DS-GA.3001 Embodied Learning and Vision

Mengye Ren

NYU

Spring 2025

embodied-learning-vision-course.github.io

Lecture Slides for Note Taking

NYU

Diffusion Models

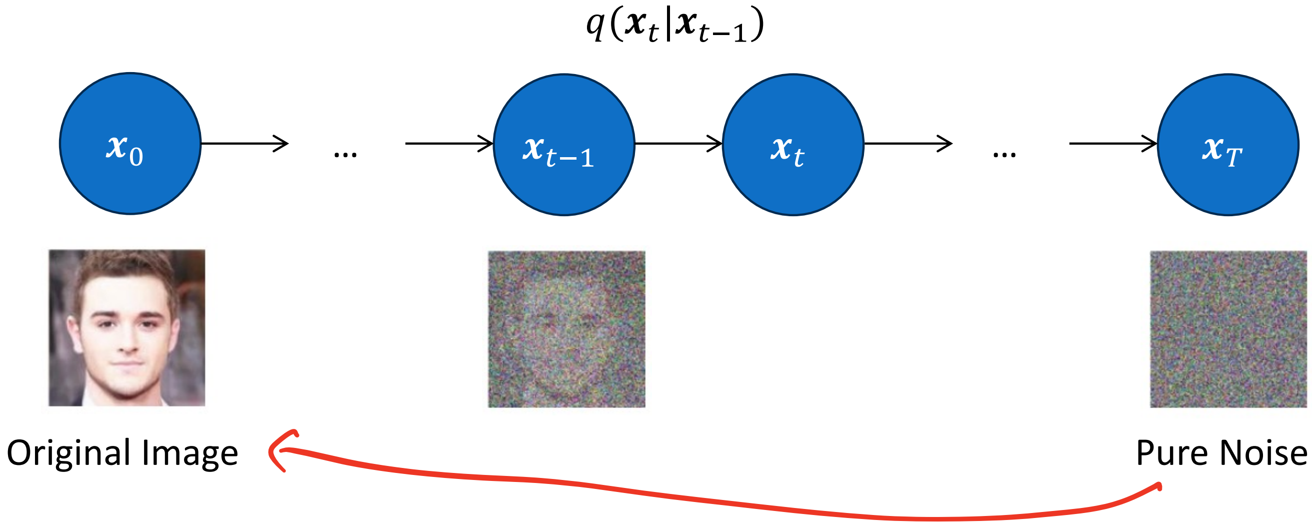
- A popular model for generative model today is diffusion model.

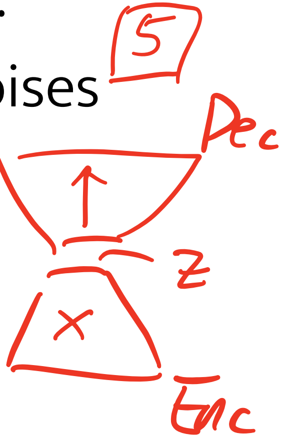
Diffusion Models

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

Diffusion Models

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



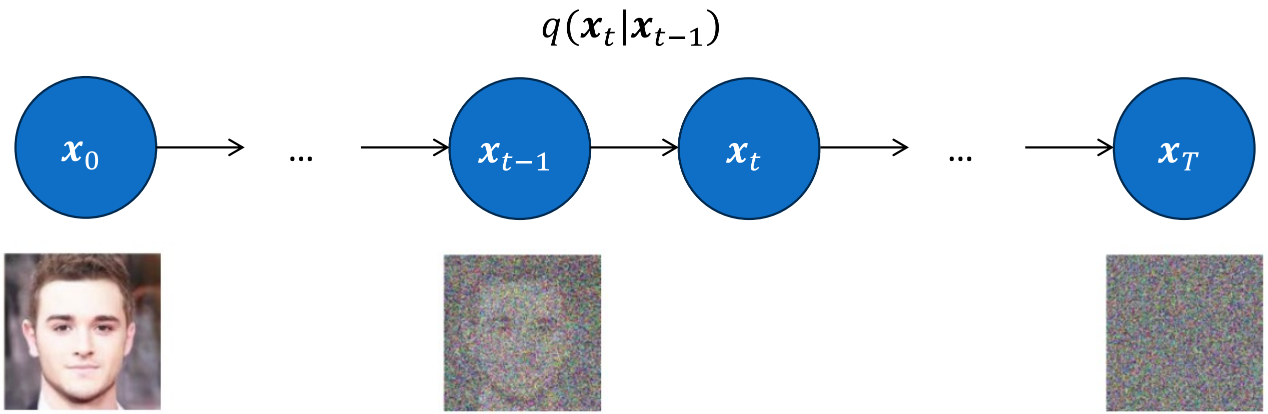


Diffusion Models

μ $\sigma^2 I$

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

~~pe~~ prev

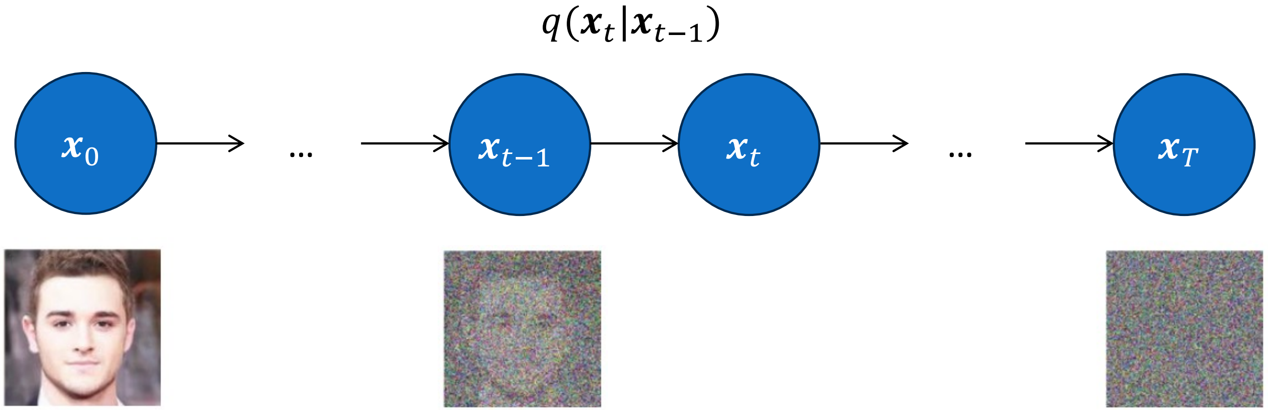


Original Image

Pure Noise

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: $x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon_t, \epsilon_t \sim \mathcal{N}(0, I)$.



Original Image

Pure Noise

Properties of the Forward Process

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

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:

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

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:

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

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

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:

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

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

Cumulative Schedule

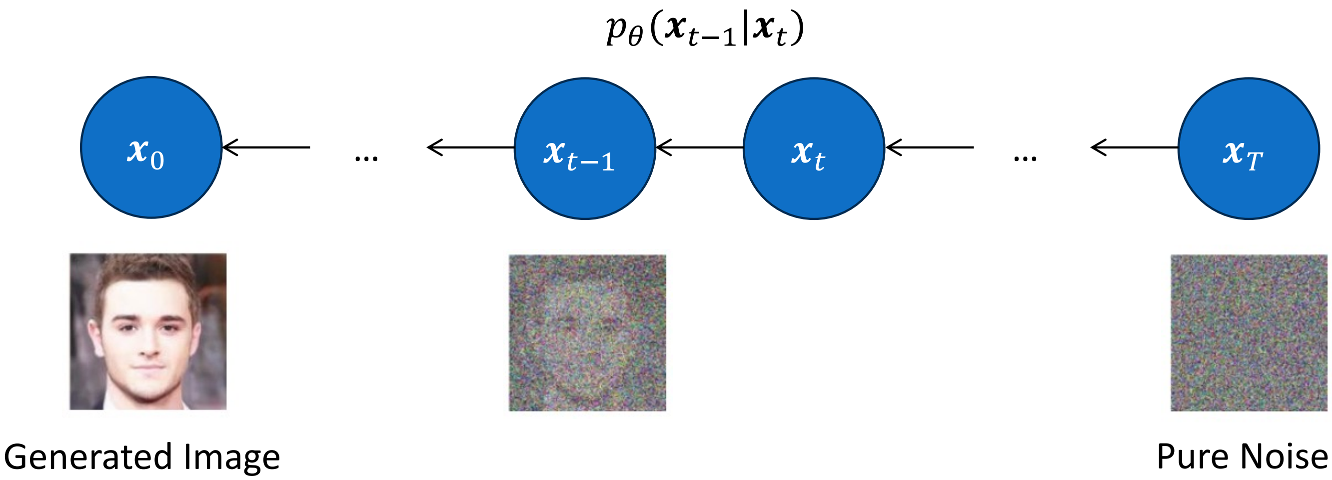
$\alpha_t = 1 - \beta_t.$

- Show it's true for x_2: $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s.$

$$
\begin{aligned}
x_2 &= \sqrt{1-\beta_2}x_1 + \sqrt{\beta_2}\epsilon_2 = \sqrt{1-\beta_2}\sqrt{1-\beta_1}x_0 + \sqrt{\beta_2}\epsilon_2 + \sqrt{1-\beta_2}\sqrt{\beta_1}\epsilon_1 \\
&= \alpha_1\alpha_2 x_0 + \sqrt{(1-\beta_2)\beta_1 + \beta_2}\epsilon \\
&= \bar{\alpha}_2 x_0 + \sqrt{1-(1-\beta_1)(1-\beta_2)}\epsilon \\
&= \bar{\alpha}_2 x_0 + \sqrt{1-\bar{\alpha}_2}\epsilon.
\end{aligned}
$$

