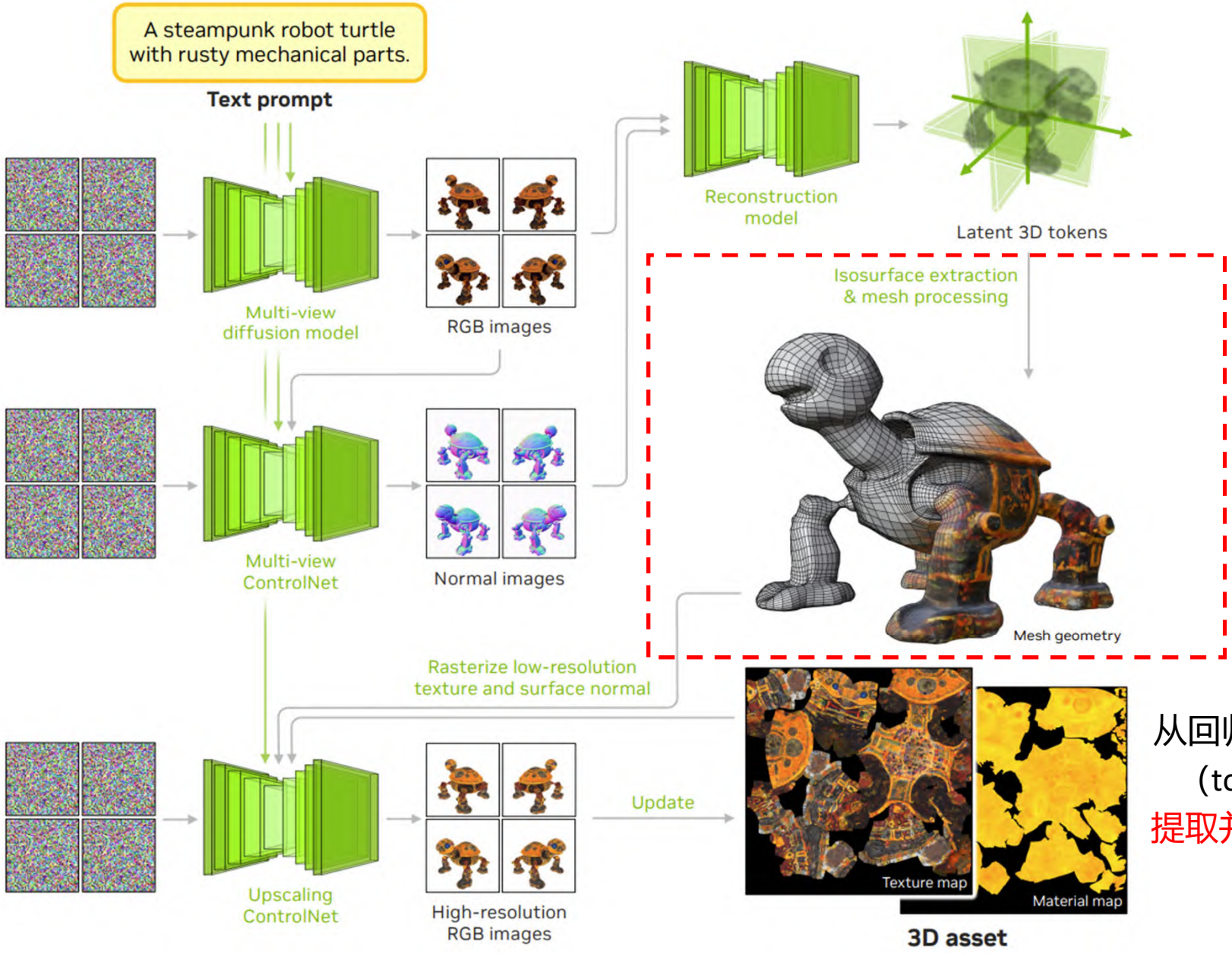
一个用于生成对应Normal Map的ControlNet

什么是ControlNet?

Need to read other papers.

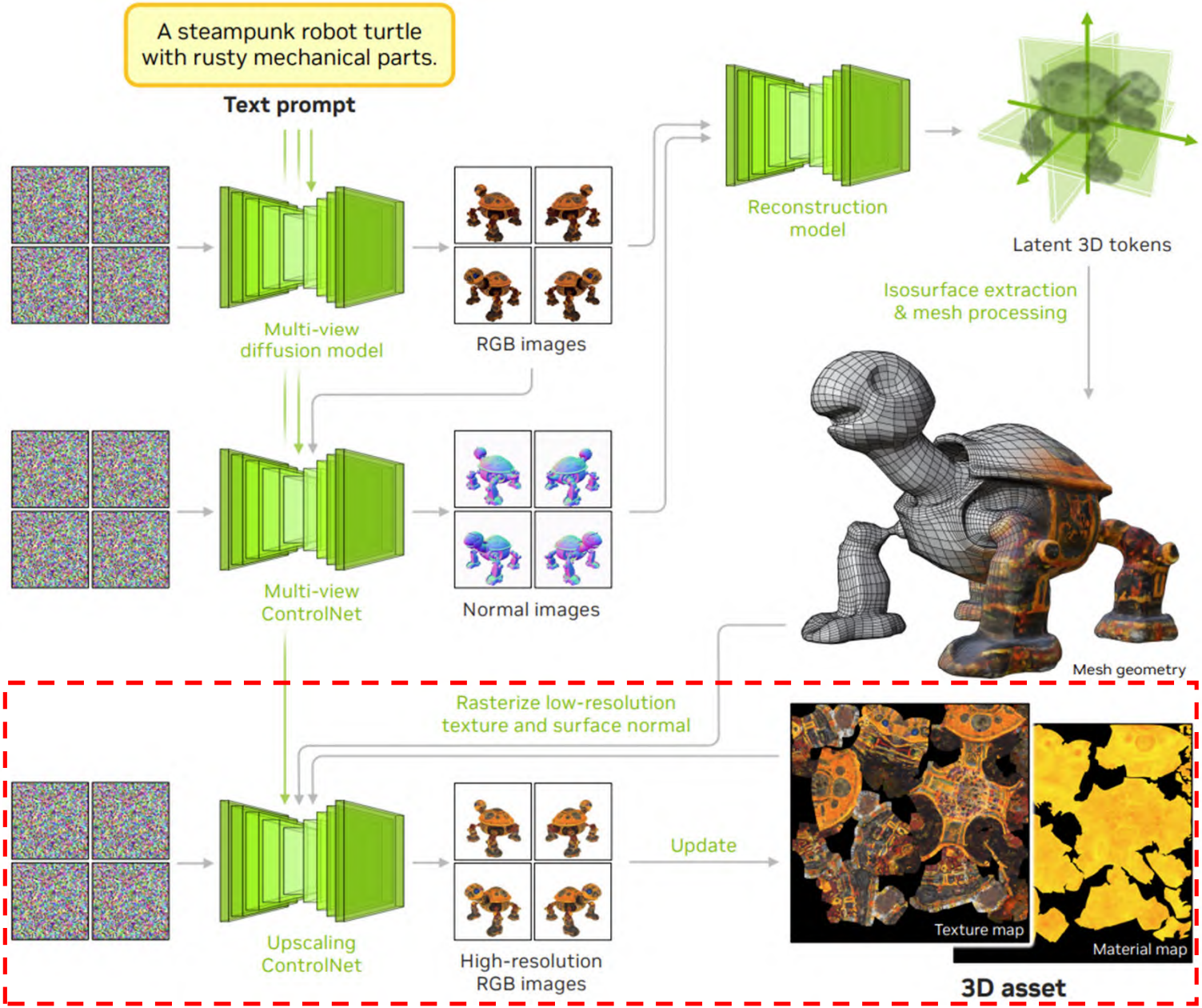


一个从多视角图片回归3D的重建网络



从回归的3D表达（tokens）中提取并优化Mesh

用ControlNet生成
超分辨率图片
来优化纹理贴图



Problems & Thinkings

We know diffusion models could generate images.

But how to generate Multi-view Images?

Details

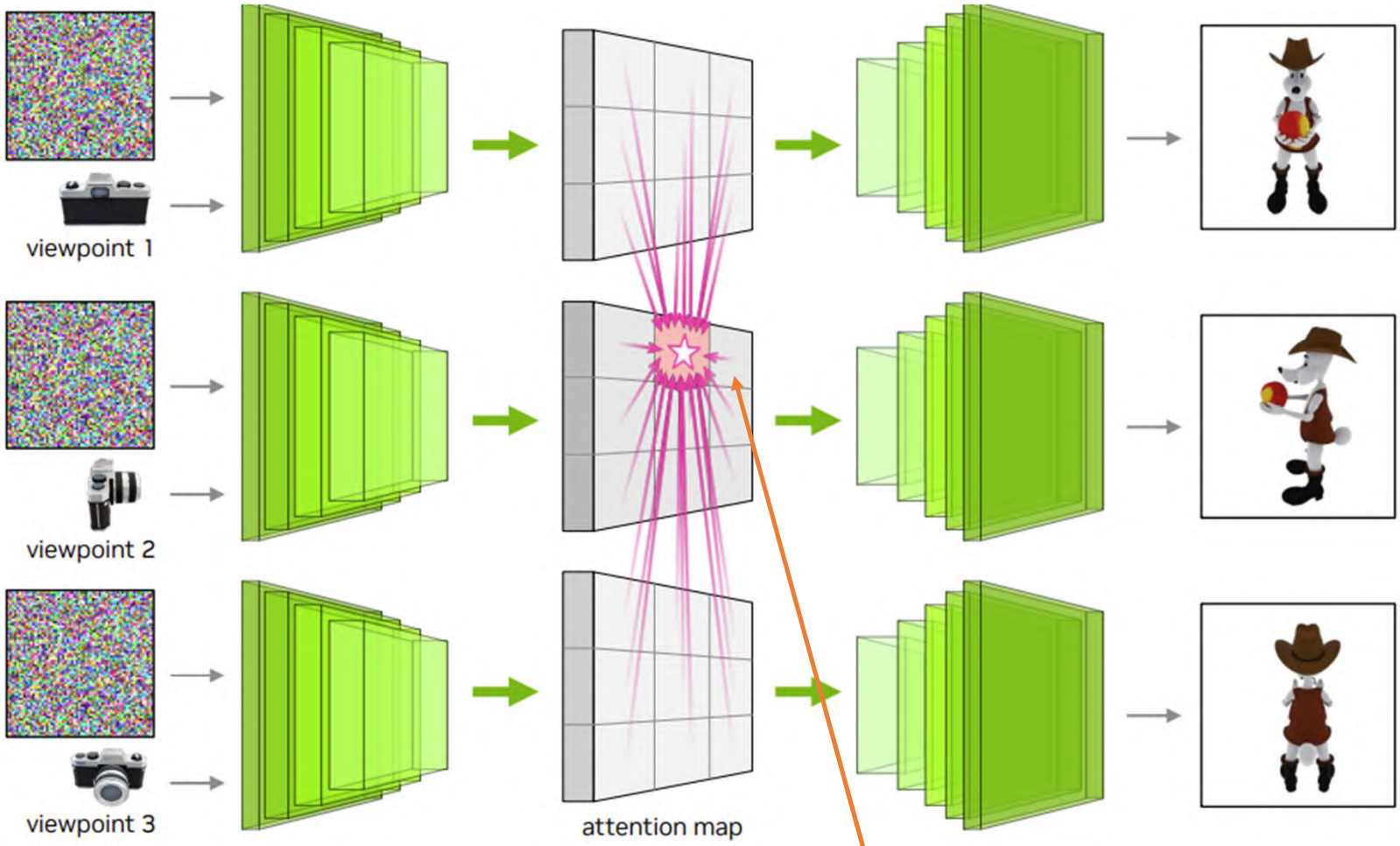


Figure 3: **Cross-view attention.** In standard diffusion models, each view is synthesized by the diffusion model independently. We extend the self-attention layer (yellow boxes) in our multi-view diffusion models to attend across other viewpoints using the same weights.

