ELV: Embodied Learning and Vision

Mengye Ren

NYU

Spring 2026

elvcourse.org

Summary: Normalizing Flow

- Requires invertibility and tractability on Jacobian determinant
- Constraint on the functional form
- Or estimation of Jacobian / iterative inverse (numerical approximation).

NYU

Inference as Integration

- Residual Flow: $\phi(x) = x + \delta u(x)$

$$\frac{\phi(x) - x}{\delta} = u(x)$$

NYU

Inference as Integration

- Residual Flow: $\phi(x) = x + \delta u(x)$

$$\frac{\phi(x) - x}{\delta} = u(x)$$

- Can be written in continuous time (continuous normalizing flow).

$$x_t = \phi_t(x_0) = x_0 + \int_0^t \delta u_s(x_s)\ ds.$$

NYU

Inference as Integration

- Residual Flow: $\phi(x) = x + \delta u(x)$

$$\frac{\phi(x) - x}{\delta} = u(x)$$

- Can be written in continuous time (continuous normalizing flow).

$$x_t = \phi_t(x_0) = x_0 + \int_0^t \delta u_s(x_s)\ ds.$$

- Integration: Euler, or directly run an ODE solver

NYU

Inference as Integration

- Residual Flow: $\phi(x) = x + \delta u(x)$

$$\frac{\phi(x) - x}{\delta} = u(x)$$

- Can be written in continuous time (continuous normalizing flow).

$$x_t = \phi_t(x_0) = x_0 + \int_0^t \delta u_s(x_s)\ ds.$$

- Integration: Euler, or directly run an ODE solver
- Backprop solving an ODE without storing intermediate states

NYU

Flow Matching

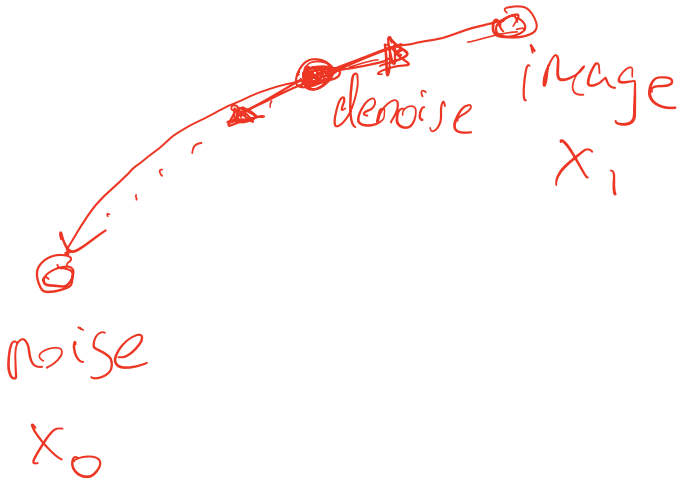
- Taking integrals during training is cumbersome, and the learning signal from the top latent is weak.

Lipman et al. Flow Matching for Generative Modeling. ICLR 2023.

NYU

Flow Matching

- Taking integrals during training is cumbersome, and the learning signal from the top latent is weak.
- Given an original input x_1 and a noise x_0, $x_t = (1-t)x_0 + tx_1$

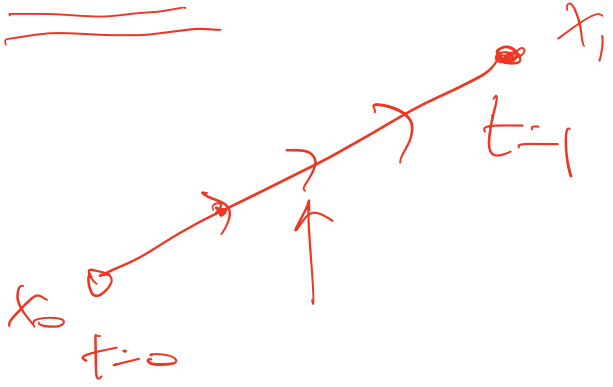


Lipman et al. Flow Matching for Generative Modeling. ICLR 2023.

NYU

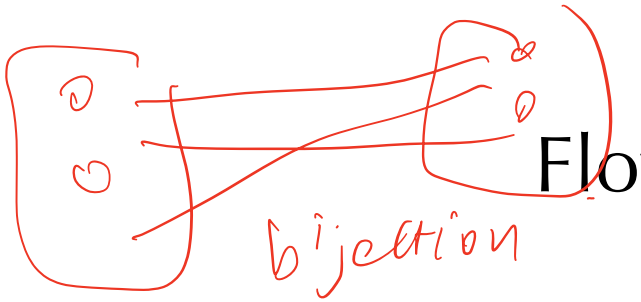
Flow Matching

- Taking integrals during training is cumbersome, and the learning signal from the top latent is weak.
- Given an original input x_1 and a noise x_0, $x_t = (1-t)x_0 + tx_1$
- Want to fit a velocity $v_t(x)$ to match $u_t = x_1 - x_0$



Lipman et al. Flow Matching for Generative Modeling. ICLR 2023.

NYU



deterministic noise.

Flow Matching

- Taking integrals during training is cumbersome, and the learning signal from the top latent is weak.
- Given an original input x_1 and a noise x_0, $x_t = (1-t)x_0 + tx_1$
- Want to fit a velocity $v_t(x)$ to match $u_t = x_1 - x_0$
- Flow Matching learning objective:

$$\mathcal{L}_{FM}(\theta) = \mathbb{E}_{t,x_0,x_1}[||v(x_t,t) - u_t(x_0,x_1)||^2]$$

targets

ideal velocity.

Euler integration → 10-20 steps.

efficient.

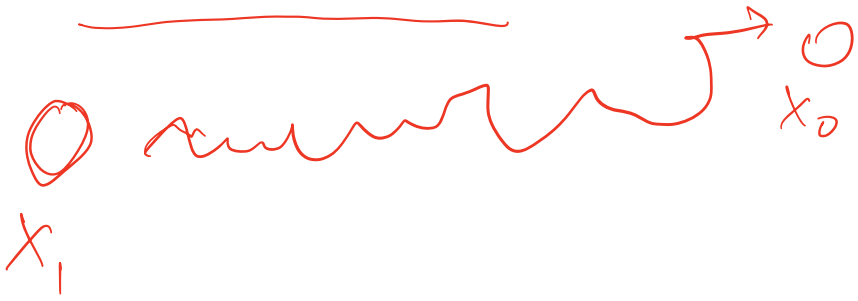
Diffusion Models

- A popular generative model formulation.

NYU

Diffusion Models

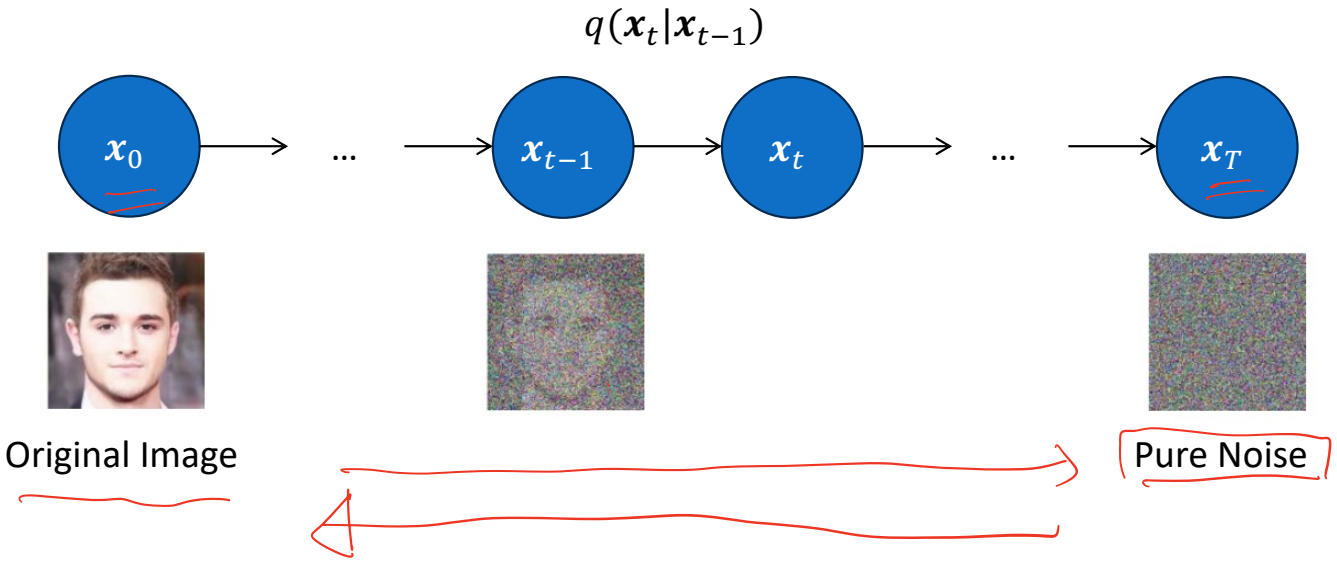
- A popular generative model formulation.
- The intuition is to iteratively denoise from Gaussian random noises into an image.



NYU

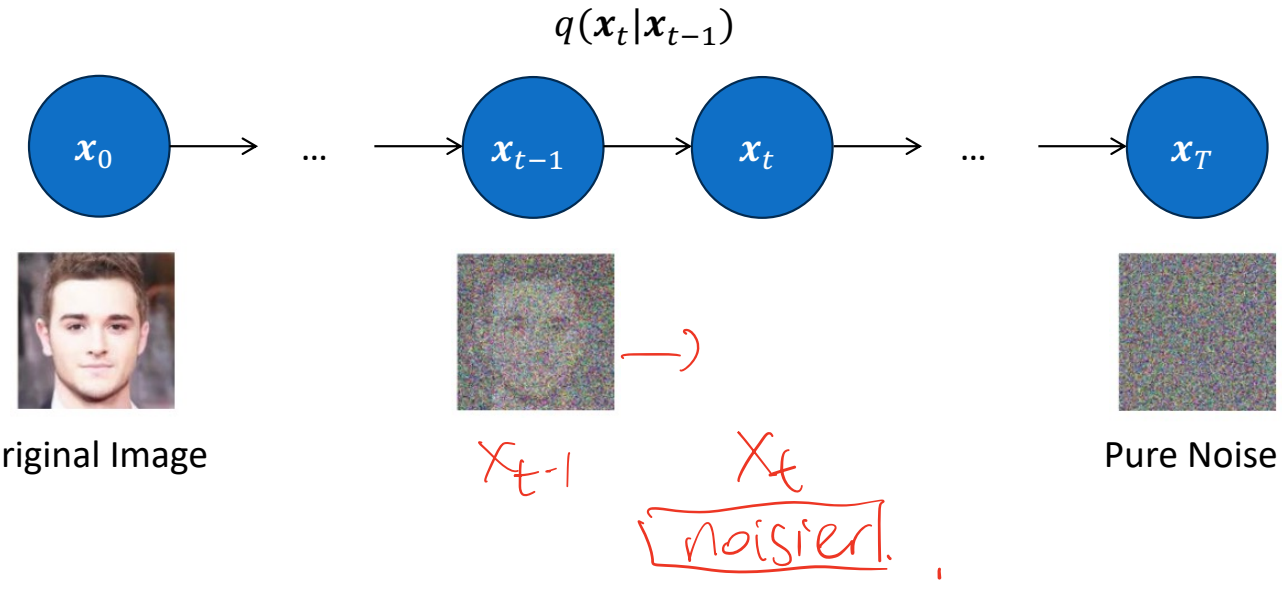
Diffusion Models

- A popular generative model formulation.
- The intuition is to iteratively denoise from Gaussian random noises into an image.



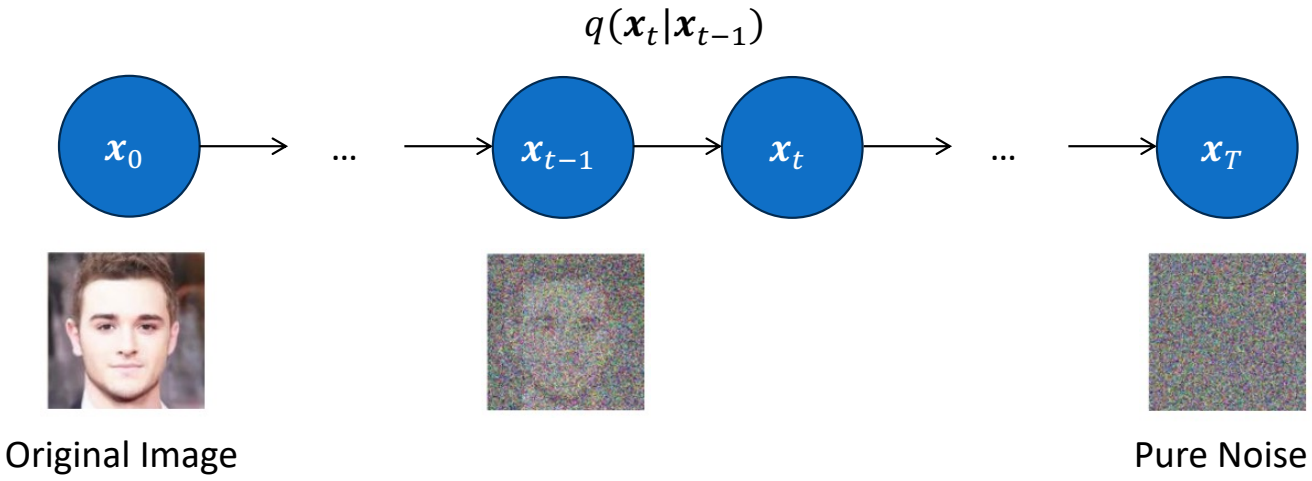
Diffusion Models

- Forward process: $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$.



Diffusion Models

- Forward process: $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$.
- You can also write: $x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon_t$, $\epsilon_t \sim \mathcal{N}(0, I)$.



NYU

Properties of the Forward Process

- Forward process: $x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon_t$

NYU

Properties of the Forward Process

- Forward process: $x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon_t$
- Write x_t as a function of x_0 with larger noises:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I).$$

Properties of the Forward Process

- Forward process: $x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon_t$
- Write x_t as a function of x_0 with larger noises:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I).$$

- Cumulative schedule: $\alpha_t = 1 - \beta_t$. $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$.

amount of original.

Properties of the Forward Process

- Forward process: $x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon_t$
- Write x_t as a function of x_0 with larger noises:

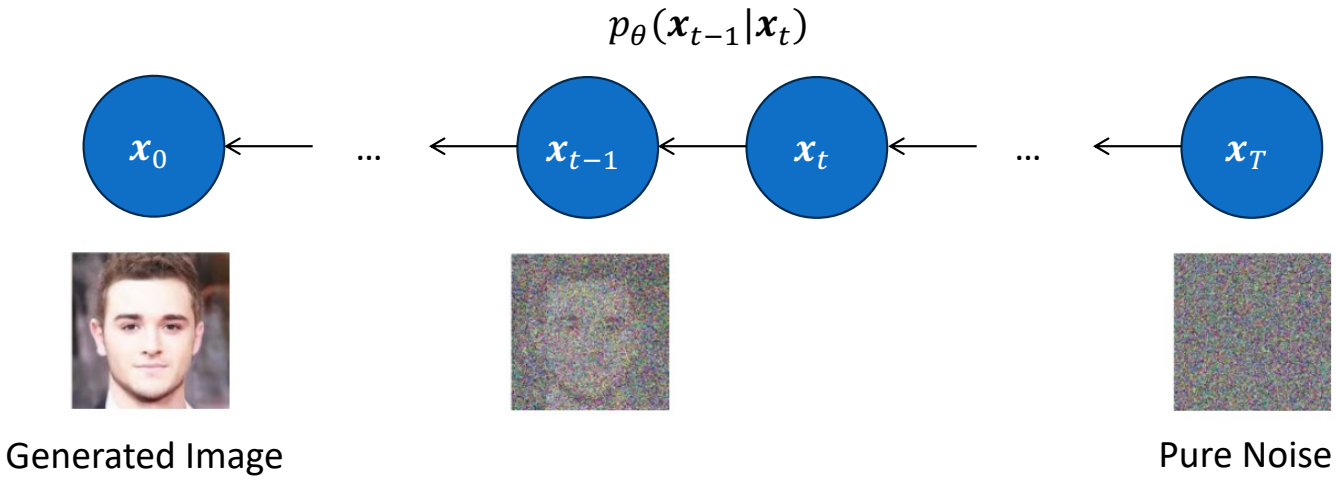
$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I).$$

- Cumulative schedule: $\alpha_t = 1 - \beta_t$. $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$.
- $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$.

NYU

Reverse Process

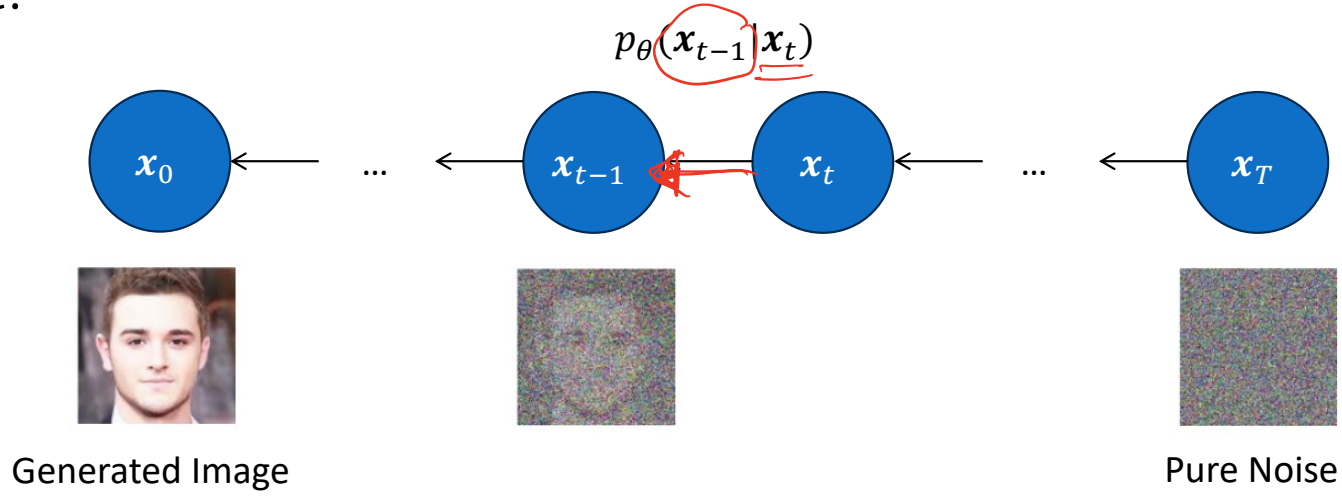
- A generative model wants to predict x_0 from x_T.



NYU

Reverse Process

- A generative model wants to predict x_0 from x_T.
- The reverse process transition is also Gaussian distributed. But we don't know what the transition will be like just by looking at the noisy image!

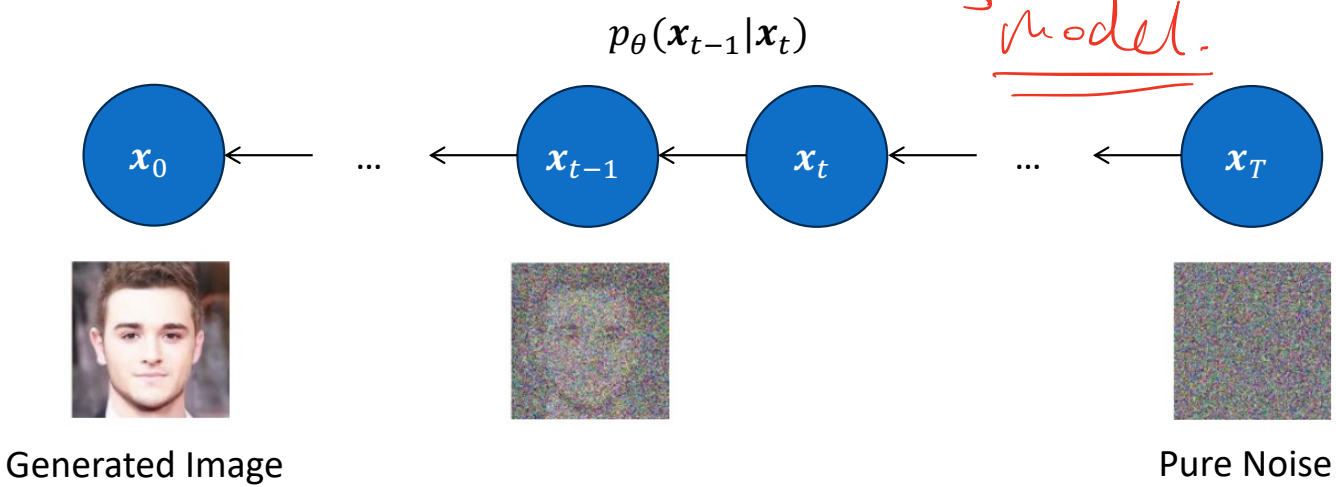


Reverse Process

- So, we need to learn a "model":

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)).$$

model.

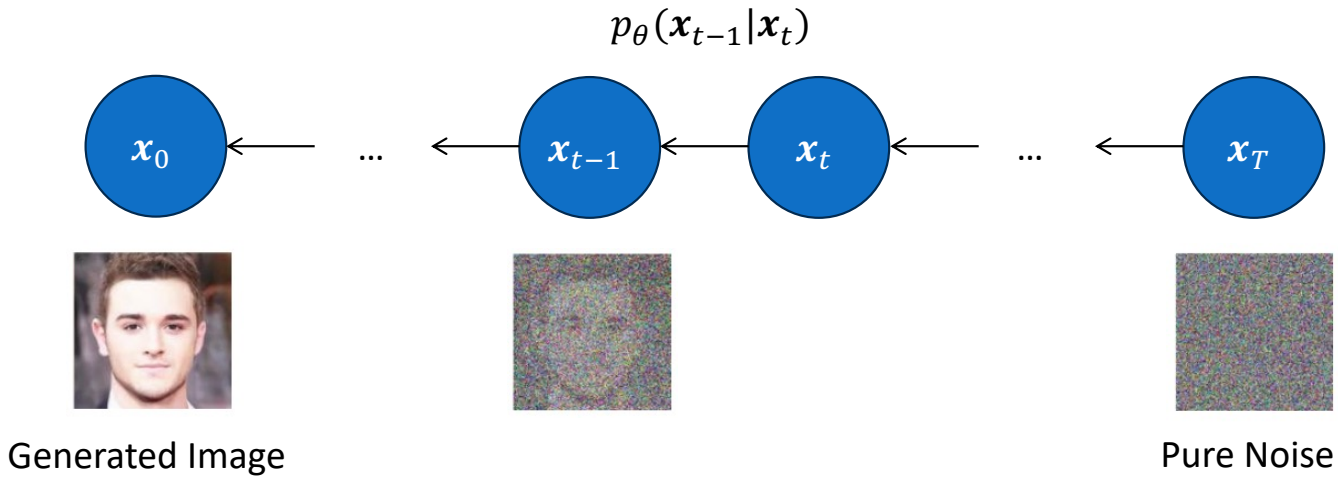


Reverse Process

- So, we need to learn a "model":

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)).$$

- μ_θ is the denoising vector.



Reverse Process

- Compute μ_θ? Derive $p(x_{t-1}|x_t)$.

NYU

Reverse Process

- Compute μ_θ? Derive $p(x_{t-1}|x_t)$.
- Bayes rule:

$$q(x_{t-1}|x_t) = \frac{q(x_t|x_{t-1})q(x_{t-1})}{q(x_t)}.$$

forward process.

Reverse Process

- Compute μ_θ? Derive $p(x_{t-1}|x_t)$.
- Bayes rule:

$$q(x_{t-1}|x_t) = \frac{q(x_t|x_{t-1})q(x_{t-1})}{q(x_t)}.$$

- But we don't know the marginal distribution $q(x_{t-1})$. We only know q_T and $q(x_t|x_{t-1})$.

learning approx. $p(x_{t-1}|x_t) \rightarrow \mu_\theta$

↳ training data $\rightarrow$ x_0's: learn.

Reverse Process

- Compute μ_θ? Derive $p(x_{t-1}|x_t)$.
- Bayes rule:

$$q(x_{t-1}|x_t) = \frac{q(x_t|x_{t-1})q(x_{t-1})}{q(x_t)}.$$

- But we don't know the marginal distribution $q(x_{t-1})$. We only know q_T and $q(x_t|x_{t-1})$.
- Solution: Condition on the original input x_0:

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1})q(x_{t-1}|x_0)}{q(x_t|x_0)}.$$

— during training

NYU

Reverse Process

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1})q(x_{t-1}|x_0)}{q(x_t|x_0)}.$$

NYU

Reverse Process

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1})q(x_{t-1}|x_0)}{q(x_t|x_0)}.$$

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t, \tilde{\beta} I).$$

NYU

Reverse Process

small jump.

big jump.

big jump.

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1})q(x_{t-1}|x_0)}{q(x_t|x_0)}.$$

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t, \tilde{\beta} I).$$

$$\tilde{\mu}_t = \frac{\sqrt{\alpha_t}\bar{\beta}_{t-1}}{\bar{\beta}_t} x_t + \frac{\sqrt{\alpha_{t-1}}\beta_t}{\bar{\beta}_t} x_0 = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right).$$

noisy

orig. img.

NYU

Reverse Process

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1})q(x_{t-1}|x_0)}{q(x_t|x_0)}.$$

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t, \tilde{\beta} I).$$

$$\tilde{\mu}_t = \frac{\sqrt{\alpha_t}\bar{\beta}_{t-1}}{\bar{\beta}_t} x_t + \frac{\sqrt{\alpha_{t-1}}\beta_t}{\bar{\beta}_t} x_0 = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon).$$

- Want: train up a μ_θ to match with $\tilde{\mu}_t$.

NYU

Training

- Sometimes it is more common to predict the denoising vector ϵ instead of μ.

$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon),$$

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t)).$$

NYU

Training

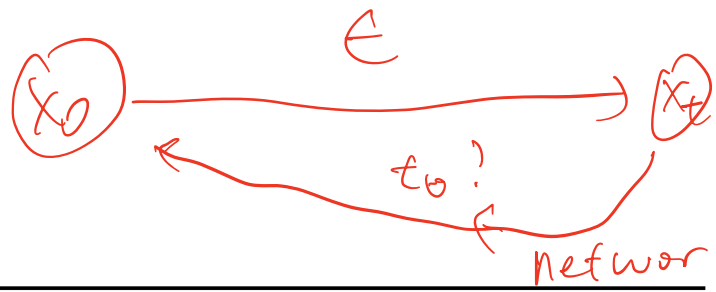
- Sometimes it is more common to predict the denoising vector ϵ instead of μ.
- Randomly pick at a time step and predict the difference between the noisy and the original.

$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon),$$

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t)).$$

NYU

Training



- Sometimes it is more common to predict the denoising vector ϵ instead of μ.
- Randomly pick at a time step and predict the difference between the noisy and the original.

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

true noise network noisy ver.

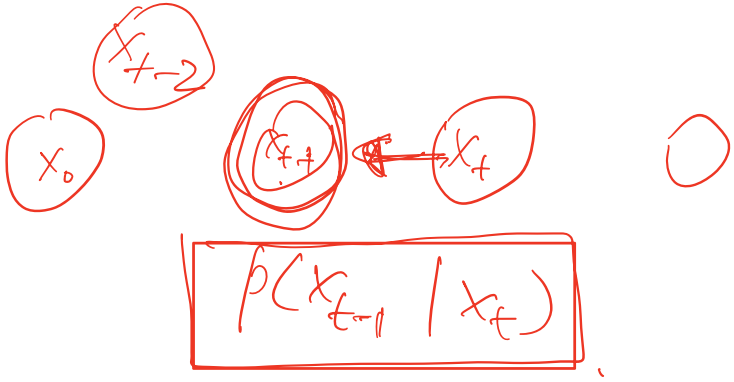
$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon),$$

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t)).$$

NYU

Sampling

- How do we sample an image?

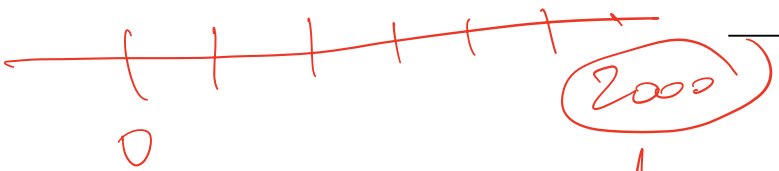


Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

add noise

Sampler



denoising

Schedule.

mean

Current image

Sampling

- How do we sample an image?
- We know μ_θ which will help us transition from x_t to x_{t-1}.

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t)).$$

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

NYU

Sampling

- How do we sample an image?
- We know μ_θ which will help us transition from x_t to x_{t-1}.