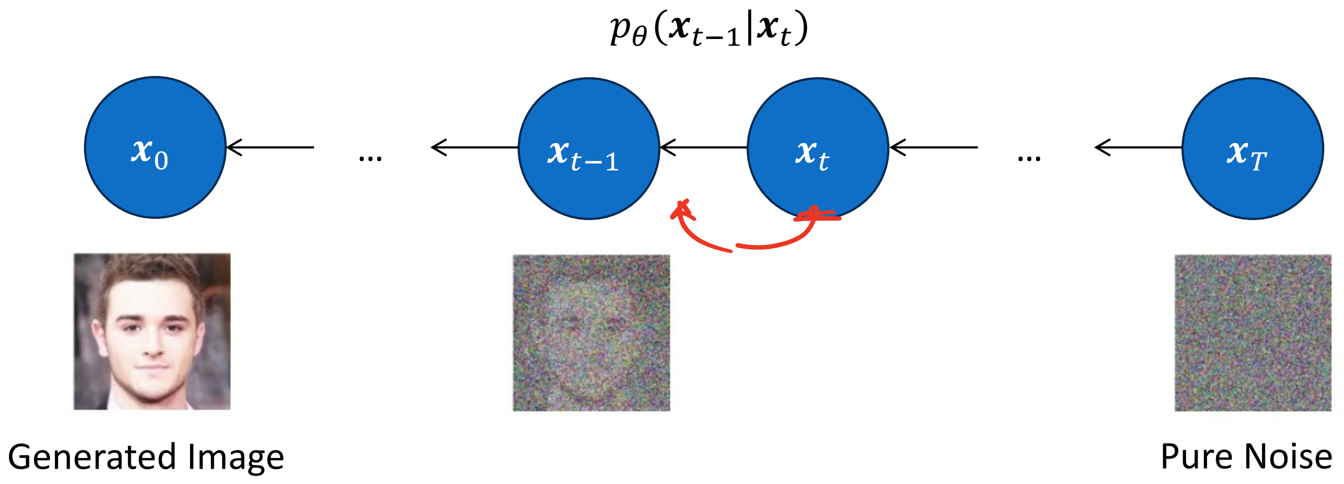
Reverse Process

- A generative model wants to predict x_0 from x_T.



Reverse Process

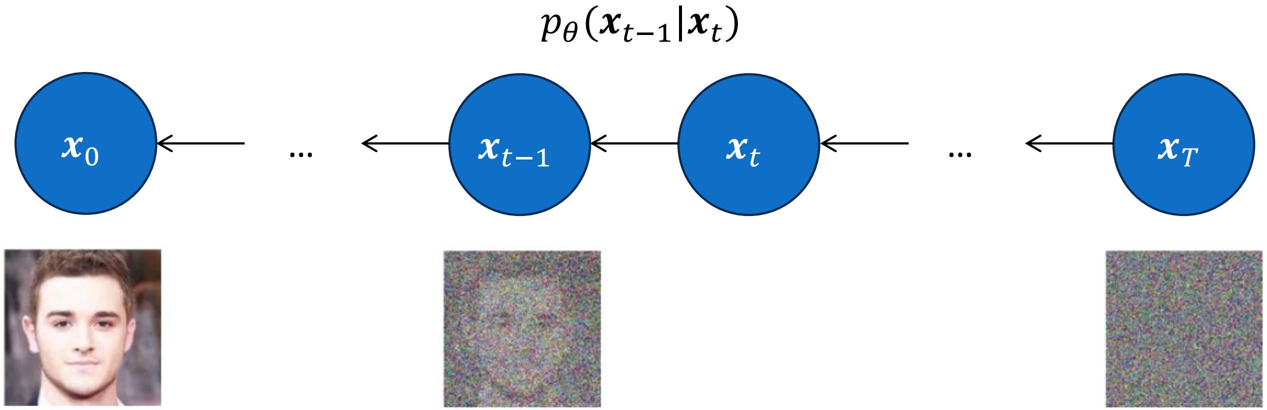
- A generative model wants to predict x_0 from x_T.
- The reverse process transition is also Gaussian distributed. But we don't know what the transition will be like just by looking at the noisy image!



Reverse Process

- So, we need to learn a “model”:

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



Generated Image

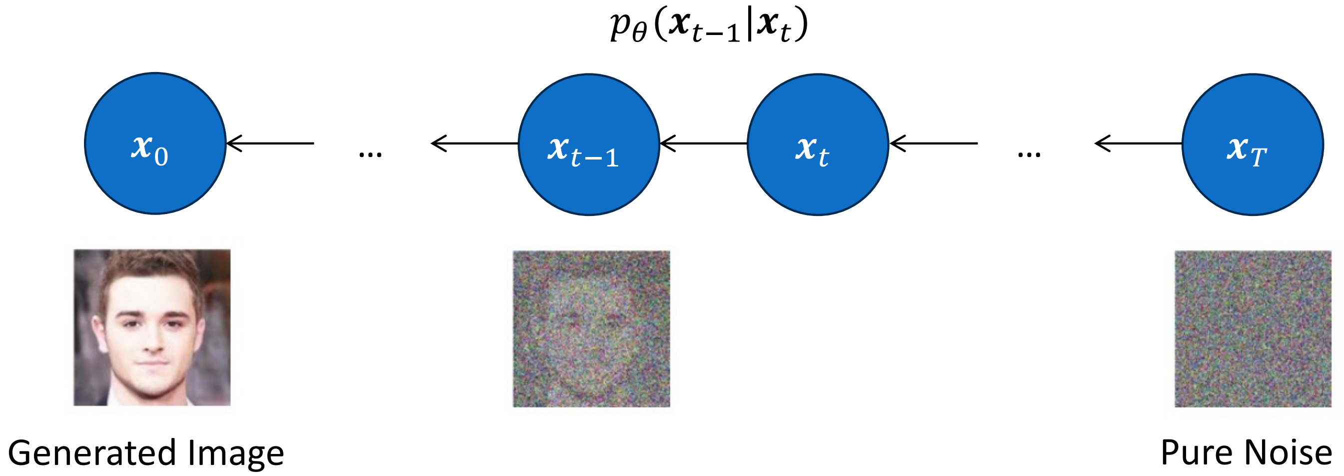
Pure Noise

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 $p(x_{t-1}|x_t)$.

Reverse Process

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

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

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

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:

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

NYU

Reverse Process

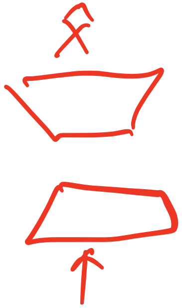
$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1})q(x_{t-1}|x_0)}{q(x_t|x_0)}.$$

Reverse Process

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

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

Reverse Process



X + noise

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

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

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

training target

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 $\tilde{\mu}_t$.

network

target

NYU

Training

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

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

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

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.

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

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

600 601

1000

Training

0

1000

- 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 the noisy and the original.

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

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

Algorithm 1 Training

repeat

$\boxed{\mathbf{x}_0} \sim q(\mathbf{x}_0)$

$\boxed{t} \sim \mathrm{Uniform}(\{1, \ldots, T\})$

$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

Take gradient descent step on

$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boxed{\boldsymbol{\epsilon}_\theta}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$

until converged

network.

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** $\mathbf{x}_0$

NYU

Sampling

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

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

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}.

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

- 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

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

sample from posterior Gaussian distribution

NYU

More on Samplers

- DDPM relies on many iterations (e.g. 1000) to produce one sample. Slower than NFs and GANs.

[Song et al. 2021]

Song et al. Denoising Diffusion Implicit Models. ICLR 2021.

NYU

More on Samplers

- DDPM relies on many iterations (e.g. 1000) to produce one sample. Slower than NFs and GANs.
- In the non-Markovian model, we can first generate x_T, and based on x_T and x_0 we can generate x_{T-1} and so on.

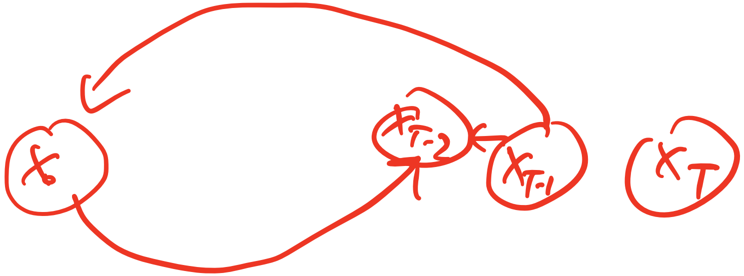
[Song et al. 2021]

Song et al. Denoising Diffusion Implicit Models. ICLR 2021.