Problems & Thinkings

Why first generate multi-view images?
Why not direct generate 3D?

Why predict Normal Maps?

What are the 3D tokens?

Discover by yourself !

About Implementation.

Look at Implementation Details

Training. We finetune the text-to-image models on renderings of 3D objects. During training, we jointly train on natural 2D images as well as 3D object renderings with randomly chosen numbers of views (1, 4, and 8). The diffusion models are trained using the $\mathbf{x}_0$ parametrization for the loss, consistent with the approach used in base model training. For multi-view ControlNets, we train the base model with multi-view surface normal images first. Subsequently, we add a ControlNet encoder taking RGB images as input and train it while freezing the base model.

About Implementation.

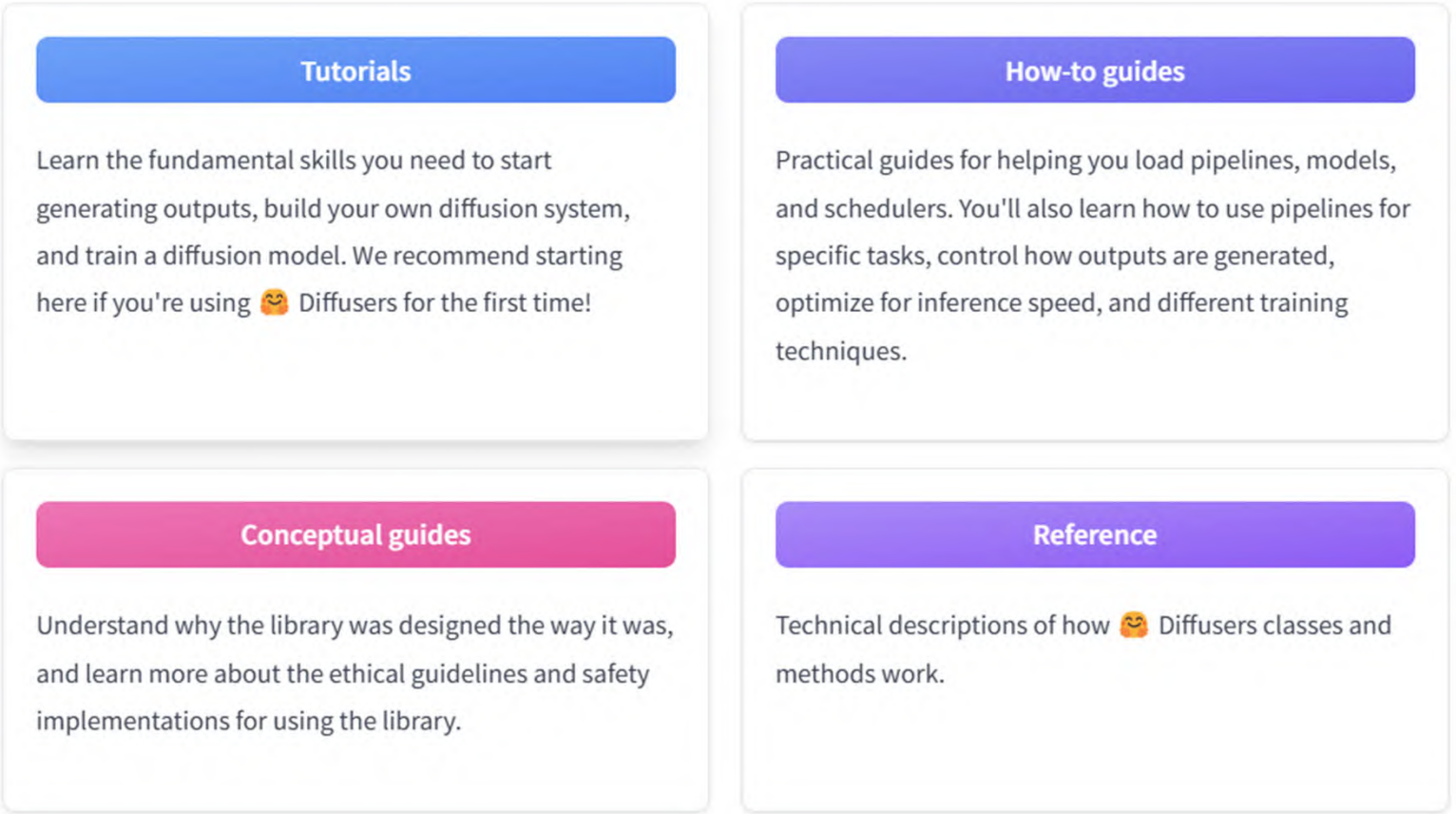
🤗 Diffusers is the go-to library for state-of-the-art pretrained diffusion models for generating images, audio, and even 3D structures of molecules. Whether you're looking for a simple inference solution or want to train your own diffusion model, 🤗 Diffusers is a modular toolbox that supports both. Our library is designed with a focus on usability over performance, simple over easy, and customizability over abstractions.

The library has three main components:

- State-of-the-art diffusion pipelines for inference with just a few lines of code. There are many pipelines in 🤗 Diffusers, check out the table in the pipeline overview for a complete list of available pipelines and the task they solve.
- Interchangeable noise schedulers for balancing trade-offs between generation speed and quality.
- Pretrained models that can be used as building blocks, and combined with schedulers, for creating your own end-to-end diffusion systems.

https://huggingface.co/docs/diffusers/en/index

About Implementation.



https://huggingface.co/docs/diffusers/en/index

Demo Show

About Final Project

I will recommend a list of recent papers.

1-3 members collaboration, choose one of the paper.

Details at Thursday, 11.28

Review Previous 3D Modeling

Recon

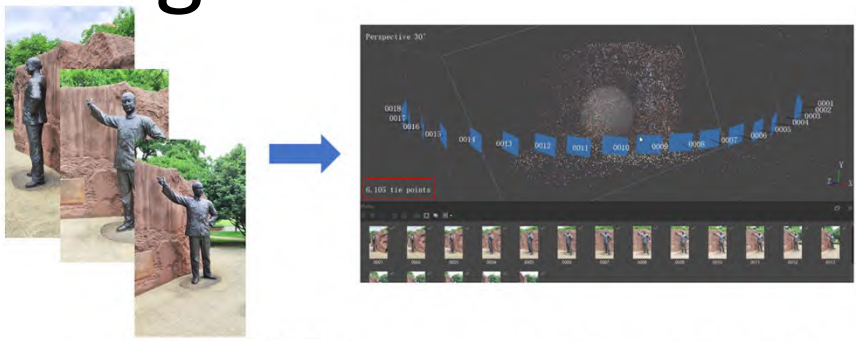
3D Surface

Light & Reflectance

Review Previous 3D Modeling

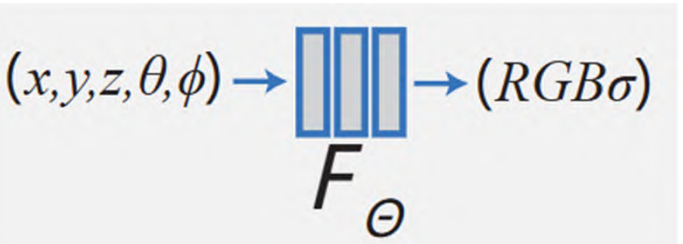
Static Scenes

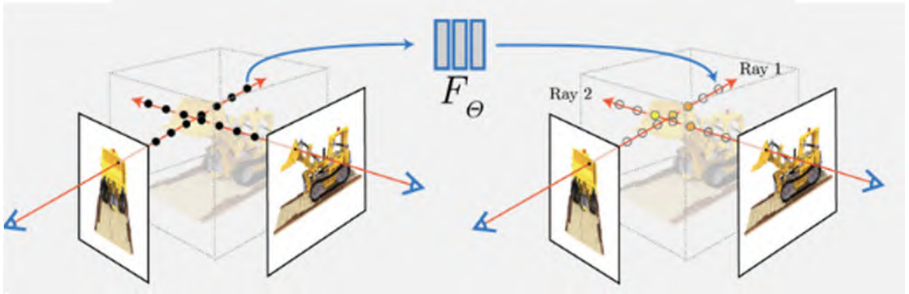
$F^{-1}(2D) = 3D$



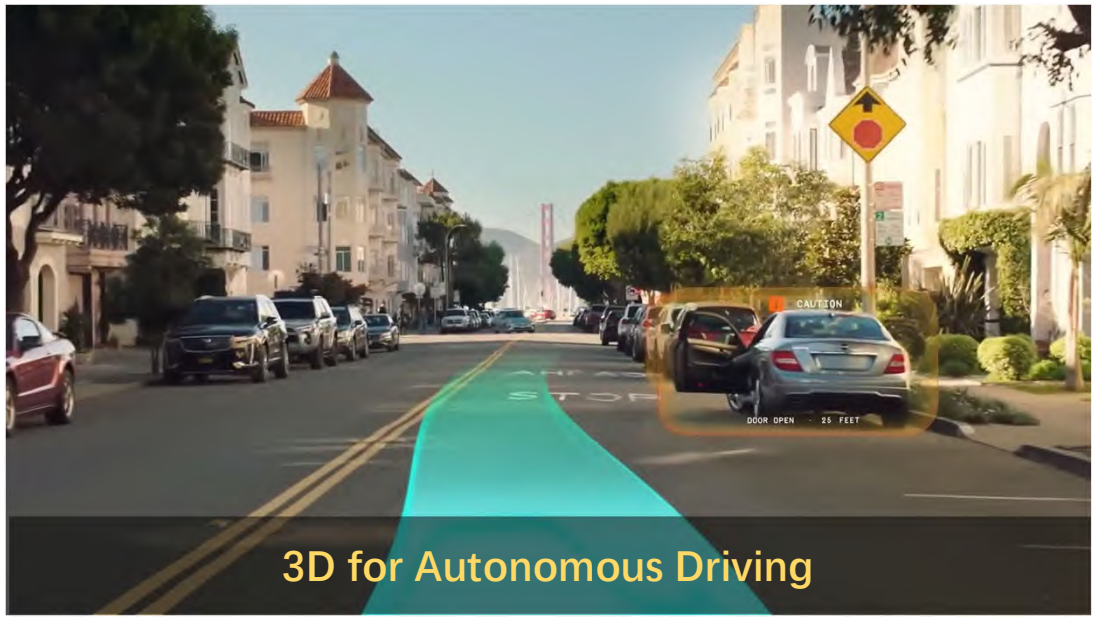
$$\min_{\{\mathbf{R}_k,\mathbf{t}_k\},\mathbf{K}} \sum_i \sum_k \|\mathbf{x}_{ik} - \pi_{\mathbf{K}}([\mathbf{R}_k|\mathbf{t}_k]\mathbf{X}_i)\|_2^2$$