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t)).$$

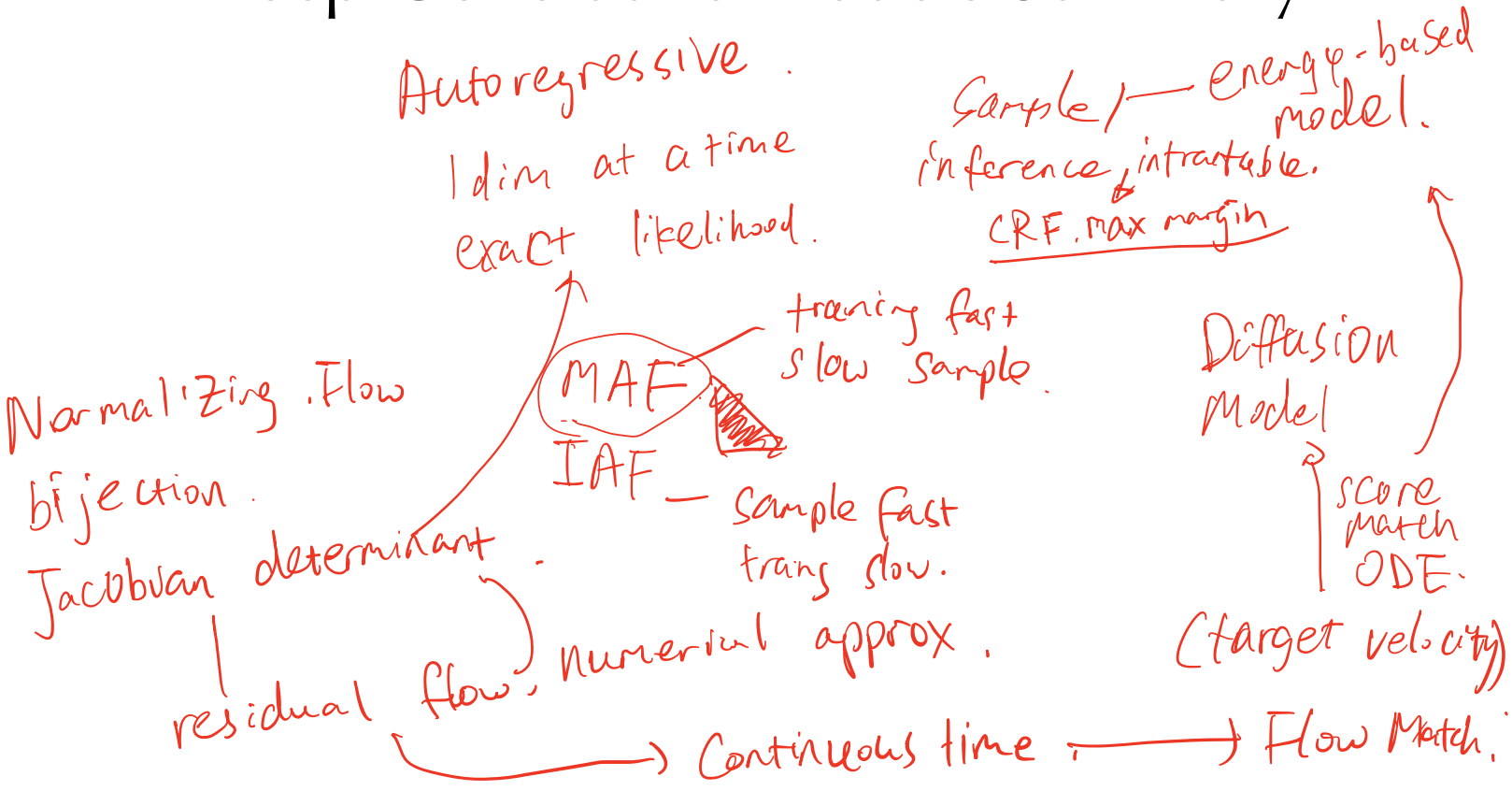
- Sample from $\mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2)$. σ_t can either be β_t or $\tilde{\beta}_t$ derived from the posterior.

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

NYU

Deep Generative Models Summary



Guided Diffusion

- We can add guidance on the diffusion updates at inference time.

Classifier Guidance / External Score Model

$$x_{t-1} \leftarrow \text{sample from } \mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log p_\phi(y|x_t), \Sigma)$$

$$\hat{\epsilon} \leftarrow \epsilon_\theta(x_t) - \sqrt{1-\bar{\alpha}_t}\, \nabla_{x_t} \log p_\phi(y|x_t)$$
$$x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}} \left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1-\bar{\alpha}_{t-1}}\hat{\epsilon}$$

gradient from another criterion

Dhariwal & Nichol. Diffusion Models Beat GANs on Image Synthesis. NeurIPS 2021.
Ho & Salimans. Classifier-Free Diffusion Guidance. 2022.

Guided Diffusion

- We can add guidance on the diffusion updates at inference time.

Classifier Guidance / External Score Model

$$x_{t-1} \leftarrow \text{sample from } \mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log p_\phi(y|x_t), \Sigma)$$

$$\hat{\epsilon} \leftarrow \epsilon_\theta(x_t) - \sqrt{1-\bar{\alpha}_t}\, \nabla_{x_t} \log p_\phi(y|x_t)$$
$$x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}} \left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1-\bar{\alpha}_{t-1}}\hat{\epsilon}$$

- We also can train a conditional diffusion model.

repeat
- $(\mathbf{x}, \mathbf{c}) \sim p(\mathbf{x}, \mathbf{c})$ ▷ Sample data with conditioning from the dataset
- $\mathbf{c} \leftarrow \varnothing$ with probability p_{uncond} ▷ Randomly discard conditioning to train unconditionally
- $\lambda \sim p(\lambda)$ ▷ Sample log SNR value
- $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- $\mathbf{z}_\lambda = \alpha_\lambda \mathbf{x} + \sigma_\lambda \boldsymbol{\epsilon}$ ▷ Corrupt data to the sampled log SNR value
- Take gradient step on $\nabla_\theta \left\| \boldsymbol{\epsilon}_\theta(\mathbf{z}_\lambda, \mathbf{c}) - \boldsymbol{\epsilon} \right\|^2$ ▷ Optimization of denoising model

until converged

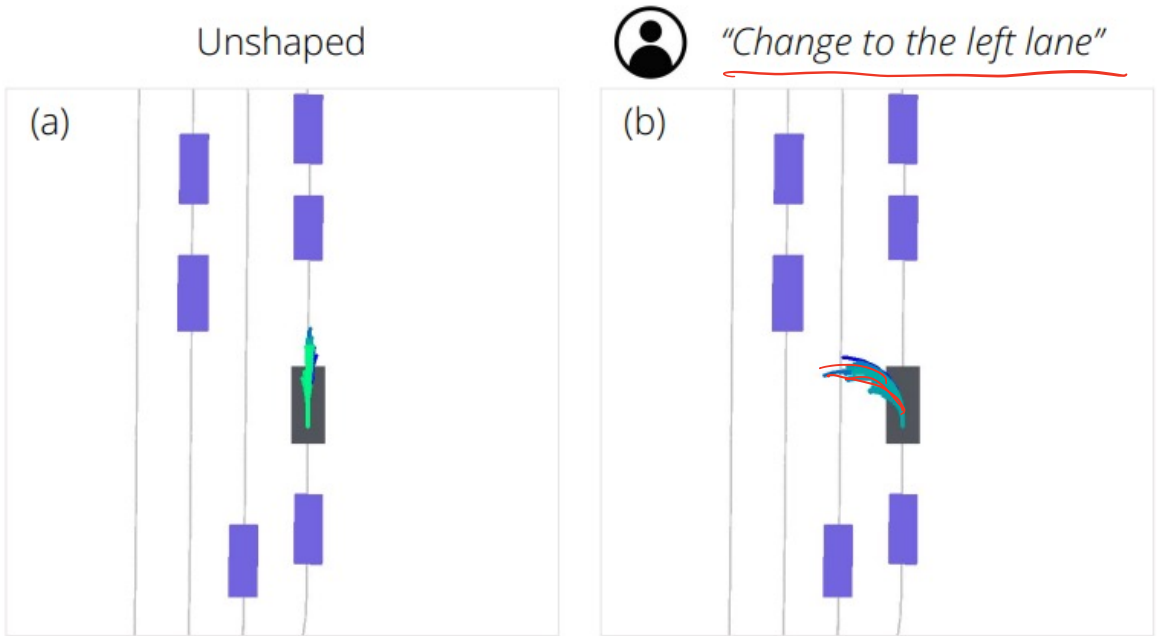
Condition.

Dhariwal & Nichol. Diffusion Models Beat GANs on Image Synthesis. NeurIPS 2021.
Ho & Salimans. Classifier-Free Diffusion Guidance. 2022.

NYU

Test-Time Guidance / Adaptation

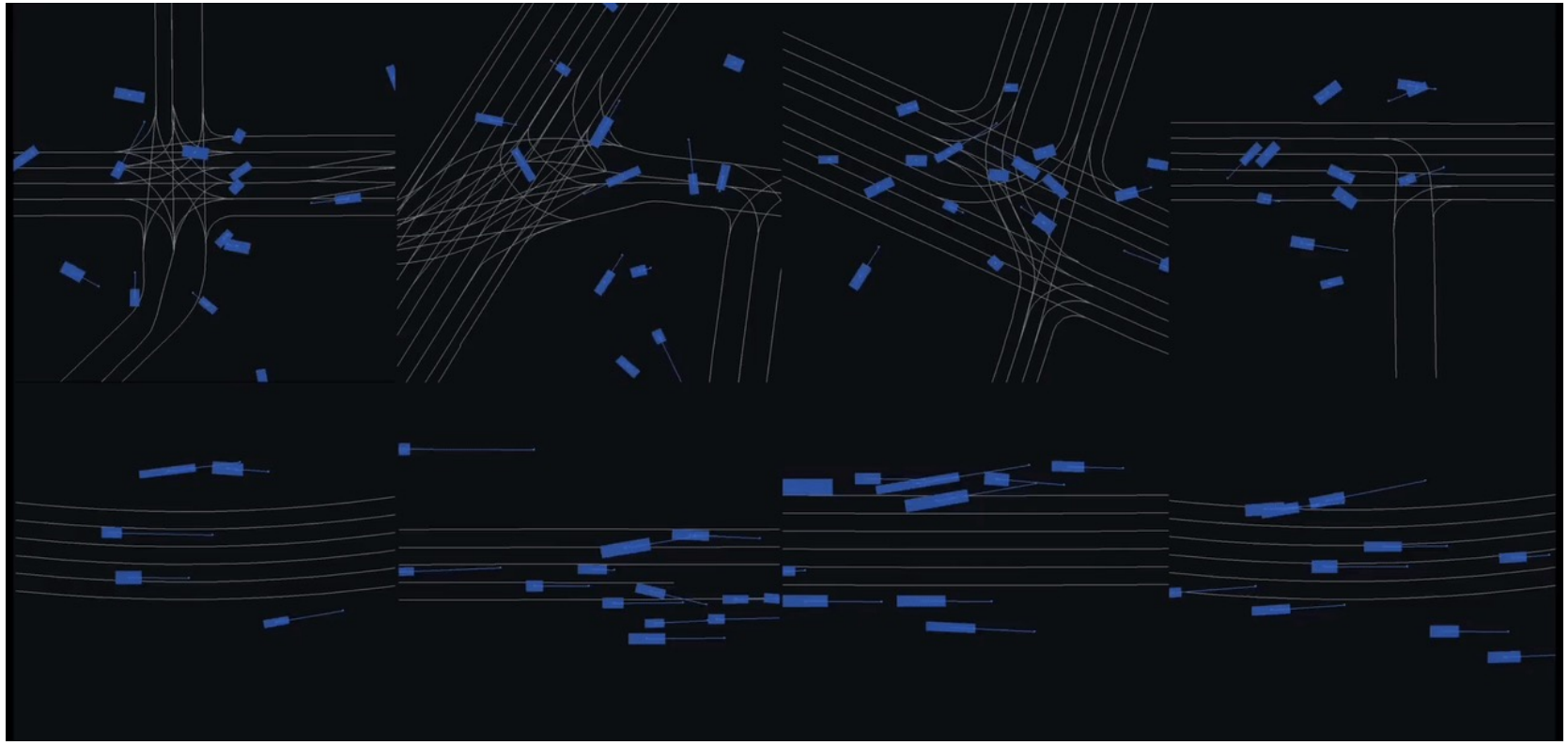
- Diffusion can be combined / guided at test time.



Yang et al. Diffusion-ES: Gradient-free Planning with Diffusion for Autonomous and Instruction-guided Driving. CVPR 2024.

NYU

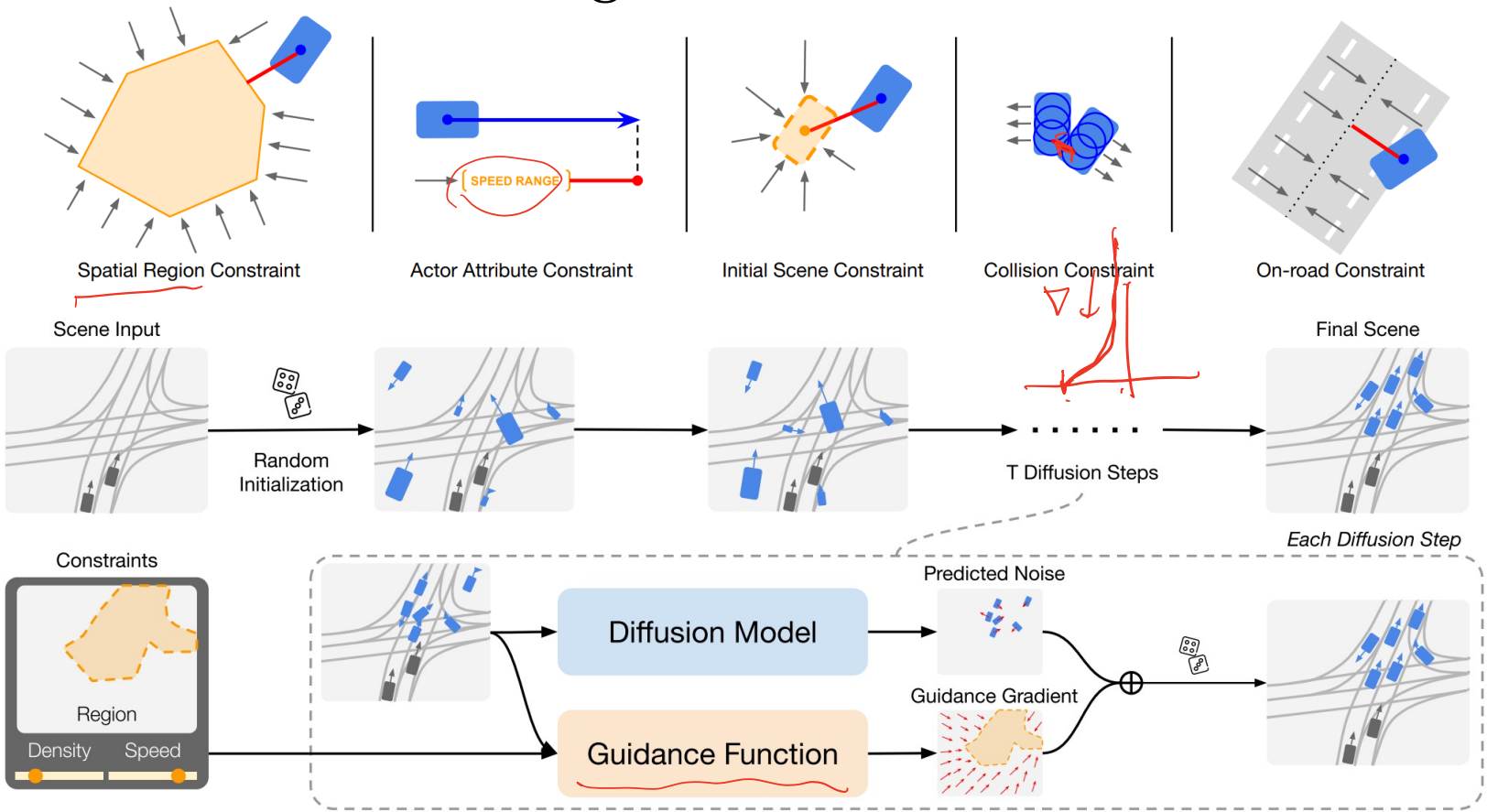
Generating Simulation Scenes



Lu et al. SceneControl: Diffusion for Controllable Traffic Scene Generation. ICRA 2024.

NYU

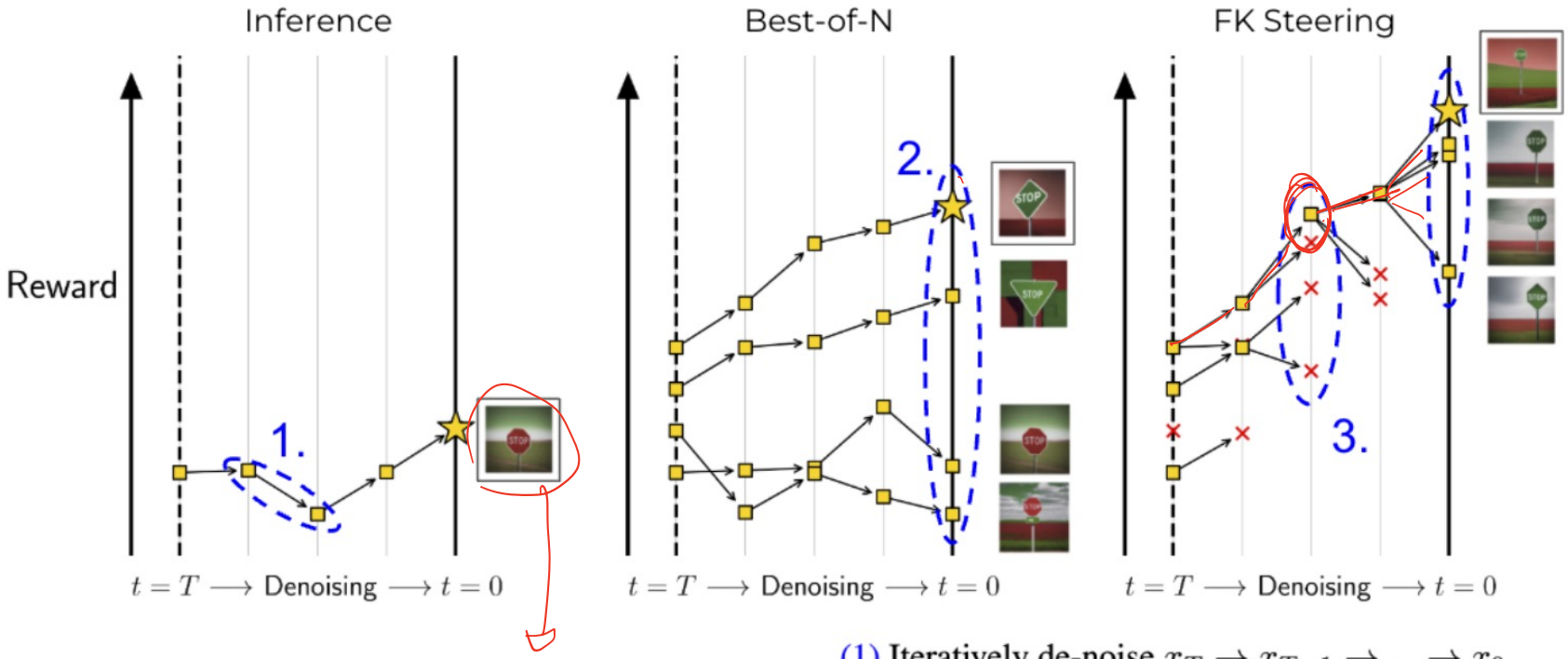
Generating Simulation Scenes



Lu et al. SceneControl: Diffusion for Controllable Traffic Scene Generation. ICRA 2024.

NYU

Non-Differentiable Rewards?



Prompt: *"a green stop sign in a red field"*

(1) Iteratively de-noise $x_T \rightarrow x_{T-1} \rightarrow \ldots \rightarrow x_0$.
(2) Generate multiple samples (*particles*).
(3) Resample promising particles at *intermediate* steps.

Singhal et al. A general framework for inference-time scaling and steering of diffusion models. ICML 2025.

NYU

Diffusion Models for Detection

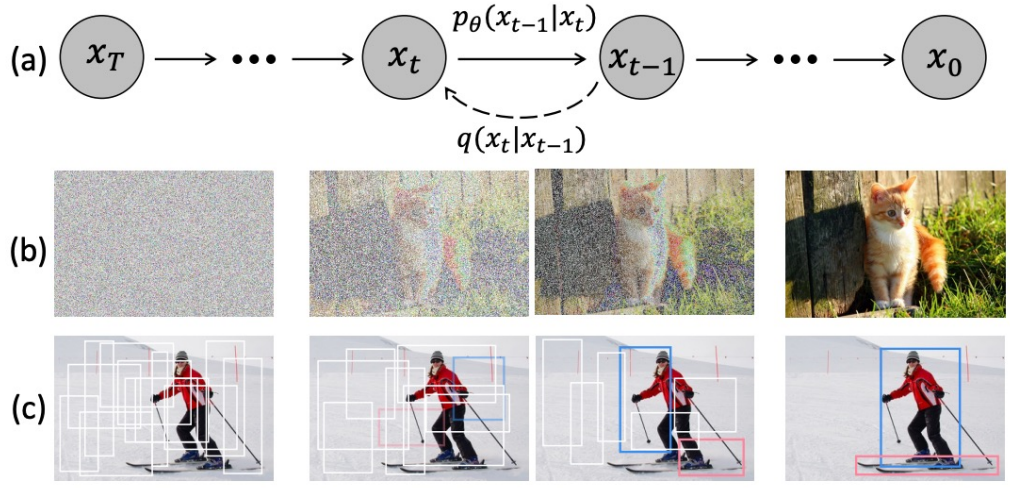
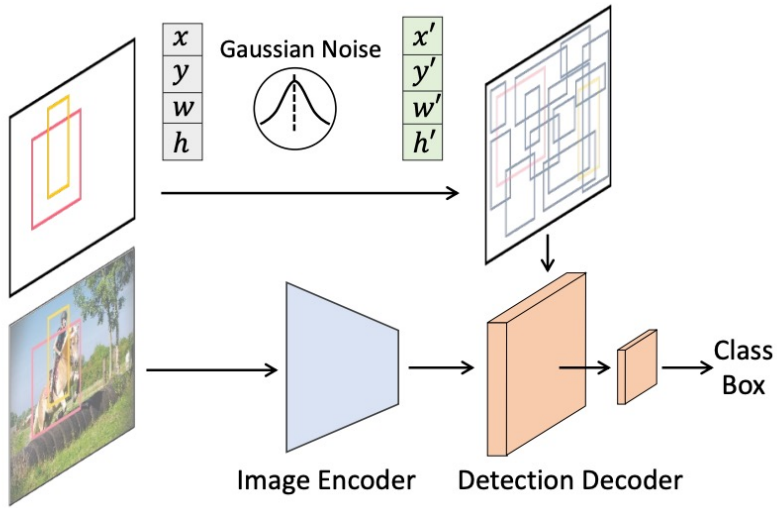


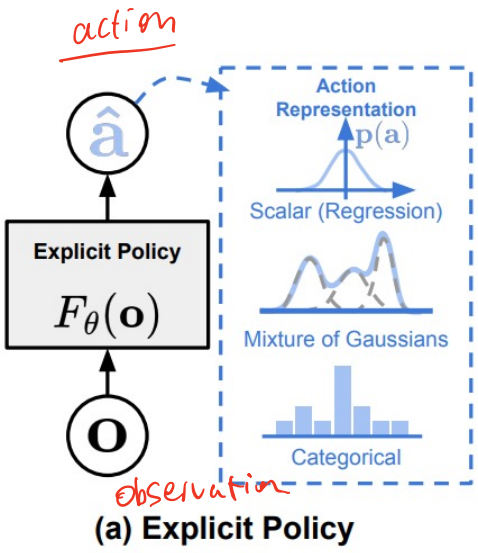
Figure 1. **Diffusion model for object detection**. (a) A diffusion model where q is the diffusion process and p_θ is the reverse process. (b) Diffusion model for image generation task. (c) We propose to formulate object detection as a denoising diffusion process from noisy boxes to object boxes.



Chen et al. DiffusionDet: Diffusion Model for Object Detection. ICCV 2023.

NYU

Planning and Control

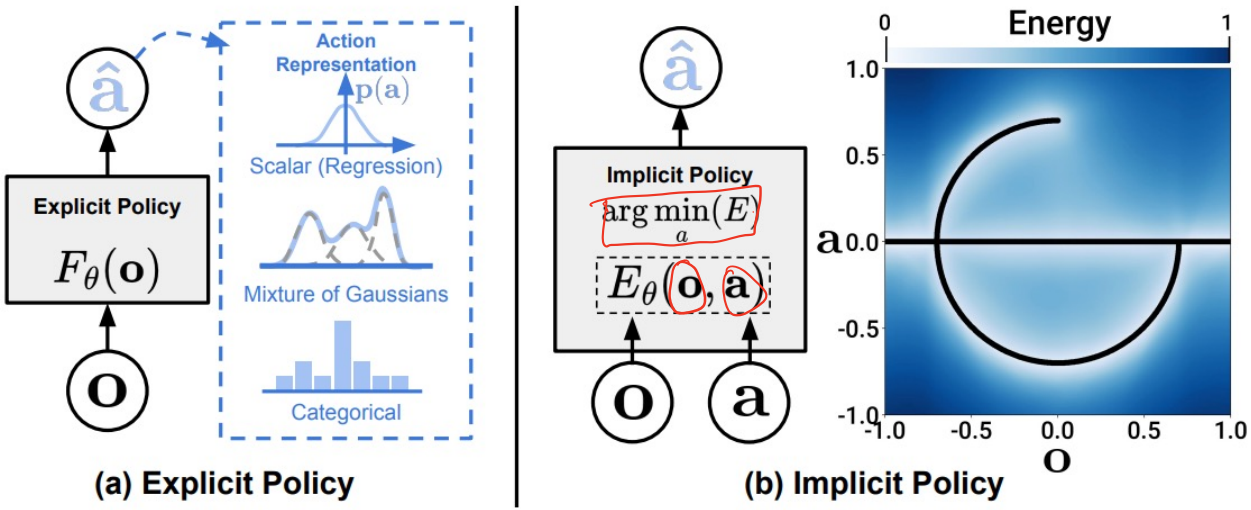


(a) Explicit Policy

Chi et al. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. RSS 2023.

NYU

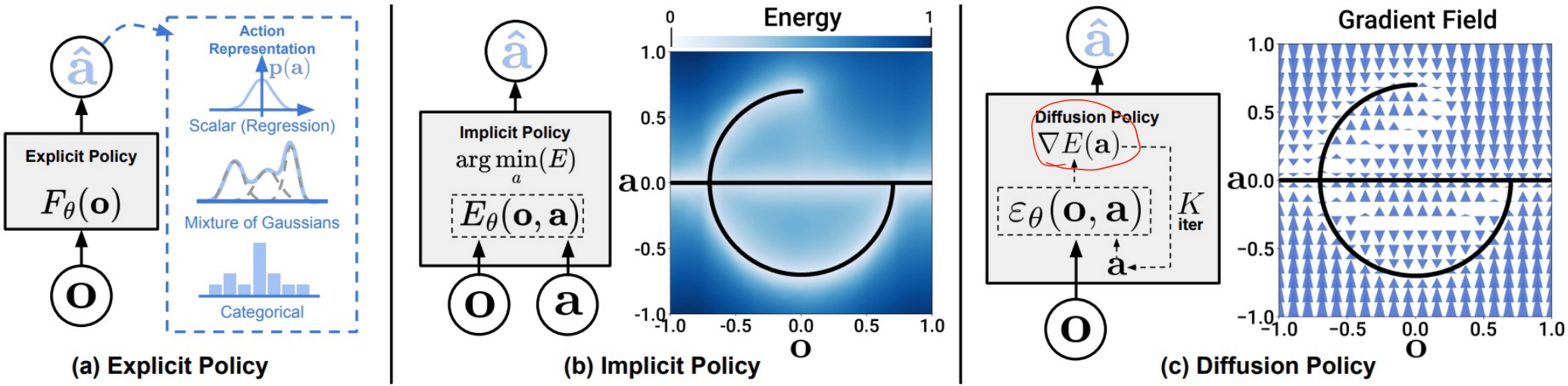
Planning and Control



Chi et al. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. RSS 2023.

NYU

Planning and Control



Chi et al. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. RSS 2023.

NYU

Summary: DL for Structured Outputs

- Expanding the output dimension has limitations.

NYU

Summary: DL for Structured Outputs

- Expanding the output dimension has limitations.
- Requires us thinking about deep generative models.
  - Graphical models
  - Autoregressive
  - Energy-based
  - Normalizing flow
  - Diffusion

NYU

Summary: DL for Structured Outputs

- Expanding the output dimension has limitations.
- Requires us thinking about deep generative models.
  - Graphical models
  - Autoregressive
  - Energy-based
  - Normalizing flow
  - Diffusion
- Understand relations, pros and cons.

NYU

Summary: DL for Structured Outputs

- Expanding the output dimension has limitations.
- Requires us thinking about deep generative models.
  - Graphical models
  - Autoregressive
  - Energy-based
  - Normalizing flow
  - Diffusion
- Understand relations, pros and cons.
- Application in embodied environments.