More on Samplers

- DDPM relies on many iterations (e.g. 1000) to produce one sample. Slower than NFs and GANs.
- In the non-Markovian model, we can first generate x_T, and based on x_T and x_0 we can generate x_{T-1} and so on.
- Joint distribution:

$$q(x_{1:T}|x_0) = q(x_T|x_0)\prod_{t=2}^{T} q(x_{t-1}|x_t, x_0).$$



[Song et al. 2021]

More on Samplers

- DDPM relies on many iterations (e.g. 1000) to produce one sample. Slower than NFs and GANs.
- In the non-Markovian model, we can first generate x_T, and based on x_T and x_0 we can generate x_{T-1} and so on.
- Joint distribution:

$$q(x_{1:T}|x_0) = q(x_T|x_0)\prod_{t=2}^{T} q(x_{t-1}|x_t, x_0).$$

- Estimate x_{t-1} based on x_0 and x_t:

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}\left(\sqrt{a_{t-1}}x_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\frac{x_t - \sqrt{\alpha_t}x_0}{\sqrt{1-\alpha_t}}, \sigma_t^2 I\right).$$

[Song et al. 2021]

Song et al. Denoising Diffusion Implicit Models. ICLR 2021.

More on DDIM Samplers

- Prediction of x_0:

$$f_\theta^{(t)}(x_t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \sqrt{1-\alpha_t} \cdot \epsilon_\theta^{(t)}(x_t)).$$

[Song et al. 2021]

Song et al. Denoising Diffusion Implicit Models. ICLR 2021.

NYU

More on DDIM Samplers

- Prediction of x_0:

$$f_\theta^{(t)}(x_t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \sqrt{1-\alpha_t} \cdot \epsilon_\theta^{(t)}(x_t)\right).$$

- Sampling process:

$$p_\theta^{(t)}(x_{t-1}|x_t) = \begin{cases} \mathcal{N}(f_\theta^{(1)}(x_t)), \sigma_1^2 I) & \text{if } t = 1 \\ q(x_{t-1}|x_t, \underbrace{f_\theta^{(t)}(x_t)}_{\hat{x}_0}) & \text{otherwise.} \end{cases}$$

[Song et al. 2021]

Song et al. Denoising Diffusion Implicit Models. ICLR 2021.

Guided Diffusion

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

Classifier Guidance / External Score Model

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

$$\hat{\epsilon} \leftarrow \epsilon_\theta(x_t) - \sqrt{1-\bar{\alpha}_t}\, \nabla_{x_t} \log p_\phi(y|x_t)$$

$$x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}} \left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1-\bar{\alpha}_{t-1}}\hat{\epsilon}$$

ImageNet Classifier.

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

Guided Diffusion

inference only.

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

Classifier Guidance / External Score Model

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

$$\hat{\epsilon} \leftarrow \epsilon_\theta(x_t) - \sqrt{1-\bar{\alpha}_t}\, \nabla_{x_t} \log p_\phi(y|x_t)$$

$$x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}} \left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1-\bar{\alpha}_{t-1}}\hat{\epsilon}$$

- 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

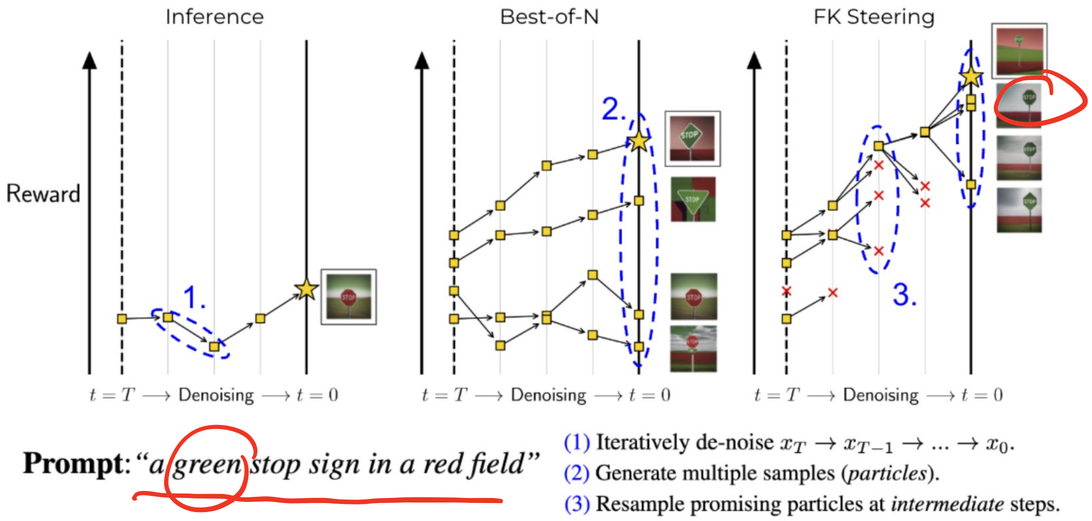
Requires training

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

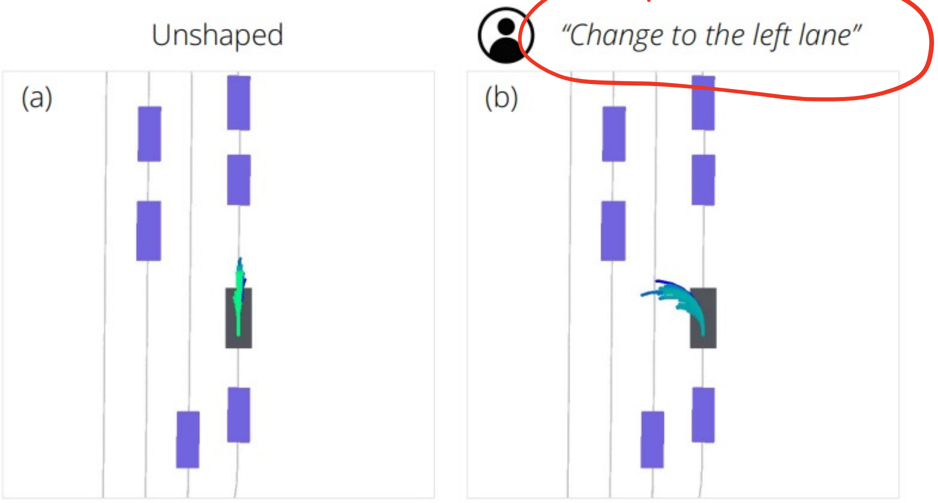
NYU

Test-Time Adaptation

- Diffusion can be combined / guided with reward functions at test time.



[Singhal et al. 2025]



[Yang et al. 2024]

Singhal et al. A General Framework for Inference-time Scaling and Steering of Diffusion Models. arXiv 2025.
Yang et al. Diffusion-ES: Gradient-free Planning with Diffusion for Autonomous and Instruction-guided Driving. CVPR 2024.

Diffusion for Detection

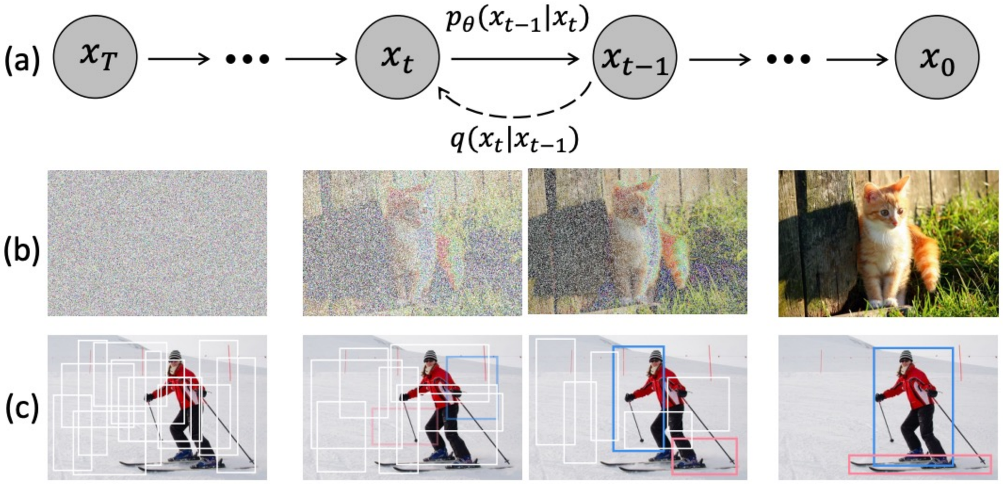
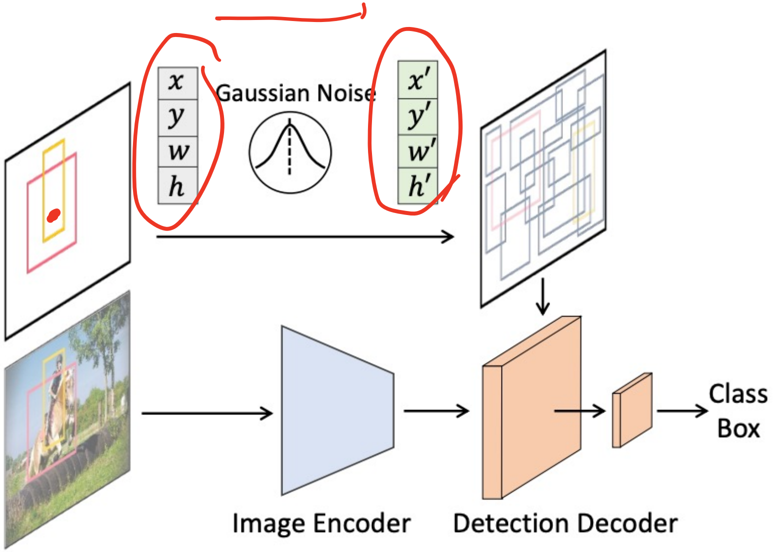


Figure 1. **Diffusion model for object detection**. (a) A diffusion model where q is the diffusion process and p_θ is the reverse process. (b) Diffusion model for image generation task. (c) We propose to formulate object detection as a denoising diffusion process from noisy boxes to object boxes.