Extrinsic Intrinsic Detection Corner Points Perspective Projection Known 3D Location





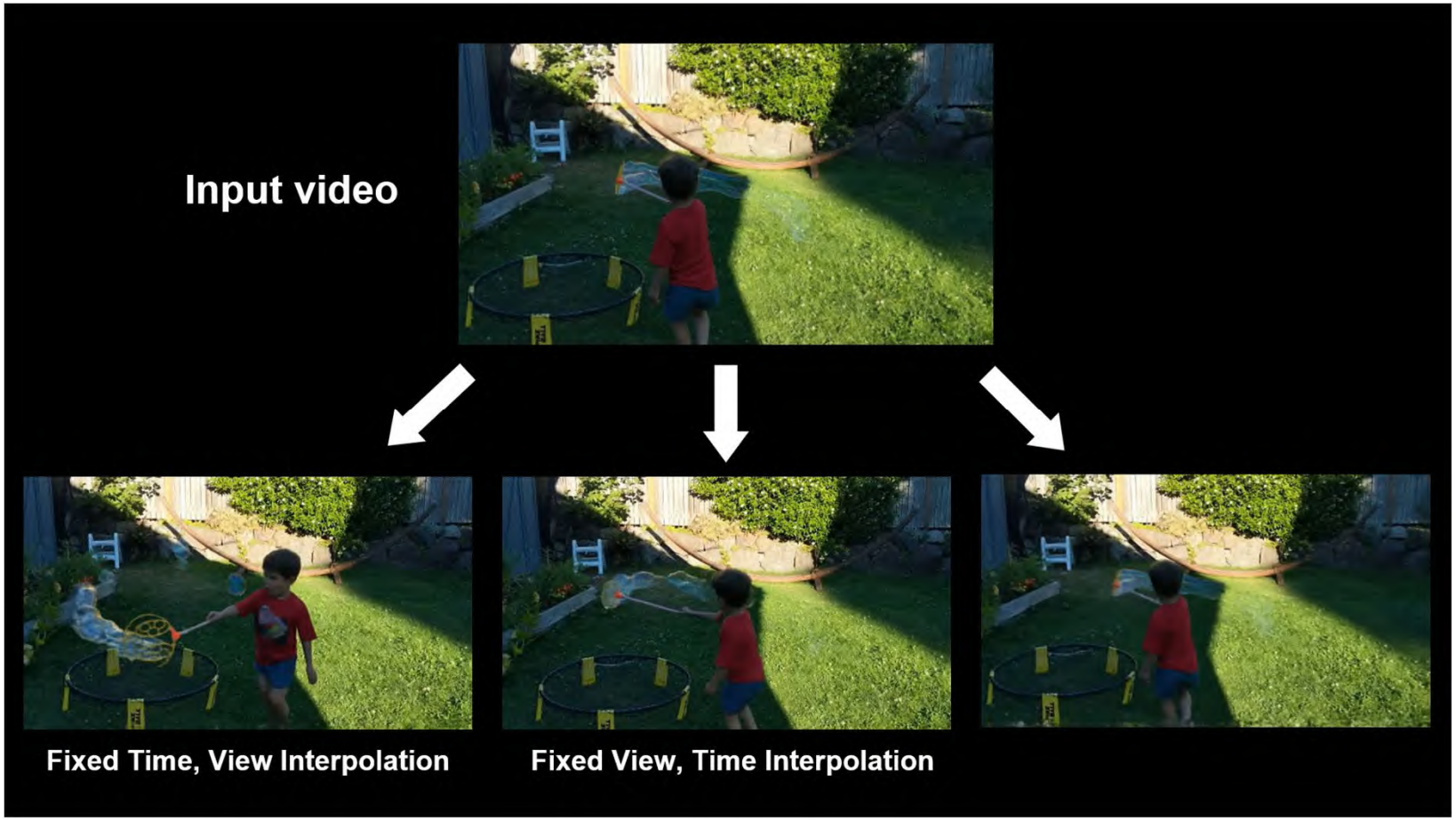
We live in a Dynamic World



3D for Autonomous Driving

3D for Interacting with the World

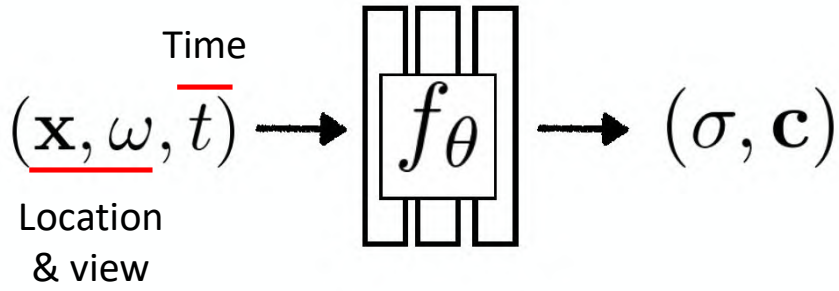
Towards Dynamic World Modeling



3D + time
or 4D

Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes. CVPR 2021.

Towards Dynamic World Modeling

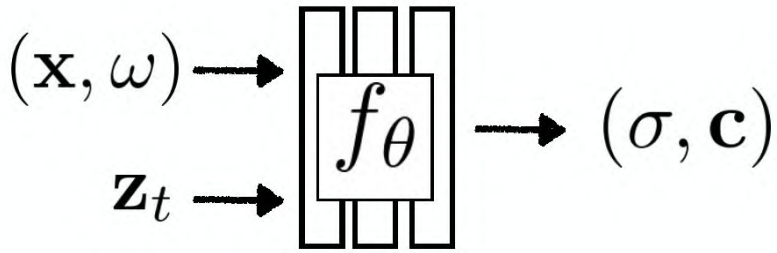


Idea 1: Just add time as an additional variable!

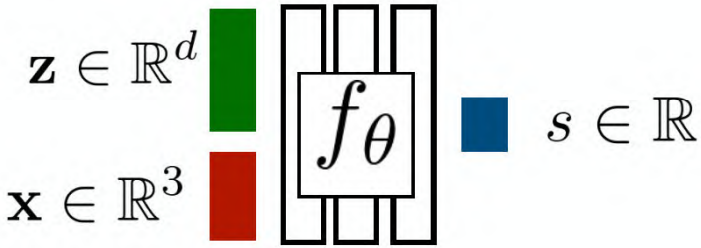
Temporally far does not imply structurally far

Credit: Shubham Tulsiani

Towards Dynamic World Modeling



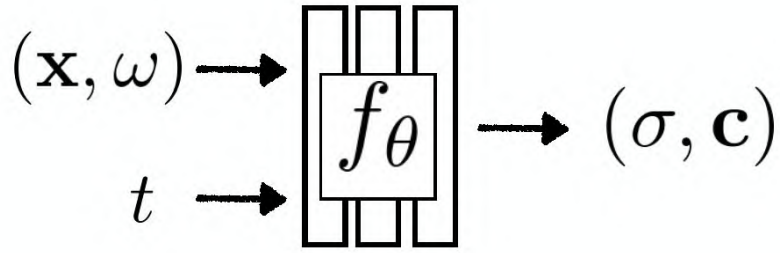
Idea 1 (alternative implementation): Time-based latent variable



Generalizable Signed Distance Field:
(latent code, position) → (distance)

$$f_\theta : \mathbb{R}^3 \times \mathbb{R}^d \to \mathbb{R}$$

Towards Dynamic World Modeling

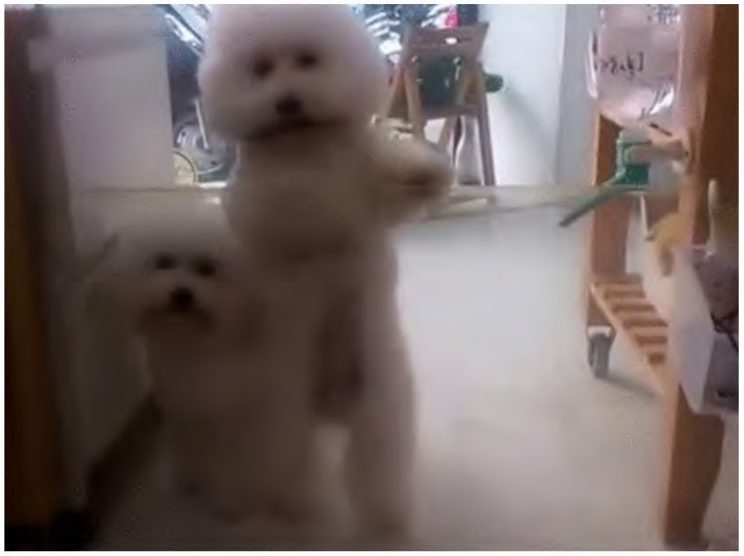


Idea 1: Time as additional input — f($\cdot, \cdot,$ **t**) is the scene at time **t**

Would typically require multi-view data at each instance

No explicit constraints that f($\cdot, \cdot,$ **t**) and f($\cdot, \cdot,$ **t'**)
represent a common scene

But there are static constraints in a dynamic scene.