NYU

Module 2:
3D Vision and Mapping

NYU

The World is 3D

- We have previously focused on using 2D images as input.

The World is 3D

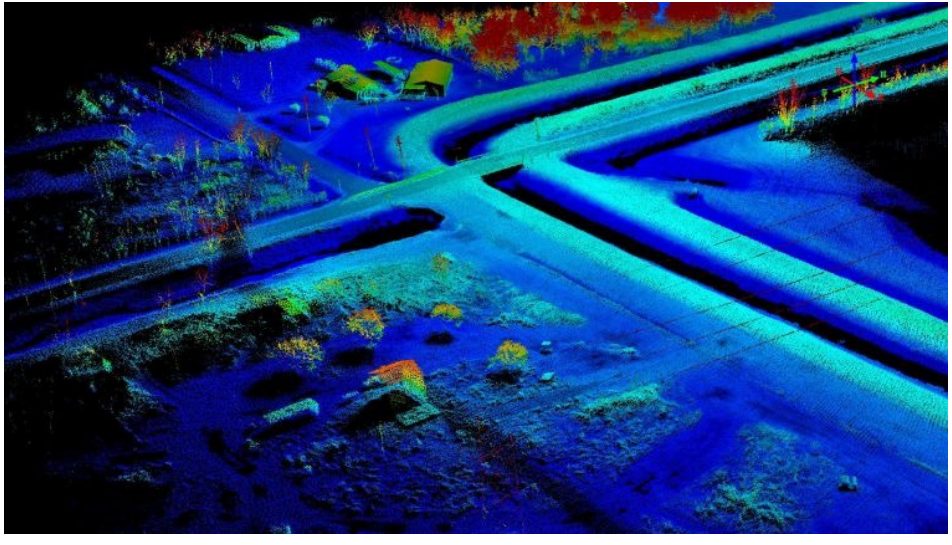
- We have previously focused on using 2D images as input.
- But, the world is 3D. Many non-rigid in 2D becomes rigid in 3D. There are also a wide range of 3D sensors.

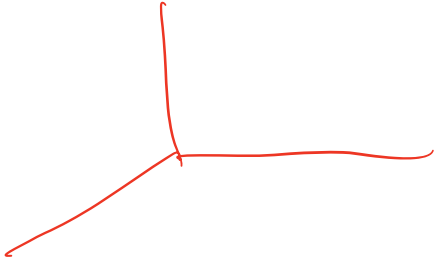
The World is 3D

- We have previously focused on using 2D images as input.
- But, the world is 3D. Many non-rigid in 2D becomes rigid in 3D. There are also a wide range of 3D sensors.
- Stereo (our binocular vision), infrared camera, LiDAR, radar, etc.

LiDAR

10k. (x, y, z) intensity.

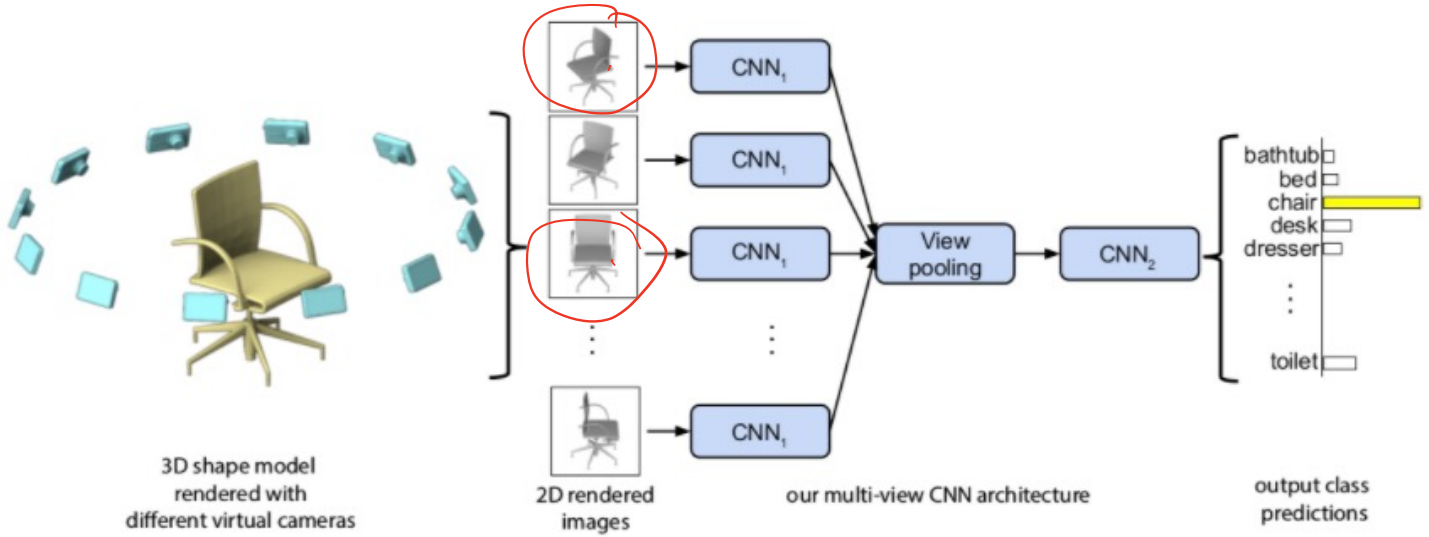




NYU

Multi-View CNN

- Treat it as a 2D problem.
- Aggregate the views together with a max-pooling layer.

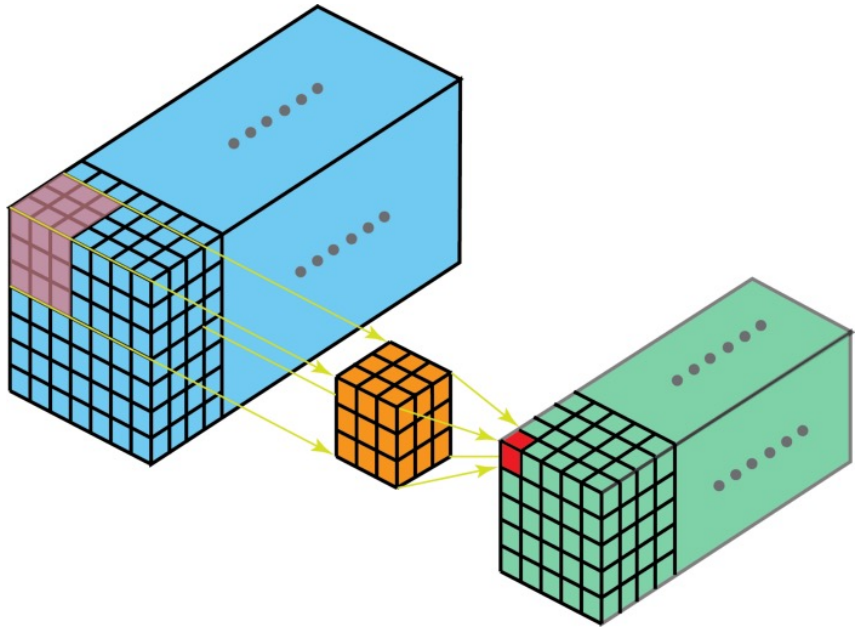


Su et al., Multi-view convolutional neural networks for 3D shape recognition, ICCV 2015.

NYU

3D Convolution on Voxels

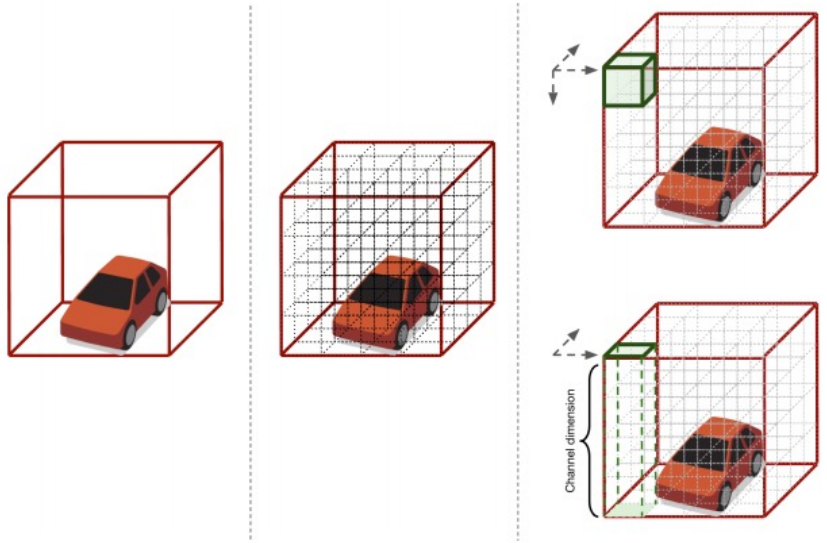
- 3D convolution on occupancy voxels.
- This can be expensive (memory + compute).



NYU

Bird's Eye View (BEV) Voxel

- Treat the z-dimension as different channels.

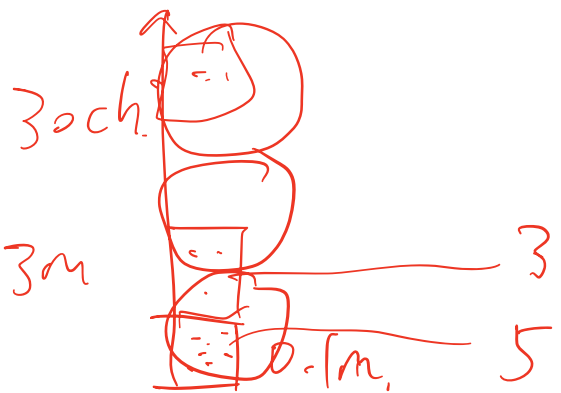


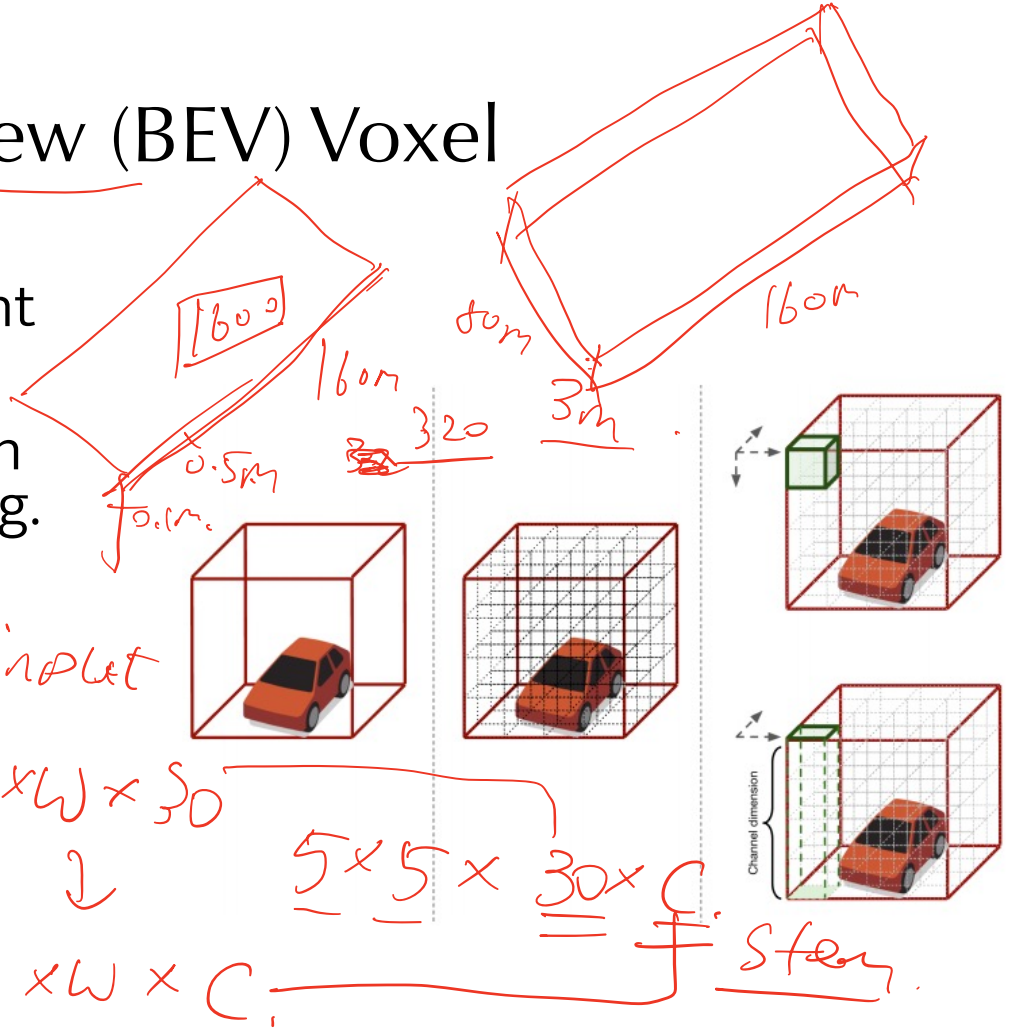
Zhang et al. Efficient convolutions for real-time semantic segmentation of 3D point clouds, 3DV 2018.1

NYU

Bird's Eye View (BEV) Voxel

- Treat the z-dimension as different channels.
- 3D convolution → 2D convolution
  Popular in self-driving domain, e.g. 80m x



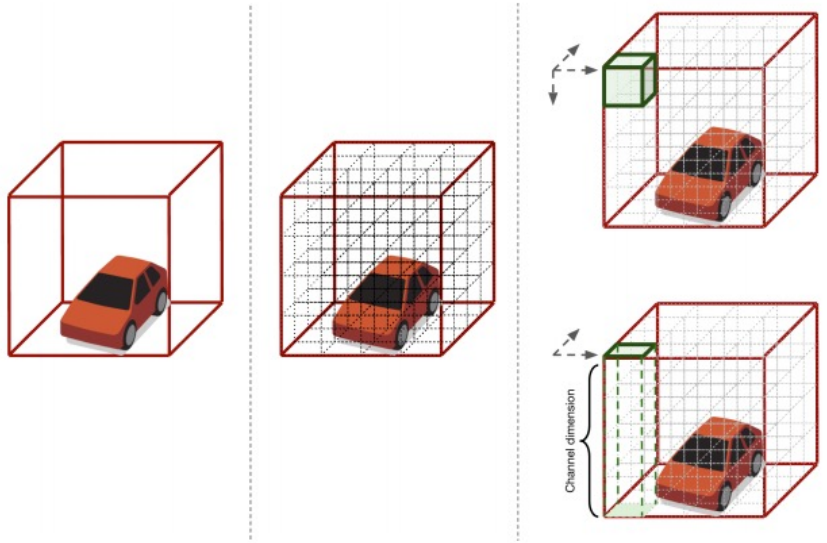


Zhang et al. Efficient convolutions for real-time semantic segmentation of 3D point clouds, 3DV 2018.1

NYU

Bird's Eye View (BEV) Voxel

- Treat the z-dimension as different channels.
- 3D convolution ⟶ 2D convolution
  Popular in self-driving domain, e.g. 80m x
- 140m x 3m (very thin!)

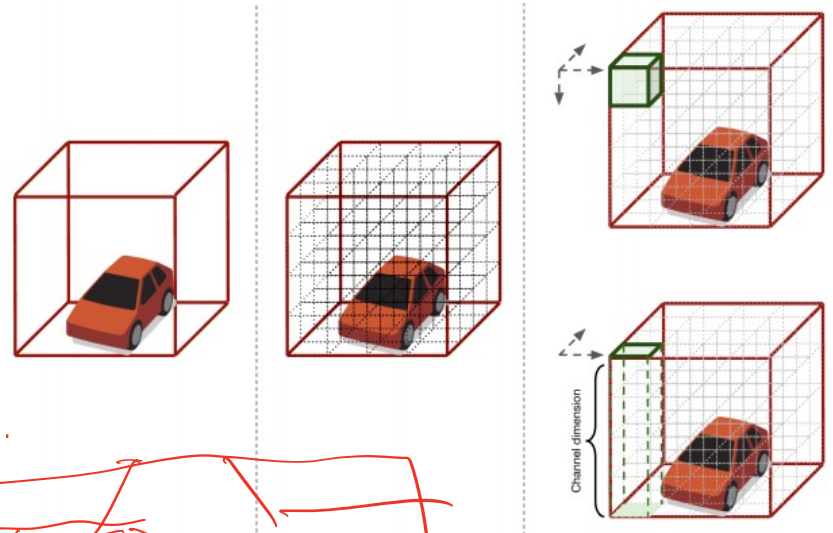


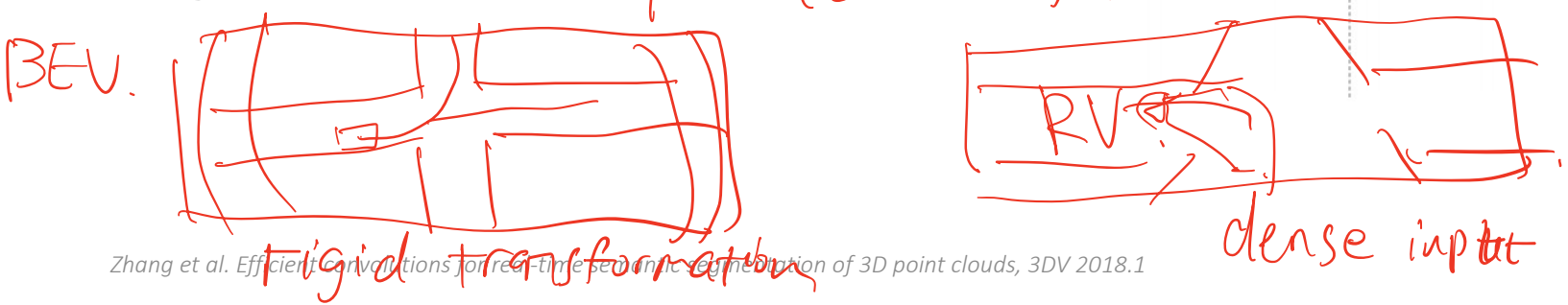
Zhang et al. Efficient convolutions for real-time semantic segmentation of 3D point clouds, 3DV 2018.1

NYU

Bird's Eye View (BEV) Voxel

- Treat the z-dimension as different channels.
- 3D convolution → 2D convolution
  Popular in self-driving domain, e.g. 80m x
- 140m x 3m (very thin!)
- Transformation in x-y plane is still rigid.



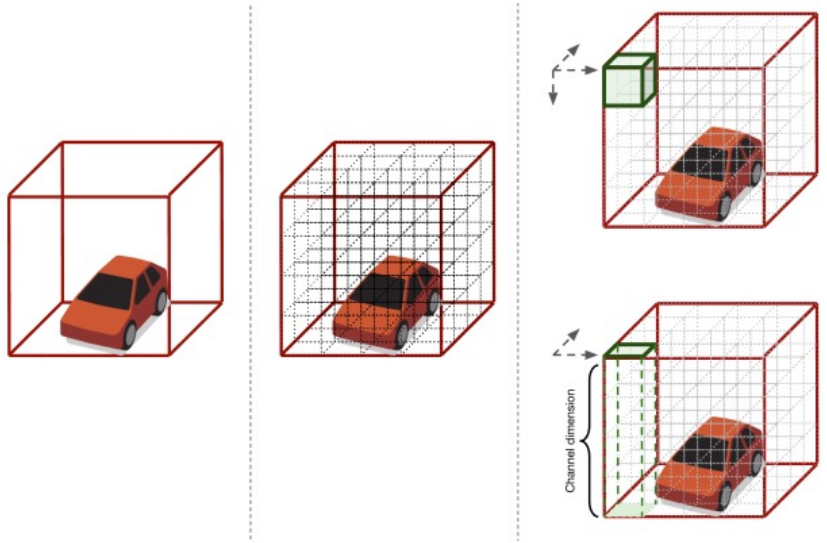


Zhang et al. Efficient convolutions for real-time semantic segmentation of 3D point clouds, 3DV 2018.1

NYU

Bird's Eye View (BEV) Voxel

- Treat the z-dimension as different channels.
- 3D convolution ⟶ 2D convolution
  Popular in self-driving domain, e.g. 80m x
- 140m x 3m (very thin!)
- Transformation in x-y plane is still rigid.
- Bird's eye view: Top down representation of the scene (rigid, sparse) vs. Range view (non-rigid, dense)

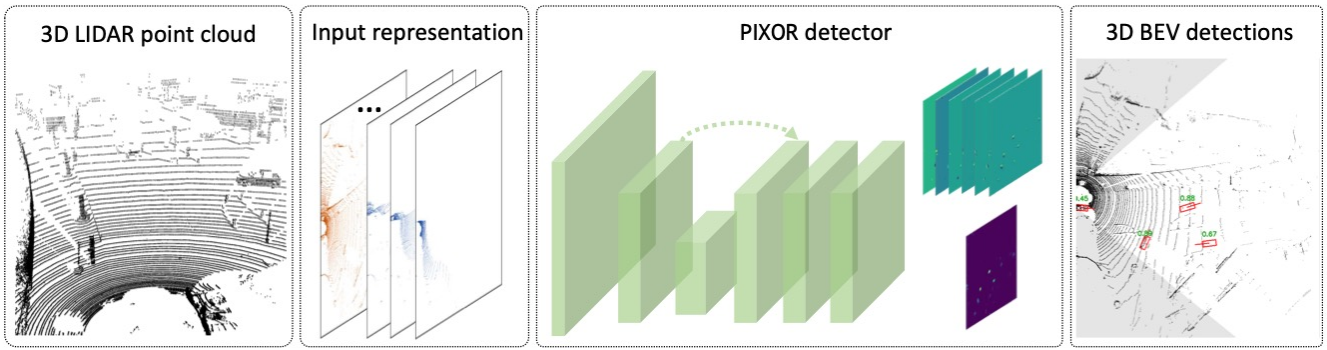


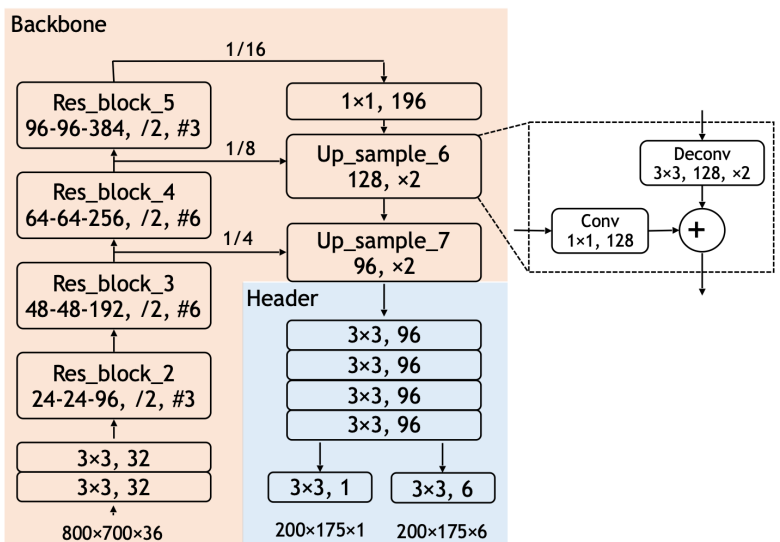
Zhang et al. Efficient convolutions for real-time semantic segmentation of 3D point clouds, 3DV 2018.1

NYU

PIXOR

- Bird's eye view object detection.



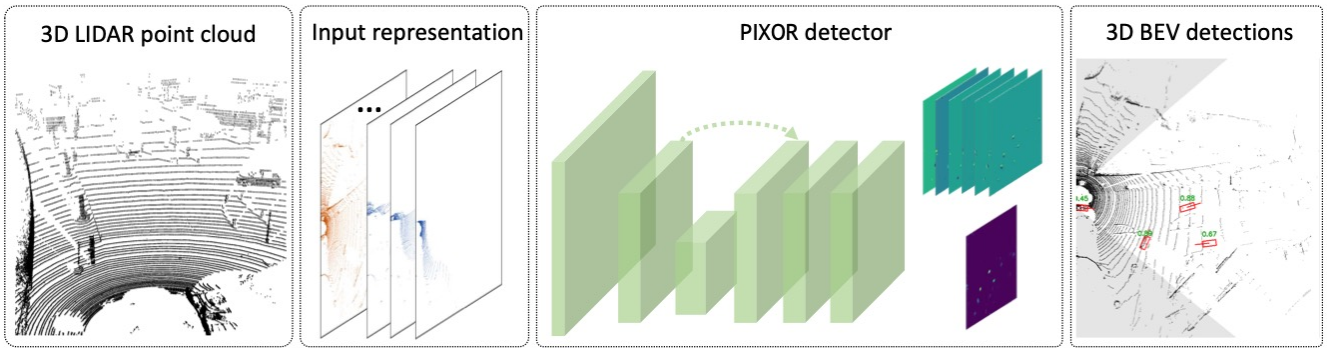


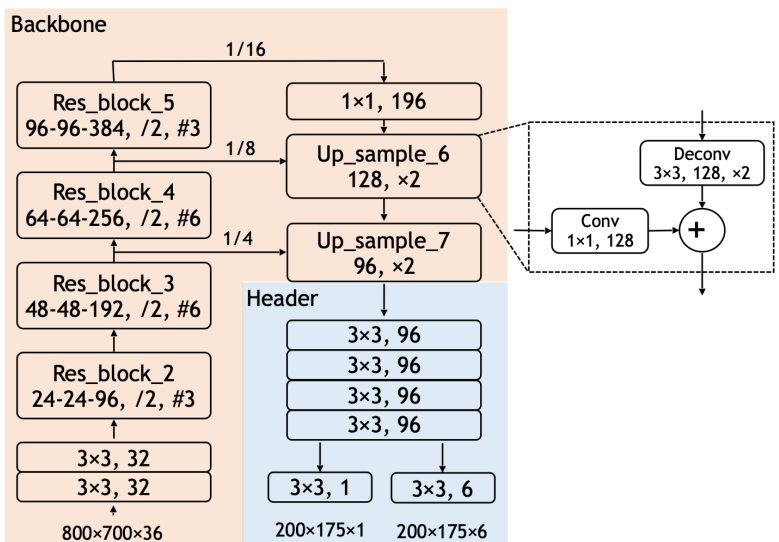
Yang et al. PIXOR: Real-time 3D Object Detection from Point Clouds. CVPR 2018.

NYU

PIXOR

- Bird's eye view object detection.
- Used the ResNet + FPN network single-stage architecture.



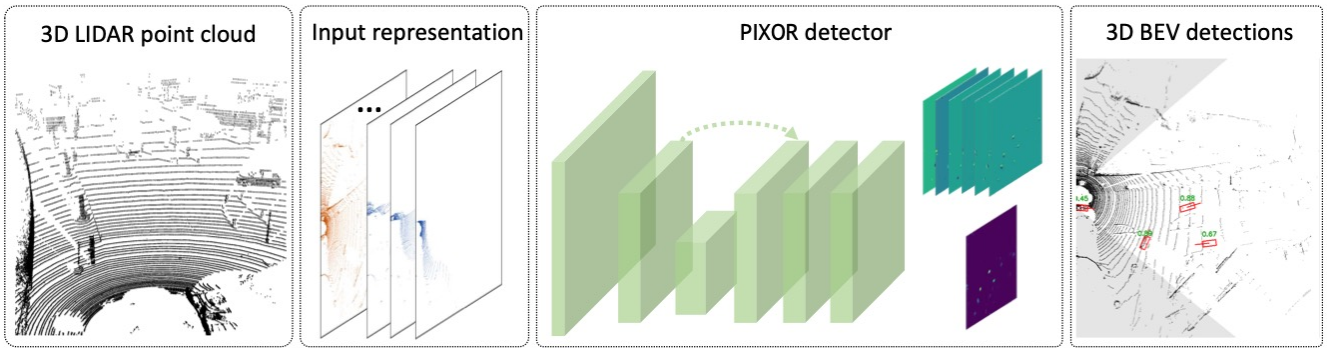


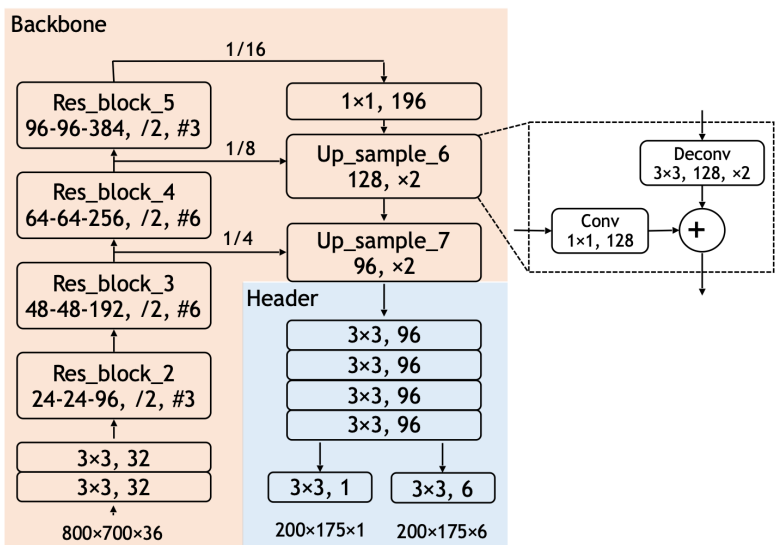
Yang et al. PIXOR: Real-time 3D Object Detection from Point Clouds. CVPR 2018.