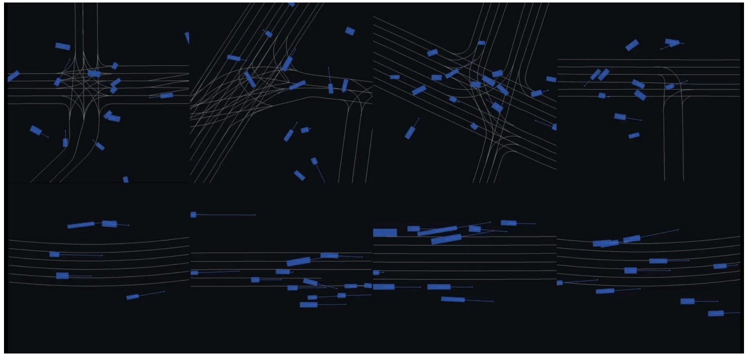


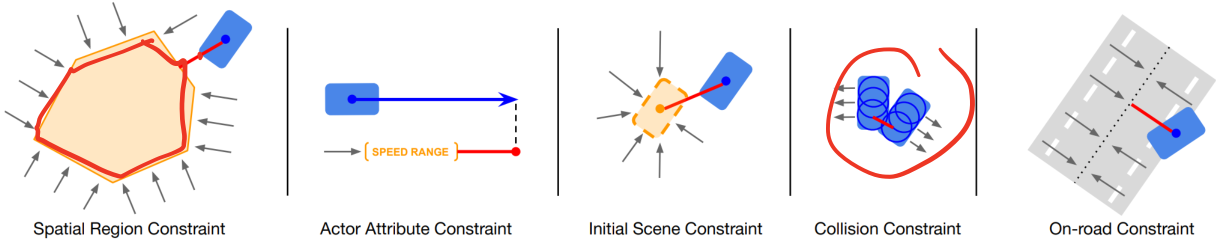
Chen et al. DiffusionDet: Diffusion Model for Object Detection. ICCV 2023.