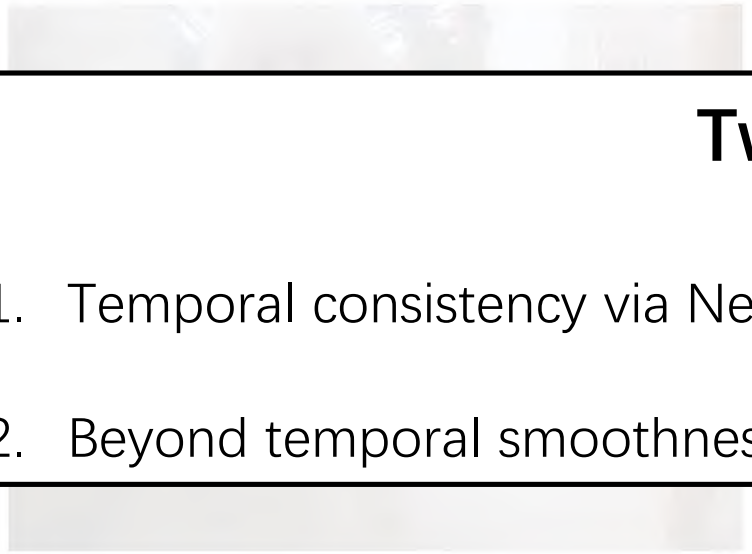


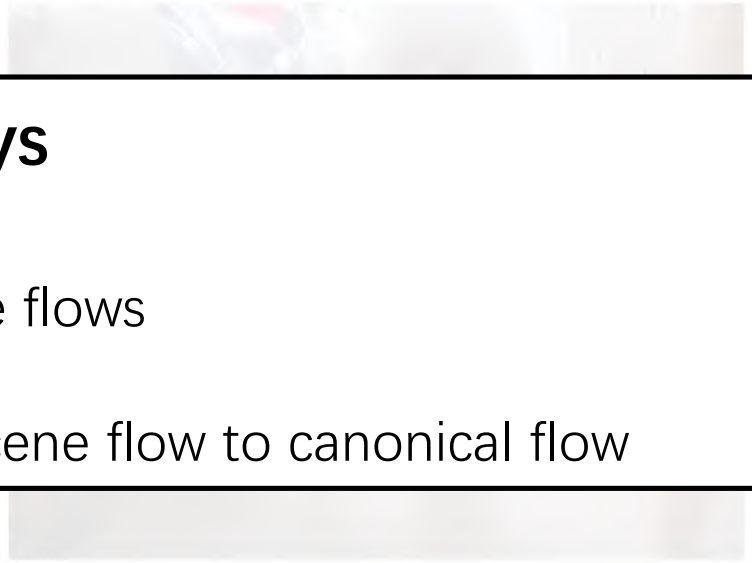
t=0

t=100

The stuff at t=0 is the ‘same’ is t=100, but possibly at a different location

But there are static constraints in a dynamic scene.





t=0

t=100

The stuff at t=0 is the 'same' is t=100, but possibly at a different location

Two Ways

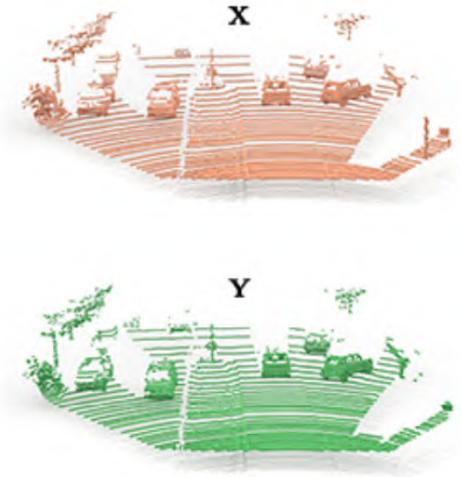
1. Temporal consistency via Neural scene flows
2. Beyond temporal smoothness: from scene flow to canonical flow

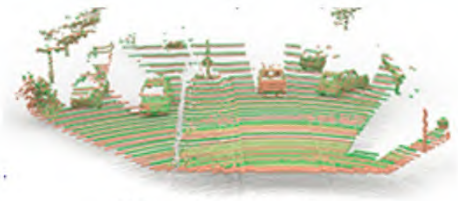
Correspondence via Flow Fields

Optical Flow

Given a pixel in frame 1, where does it go to in frame 2?

$\mathbb{R}^2 \rightarrow \mathbb{R}^2$ A 2D flow field



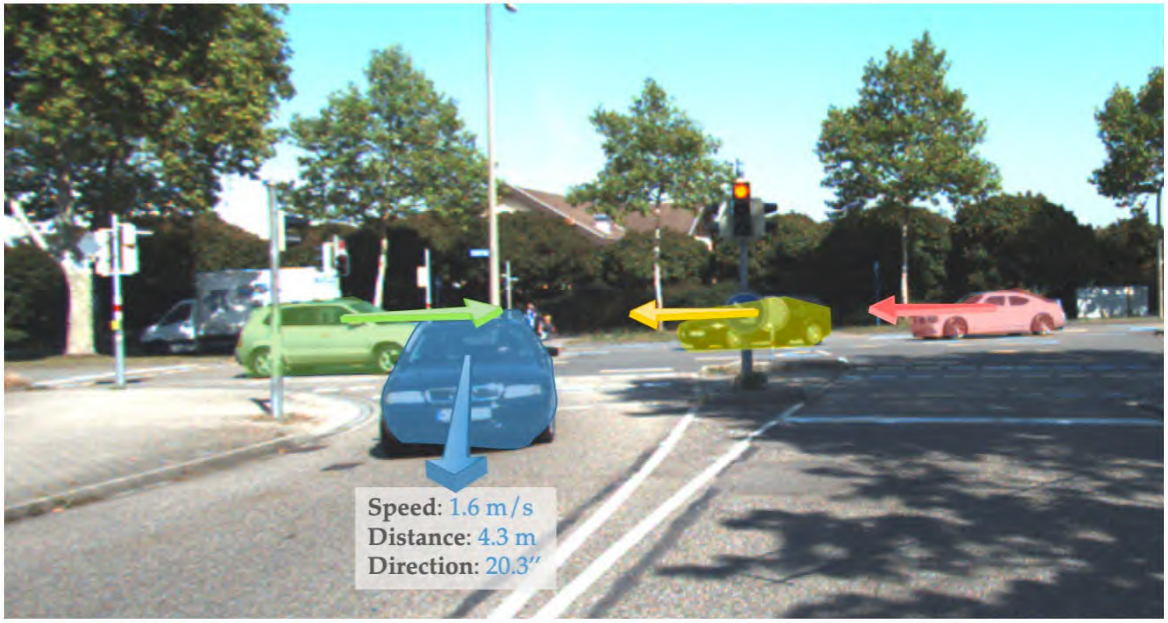


Per-point scene flow

Given a 3D point at time t1, where does it go at time t2?

$\mathbb{R}^3 \rightarrow \mathbb{R}^3$ A 3D flow field

3D Scene Flow



Observation: 3D flow typically
comprised of piecewise rigid motions

Image Courtesy: Zhe Cao

Parameterizing Scene Flow Fields

For an object rotating around its centre:

$$\mathbf{x}_{t+1} = R(\mathbf{x}_t - \bar{\mathbf{x}}) + \bar{\mathbf{x}}$$

Option 1: Predict a per-point translation

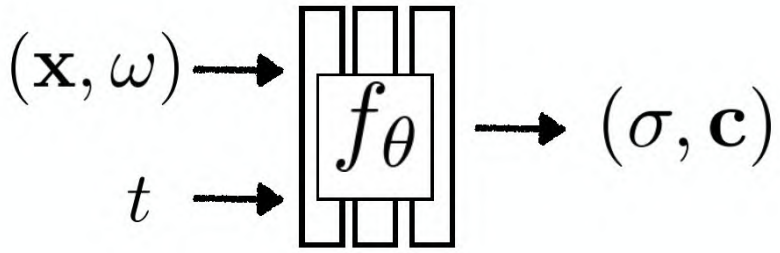
(requires different predictions for different points)

Option 2: Predict a per-point translation **and** rotation

(allows **same** prediction for **all** points within a rigid object)

Image Courtesy: Zhe Cao

Towards Dynamic World Modeling



Can we add explicit constraints regarding scene persistence?

Neural Scene Flow Fields

$$(\mathbf{c}_i, \sigma_i, \mathcal{F}_i, \mathcal{W}_i) = F_\Theta^{\text{dy}}(\mathbf{x}, \mathbf{d}, i)$$

Basically, a time-dependent NeRF representation (***i*** is indexing time)

Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes. *Li et. al.*

Neural Scene Flow Fields

$$(\mathbf{c}_i, \sigma_i, \mathcal{F}_i, \mathcal{W}_i) = F_\Theta^{\text{dy}}(\mathbf{x}, \mathbf{d}, i)$$

$\mathcal{F}_i$ indicates the 3D flow to next/ previous frame for point **x**

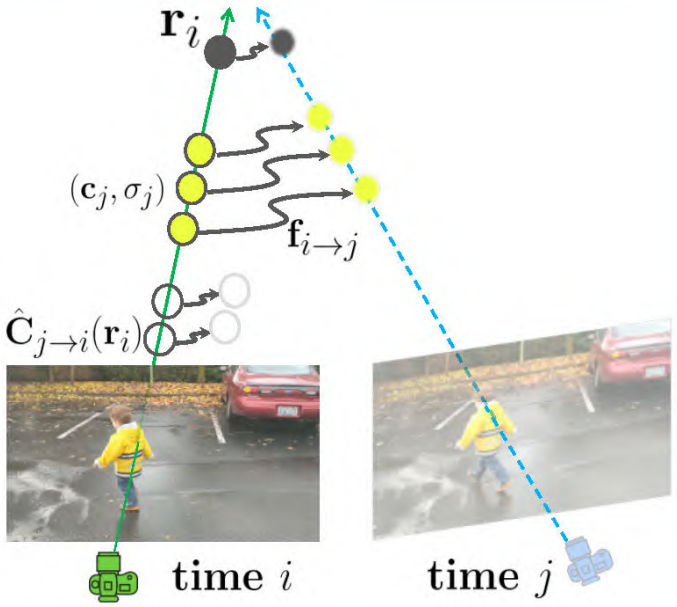
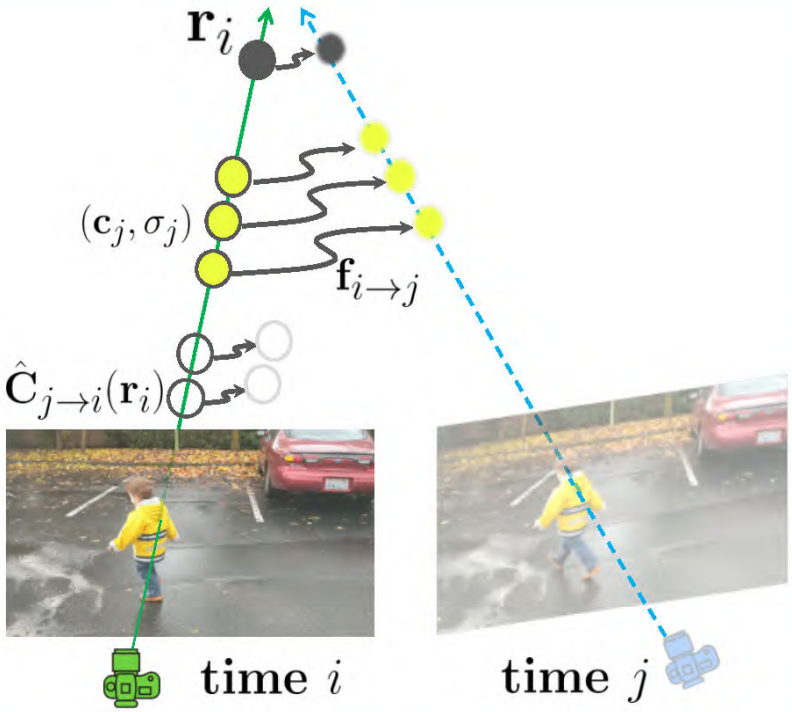


Image Courtesy: Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes. *Li et. al.*