NYU

PIXOR

- Bird's eye view object detection.
- Used the ResNet + FPN network single-stage architecture.
- Detection: Classification + regression cosθ,sinθ,dx,dy,w,l.



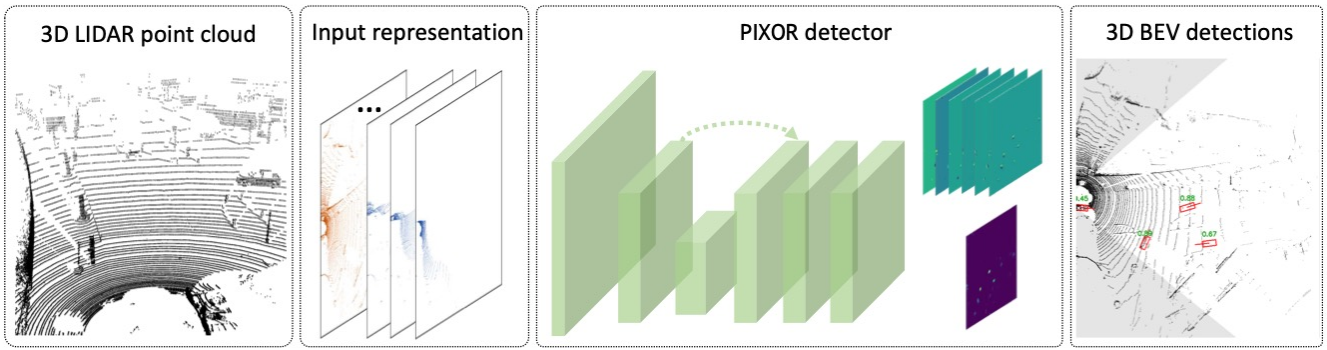


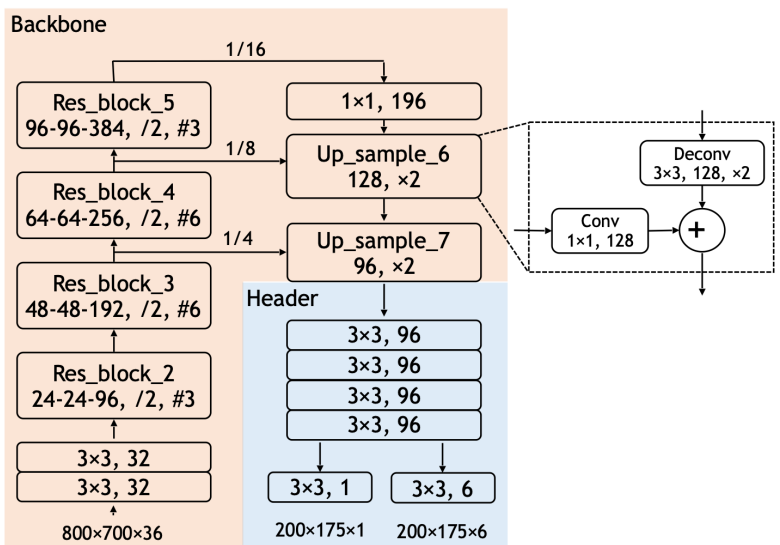
Yang et al. PIXOR: Real-time 3D Object Detection from Point Clouds. CVPR 2018.

NYU

PIXOR

- Bird's eye view object detection.
- Used the ResNet + FPN network single-stage architecture.
- Detection: Classification + regression cosθ,sinθ,dx,dy,w,l.
- First real-time 3D detection network.





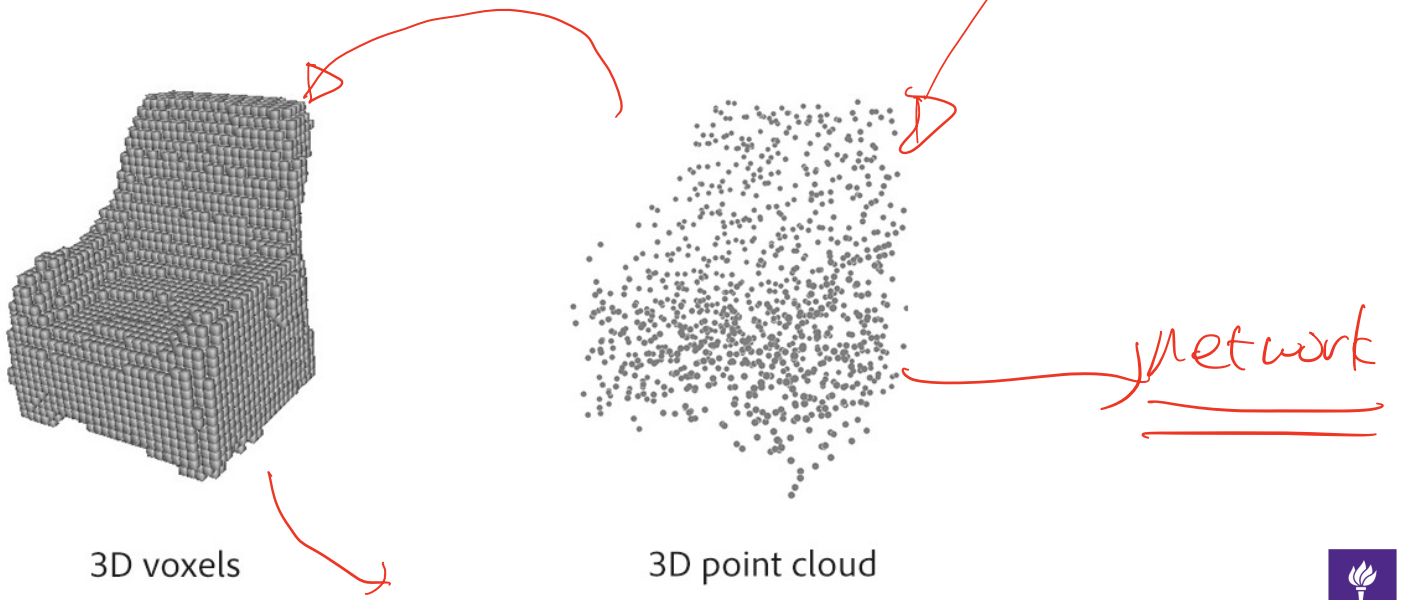
Yang et al. PIXOR: Real-time 3D Object Detection from Point Clouds. CVPR 2018.

NYU

Point Cloud

samples of 3D world

- Point cloud is native for many 3D-depth sensors: RGBD sensor, LiDAR sensor, etc.

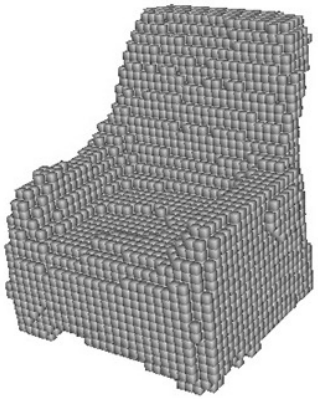


NYU

Point Cloud

- Point cloud is native for many 3D-depth sensors: RGBD sensor, LiDAR sensor, etc.
- List of 3D points: $[(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_N, y_N, z_N)]$

20000... or more.



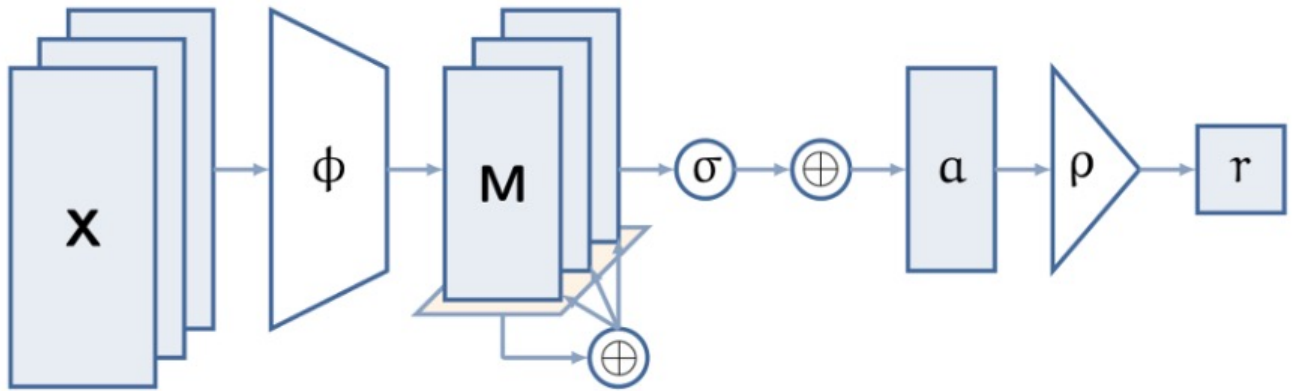
3D voxels

3D point cloud

NYU

Permutation Invariance

- Point cloud is a set.

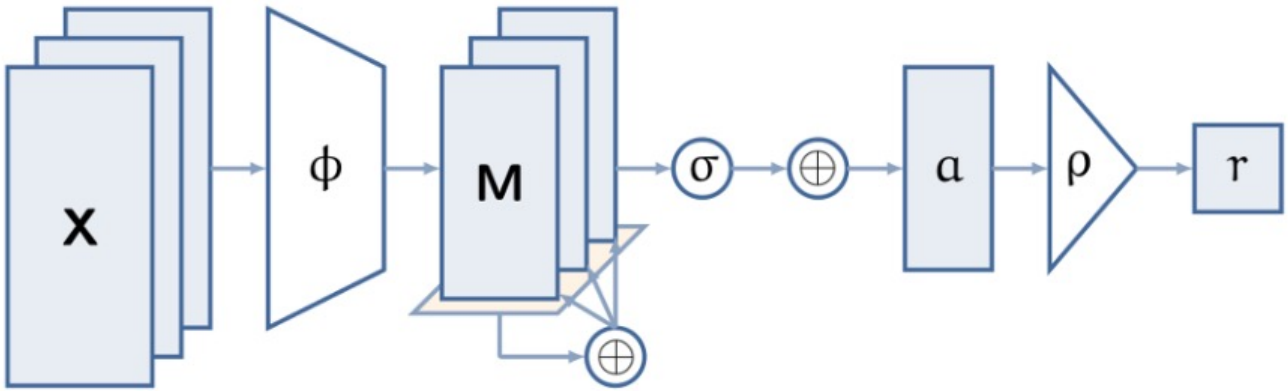


Zaheer et al. Deep Sets. NIPS 2017. https://argmax.ai/blog/setinvariance/

NYU

Permutation Invariance

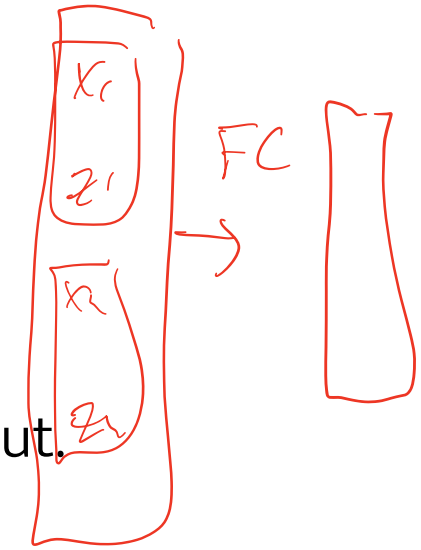
- Point cloud is a set.
- Permutation does not affect the classification in the output.



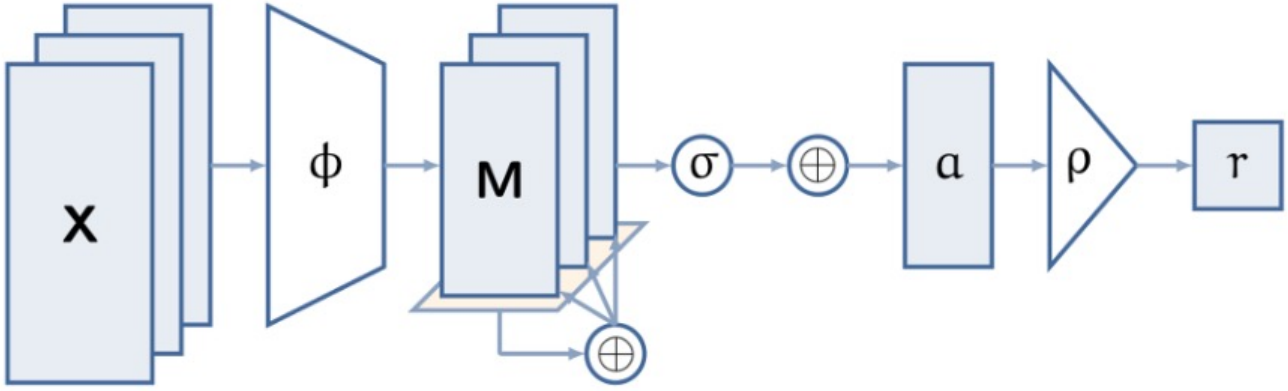
Zaheer et al. Deep Sets. NIPS 2017. https://argmax.ai/blog/setinvariance/

NYU

Permutation Invariance



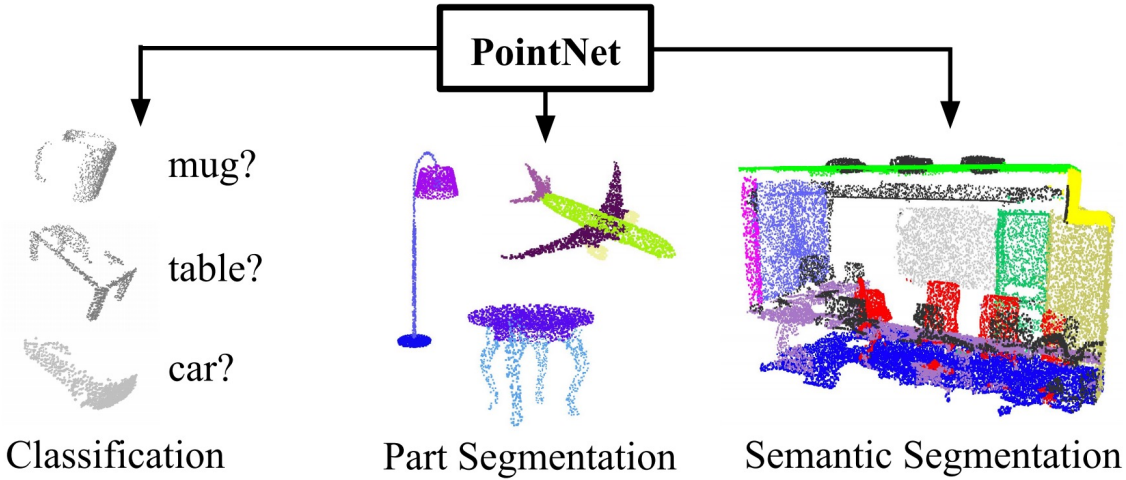
- Point cloud is a set.
- Permutation does not affect the classification in the output.
- What operations are permutation invariant?



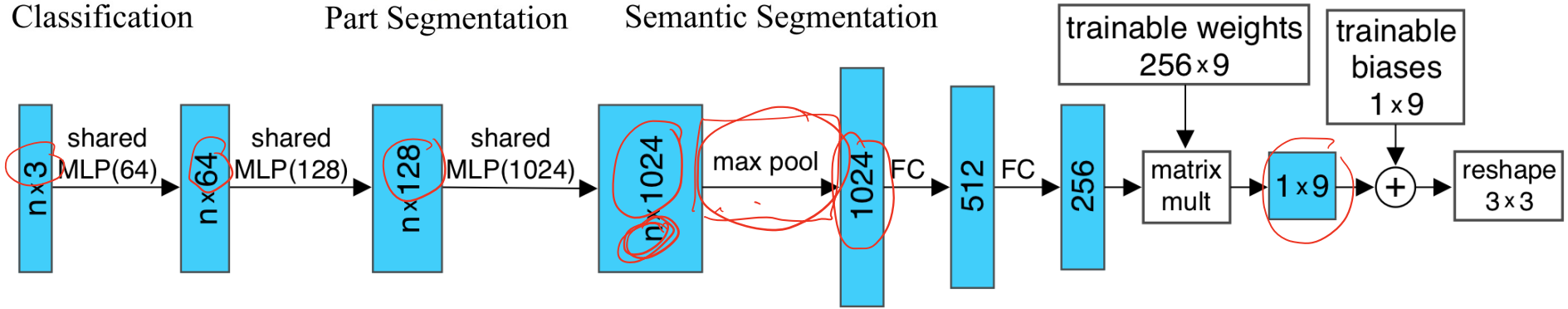
Zaheer et al. Deep Sets. NIPS 2017. https://argmax.ai/blog/setinvariance/

NYU

PointNet

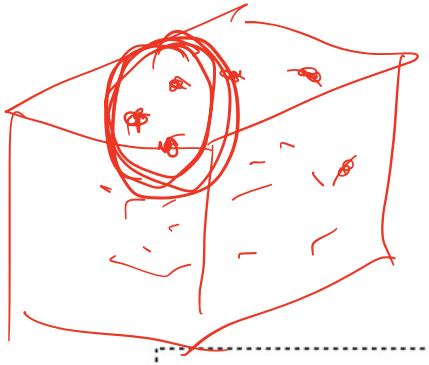


- Apply an MLP on each point.
- Max pool the features across all points.

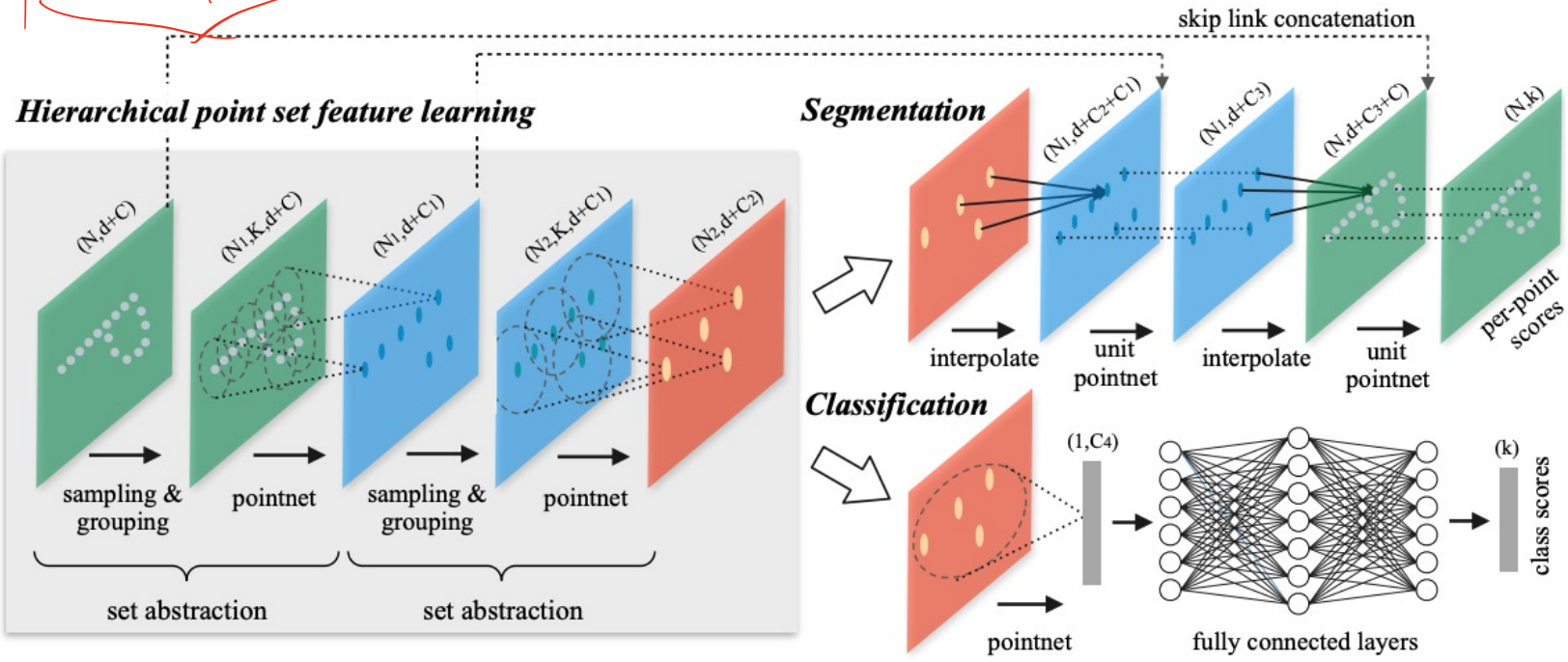


Qi et al., PointNet: Deep learning on point sets for 3D classification and segmentation, CVPR 2017.

NYU



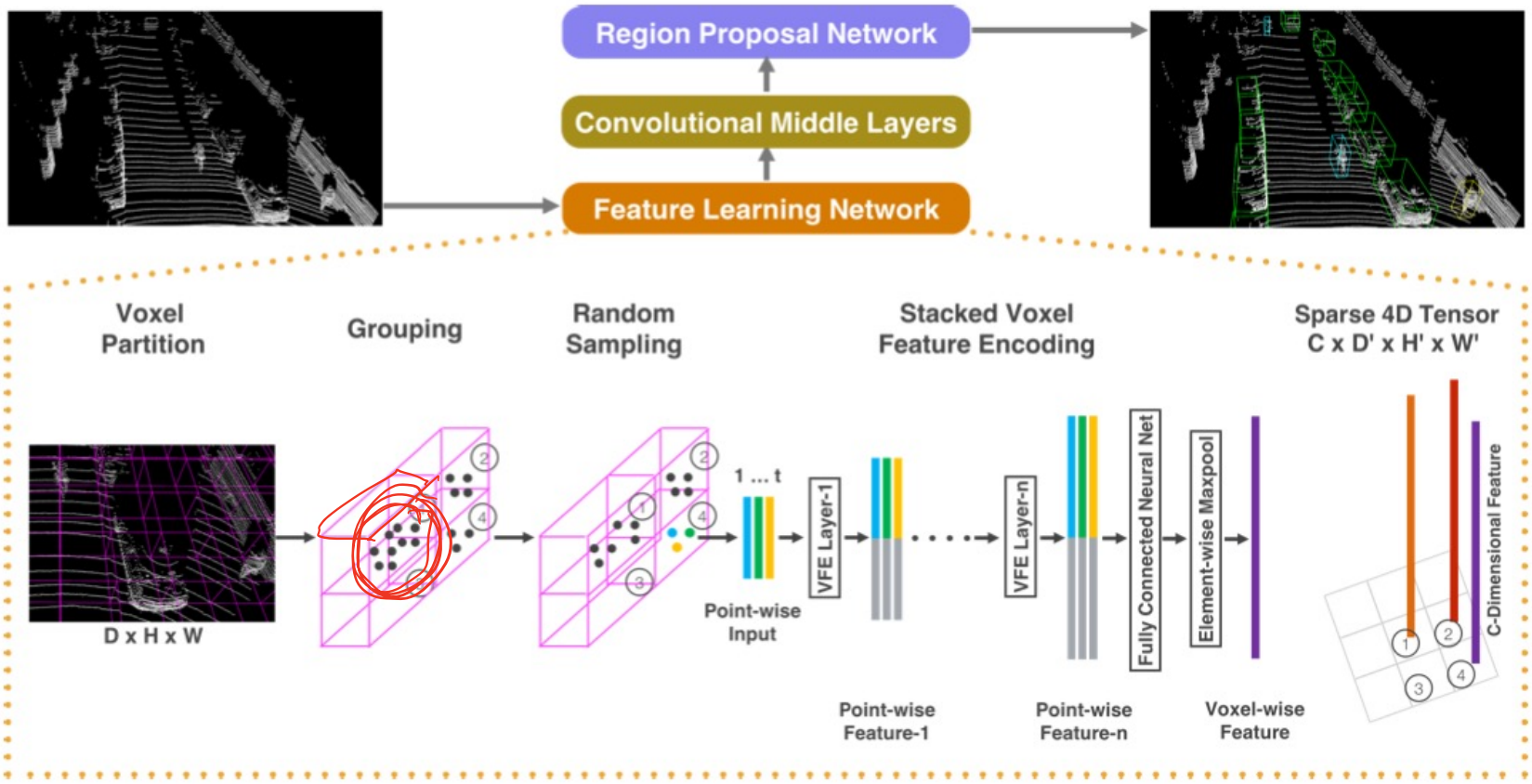
PointNet++



Qi et al., PointNet++: Deep hierarchical feature learning on point sets in a metric space, NIPS 2017.

NYU

VoxelNet

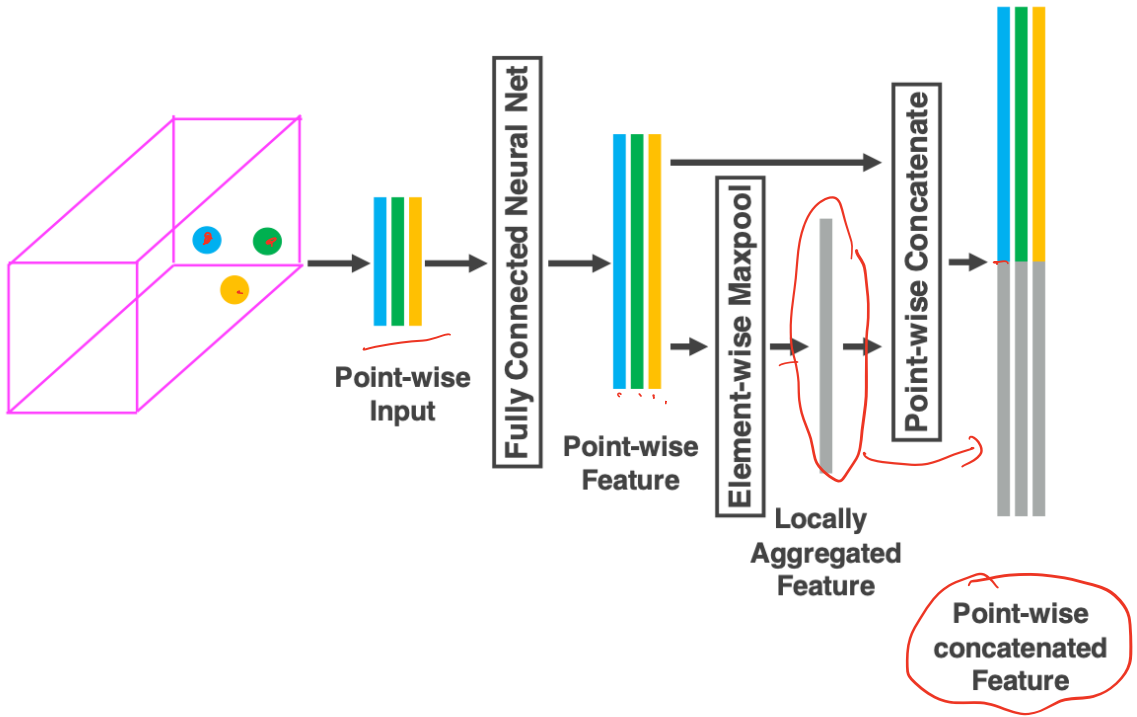


Zhou and Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. CVPR 2018.

NYU

VoxelNet

- Zooming inside voxel feature encoding (VFE)

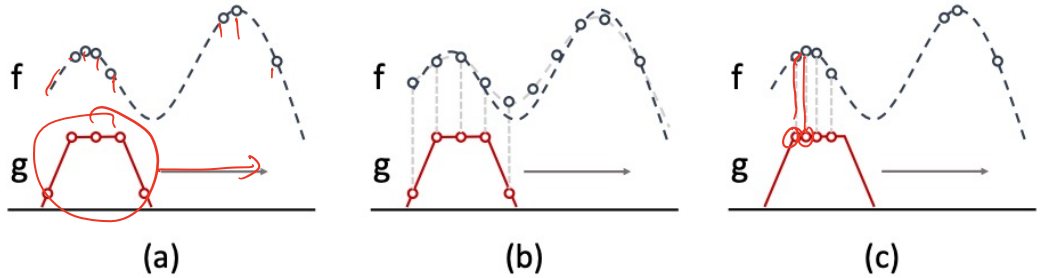


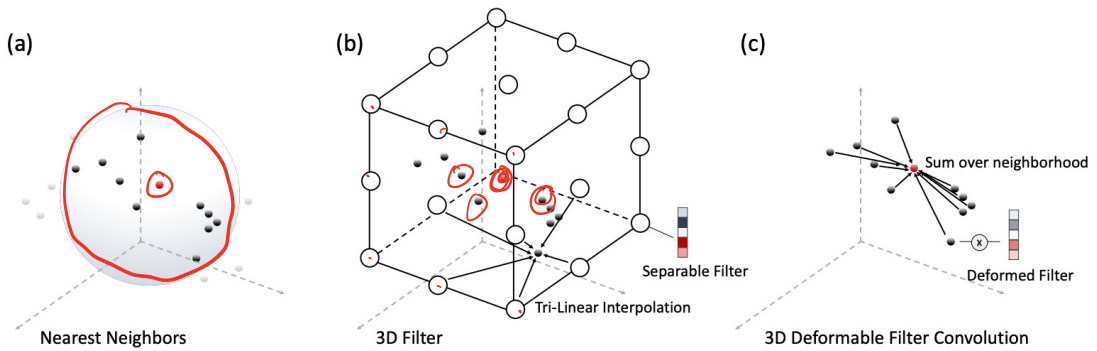
Zhou and Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. CVPR 2018.