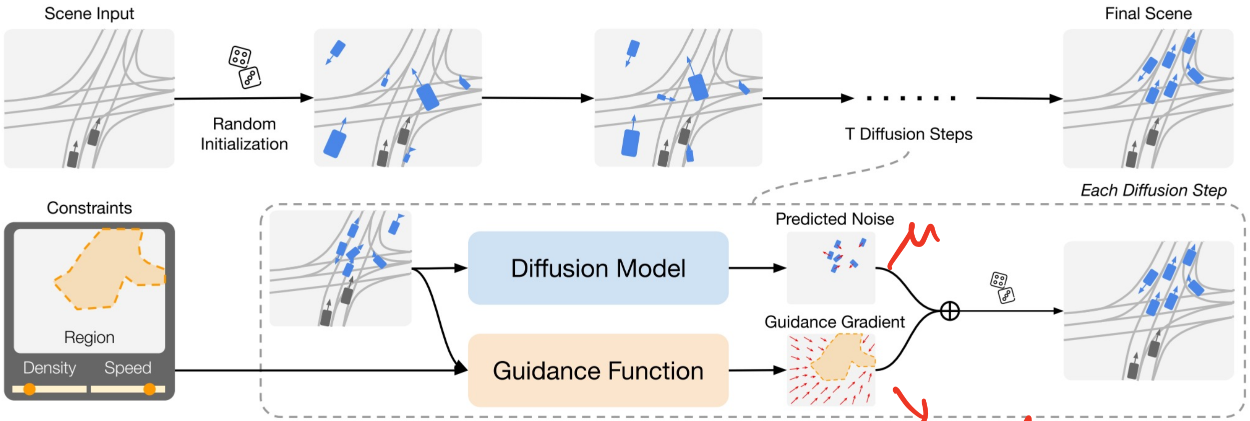
NYU

Diffusion for Generating Simulation Scenes

Hand Crafted





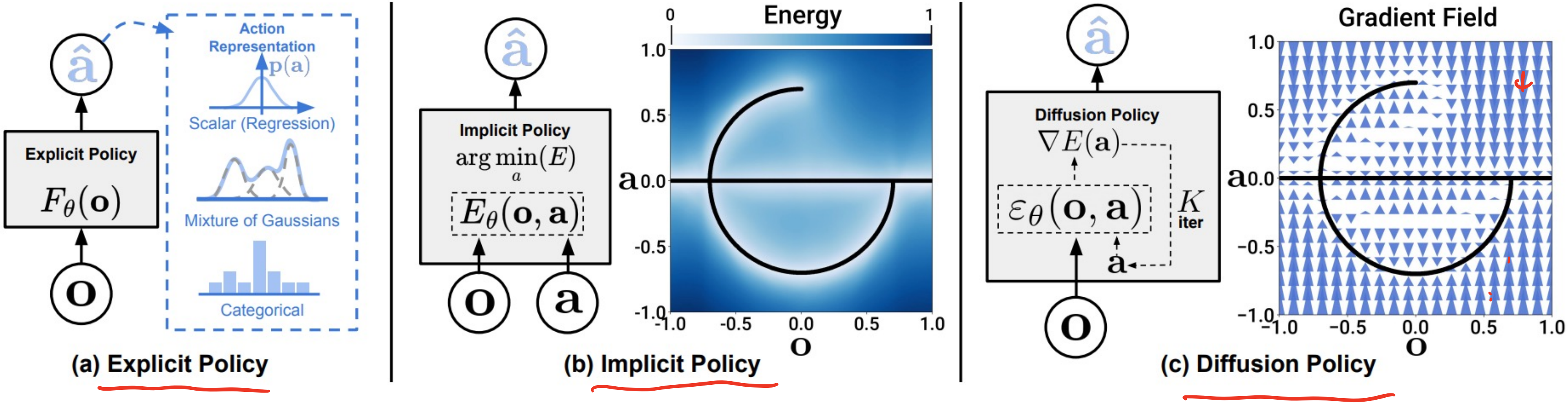


guidance

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

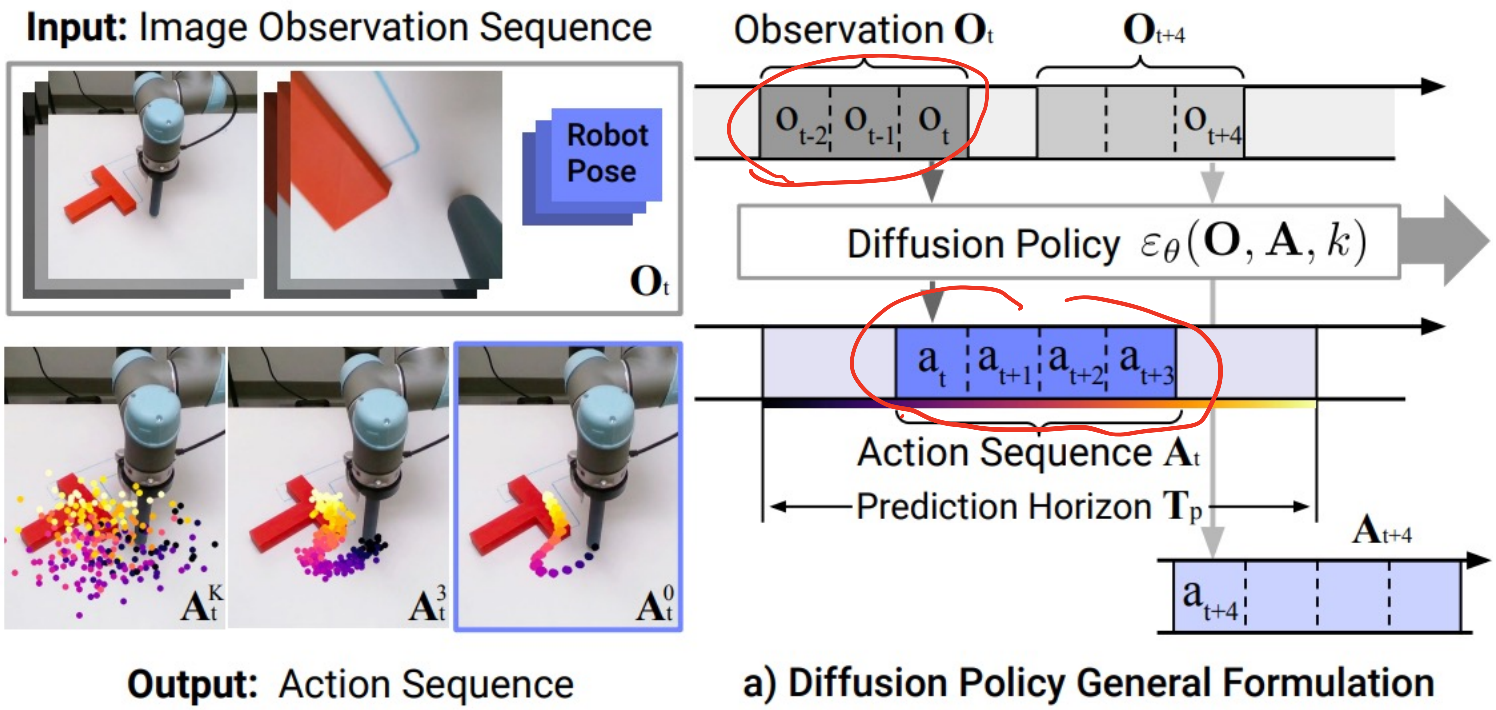
NYU

Diffusion for Planning and Control



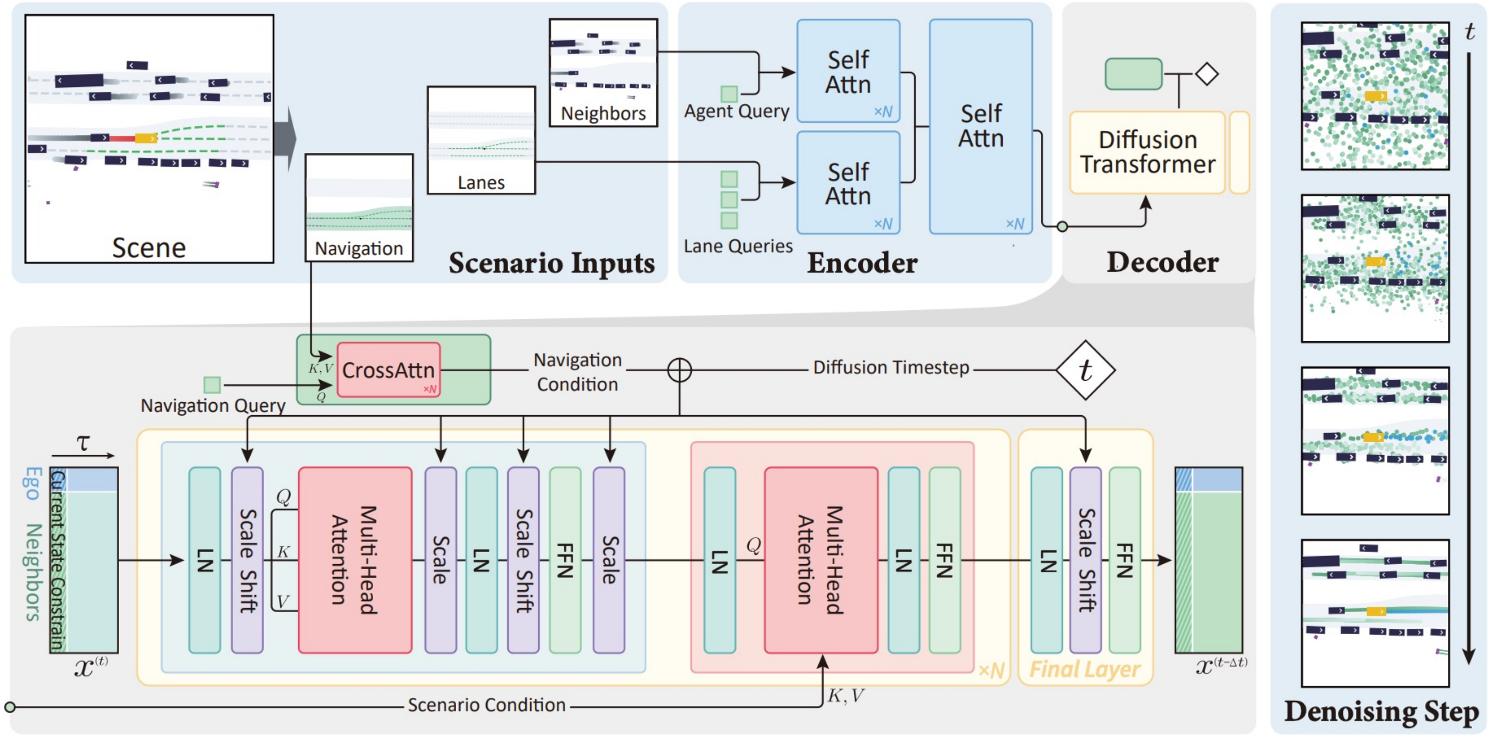
Chi et al. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. RSS 2023.

Diffusion for Planning and Control



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

Diffusion Planner for Self-Driving



https://openreview.net/forum?id=wM2sfVgMDH

Diffusion-Based Planning for Autonomous Driving with Flexible Guidance. OpenReview 2024.

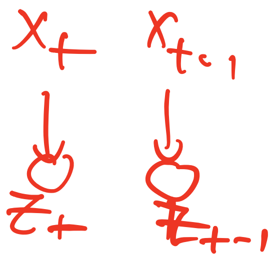
Summary: DL for Structured Outputs

Summary: DL for Structured Outputs

- Expanding the output dimension has limitations.

Summary: DL for Structured Outputs

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



Summary: DL for Structured Outputs

- Expanding the output dimension has limitations.
- Requires us thinking about generative models.
  - Graphical models
  - Autoregressive
  - Energy-based
  - Diffusion
- Understand pros and cons. Experiment with each option.

Summary: DL for Structured Outputs

- Expanding the output dimension has limitations.
- Requires us thinking about generative models.
  - Graphical models
  - Autoregressive
  - Energy-based
  - Diffusion
- Understand pros and cons. Experiment with each option.
- Application in embodied environments.