Enforcing Temporal Persistence



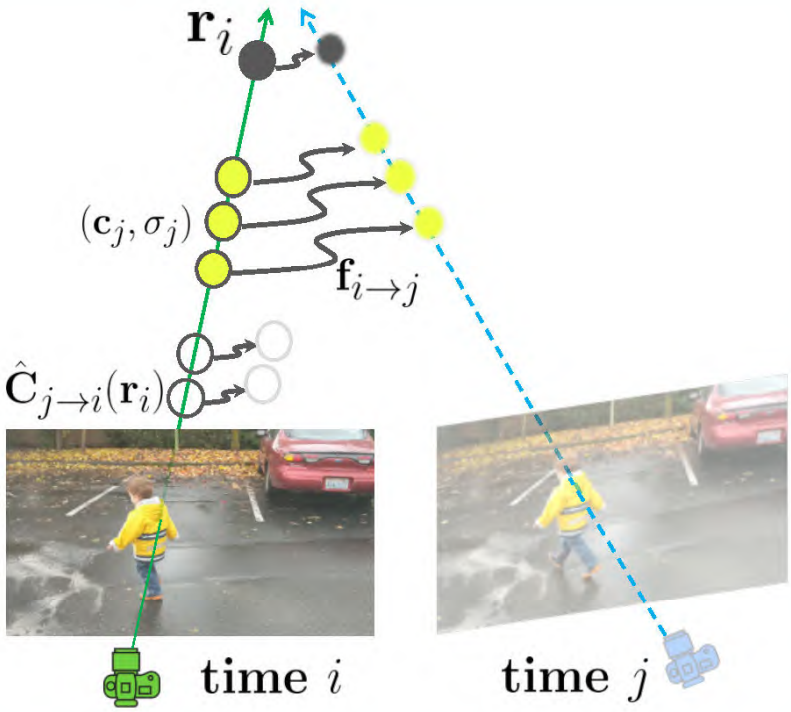
$$\hat{\mathbf{C}}_{j \to i}(\mathbf{r}_i) = \int_{t_n}^{t_f} T_j(t)\, \sigma_j(\mathbf{r}_{i \to j}(t))\, \mathbf{c}_j(\mathbf{r}_{i \to j}(t), \mathbf{d}_i) dt$$

$$\text{where } \mathbf{r}_{i \to j}(t) = \mathbf{r}_i(t) + \mathbf{f}_{i \to j}(\mathbf{r}_i(t)).$$

Rendering for a ray ***r*** at time ***i*** using the NeRF for time ***j***

Image Courtesy: Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes. *Li et. al.*

Enforcing Temporal Persistence



$$\hat{\mathbf{C}}_{j \to i}(\mathbf{r}_i) = \int_{t_n}^{t_f} T_j(t)\, \sigma_j(\mathbf{r}_{i \to j}(t))\, \mathbf{c}_j(\mathbf{r}_{i \to j}(t), \mathbf{d}_i) dt$$

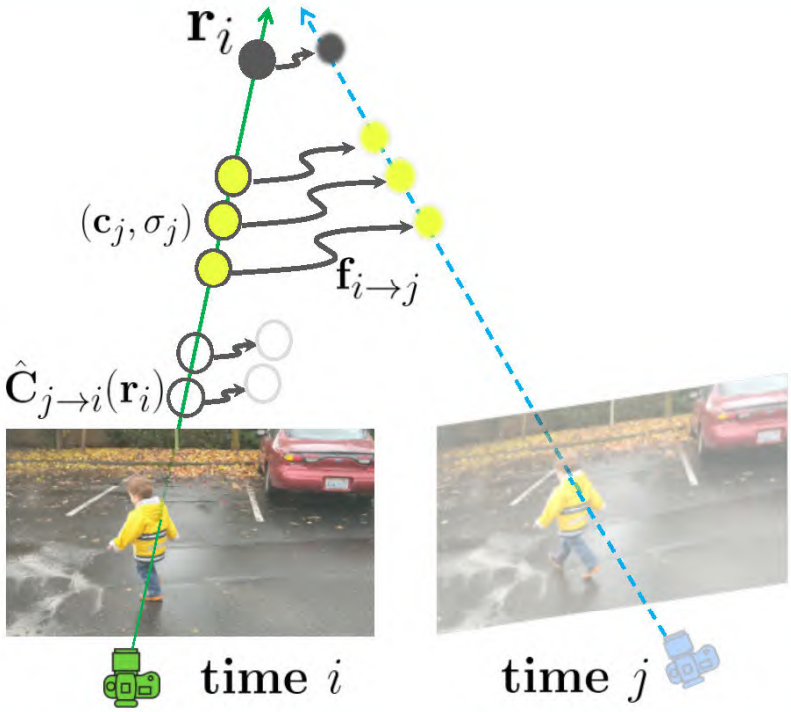
$$\text{where } \mathbf{r}_{i \to j}(t) = \mathbf{r}_i(t) + \mathbf{f}_{i \to j}(\mathbf{r}_i(t)).$$

$$\mathcal{L}_{\text{pho}} = \sum_{\mathbf{r}_i} \sum_{j \in \mathcal{N}(i)} ||\hat{\mathbf{C}}_{j \to i}(\mathbf{r}_i) - \mathbf{C}_i(\mathbf{r}_i)||_2^2$$

This should match the observed image at time ***i***

Image Courtesy: Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes. *Li et. al.*

Scene Flow Fields: additional regularizations

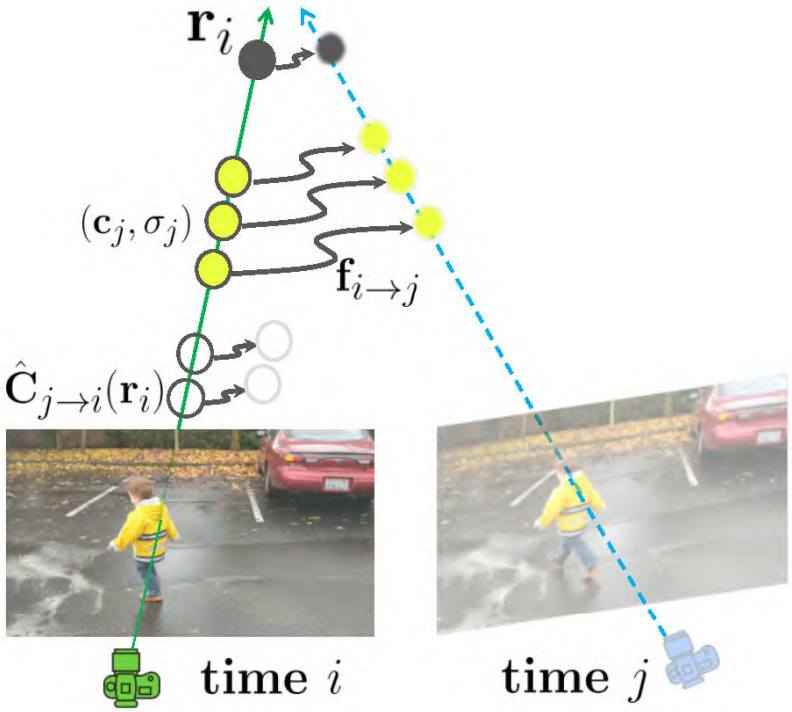


$$\mathcal{L}_{\text{cyc}} = \sum_{\mathbf{x}_i} \sum_{j \in i \pm 1} ||\mathbf{f}_{i \to j}(\mathbf{x}_i) + \mathbf{f}_{j \to i}(\mathbf{x}_{i \to j})||_1$$

The scene flow should be cycle consistent

Image Courtesy: Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes. *Li et. al.*

Scene Flow Fields: additional regularizations



$$\mathcal{L}_{\text{geo}} = \sum_{\mathbf{r}_i} \sum_{j \in \{i \pm 1\}} ||\hat{\mathbf{p}}_{i \to j}(\mathbf{r}_i) - \mathbf{p}_{i \to j}(\mathbf{r}_i))||_1$$

The scene flow match with off-the-shelf optical flow

Image Courtesy: Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes. *Li et. al.*

Neural Scene Flow Fields

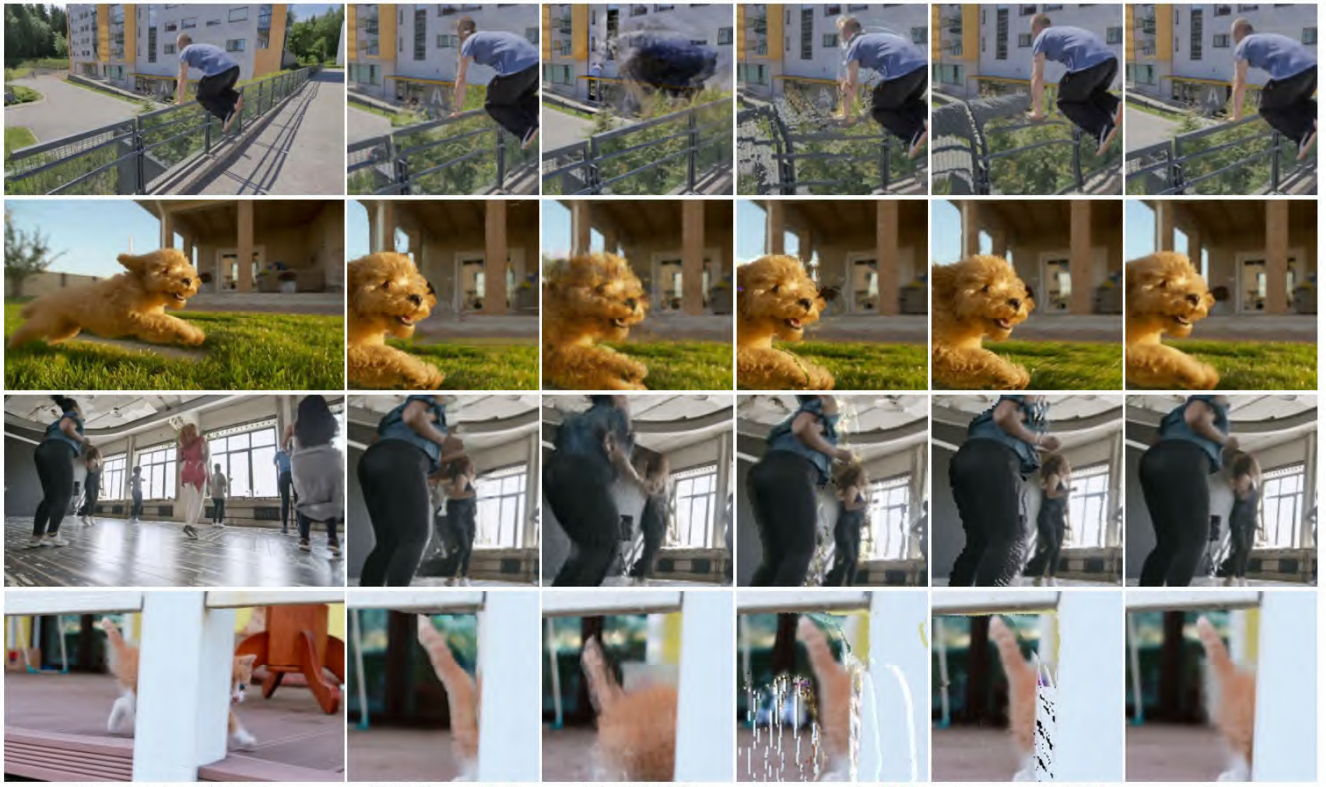


Figure 9: **Qualitative comparisons on monocular video clips.** When compared to baselines, our approach more correctly synthesizes hidden content in disocclusions (shown in the last three rows), and locations with complex scene structure such as the fence in the first row.