NYU

Deformable Convolution in Point Cloud

- Can we convolve a point cloud with a spatially defined kernel function?



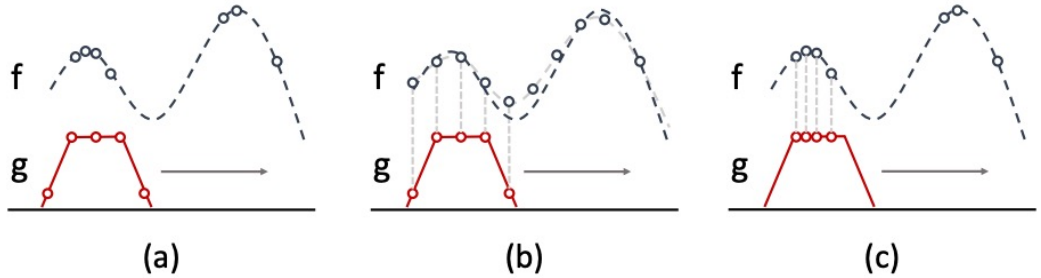


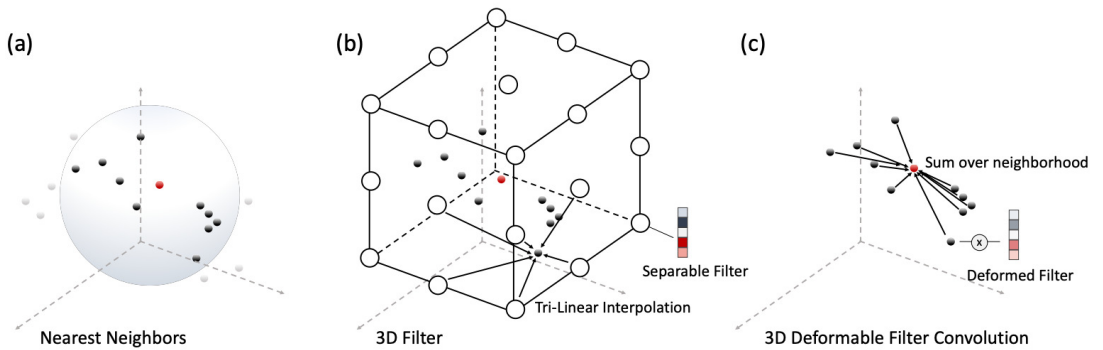
Xiong et al. Deformable Filter Convolution for Point Cloud Reasoning. 2019.

NYU

Deformable Convolution in Point Cloud

- Can we convolve a point cloud with a spatially defined kernel function?
- Resample the kernel at the point location.



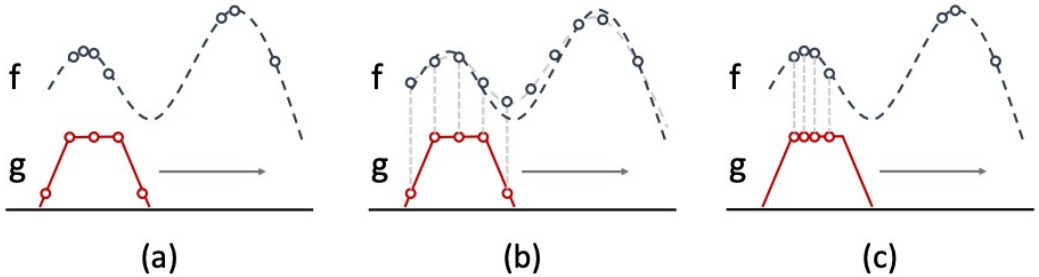


Xiong et al. Deformable Filter Convolution for Point Cloud Reasoning. 2019.

NYU

Deformable Convolution in Point Cloud

- Can we convolve a point cloud with a spatially defined kernel function?
- Resample the kernel at the point location.
- Compute the weighted sum around a neighborhood.



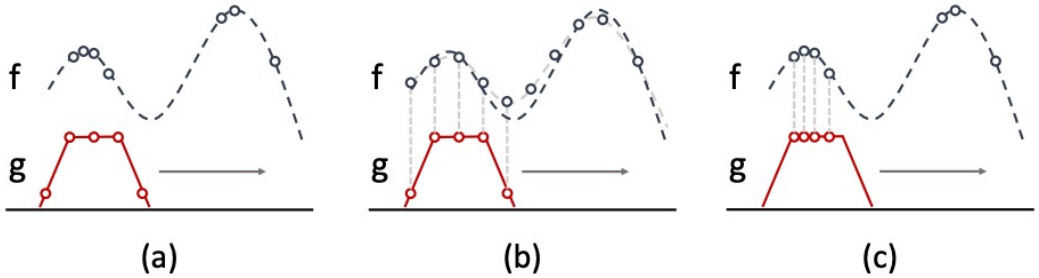
Xiong et al. Deformable Filter Convolution for Point Cloud Reasoning. 2019.

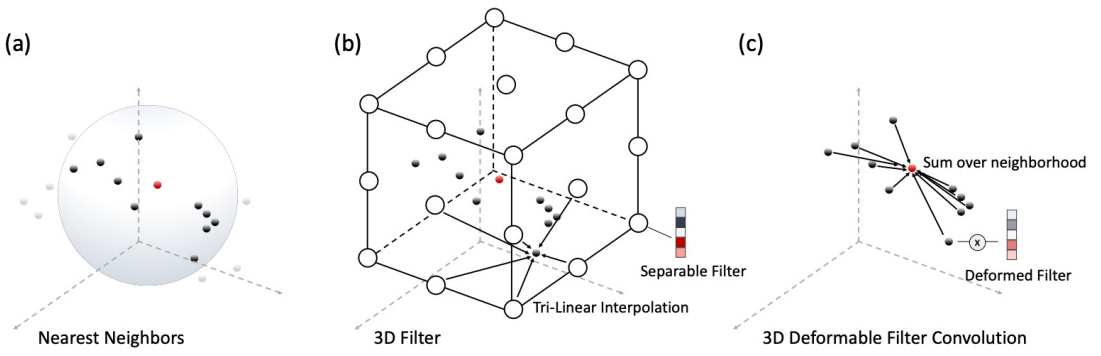
NYU

(x, y, z).

Deformable Convolution in Point Cloud

- Can we convolve a point cloud with a spatially defined kernel function?
- Resample the kernel at the point location.
- Compute the weighted sum around a neighborhood.
- Translational equivariance.



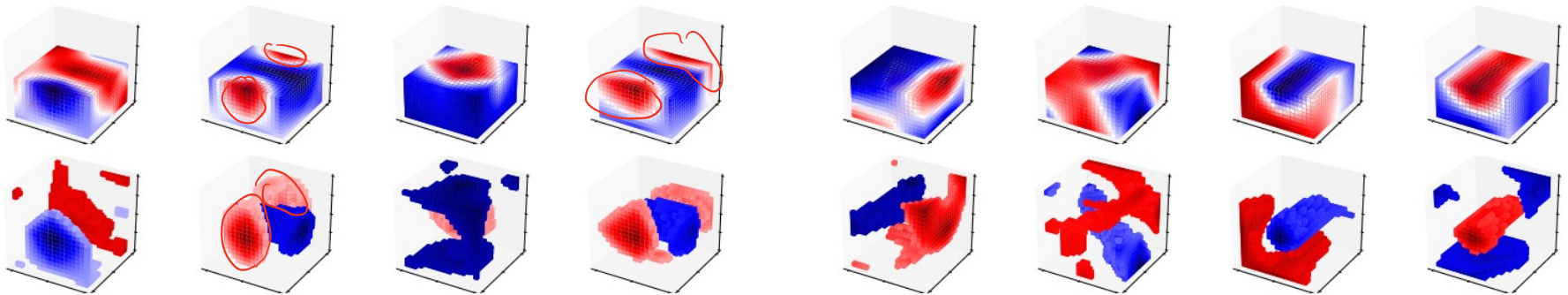


Xiong et al. Deformable Filter Convolution for Point Cloud Reasoning. 2019.

NYU

3D Filters

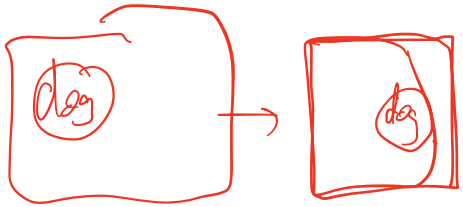
- Visualizing 3D convolution kernels.



Xiong et al. Deformable Filter Convolution for Point Cloud Reasoning. 2019.

NYU

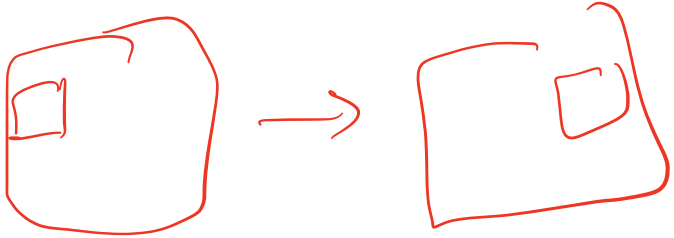
More on Equivariance



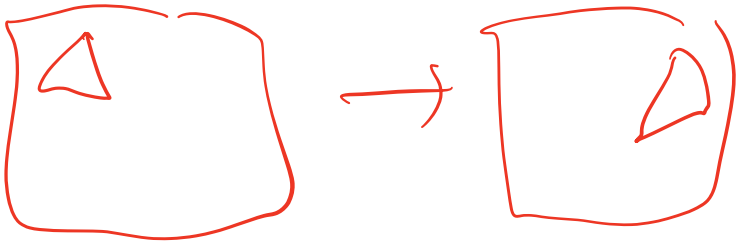
- Equivariance is a symmetry property on functions and transformations.

$$f(\mathcal{T}x) = \mathcal{T}'f(x)$$

translation



rotation



H x W x C.

rotate.

new feature map.

More on Equivariance

- Equivariance is a symmetry property on functions and transformations.

$$f(\mathcal{T}x) = \mathcal{T}' f(x)$$

- Useful property to constrain the hidden representation space.

data efficiency, ⟷ Compute efficiency (not always).

More on Equivariance

- Equivariance is a symmetry property on functions and transformations.

$$f(\mathcal{T}x) = \mathcal{T}'f(x)$$

- Useful property to constrain the hidden representation space.
- Convolution has the property of translational equivariance. Translated input = translated output. (up to discretization)

More on Equivariance

- Equivariance is a symmetry property on functions and transformations.

$$f(\mathcal{T}x) = \mathcal{T}'f(x)$$

- Useful property to constrain the hidden representation space.
- Convolution has the property of translational equivariance. Translated input = translated output. (up to discretization)
- 3D convolution handles translational equivariance.

NYU

More on Equivariance

- Equivariance is a symmetry property on functions and transformations.

$$f(\mathcal{T}x) = \mathcal{T}'f(x)$$

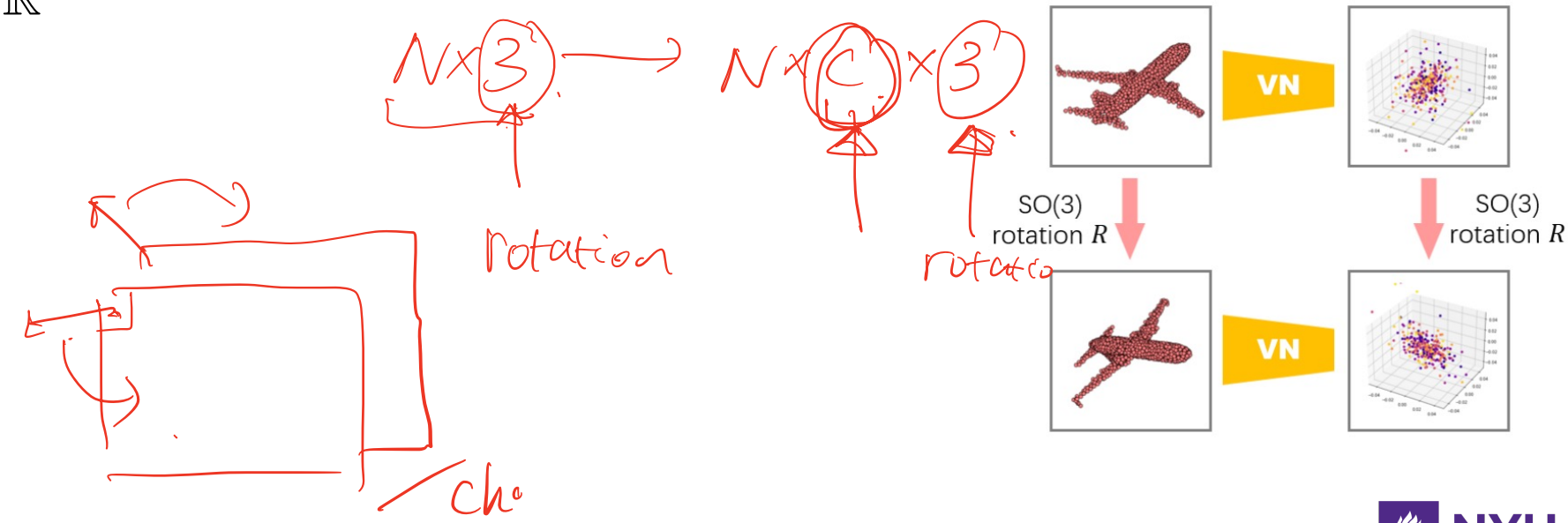
- Useful property to constrain the hidden representation space.
- Convolution has the property of translational equivariance. Translated input = translated output. (up to discretization)
- 3D convolution handles translational equivariance.
- What about rotation?

NYU

3D Rotational Equivariance

- For object centered point clouds, rotation are more important.
- Imagine every feature has a vector / arrow.

$V \in \mathbb{R}^{N \times C \times 3}$



Deng et al. Vector Neurons: A General Framework for SO(3)-Equivariant Networks. ICCV 2021.

NYU

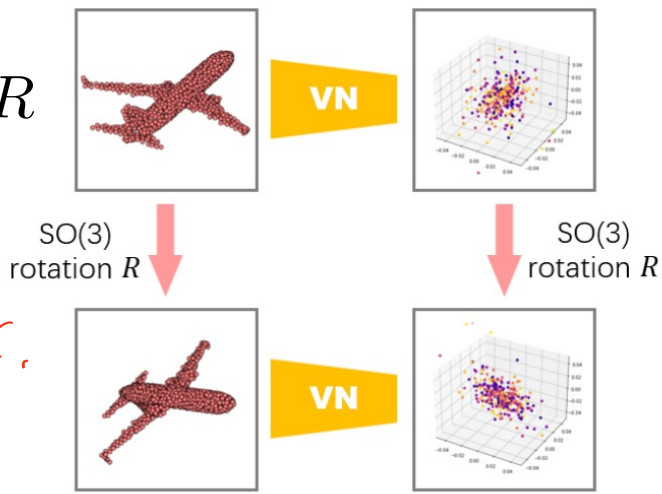
3D Rotational Equivariance

- For object centered point clouds, rotation are more important.
- Imagine every feature has a vector / arrow.
  $V \in \mathbb{R}^{N \times C \times 3}$
- Linear layers are already good $W(VR) = (WV)R$

linear. layer.

$=$ $\neq$

rotation matrix.

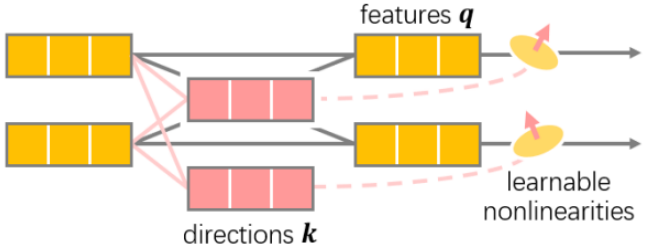


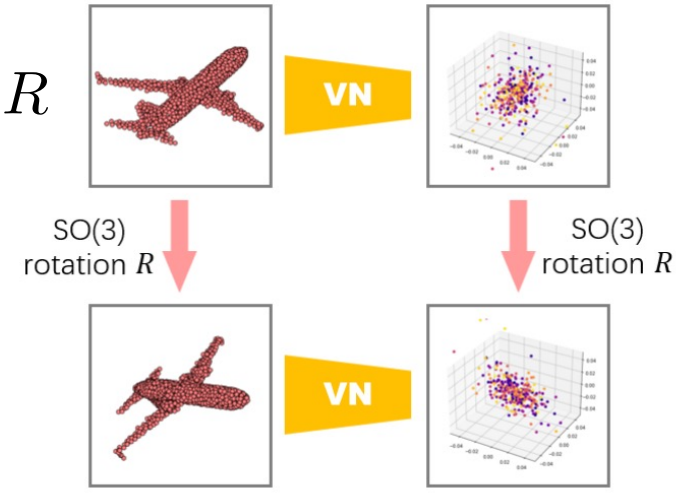
3D Rotational Equivariance

- For object centered point clouds, rotation are more important.
- Imagine every feature has a vector / arrow.
  $V \in \mathbb{R}^{N \times C \times 3}$
- Linear layers are already good $W(VR) = (WV)R$
- Need a different nonlinearity

$$\boldsymbol{q} = \mathbf{W}\boldsymbol{V}, \quad \boldsymbol{k} = \mathbf{U}\boldsymbol{V}$$

$$\boldsymbol{v}' = \begin{cases} \boldsymbol{q} & \text{if } \langle \boldsymbol{q}, \boldsymbol{k} \rangle \geqslant 0 \\ \boldsymbol{q} - \left\langle \boldsymbol{q}, \frac{\boldsymbol{k}}{\|\boldsymbol{k}\|} \right\rangle \frac{\boldsymbol{k}}{\|\boldsymbol{k}\|} & \text{otherwise,} \end{cases}$$





Deng et al. Vector Neurons: A General Framework for SO(3)-Equivariant Networks. ICCV 2021.

NYU

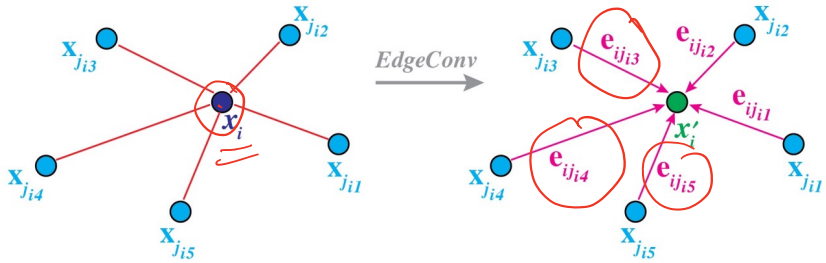
More General Equivariance

- Making VN also translational equivariant.

Edge. Conv → VN

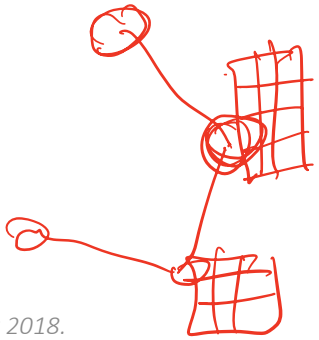
translational eq.

rot eq.



[Wang et al. 2019]

- For higher order equivariance (e.g. matrices, tensors), check out Tensor Field Networks.
- Convert features into spherical harmonics.



Wang et al. Dynamic Graph CNN for Learning on Point Clouds. TOG 2019.
Thomas et al. Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds. arXiv 2018.

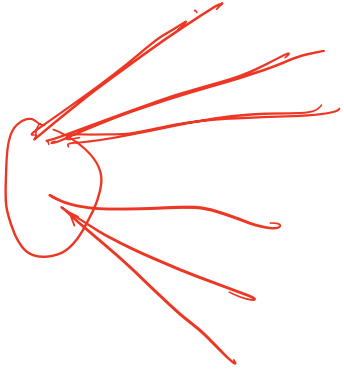
NYU

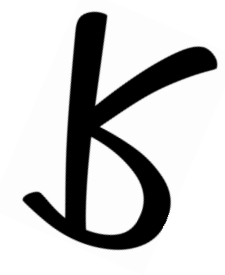
Challenges and Tradeoffs

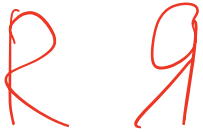
- Computational and memory efficiency vs. training data efficiency

Challenges and Tradeoffs

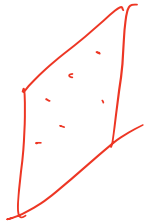
- Computational and memory efficiency vs. training data efficiency
- Mental rotation: Rotate in the latent space may not be as simple as a geometric transformation. The skill may still be experience driven.



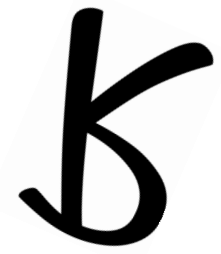




Challenges and Tradeoffs



- Computational and memory efficiency vs. training data efficiency
- Mental rotation: Rotate in the latent space may not be as simple as a geometric transformation. The skill may still be experience driven.



- Neuroscience evidence: No spatial equivariance in connections. The LGN performs “pre-pre-training” from simulating retina waves [Blankenship & Feller, 2009].

NYU

Multi-Sensor Fusion

- LiDAR is precise in depth perception, but the point cloud format is sparse and non-uniform (dense around the ego-car and sparse in long distance.)

NYU

Multi-Sensor Fusion

- LiDAR is precise in depth perception, but the point cloud format is sparse and non-uniform (dense around the ego-car and sparse in long distance.)
- Camera provides high resolution 2D view and good for long distance but lacks 3D. Can we achieve the best of both worlds?

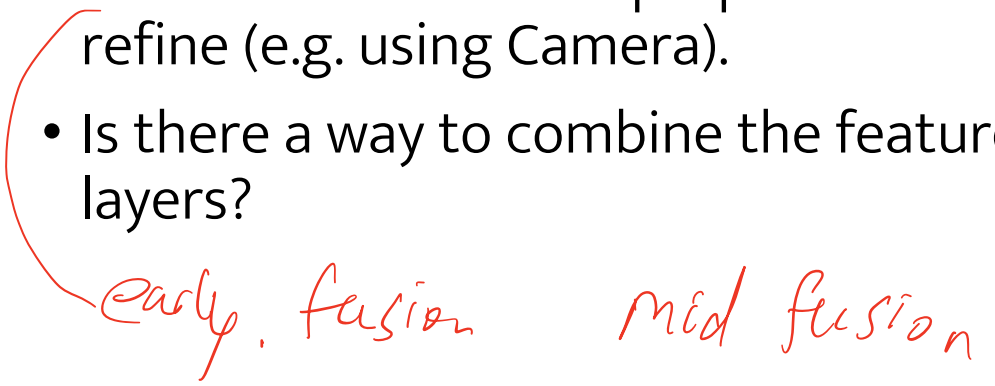
NYU

Multi-Sensor Fusion

- LiDAR is precise in depth perception, but the point cloud format is sparse and non-uniform (dense around the ego-car and sparse in long distance.)
- Camera provides high resolution 2D view and good for long distance but lacks 3D. Can we achieve the best of both worlds?
- Late fusion: Generate proposals from one branch (e.g. LiDAR) and refine (e.g. using Camera).

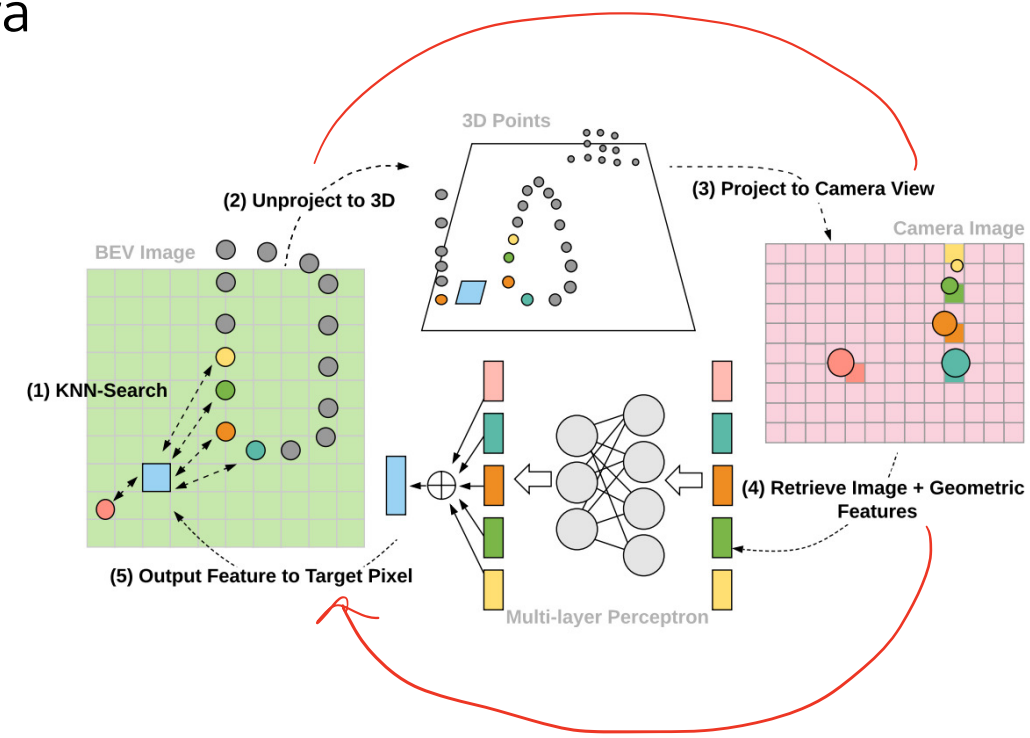
Multi-Sensor Fusion

- LiDAR is precise in depth perception, but the point cloud format is sparse and non-uniform (dense around the ego-car and sparse in long distance.)
- Camera provides high resolution 2D view and good for long distance but lacks 3D. Can we achieve the best of both worlds?
- Late fusion: Generate proposals from one branch (e.g. LiDAR) and refine (e.g. using Camera).
- Is there a way to combine the features from both modality in lower layers?



Camera-LiDAR Projection

- Unproject LiDAR points to camera view (i.e. Range View)
- Query the closest camera RGB features for each LiDAR point.
- For empty space in BEV, we can interpolate from neighboring points using kNN.
- Continuous Fusion: $h_i = \sum_j MLP([f_j, x_j - x_i])$.

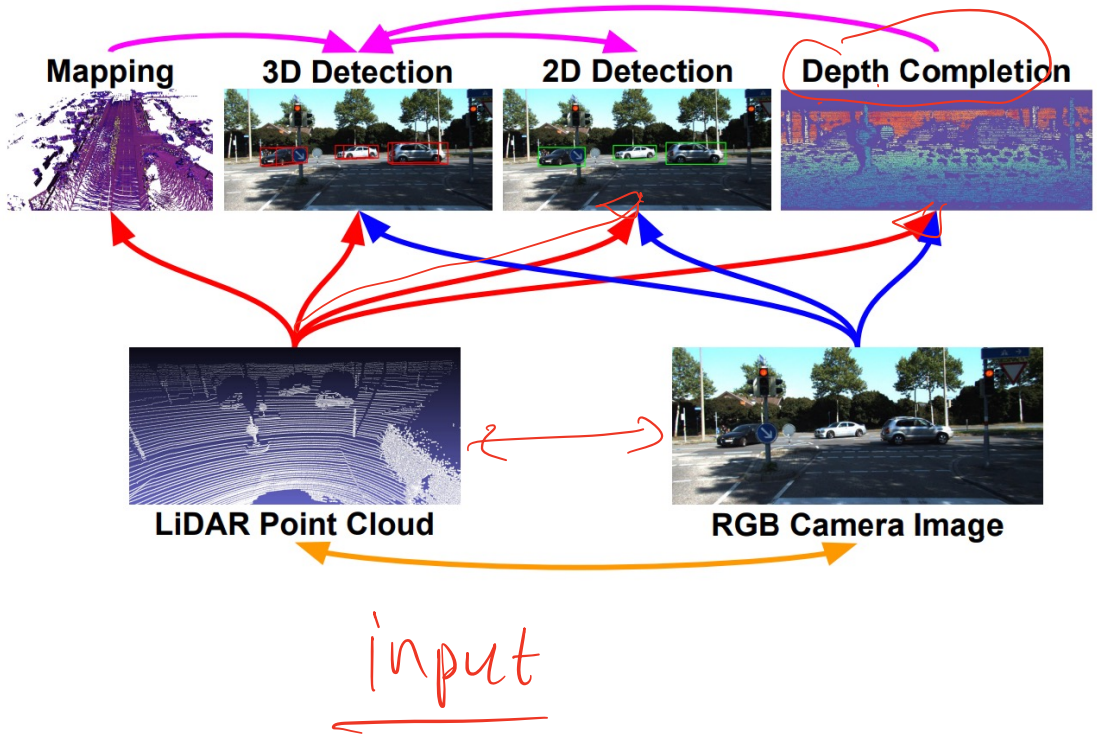


Liang et al. Deep Continuous Fusion for Multi-Sensor 3D Object Detection. ECCV 2018.