The World is 3D

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

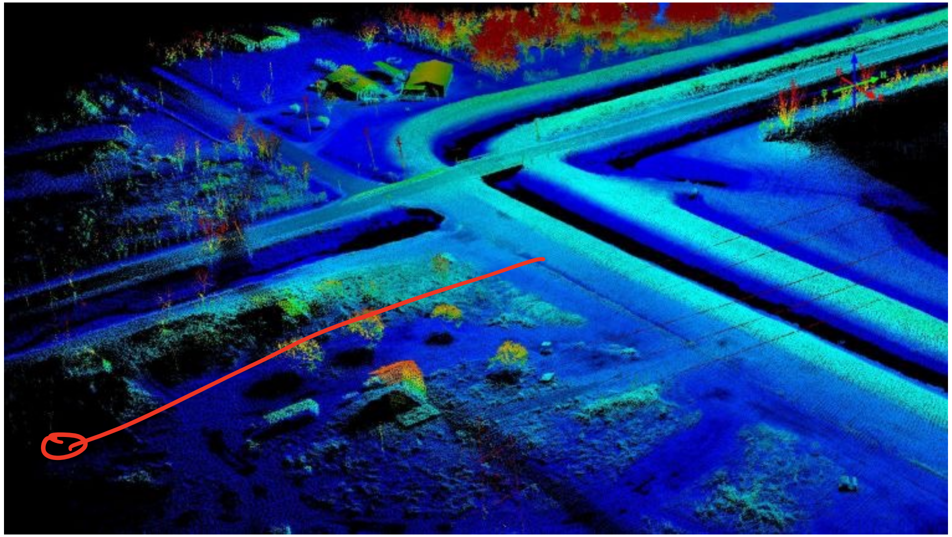
The World is 3D

- 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

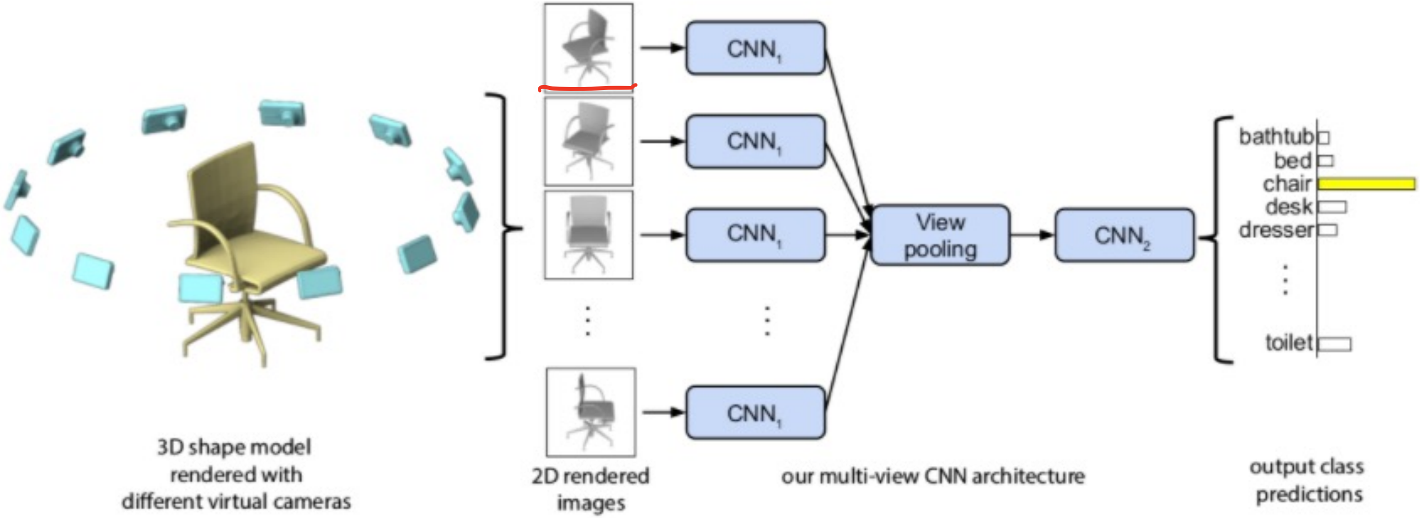
- We have previously focused on using 2D images as input.
- But, the world is 3D. Many non-rigid in 2D becomes rigid in 3D. There are also a wide range of 3D sensors.
- Stereo (our binocular vision), infrared camera, LiDAR, radar, etc.

LiDAR



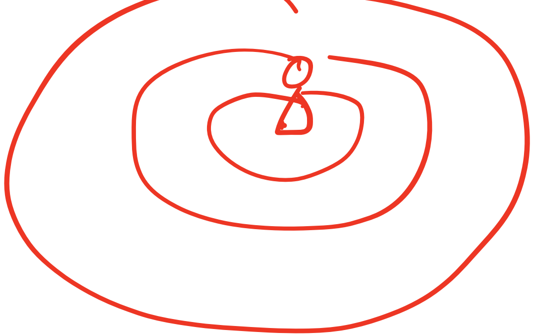
Multi-View CNN

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

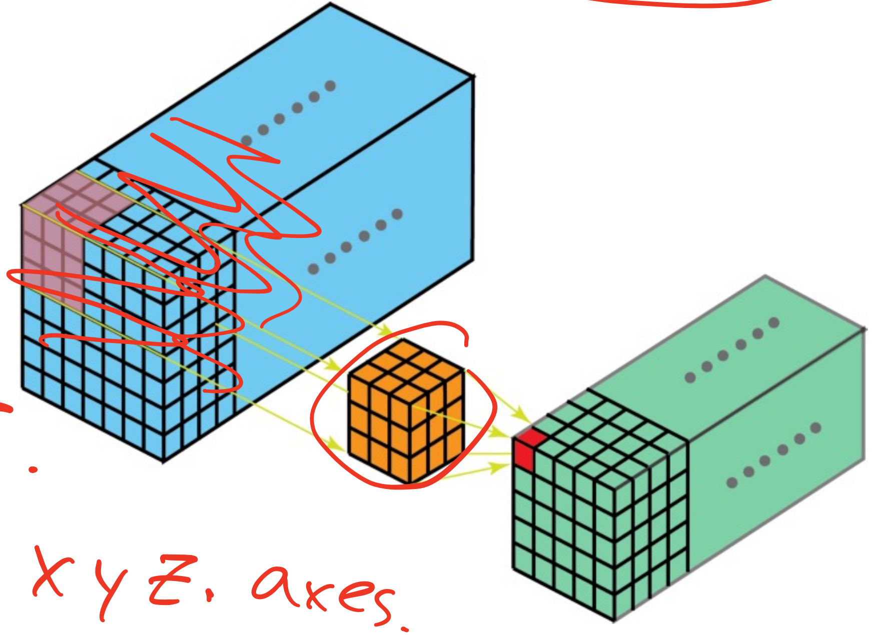


Su et al., Multi-view convolutional neural networks for 3D shape recognition, ICCV 2015.

3D Convolution on Voxels



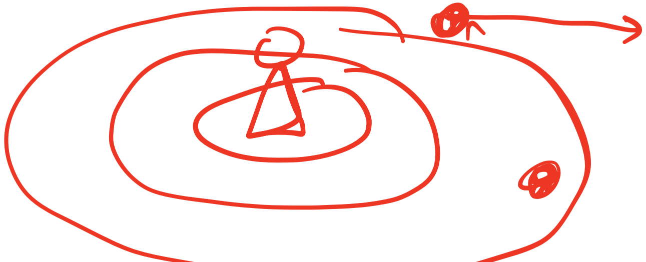
- 3D convolution on occupancy voxels.
- This can be expensive (memory + compute).



translational equivariant

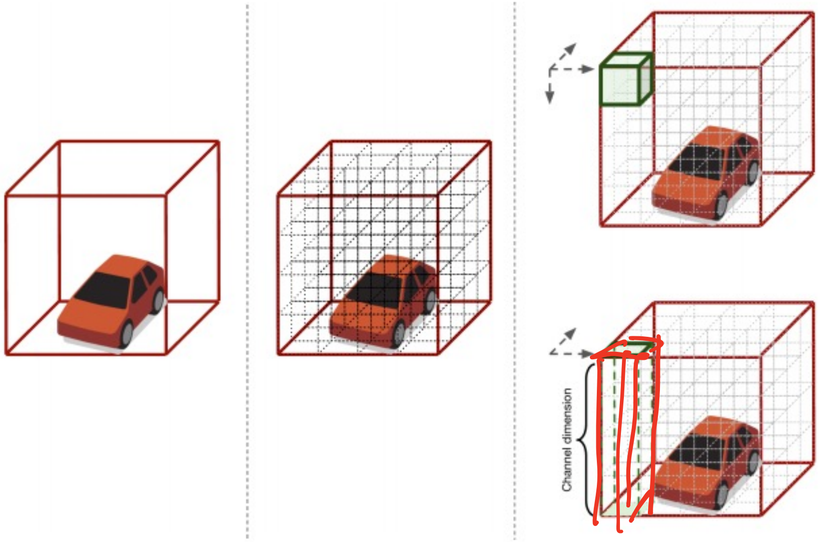
$$f(T(x)) = T(f(x))$$

xyz axes.



Bird's Eye View (BEV) Voxel

- Treat the z-dimension as different channels.

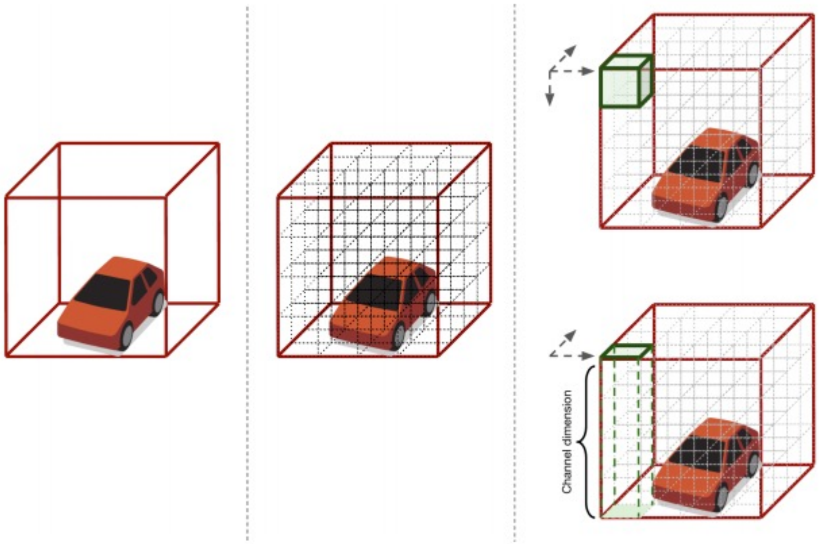


Zhang et al. Efficient convolutions for real-time semantic segmentation of 3D point clouds, 3DV 2018.1

NYU

Bird's Eye View (BEV) Voxel

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



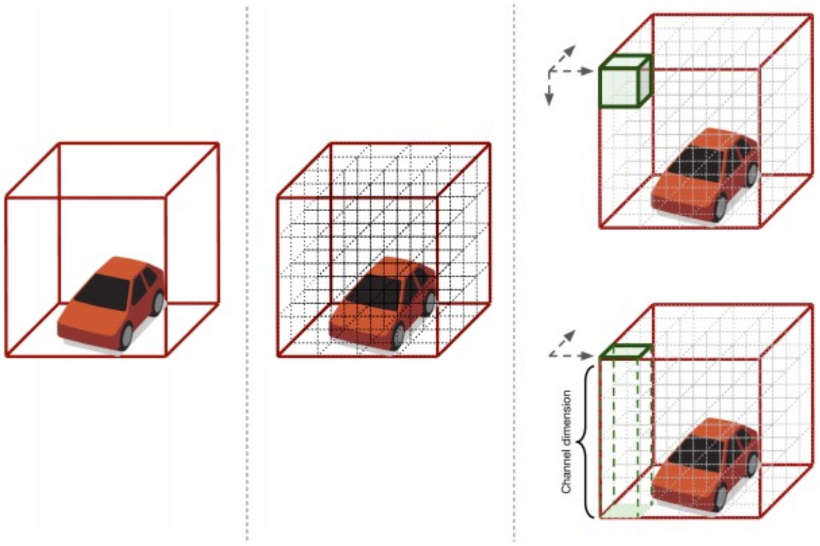
Zhang et al. Efficient convolutions for real-time semantic segmentation of 3D point clouds, 3DV 2018.1

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!)

x 80m.