Image Courtesy: Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes. *Li et. al.*

Neural Scene Flow Fields

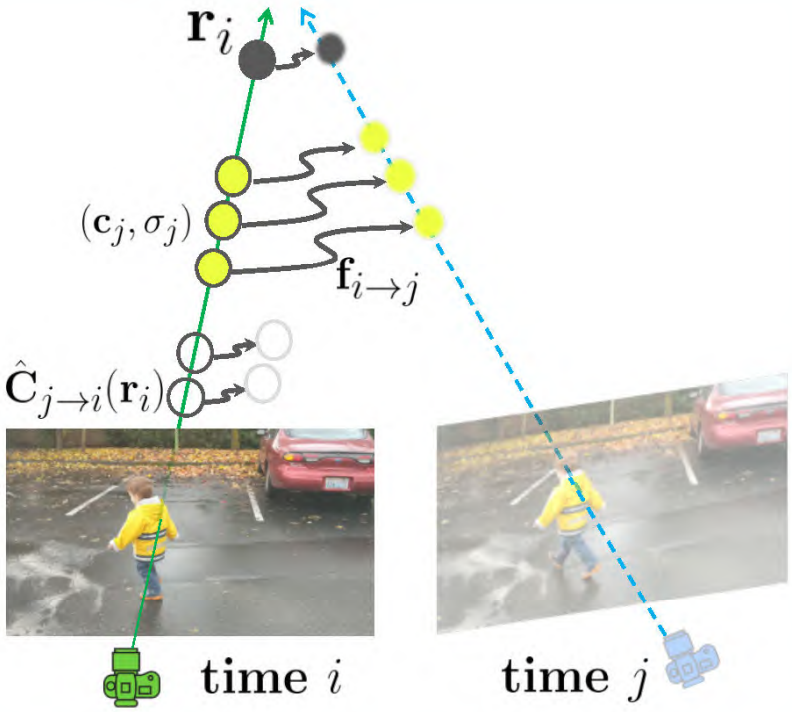
Input

Fixed Time, View Interpolation

Fixed View, Time Interpolation

Space-Time Interpolation

Neural Scene Flow Fields



Only model flow w.r.t neighboring frames

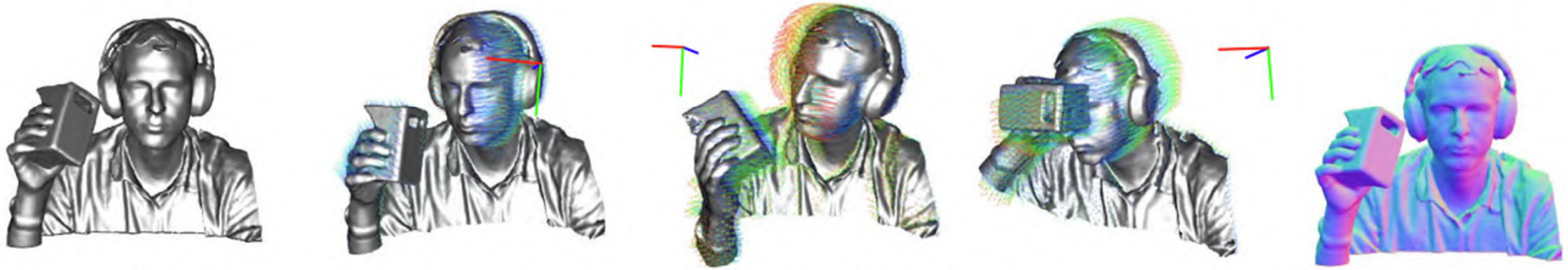
No guarantees for persistence over long horizons

Drift: long-range correspondences maybe inaccurate

Image Courtesy: Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes. *Li et. al.*

A ‘Canonical’ Representation

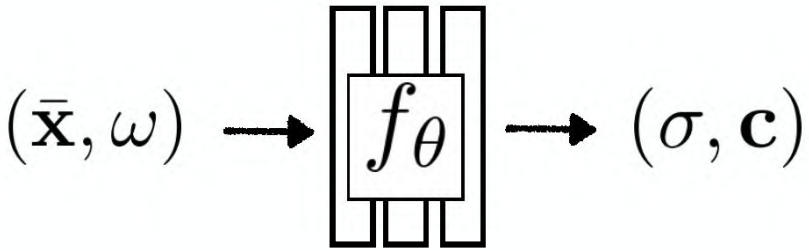
(a) Initial Frame at $t = 0s$ (b) Raw (noisy) depth maps for frames at $t = 1s, 10s, 15s, 20s$ (c) Node Distance



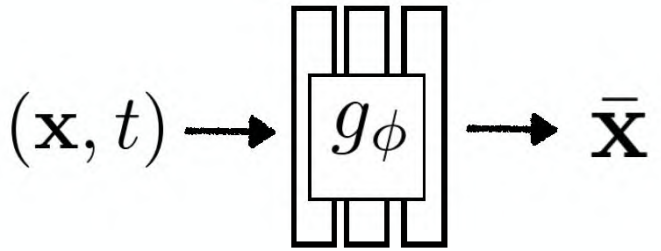
(d) Canonical Model (e) Canonical model warped into its live frame (f) Model Normals

DynamicFusion: Reconstruction and Tracking of Non-rigid Scenes in Real-Time. CVPR 2015 Best Paper.

Towards Neural Canonical Representations



Model a single (canonical) scene representation



Predict flow to the canonical scene at each ***t***

Towards Neural Canonical Representations

Non-Rigid Neural Radiance Fields: Reconstruction and Novel View Synthesis of a Dynamic Scene From Monocular Video

Edgar Tretschk[1] Ayush Tewari[1] Vladislav Golyanik[1] Michael Zollhoefer[2] Christoph Lassner[2] Christian Theobalt[1]
[1]Max Planck Institute for Informatics, Saarland Informatics Campus [2]Facebook Reality Labs

(Accepted at ICCV 2021)

Nerfies: Deformable Neural Radiance Fields

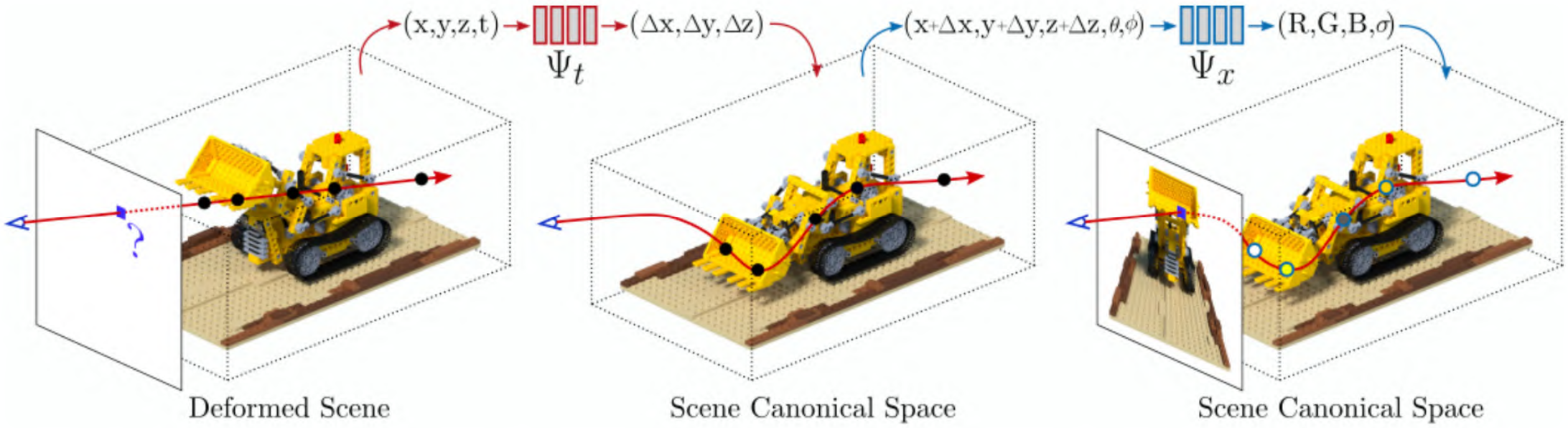
Keunhong Park[1], Utkarsh Sinha[2], Jonathan T. Barron[2], Sofien Bouaziz[2], Dan B Goldman[2], Steven M. Seitz[1,2], Ricardo Martin-Brualla[2]
[1]University of Washington, [2]Google Research

D-NeRF: Neural Radiance Fields for Dynamic Scenes

Albert Pumarola, Enric Corona, Gerard Pons-Moll, Francesc Moreno-Noguer

Several concurrent works with instantiations of the same base idea!

Neural Canonical Representations



Learn two networks — one for flow, one for canonical representation

Learn both using only a rendering objective!

Minor technical concern: why not transform the ray direction using deformation?

Image Courtesy: D-NeRF: Neural Radiance Fields for Dynamic Scenes. *Pumarola et. al.*