NYU

Supervised Dense Depth

- Drawback of continuous fusion: Sparse LiDAR can cause the fusion process to be less accurate. Relies on kNN.

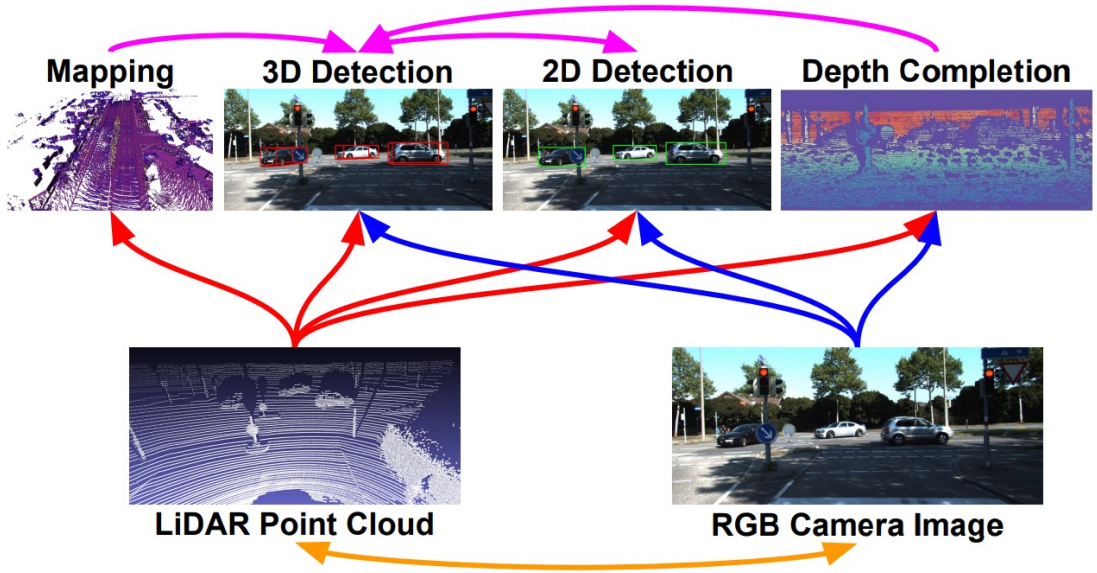


Liang et al. Multi-Task Multi-Sensor Fusion for 3D Object Detection. CVPR 2019.

NYU

Supervised Dense Depth

- Drawback of continuous fusion: Sparse LiDAR can cause the fusion process to be less accurate. Relies on kNN.
- Why not predict a dense depth to pair with the camera image?

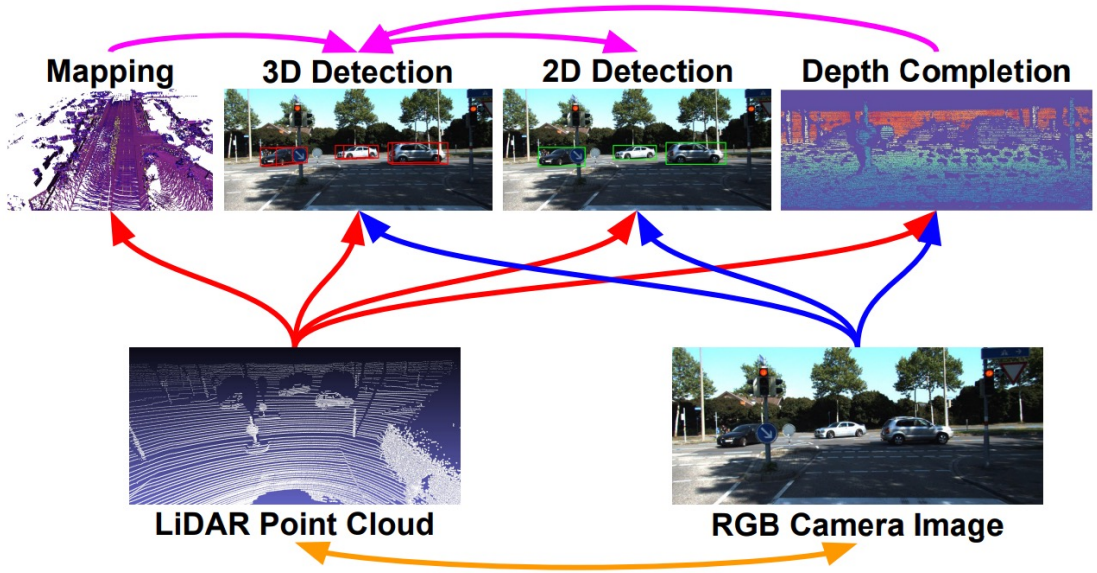


Liang et al. Multi-Task Multi-Sensor Fusion for 3D Object Detection. CVPR 2019.

NYU

Supervised Dense Depth

- Drawback of continuous fusion: Sparse LiDAR can cause the fusion process to be less accurate. Relies on kNN.
- Why not predict a dense depth to pair with the camera image?
- Depth completion module is supervised by sparse LiDAR and is used for dense fusion.



Liang et al. Multi-Task Multi-Sensor Fusion for 3D Object Detection. CVPR 2019.

NYU

3D Perception

- With the ease of use of automatic differentiation libraries, we can compose a computation graph in millions of ways.

NYU

3D Perception

- With the ease of use of automatic differentiation libraries, we can compose a computation graph in millions of ways.
- We can design layers and operators to accomodate different types of inputs and outputs. 3D, point cloud, sparse data, etc.

NYU

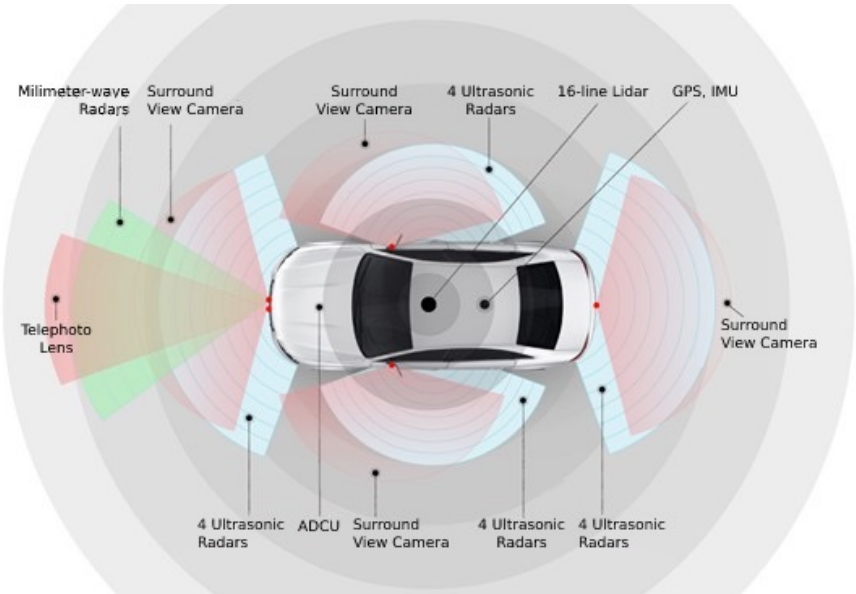
3D Perception

- With the ease of use of automatic differentiation libraries, we can compose a computation graph in millions of ways.
- We can design layers and operators to accomodate different types of inputs and outputs. 3D, point cloud, sparse data, etc.
- We can fuse different modalities together too, by leveraging geometric relationships.

NYU

2D to 3D

- Not all embodied agents have the luxury to have a full set of sensors.
- Can we infer the geometric structure with 2D perception?

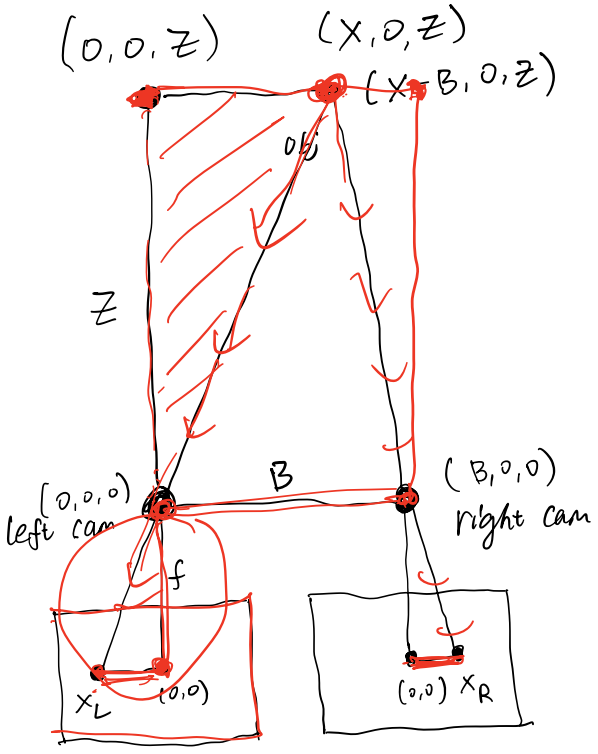


NYU

$(x, y) \rightarrow z$

Classic Vision on Depth and Disparity

- One source of depth is from the displacement of pixels in a stereo setup.
- But we need to estimate disparity.



$$x_L = \frac{Xf}{z}$$

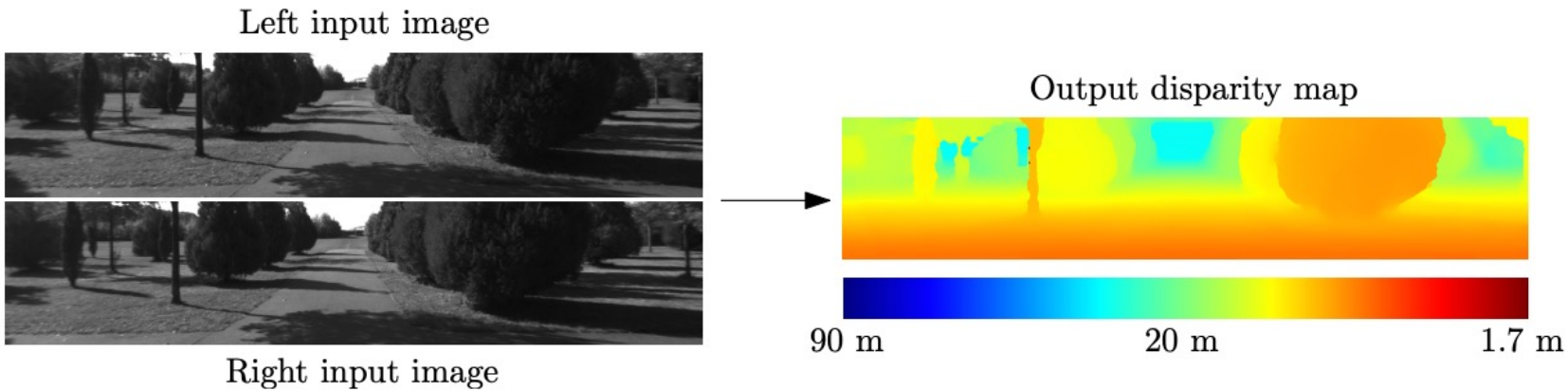
$$x_R = \frac{(X-B)f}{z}$$

$$\delta = x_L - x_R = \frac{[X-(X-B)]f}{z} = \frac{Bf}{z}$$

$$z = \frac{Bf}{\delta}$$

From 2D to 3D: Depth Network

- A network that can output disparity.
- Using LiDAR or depth camera as groundtruth supervision.



Zbontar & LeCun. Computing the Stereo Matching Cost with a Convolutional Neural Network. CVPR 2015.

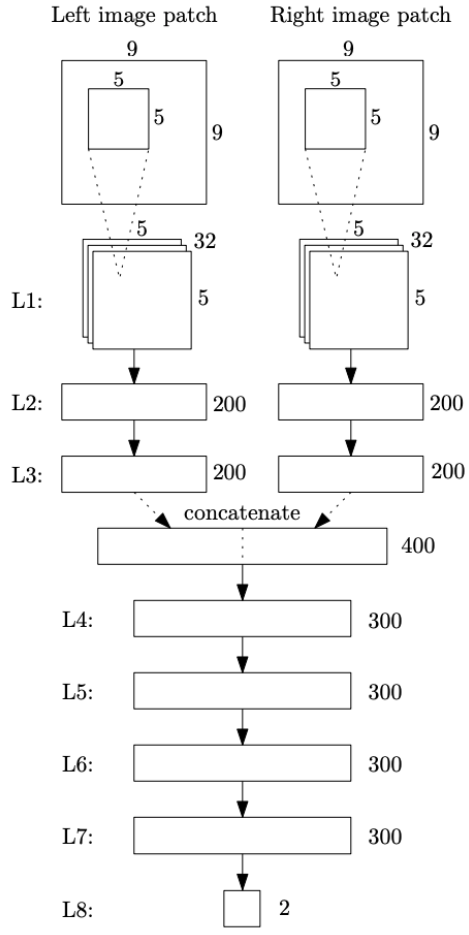
NYU

The Energy-Based Approach

- The energy penalize matching with high cost (unary), and when neighboring pixels have disparity differences greater or equal to one (pairwise).
- Cost network: Train with binary classification

Energy $$E(D) = \sum_{\mathbf{p}} \Big(C^4_{\text{CBCA}}(\mathbf{p}, D(\mathbf{p})) + \sum_{\mathbf{q} \in \mathcal{N}_\mathbf{p}} P_1 \times 1\{|D(\mathbf{p}) - D(\mathbf{q})| = 1\} + \sum_{\mathbf{q} \in \mathcal{N}_\mathbf{p}} P_2 \times 1\{|D(\mathbf{p}) - D(\mathbf{q})| > 1\} \Big),$$

$$D(\mathbf{p}) = \underset{d}{\operatorname{argmin}}\, C(\mathbf{p}, d).$$



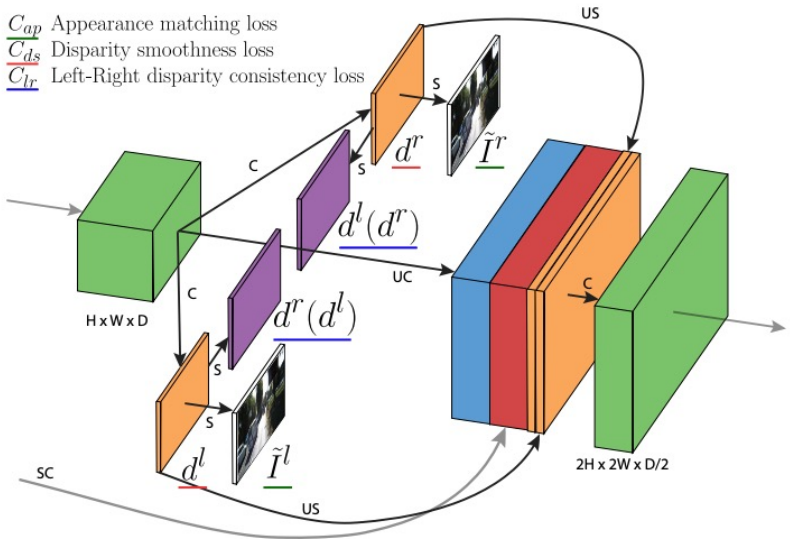
Zbontar & LeCun. Computing the Stereo Matching Cost with a Convolutional Neural Network. CVPR 2015.

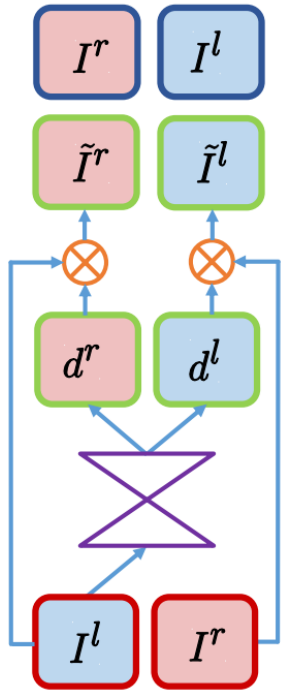
NYU

Self-Supervised Depth

- Appearance matching loss

$$C_{ap}^{l} = \frac{1}{N}\sum_{i,j}\alpha\frac{1-\text{SSIM}(I_{ij}^{l},\tilde{I}_{ij}^{l})}{2} + (1-\alpha)\left\|I_{ij}^{l}-\tilde{I}_{ij}^{l}\right\|.$$





Godard et al. Unsupervised Monocular Depth Estimation with Left-Right Consistency. CVPR 2017.

NYU

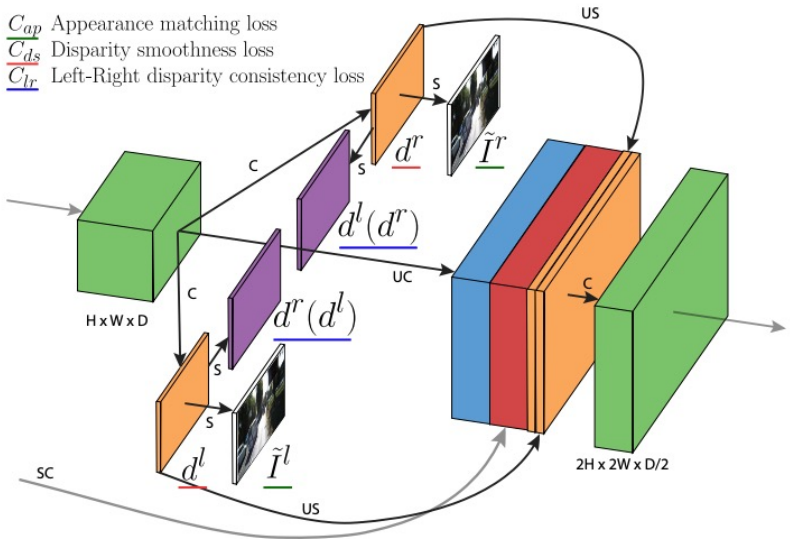
Self-Supervised Depth

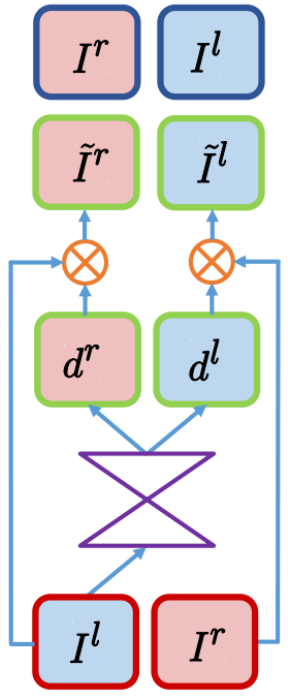
- Appearance matching loss

$$C_{ap}^{l} = \frac{1}{N}\sum_{i,j} \alpha \frac{1-\text{SSIM}(I_{ij}^{l}, \tilde{I}_{ij}^{l})}{2} + (1-\alpha)\left\| I_{ij}^{l} - \tilde{I}_{ij}^{l} \right\|.$$

- Disparity smoothness loss

$$C_{ds}^{l} = \frac{1}{N}\sum_{i,j} \left|\partial_x d_{ij}^{l}\right| e^{-\left\|\partial_x I_{ij}^{l}\right\|} + \left|\partial_y d_{ij}^{l}\right| e^{-\left\|\partial_y I_{ij}^{l}\right\|}.$$





Godard et al. Unsupervised Monocular Depth Estimation with Left-Right Consistency. CVPR 2017.

NYU

Self-Supervised Depth

- Appearance matching loss

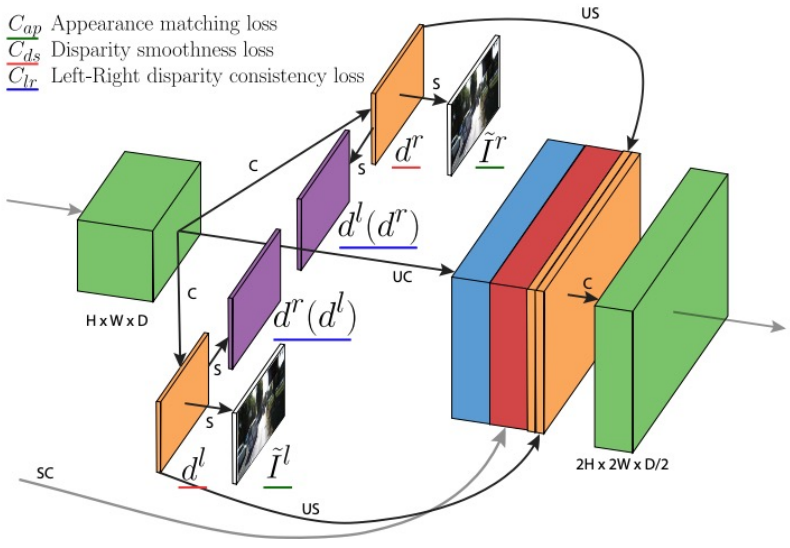
$$C_{ap}^{l} = \frac{1}{N}\sum_{i,j}\alpha\frac{1-\text{SSIM}(I_{ij}^{l},\tilde{I}_{ij}^{l})}{2} + (1-\alpha)\left\|I_{ij}^{l}-\tilde{I}_{ij}^{l}\right\|.$$

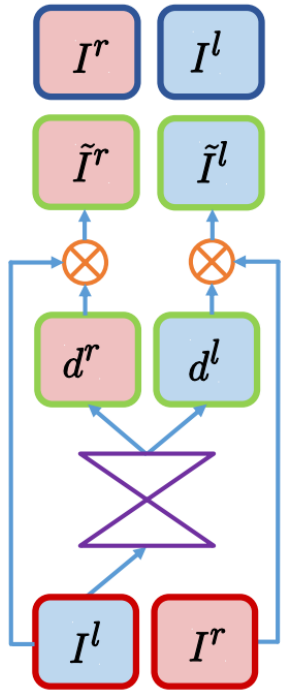
- Disparity smoothness loss

$$C_{ds}^{l} = \frac{1}{N}\sum_{i,j}\left|\partial_x d_{ij}^{l}\right|e^{-\left\|\partial_x I_{ij}^{l}\right\|} + \left|\partial_y d_{ij}^{l}\right|e^{-\left\|\partial_y I_{ij}^{l}\right\|}.$$

- Left-right disparity consistency loss

$$C_{lr}^{l} = \frac{1}{N}\sum_{i,j}\left|d_{ij}^{l}-d_{ij+d_{ij}^{l}}^{r}\right|.$$





Godard et al. Unsupervised Monocular Depth Estimation with Left-Right Consistency. CVPR 2017.

NYU

Motion, Optical Flow

- Another task that takes into a pair of image is to estimate the motion of pixels across two consecutive video frames.

Motion, Optical Flow

- Another task that takes into a pair of image is to estimate the motion of pixels across two consecutive video frames.
- Classic method uses brightness constancy assumption.

NYU

Motion, Optical Flow

- Another task that takes into a pair of image is to estimate the motion of pixels across two consecutive video frames.
- Classic method uses brightness constancy assumption.

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t).$$

NYU

Motion, Optical Flow

- Another task that takes into a pair of image is to estimate the motion of pixels across two consecutive video frames.
- Classic method uses brightness constancy assumption.

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t).$$

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t.$$

NYU

Motion, Optical Flow

- Another task that takes into a pair of image is to estimate the motion of pixels across two consecutive video frames.
- Classic method uses brightness constancy assumption.

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t).$$

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t.$$

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t = 0.$$

NYU

Motion, Optical Flow

- Another task that takes into a pair of image is to estimate the motion of pixels across two consecutive video frames.
- Classic method uses brightness constancy assumption.

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t).$$

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t.$$

$$I_x = \frac{\partial I}{\partial x}$$

$$I_y = \frac{\partial I}{\partial y}$$

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t = 0.$$

NYU

Motion, Optical Flow

- Another task that takes into a pair of image is to estimate the motion of pixels across two consecutive video frames.
- Classic method uses brightness constancy assumption.

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t).$$

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t.$$

$$I_x = \frac{\partial I}{\partial x}$$

$$I_y = \frac{\partial I}{\partial y}$$

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t = 0.$$

$$\boxed{I_x u + I_y v + I_t = 0.}$$

NYU

Classical Approach

- Under-constrained system

$$I_x u + I_y v + I_t = 0.$$

NYU

Classical Approach

- Under-constrained system
- Use a local patch and assume smooth motion

$$I_x u + I_y v + I_t = 0.$$

$$\mathbf{Au} = \mathbf{b}$$

$$\begin{pmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{N^2}) & I_y(\mathbf{p}_{N^2}) \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = - \begin{pmatrix} I_t(\mathbf{p}_1) \\ \vdots \\ I_t(\mathbf{p}_{N^2}) \end{pmatrix}$$

NYU

Classical Approach

- Under-constrained system
- Use a local patch and assume smooth motion
- Rigid, contains many assumptions

$$I_x u + I_y v + I_t = 0.$$

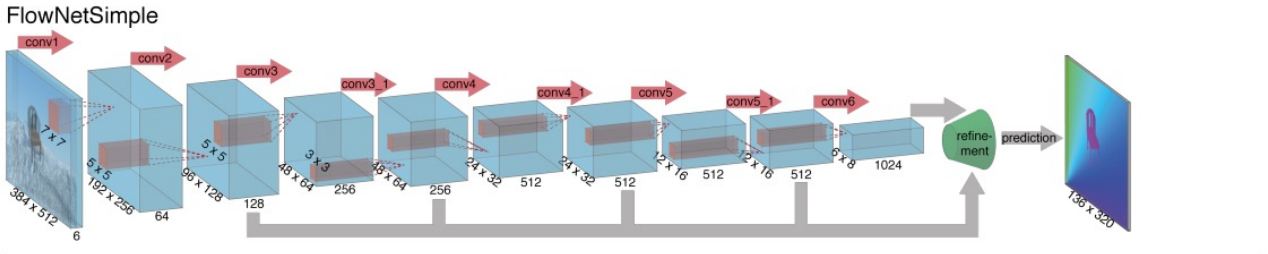
$$\mathbf{A}\mathbf{u} = \mathbf{b}$$

$$\begin{pmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{N^2}) & I_y(\mathbf{p}_{N^2}) \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = - \begin{pmatrix} I_t(\mathbf{p}_1) \\ \vdots \\ I_t(\mathbf{p}_{N^2}) \end{pmatrix}$$

NYU

Correlation Volume Approach

- Simple Approach: Concatenate the two images together.

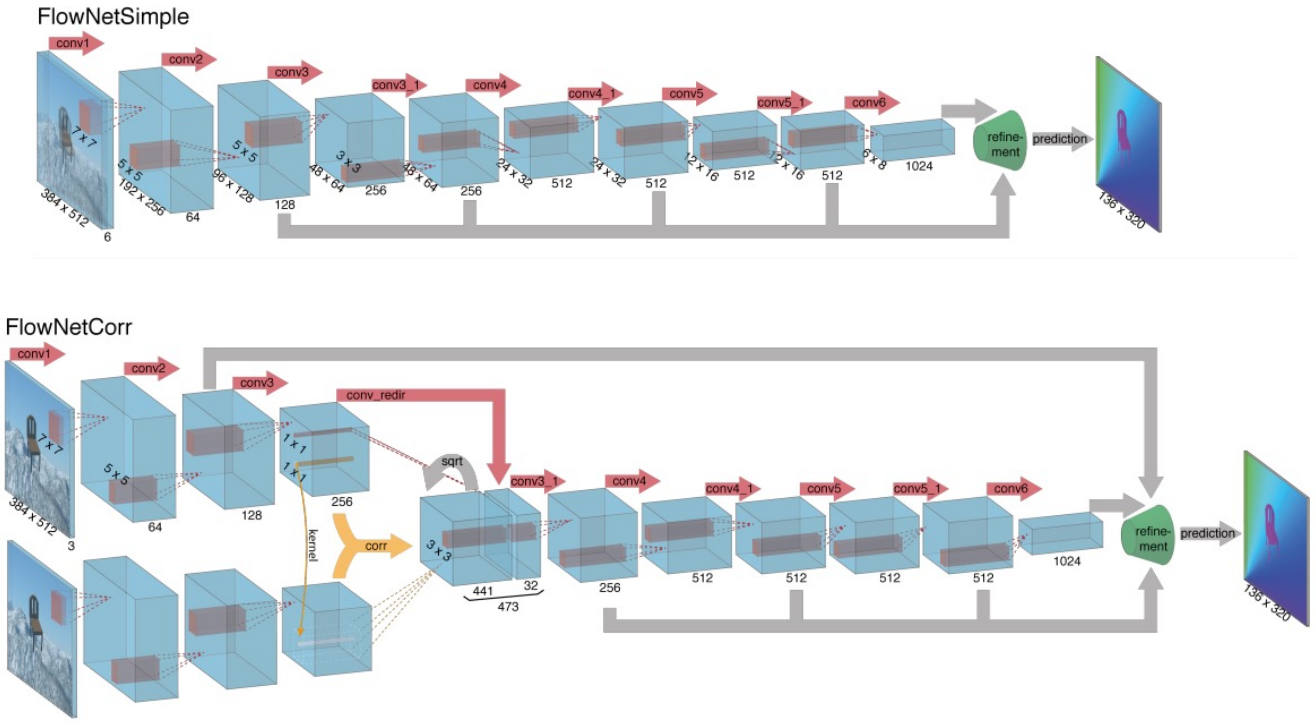


Fischer et al. FlowNet: Learning Optical Flow with Convolutional Networks. ICCV 2015.