3m

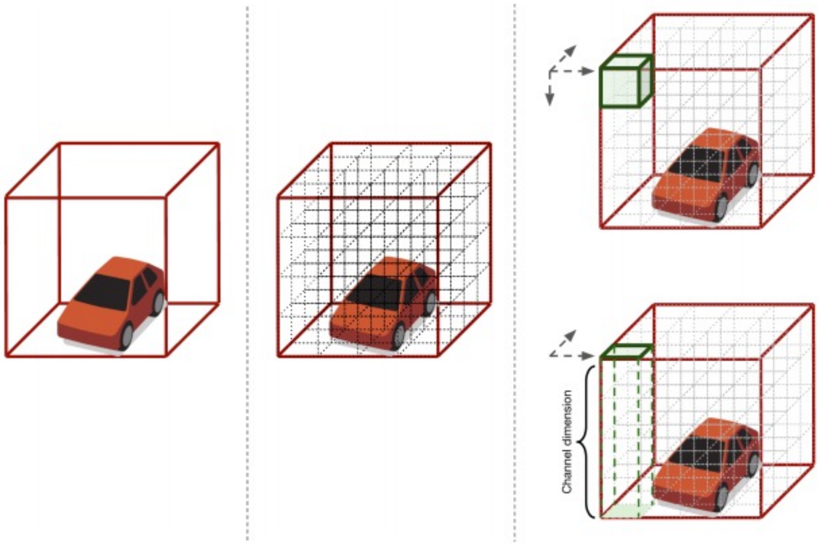


Zhang et al. Efficient convolutions for real-time semantic segmentation of 3D point clouds, 3DV 2018.1

NYU

Bird's Eye View (BEV) Voxel

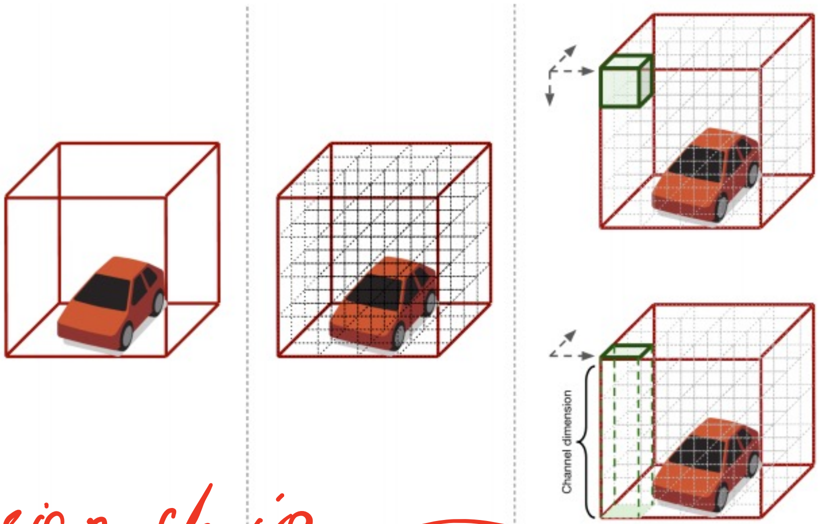
- Treat the z-dimension as different channels.
- 3D convolution ⟶ 2D convolution Popular in self-driving domain, e.g. 80m x
- 140m x 3m (very thin!)
- Transformation in x-y plane is still rigid.



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

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)



design choice

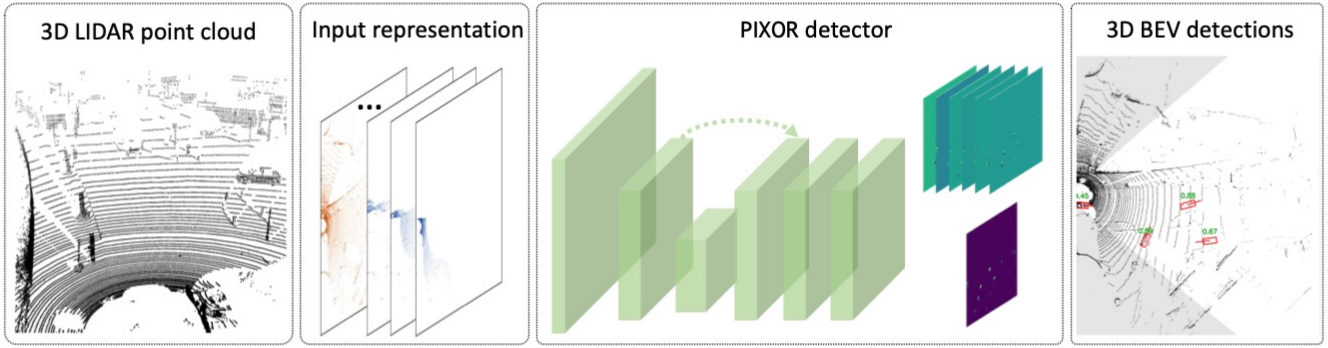
dense sparser

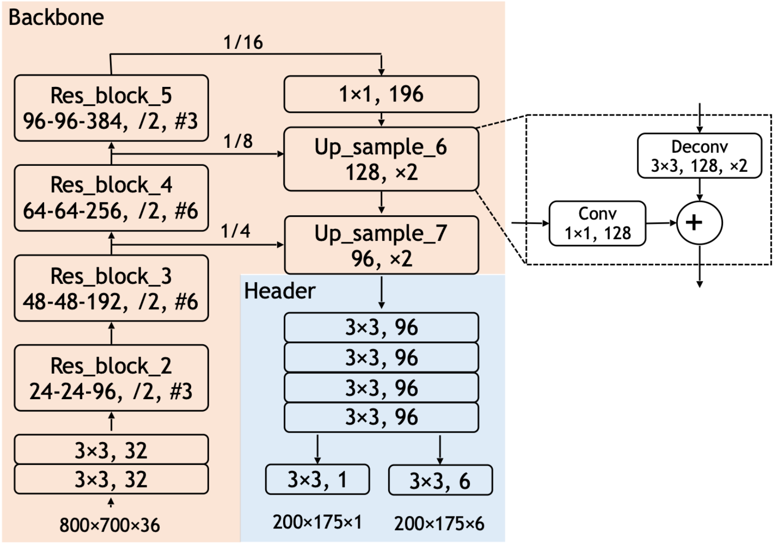
Zhang et al. Efficient convolutions for real-time semantic segmentation of 3D point clouds, 3DV 2018.1

NYU

PIXOR

- Bird's eye view object detection.

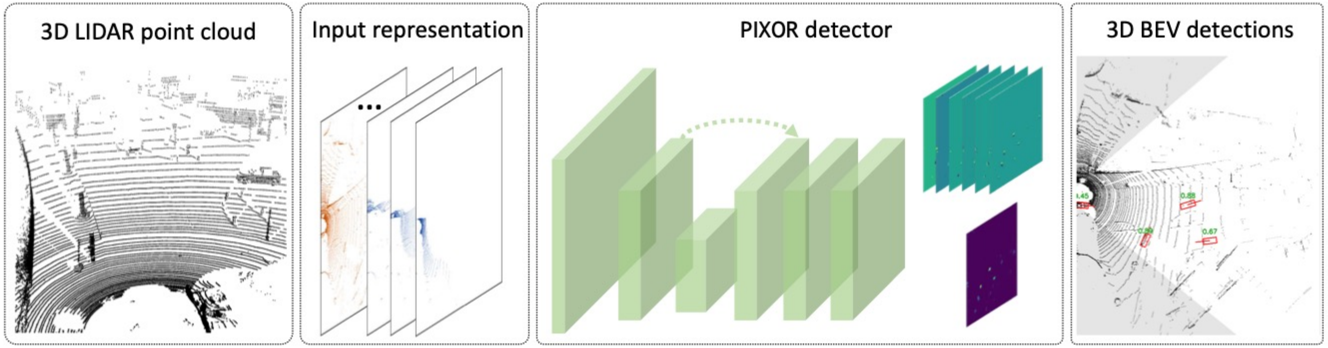


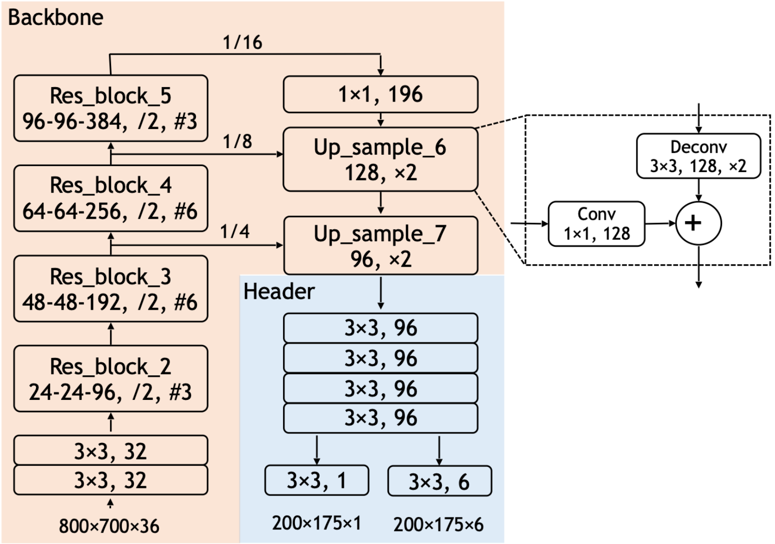


Yang et al. PIXOR: Real-time 3D Object Detection from Point Clouds. CVPR 2018.