Neural Canonical Representations

D-NeRF

Closest Input View

Closest Input Time

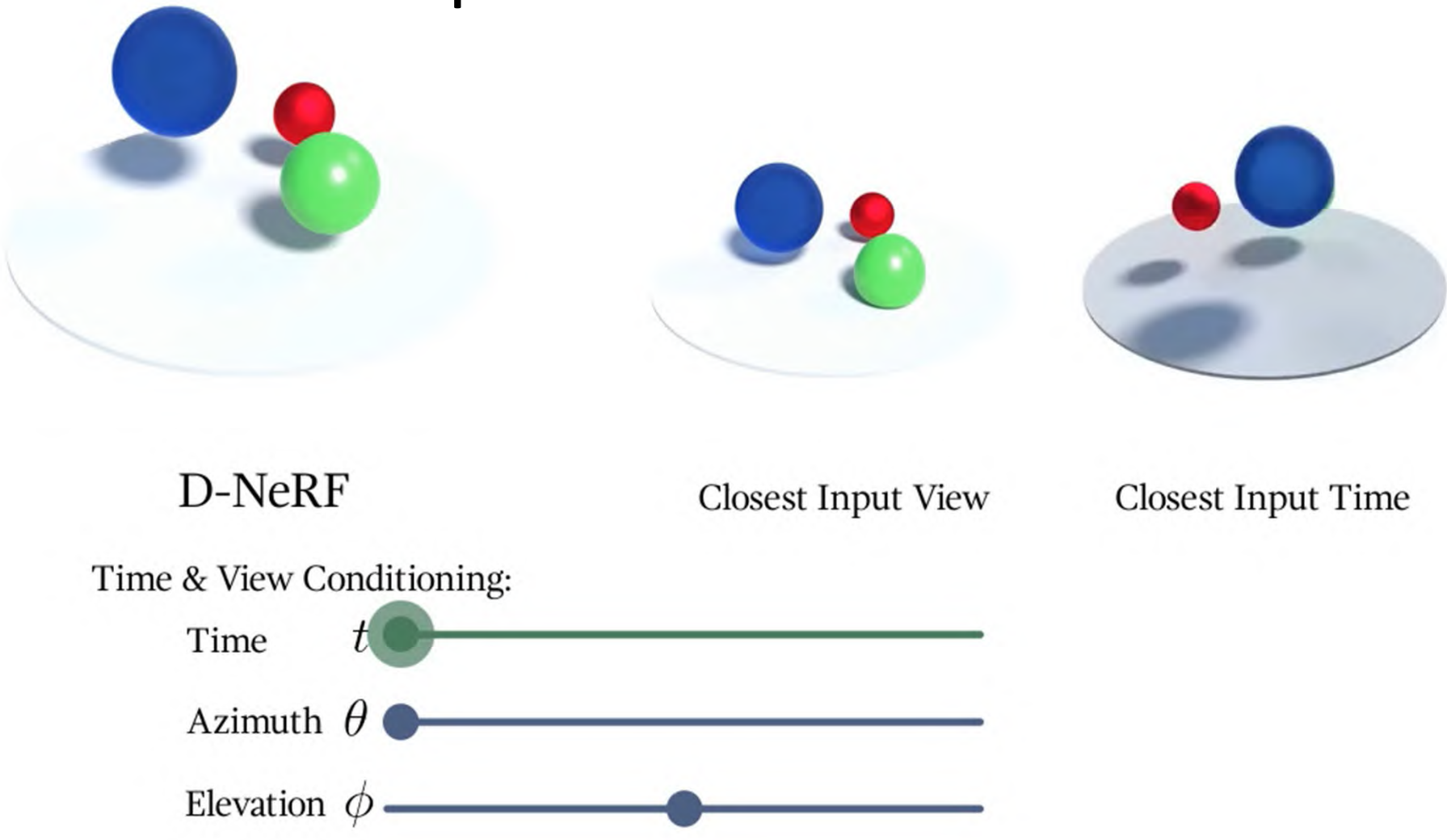
Time & View Conditioning:

Time t

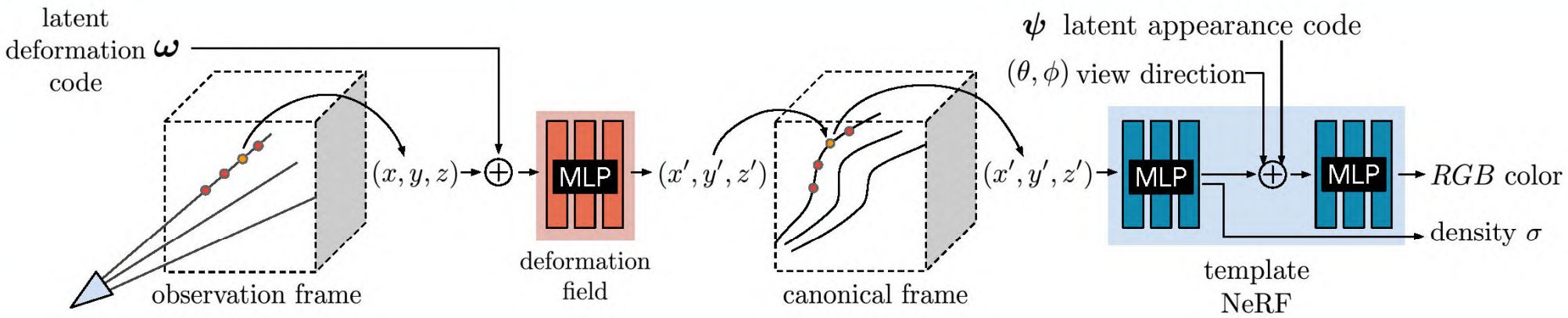
Azimuth θ

Elevation ϕ

Neural Canonical Representations



Neural Canonical Representations



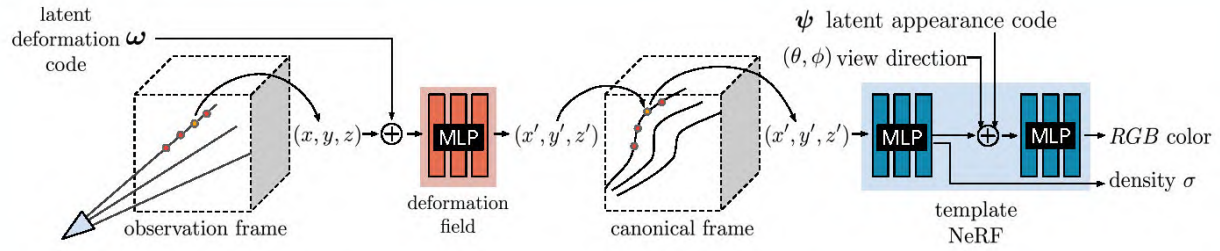
$$\text{SE(3) field } W : (\mathbf{x}, \boldsymbol{\omega}_i) \rightarrow \text{SE(3)}$$

Optimizable latent code as input for deformation prediction (instead of $\boldsymbol{t}$)

Additionally, an appearance latent code to allow lighting etc. changes
(inspired by 'NeRF in the wild')

Image Courtesy: Nerfies: Deformable Neural Radiance Fields. *Park et. al.*

Neural Canonical Representations



Many regularizations in addition to data terms

$$\mathbf{J}_T(\mathbf{x}) \longrightarrow \mathbf{J}_T = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T \longrightarrow L_{\text{elastic}}(\mathbf{x}) = \|\log \boldsymbol{\Sigma} - \log \mathbf{I}\|_F^2 = \|\log \boldsymbol{\Sigma}\|_F^2$$

Locally rigid transformation (accounting for the spatially varying SE3 field)

$$L_{\text{bg}} = \frac{1}{K}\sum_{k=1}^{K}\left\|T(\mathbf{x}_k) - \mathbf{x}_k\right\|_2$$

Points in background should be static

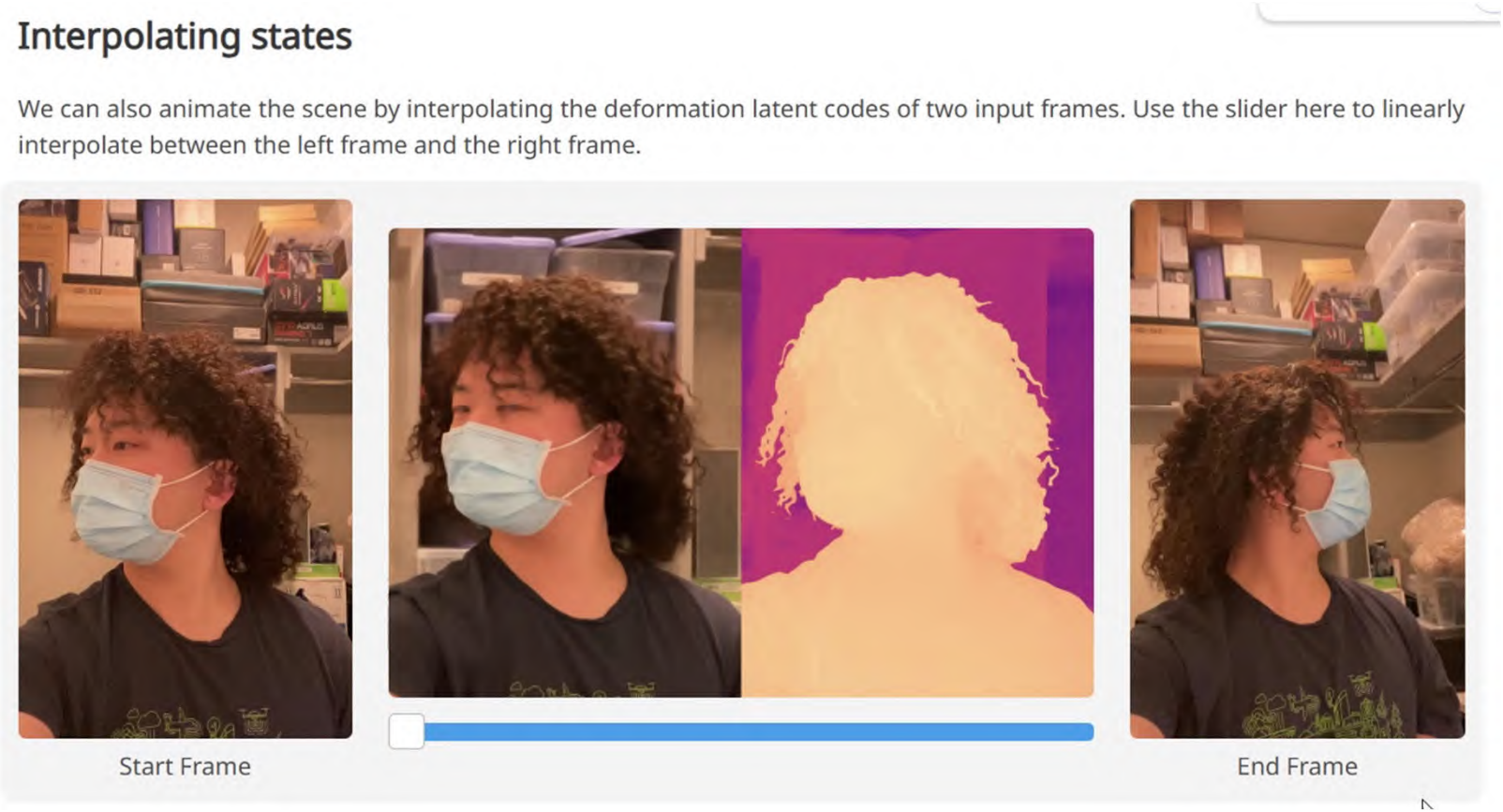
Image Courtesy: Nerfies: Deformable Neural Radiance Fields. *Park et. al.*

Neural Canonical Representations

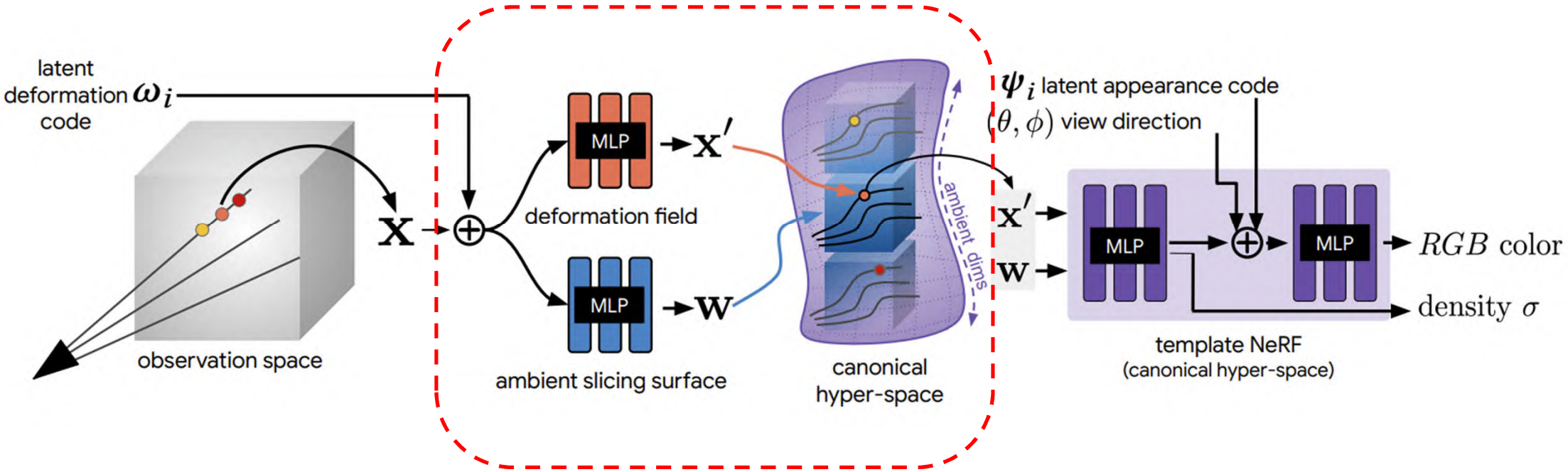
(a) Capture Process (b) Input (c) Nerfie (d) Nerfie Depth

NERFIES turns selfie videos from your phone into free-viewpoint portraits.