NYU

Correlation Volume Approach

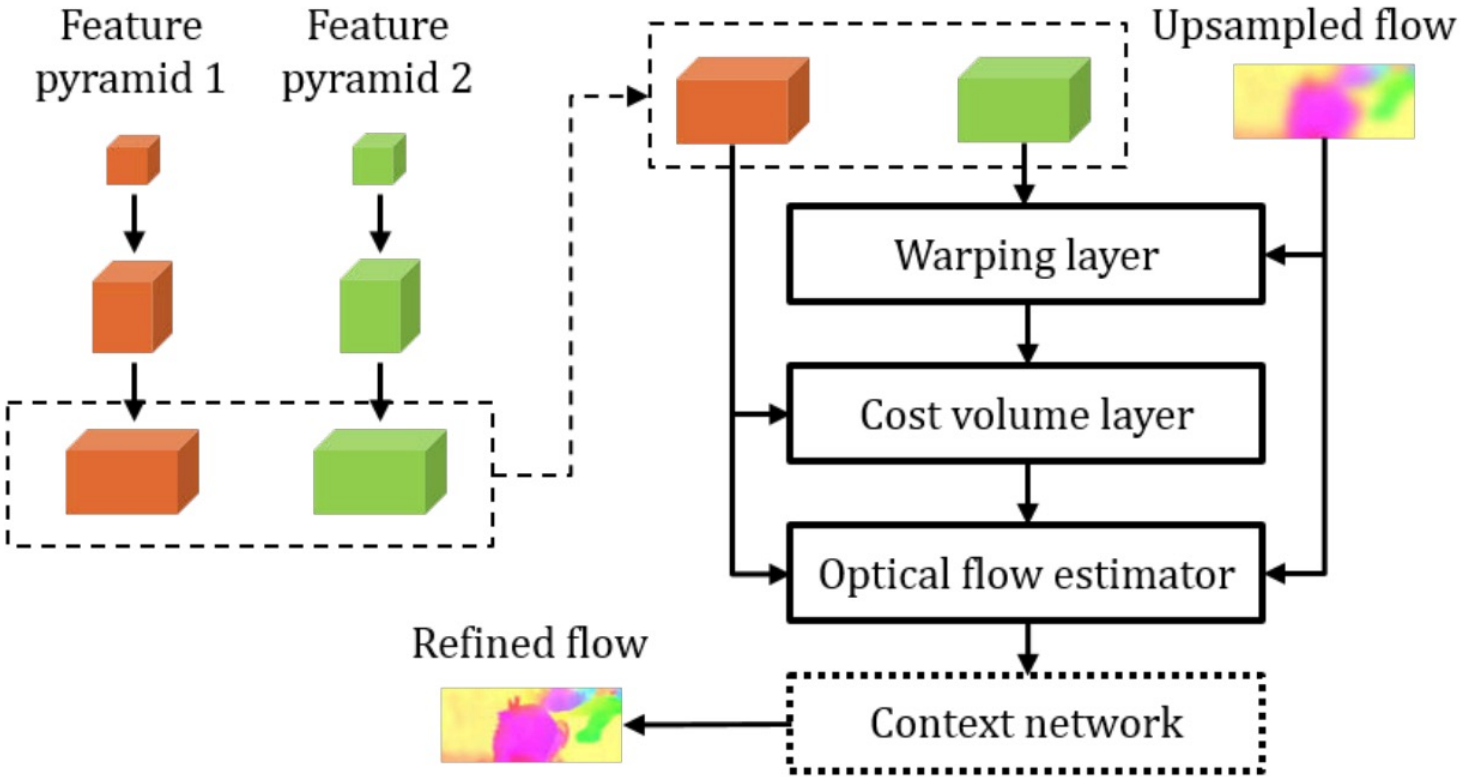
- Simple Approach: Concatenate the two images together.
- Correlation: Extract some levels of features, and convolve one feature on top of another.



Fischer et al. FlowNet: Learning Optical Flow with Convolutional Networks. ICCV 2015.

NYU

Iterative Refining through Feature Pyramid



Sun et al. PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume.

NYU

Unsupervised Flow

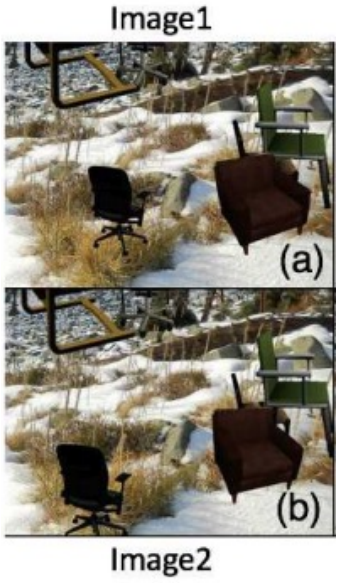
- Photometric Consistency (Appearance)

Jonschkowski et al. What Matters in Unsupervised Optical Flow. ECCV 2020

NYU

Unsupervised Flow

- Photometric Consistency (Appearance)
- Occlusion Estimation
  - Forward-backward consistency



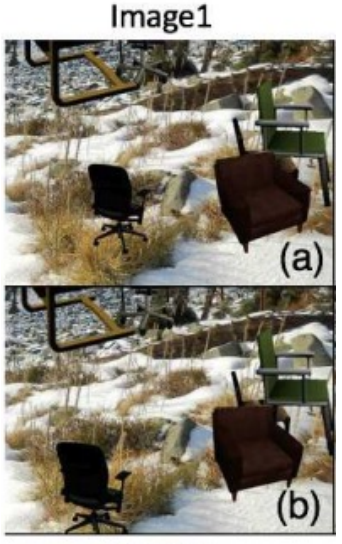
Wang et al., 2018

Jonschkowski et al. What Matters in Unsupervised Optical Flow. ECCV 2020

NYU

Unsupervised Flow

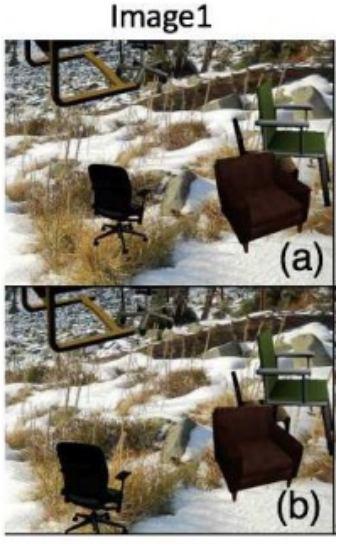
- Photometric Consistency (Appearance)
- Occlusion Estimation
  - Forward-backward consistency
- Smoothness



Wang et al., 2018

Unsupervised Flow

- Photometric Consistency (Appearance)
- Occlusion Estimation
  - Forward-backward consistency
- Smoothness
- Self-supervision: Ensure consistent flow at different augmentation (e.g. crops)



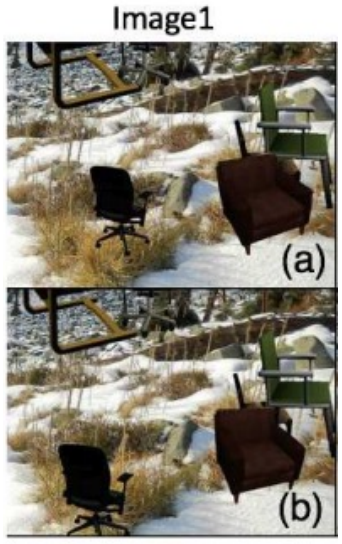
Wang et al., 2018

Jonschkowski et al. What Matters in Unsupervised Optical Flow. ECCV 2020

NYU

Unsupervised Flow

- Photometric Consistency (Appearance)
- Occlusion Estimation
  - Forward-backward consistency
- Smoothness
- Self-supervision: Ensure consistent flow at different augmentation (e.g. crops)
- Can 3D information help us reason about motion?



Wang et al., 2018

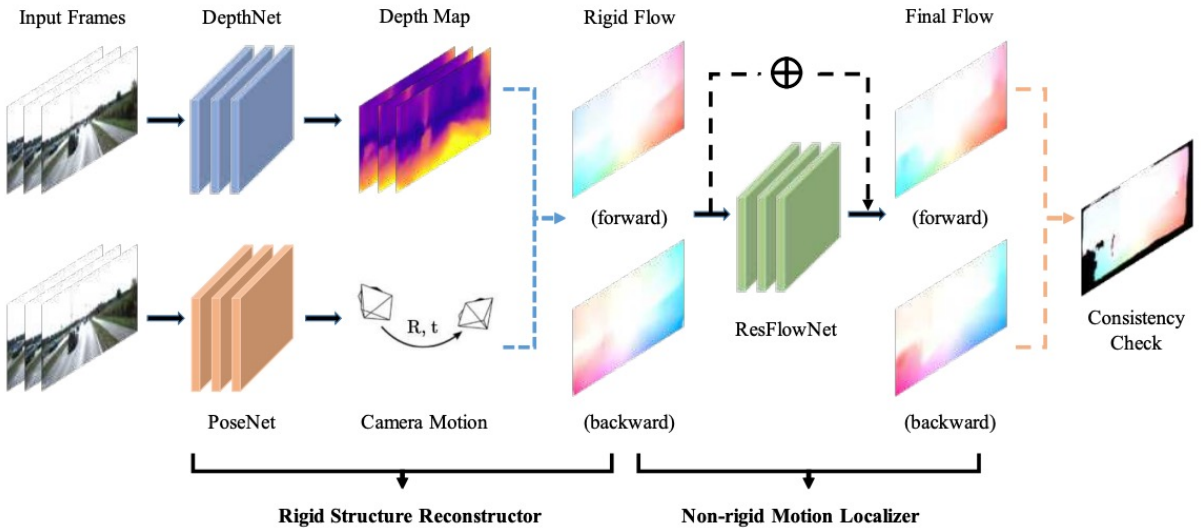
Jonschkowski et al. What Matters in Unsupervised Optical Flow. ECCV 2020

NYU

Depth, Flow, and Pose Movement

- The static objects follow rigid flow: determined by camera motion and depth.

$$f^{rig}_{t\mapsto s}(p_t) = KT_{t\mapsto s}D_t(p_t)K^{-1}p_t - p_t.$$



Yin & Shi. GeoNet: Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose. CVPR 2018.

NYU

Training Losses

- Appearance Loss (Warping):

$$\mathcal{L}_{rw} = \alpha \frac{1 - SSIM(I_t, \tilde{I}_s^{rig})}{2} + (1 - \alpha) \| I_t - \tilde{I}_s^{rig} \|_1.$$

$$\mathcal{L}_{fw} = \alpha \frac{1 - SSIM(I_t, \tilde{I}_s^{full})}{2} + (1 - \alpha) \| I_t - \tilde{I}_s^{full} \|_1.$$

Yin & Shi. GeoNet: Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose. CVPR 2018.

NYU

Training Losses

- Appearance Loss (Warping):

$$\mathcal{L}_{rw} = \alpha \frac{1 - SSIM(I_t, \tilde{I}_s^{rig})}{2} + (1 - \alpha) \| I_t - \tilde{I}_s^{rig} \|_1.$$

$$\mathcal{L}_{fw} = \alpha \frac{1 - SSIM(I_t, \tilde{I}_s^{full})}{2} + (1 - \alpha) \| I_t - \tilde{I}_s^{full} \|_1.$$

- Smoothness Loss:

$$\mathcal{L} = \sum_{p_t} |\nabla D(p_t)| \cdot (\exp(-|\nabla I(p(t)|))^T.$$

Yin & Shi. GeoNet: Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose. CVPR 2018.

NYU

Training Losses

- Appearance Loss (Warping):

$$\mathcal{L}_{rw} = \alpha \frac{1-SSIM(I_t, \tilde{I}_s^{rig})}{2} + (1-\alpha)\|I_t - \tilde{I}_s^{rig}\|_1.$$

$$\mathcal{L}_{fw} = \alpha \frac{1-SSIM(I_t, \tilde{I}_s^{full})}{2} + (1-\alpha)\|I_t - \tilde{I}_s^{full}\|_1.$$

- Smoothness Loss:

$$\mathcal{L} = \sum_{p_t} |\nabla D(p_t)| \cdot (\exp(-|\nabla I(p(t)|))^T.$$

- Forward-Backward Consistency:

$$\mathcal{L} = \sum_{p_t} [\delta(p_t)] \cdot \|\Delta f_{t \mapsto s}^{full}(p_t)\|_1.$$

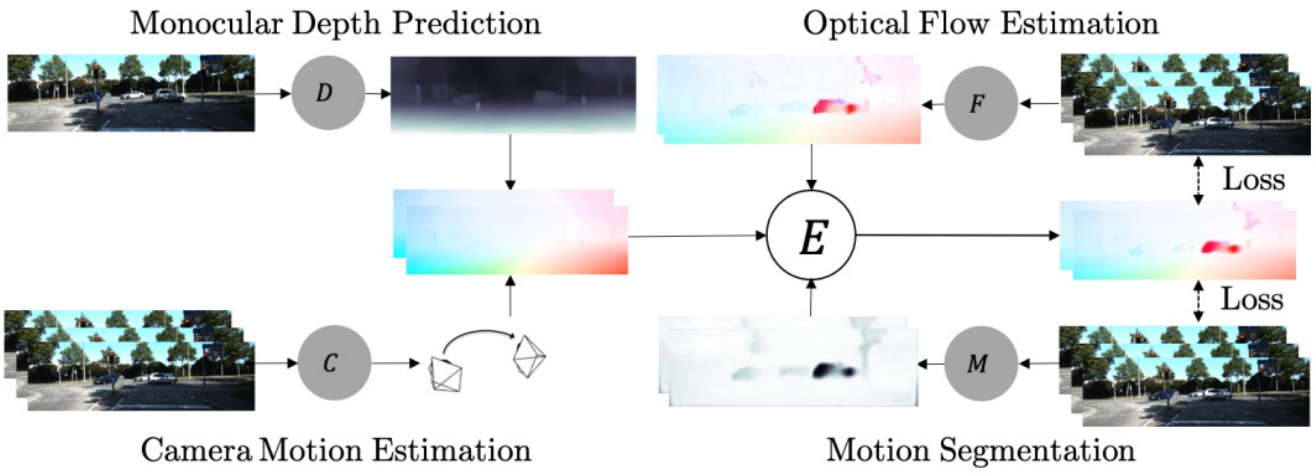
$$\delta(p_t) = \|\Delta f_{t \mapsto s}^{full}(p_t)\|_2 < \max\{\alpha, \beta \|f_{t \mapsto s}^{full}(p_t)\|_2\}.$$

Yin & Shi. GeoNet: Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose. CVPR 2018.

NYU

Summary

- Leverage cross correlation structure for spatial similarity matching.

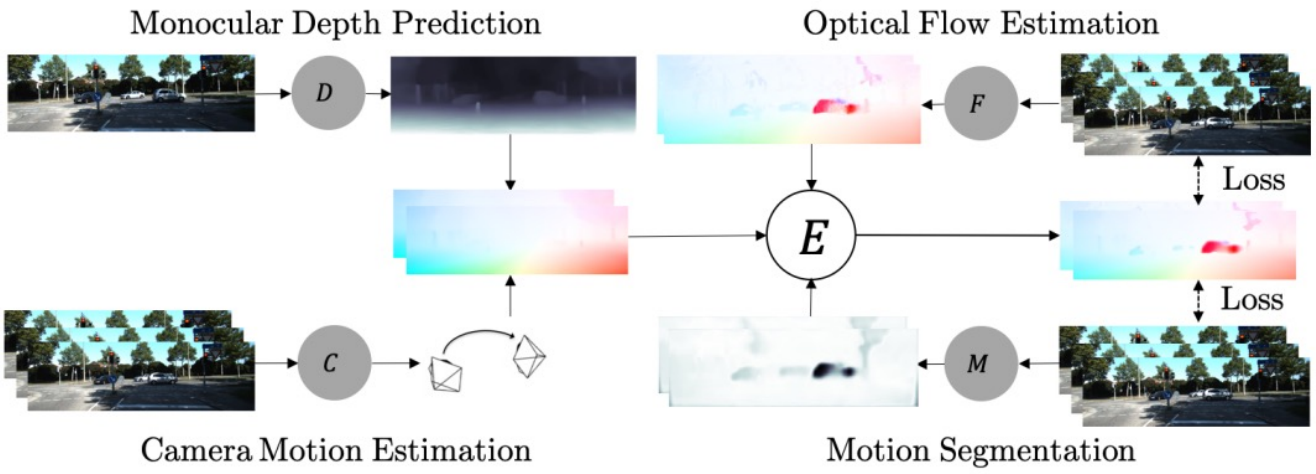


Ranjan et al. Competitive Collaboration: Joint Unsupervised Learning of Depth, Camera, Motion, Optical Flow and Motion Segmentation. CVPR 2019

NYU

Summary

- Leverage cross correlation structure for spatial similarity matching.
- Can be used towards: depth, flow, and pose prediction.

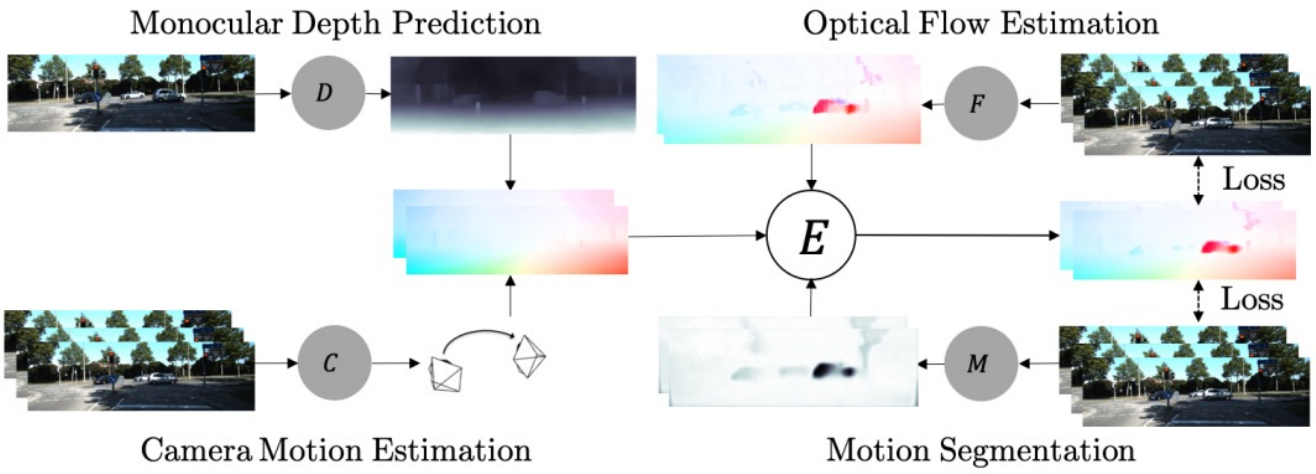


Ranjan et al. Competitive Collaboration: Joint Unsupervised Learning of Depth, Camera, Motion, Optical Flow and Motion Segmentation. CVPR 2019

NYU

Summary

- Leverage cross correlation structure for spatial similarity matching.
- Can be used towards: depth, flow, and pose prediction.
- Can form triangulation for self-supervision.

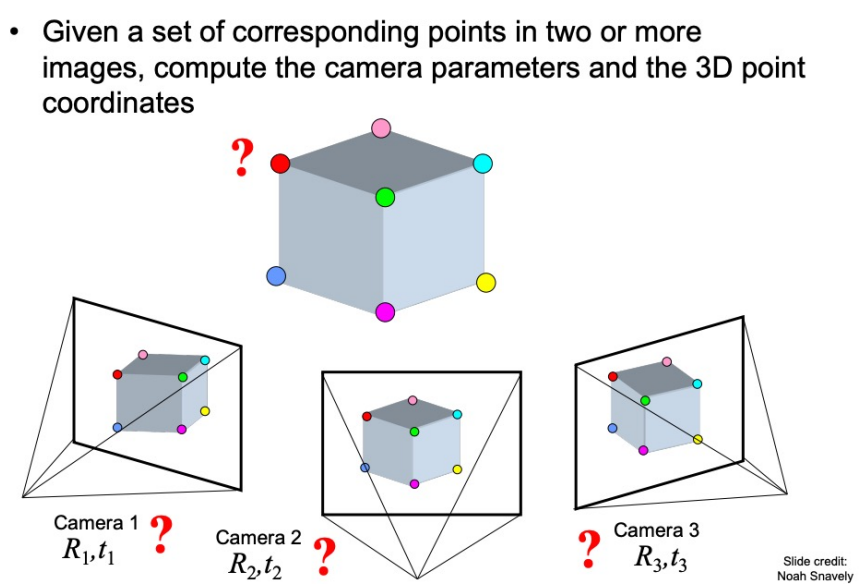


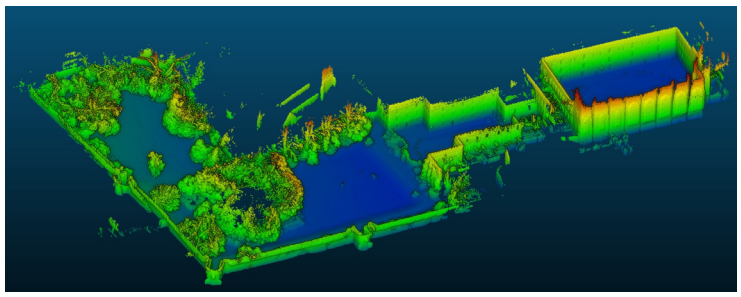
Ranjan et al. Competitive Collaboration: Joint Unsupervised Learning of Depth, Camera, Motion, Optical Flow and Motion Segmentation. CVPR 2019

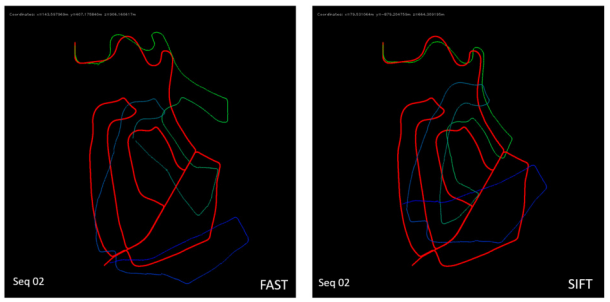
NYU

Classical Mapping

- Estimating 3D structure and location from 2D observations.
- Simultaneous Localization and Mapping.
- Common Techniques: Extended Kalman Filter, GraphSLAM







Garg & Jain

NYU

Common Drawbacks

- Probabilistic inference can take long to compute, and mapping takes a large memory storage.

NYU

Common Drawbacks

- Probabilistic inference can take long to compute, and mapping takes a large memory storage.
- Great for 3D reconstruction but downstream tasks may not need a full precision explicit map.

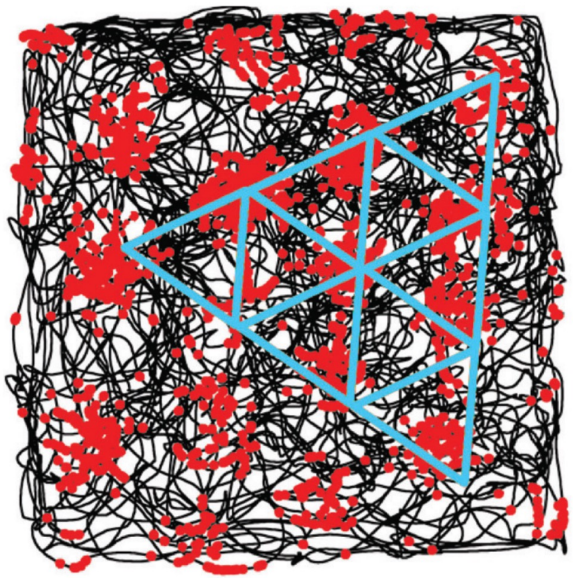
Common Drawbacks

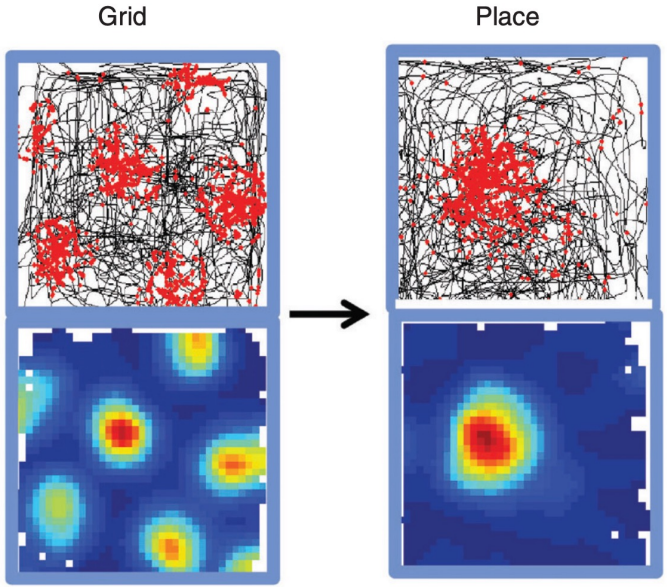
- Probabilistic inference can take long to compute, and mapping takes a large memory storage.
- Great for 3D reconstruction but downstream tasks may not need a full precision explicit map.
- May not fully understand dynamic objects (averaging across multiple scans).

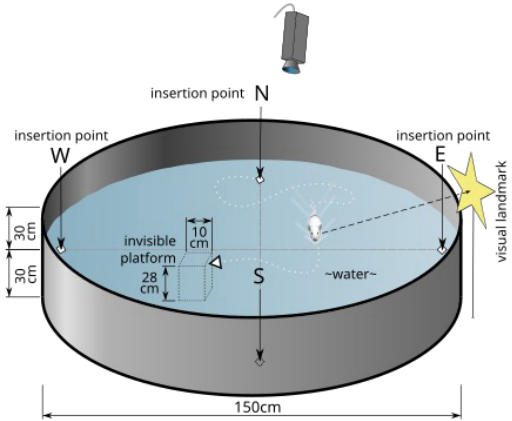
Common Drawbacks

- Probabilistic inference can take long to compute, and mapping takes a large memory storage.
- Great for 3D reconstruction but downstream tasks may not need a full precision explicit map.
- May not fully understand dynamic objects (averaging across multiple scans).
- Is there a more end-to-end version?

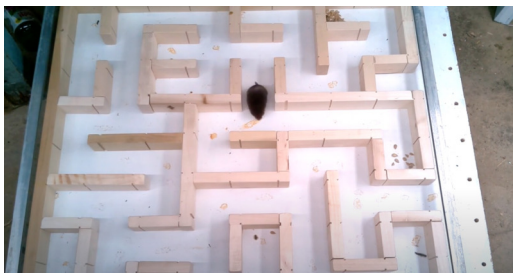
Mapping in the Brain: Grid and Place Cells







Morris water maze



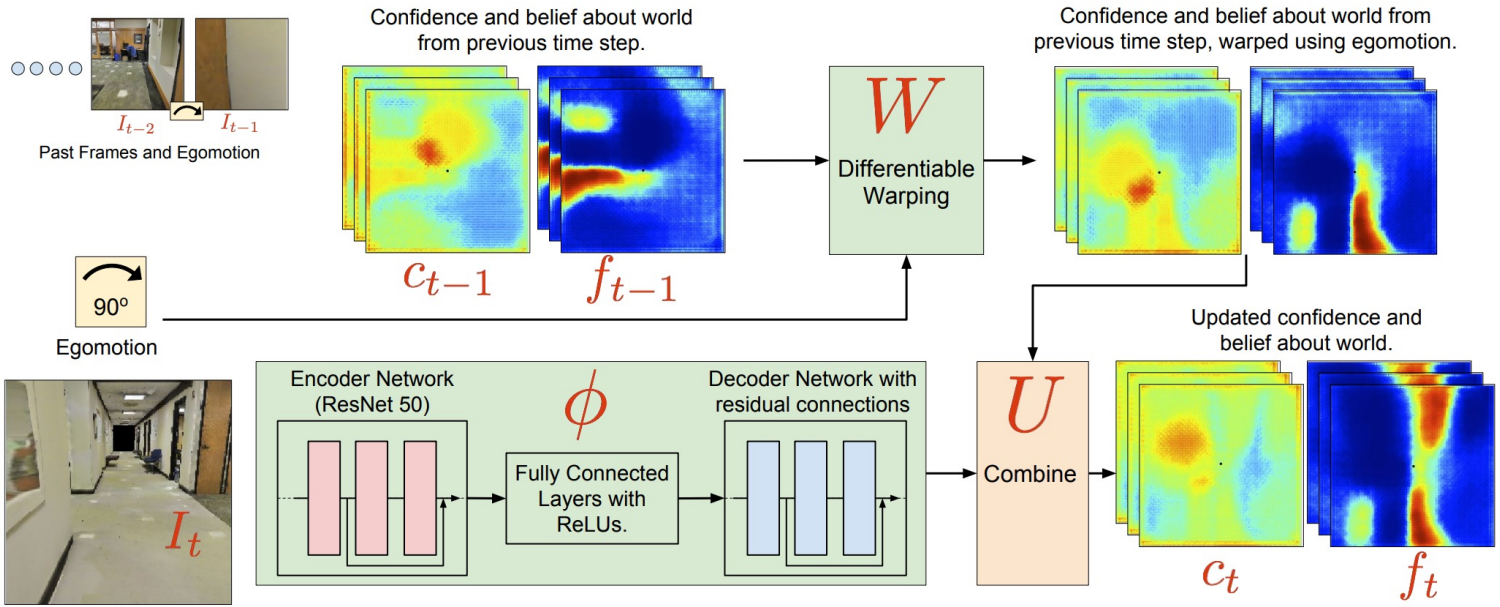
Matthias Wandel, 2018

May-Britt Moser, David C. Rowland, and Edvard I. Moser. Place Cells, Grid Cells, and Memory.

NYU

Neural Mapping

- Can we learn a mapping representation?
- Metric space, top-down warping (known egomotion).

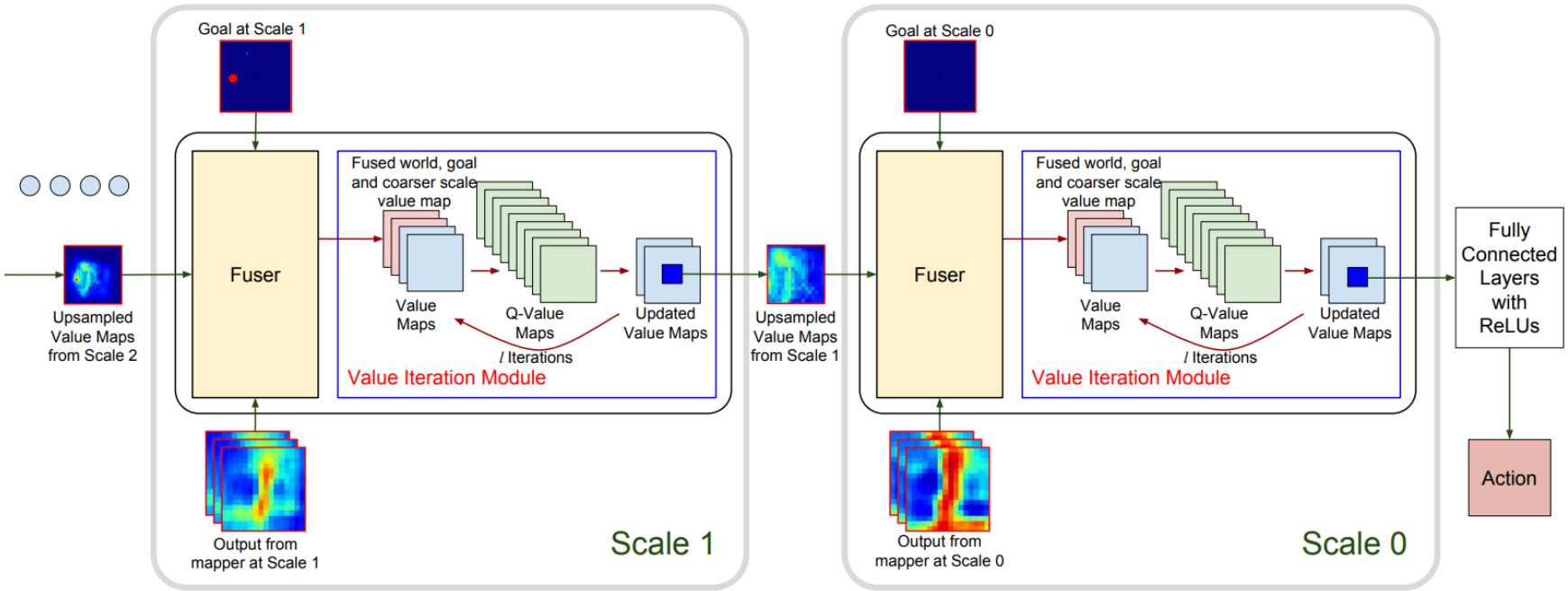


Gupta et al. Cognitive Mapping and Planning for Visual Navigation. IJCV 2017.

NYU

Hierarchical Planning

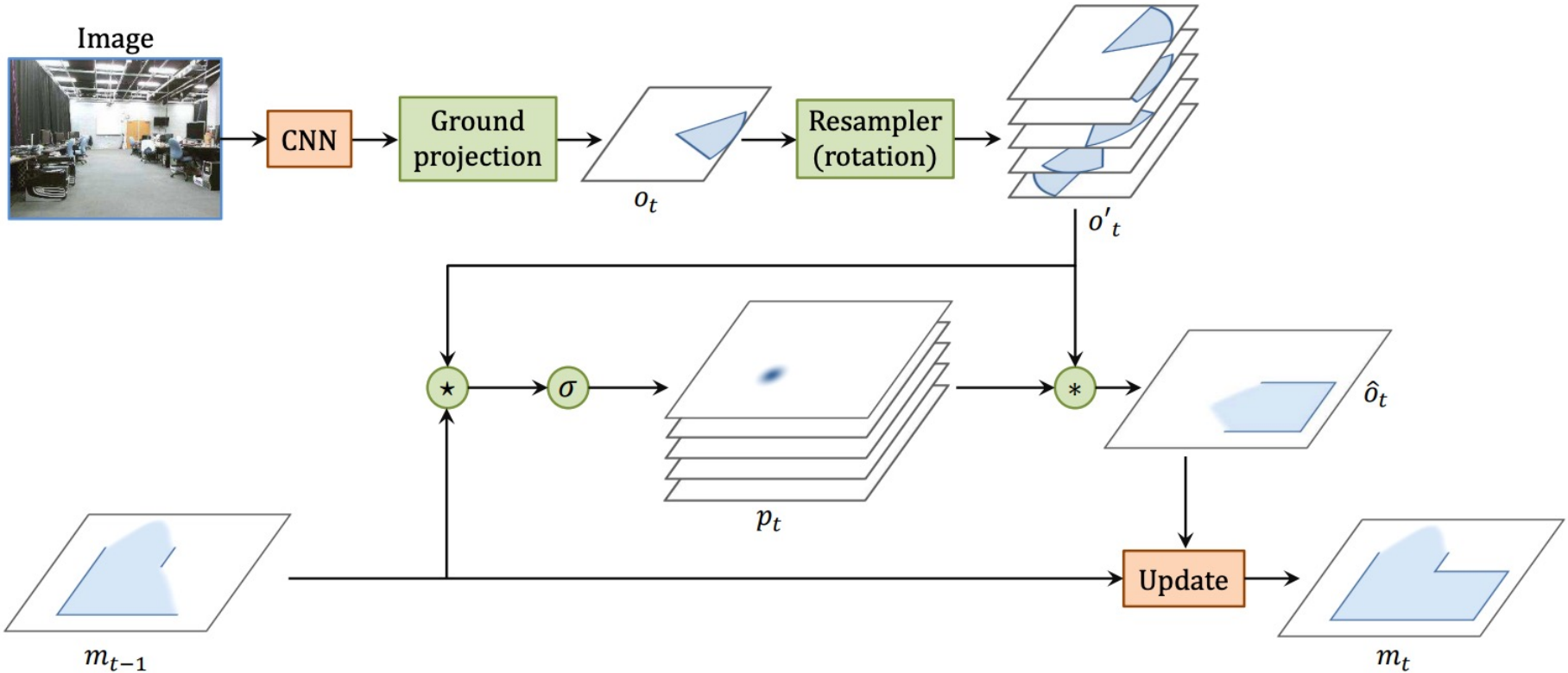
- How do we use the learned map (allocentric) feature of the world?



Gupta et al. Cognitive Mapping and Planning for Visual Navigation. IJCV 2017.

NYU

Simultaneous Localization and Registration



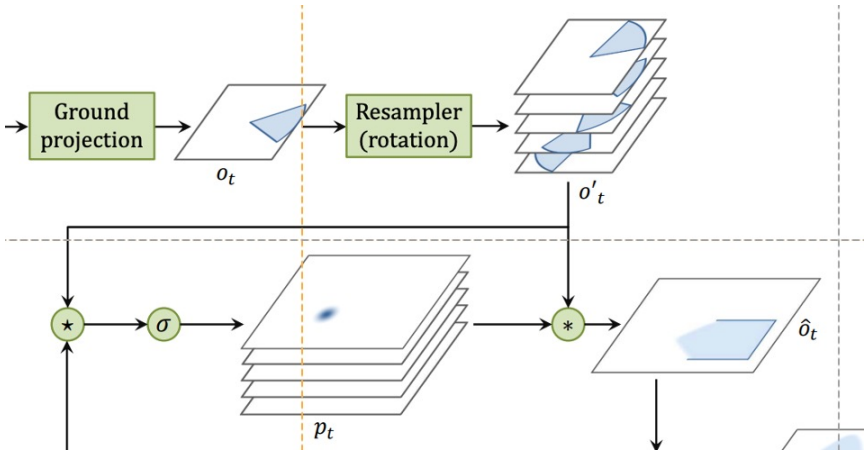
Henriques & Vedaldi. MapNet: An Allocentric Spatial Memory for Mapping Environments. CVPR 2018.

NYU

Simultaneous Localization and Registration

- The observations o_t are transformed into a stack o'_t by applying a rotation resampler.

$$o'_{ijkl} = [R(o, 2\pi l/r)]_{ijk}.$$



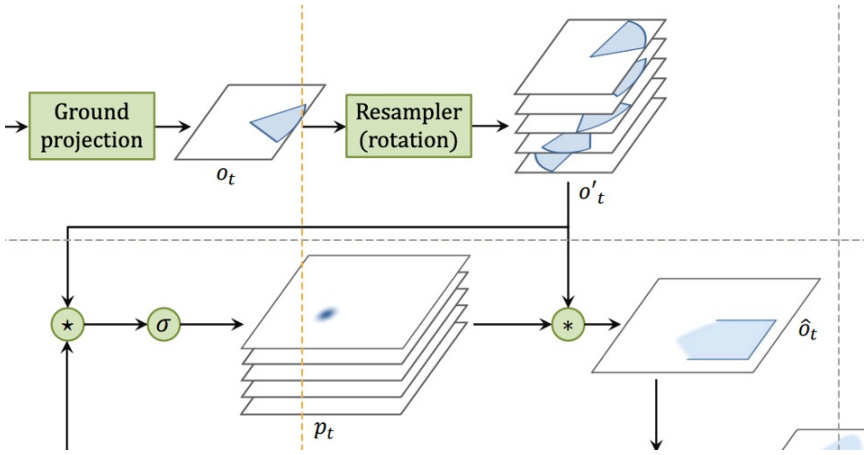
Simultaneous Localization and Registration

- The observations o_t are transformed into a stack o'_t by applying a rotation resampler.

$$o'_{ijkl} = [R(o, 2\pi l/r)]_{ijk}.$$

- o'_t convolve with the base feature.

$$p_t = \mathrm{Softmax}(m_{t-1} * o'_t).$$



Simultaneous Localization and Registration

- The observations o_t are transformed into a stack o'_t by applying a rotation resampler.

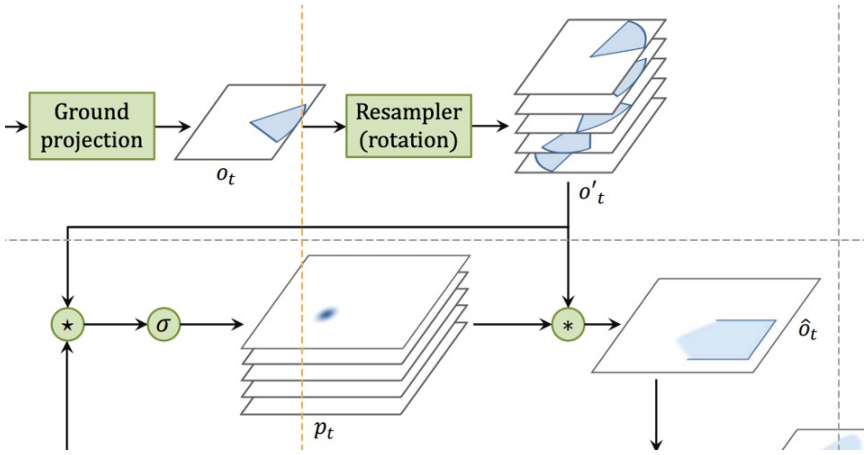
$$o'_{ijkl} = [R(o, 2\pi l/r)]_{ijk}.$$

- o'_t convolve with the base feature.

$$p_t = \mathrm{Softmax}(m_{t-1} * o'_t).$$

- Transform observations into allocentric

$$\hat{o}_t = \sum_{uvw} p_{uvw} T(o|u, v, w).$$



NYU

Simultaneous Localization and Registration

- The observations o_t are transformed into a stack o'_t by applying a rotation resampler.

$$o'_{ijkl} = [R(o, 2\pi l/r)]_{ijk}.$$

- o'_t convolve with the base feature.

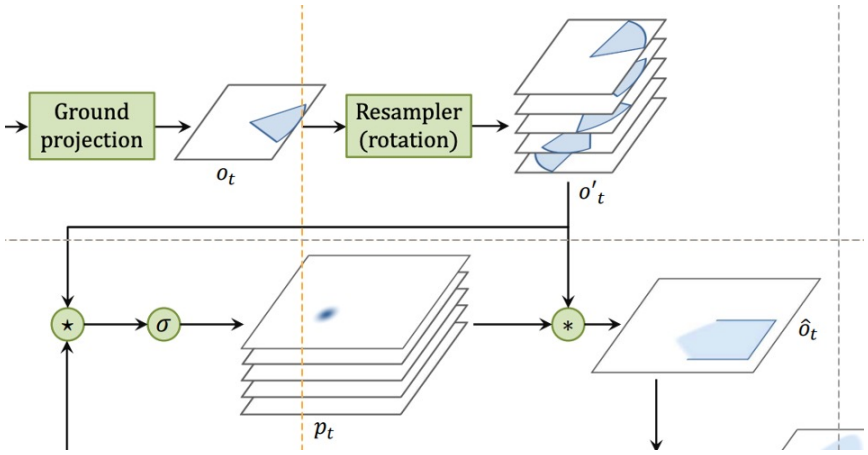
$$p_t = \mathrm{Softmax}(m_{t-1} * o'_t).$$

- Transform observations into allocentric

$$\hat{o}_t = \sum_{uvw} p_{uvw} T(o|u, v, w).$$

- Update belief:

$$m_{i,j,t+1} = \mathrm{LSTM}(m_{i,j,t}, \hat{o}_{i,j,t}).$$



Henriques & Vedaldi. MapNet: An Allocentric Spatial Memory for Mapping Environments. CVPR 2018.

NYU

Simultaneous Localization and Registration

- The observations o_t are transformed into a stack o'_t by applying a rotation resampler.

$$o'_{ijkl} = [R(o, 2\pi l/r)]_{ijk}.$$

- o'_t convolve with the base feature.

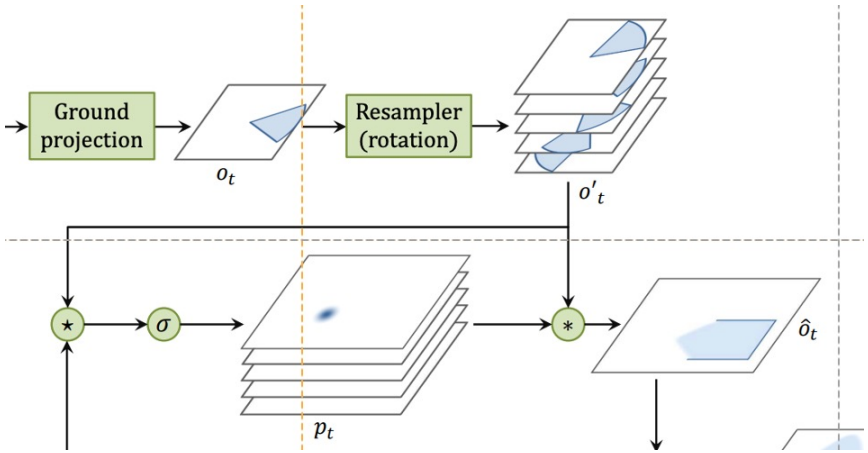
$$p_t = \mathrm{Softmax}(m_{t-1} * o'_t).$$

- Transform observations into allocentric

$$\hat{o}_t = \sum_{uvw} p_{uvw} T(o|u, v, w).$$

- Update belief:

$$m_{i,j,t+1} = \mathrm{LSTM}(m_{i,j,t}, \hat{o}_{i,j,t}).$$



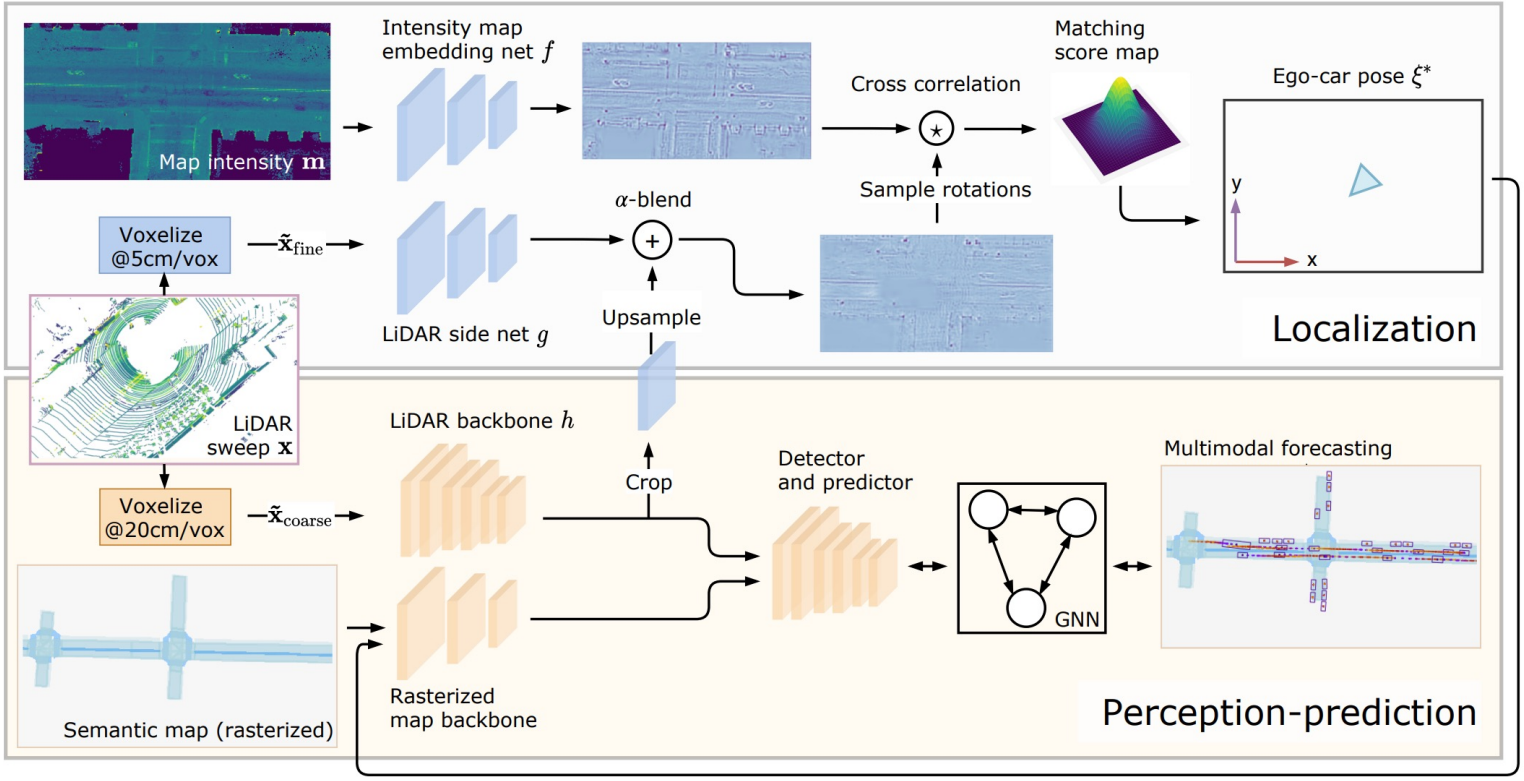
Loss:

$$\mathcal{L}(p) = -\log \sum_t p_{H_t W_t R_t t}.$$

Henriques & Vedaldi. MapNet: An Allocentric Spatial Memory for Mapping Environments. CVPR 2018.

NYU

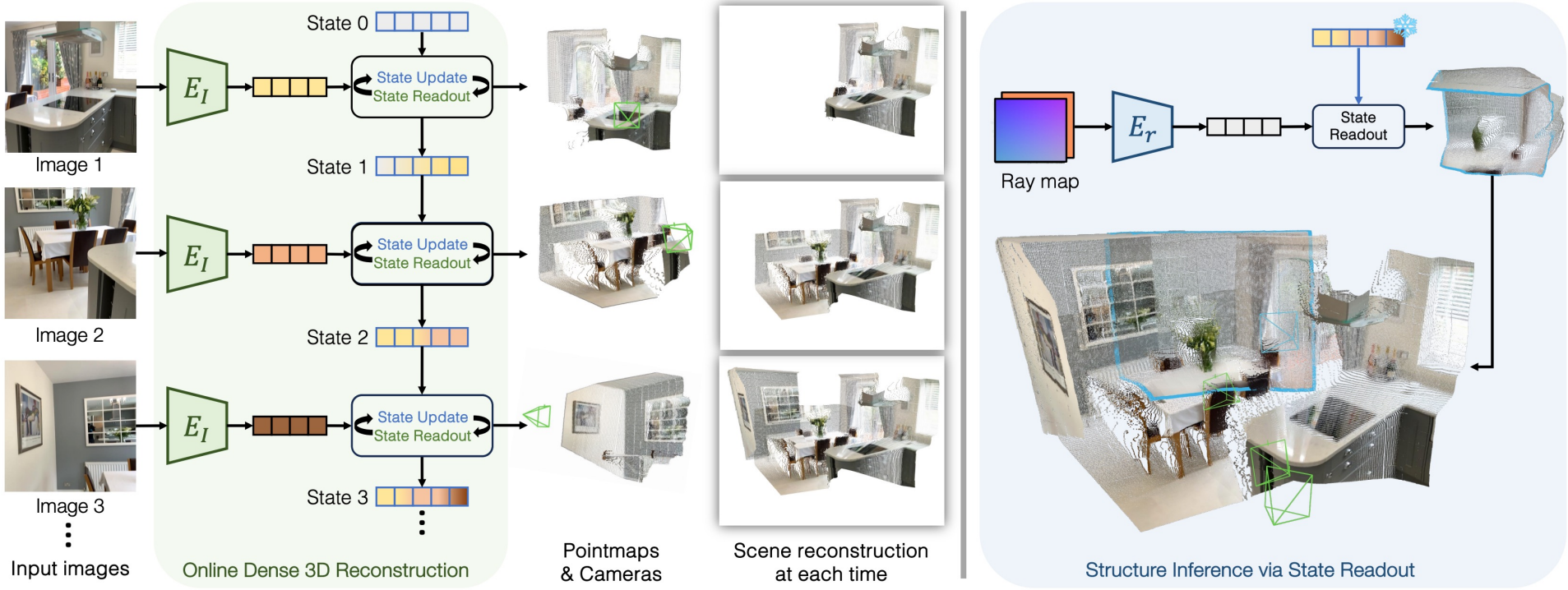
Joint Localization, Perception and Prediction



Philips et al. Deep Multi-Task Learning for Joint Localization, Perception, and Prediction CVPR 2021.

NYU

Continuous 3D Perception and Mapping

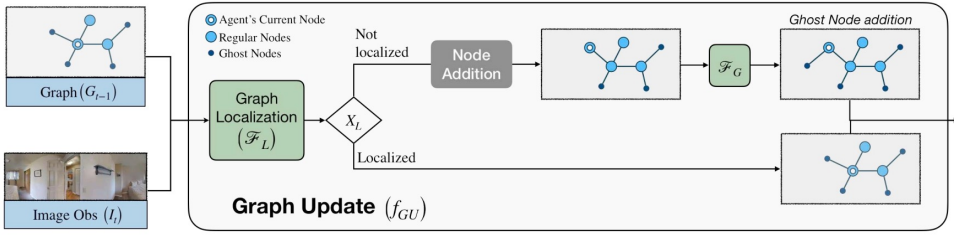


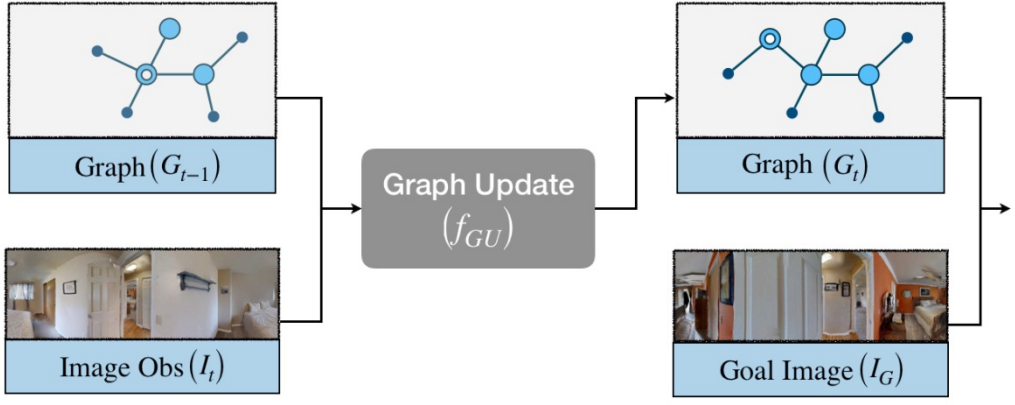
Wang et al. Continuous 3D Perception Model with Persistent State. arXiv 2025.

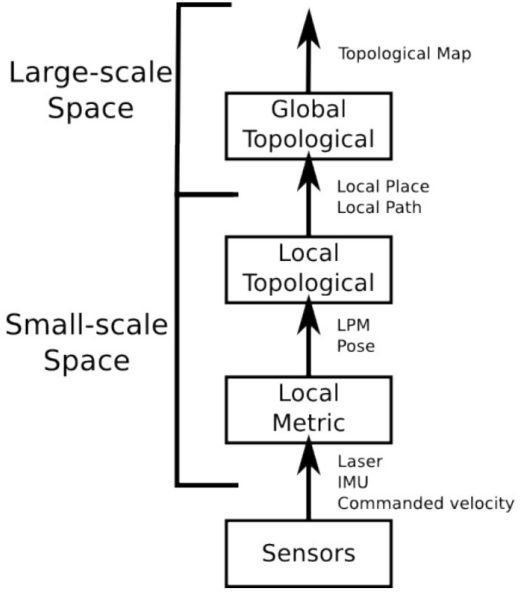
NYU

Topological Mapping

- High-level graph representation
- Each node contains more summarized information
- Enables global planning







Johnson, 2018

Johnson. Topological Mapping and Navigation in Real-World Environments. 2018.
Chaplot et al. Neural Topological SLAM for Visual Navigation. CVPR 2020.

NYU

Summary - 3D Vision

- Covers 3D, equivariance, motion, depth, and mapping.

Summary - 3D Vision

- Covers 3D, equivariance, motion, depth, and mapping.
- Still needs high-level features (recognizing the object and semantics): Spatial pyramid.

NYU

Summary - 3D Vision

- Covers 3D, equivariance, motion, depth, and mapping.
- Still needs high-level features (recognizing the object and semantics): Spatial pyramid.
- Can be made unsupervised

NYU

Summary - 3D Vision

- Covers 3D, equivariance, motion, depth, and mapping.
- Still needs high-level features (recognizing the object and semantics): Spatial pyramid.
- Can be made unsupervised
- Design end-to-end modules that contain rich features.

NYU

Summary - 3D Vision

- Covers 3D, equivariance, motion, depth, and mapping.
- Still needs high-level features (recognizing the object and semantics): Spatial pyramid.
- Can be made unsupervised
- Design end-to-end modules that contain rich features.
- Design joint learning frameworks.

NYU

Summary - 3D Vision

- Covers 3D, equivariance, motion, depth, and mapping.
- Still needs high-level features (recognizing the object and semantics): Spatial pyramid.
- Can be made unsupervised
- Design end-to-end modules that contain rich features.
- Design joint learning frameworks.
- Using geometric transformation to ground representations.

NYU

Summary - 3D Vision

- Covers 3D, equivariance, motion, depth, and mapping.
- Still needs high-level features (recognizing the object and semantics): Spatial pyramid.
- Can be made unsupervised
- Design end-to-end modules that contain rich features.
- Design joint learning frameworks.
- Using geometric transformation to ground representations.
- Useful for planning (a few weeks from now).

NYU