Neural Canonical Representations

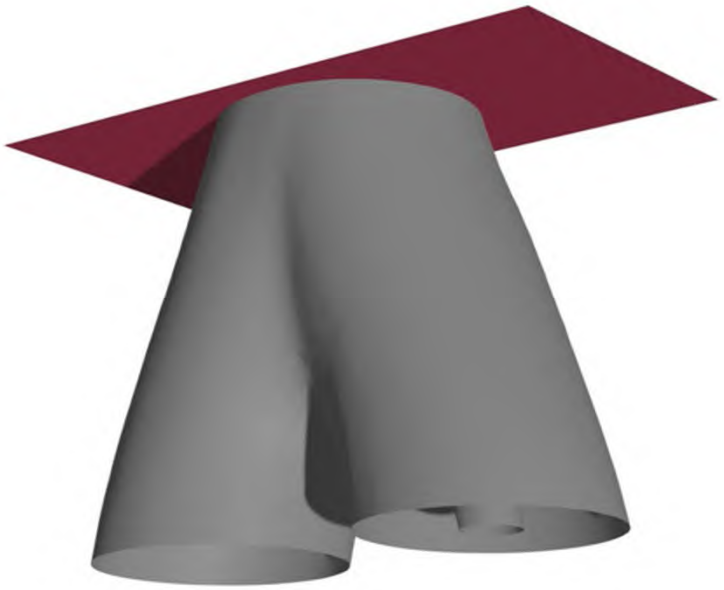


Allowing for Topology Variation

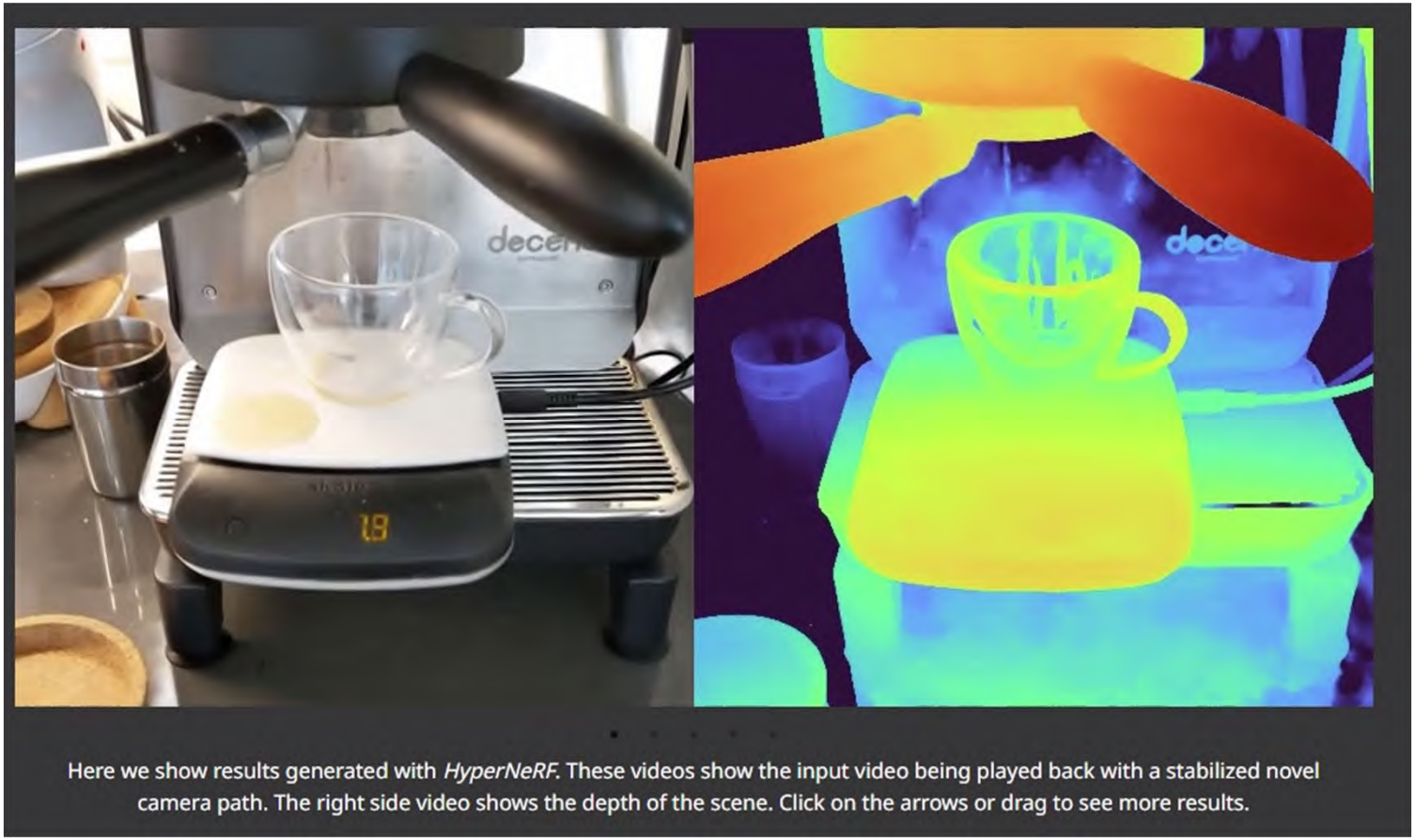


A Higher-Dimensional Representation for Topologically Varying Neural Radiance Fields. SIGGRAPH Asia 2021.

Allowing for Topology Variation



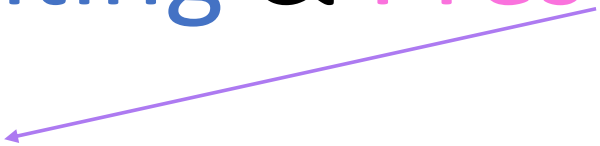
Allowing for Topology Variation



Here we show results generated with *HyperNeRF*. These videos show the input video being played back with a stabilized novel camera path. The right side video shows the depth of the scene. Click on the arrows or drag to see more results.

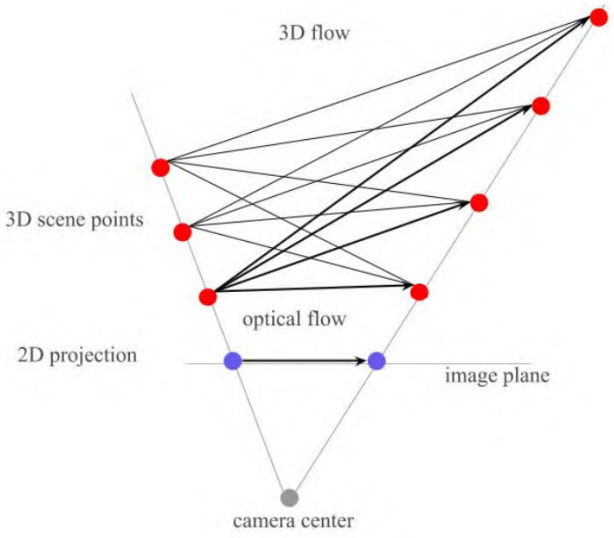
Summary

Reading, Implementing & Presenting

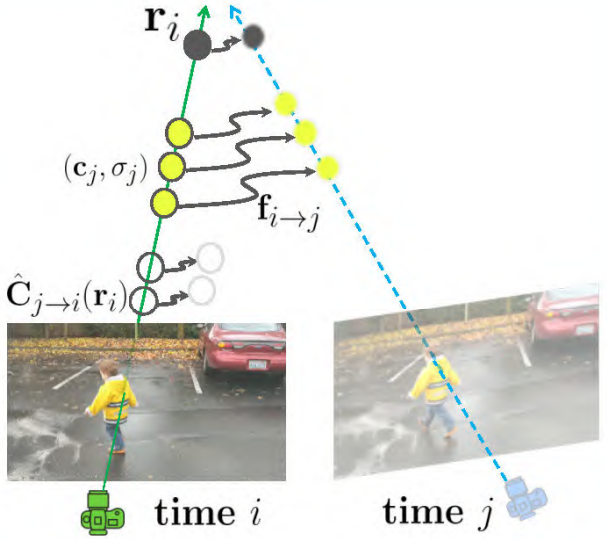


Do what -> Motivation -> Pipeline -> Implementation -> Thinking

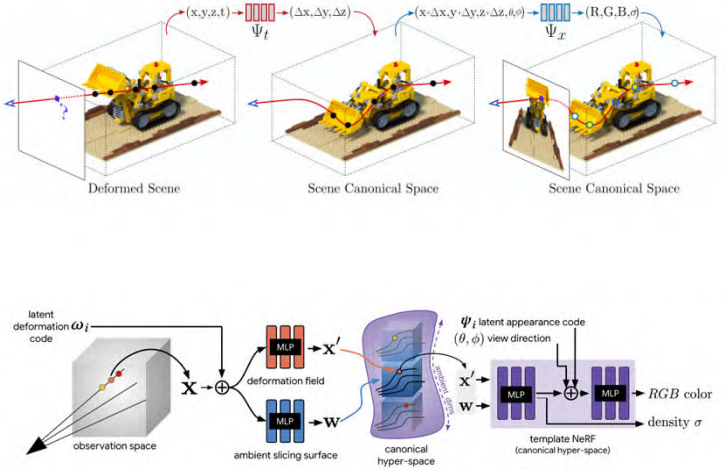
Summary



Scene Flow for 3D Correspondence across time



Joint Inference of Scene Flow and per-time Neural Representations



Learning Neural Canonical Spaces and Deformation Fields

中国科学技术大学

University of Science and Technology of China

谢谢观看!