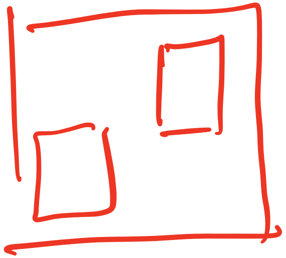
PIXOR

- 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.

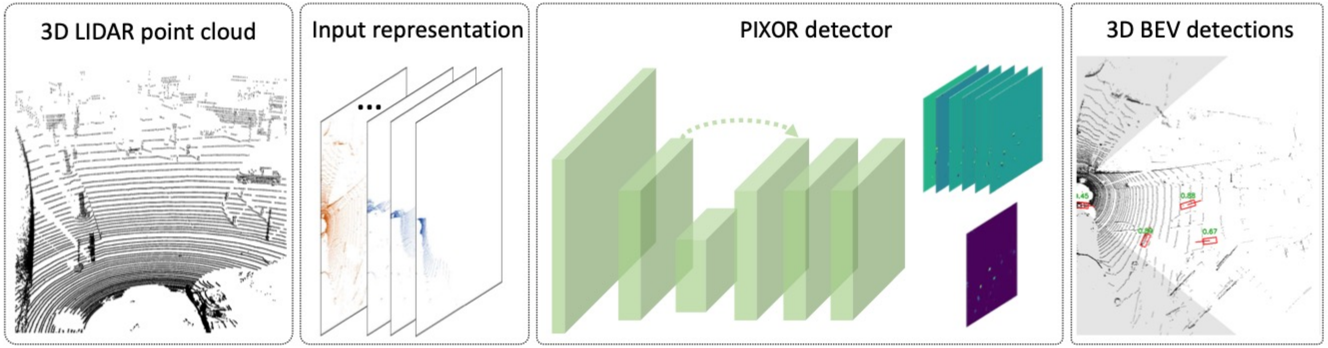


140m x 80m

PIXOR

224 x 224

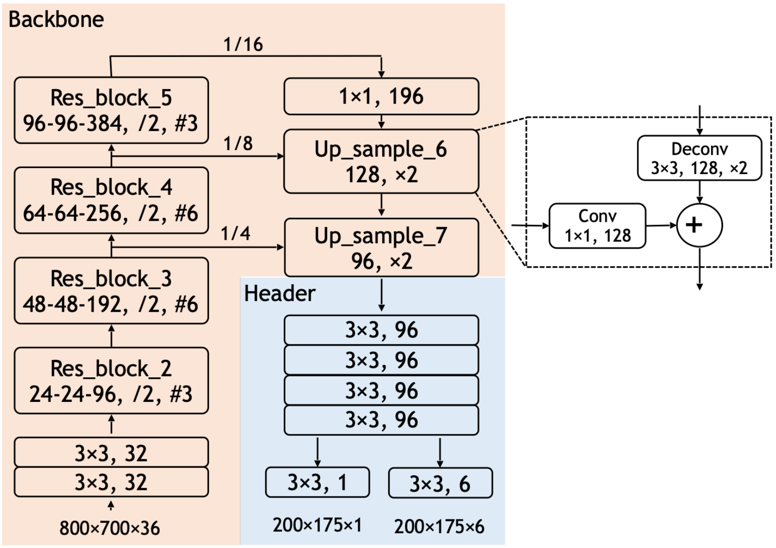
600 x 1000



- Bird's eye view object detection.
- Used the ResNet + FPN network single-stage architecture.
- Detection: Classification + regression $\cos\theta, \sin\theta, dx, dy, w, l$.

$0, 2\pi$

$\operatorname{atan}\left(\frac{\sin\theta}{\cos\theta}\right)$.

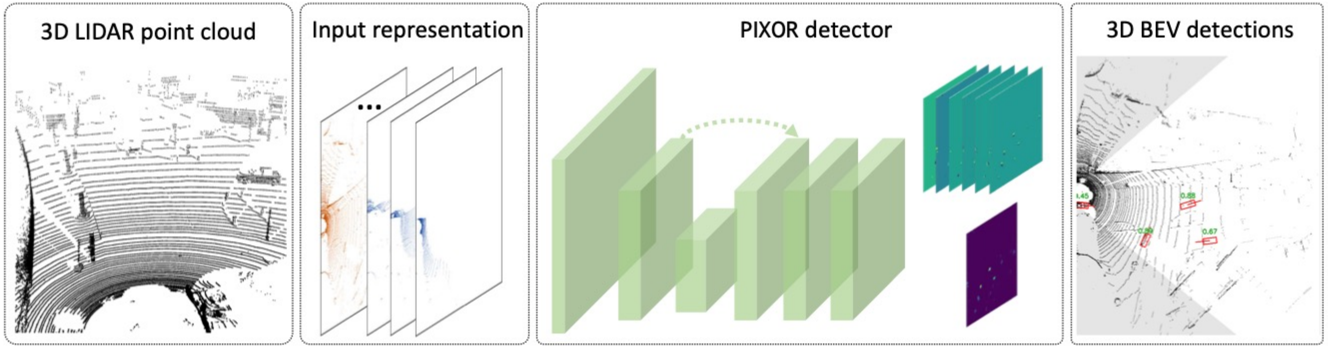


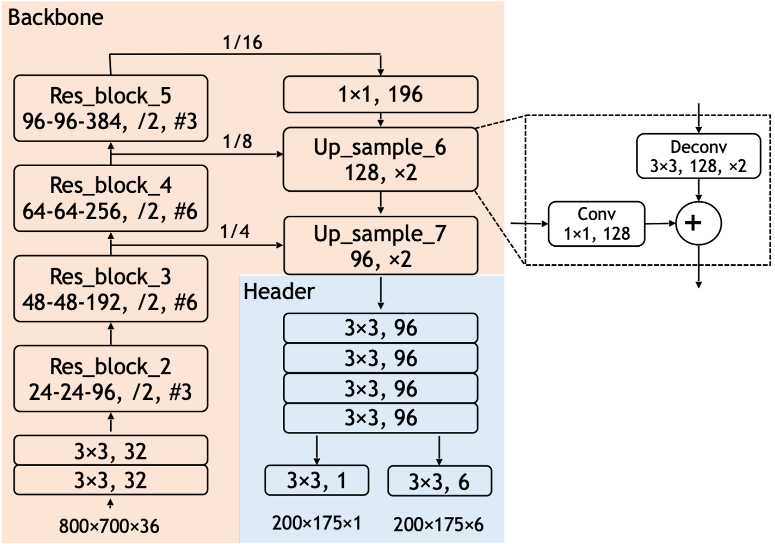
Yang et al. PIXOR: Real-time 3D Object Detection from Point Clouds. CVPR 2018.

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.



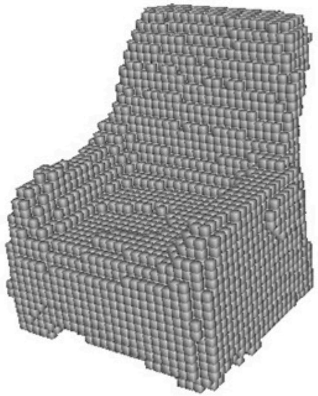


Yang et al. PIXOR: Real-time 3D Object Detection from Point Clouds. CVPR 2018.

NYU

Point Cloud

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

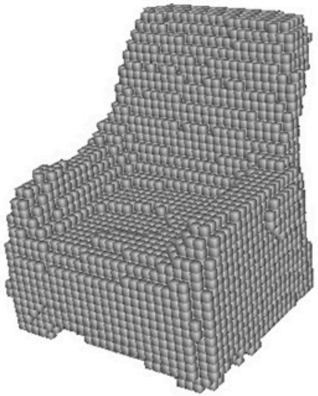


3D voxels

3D point cloud

Point Cloud

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



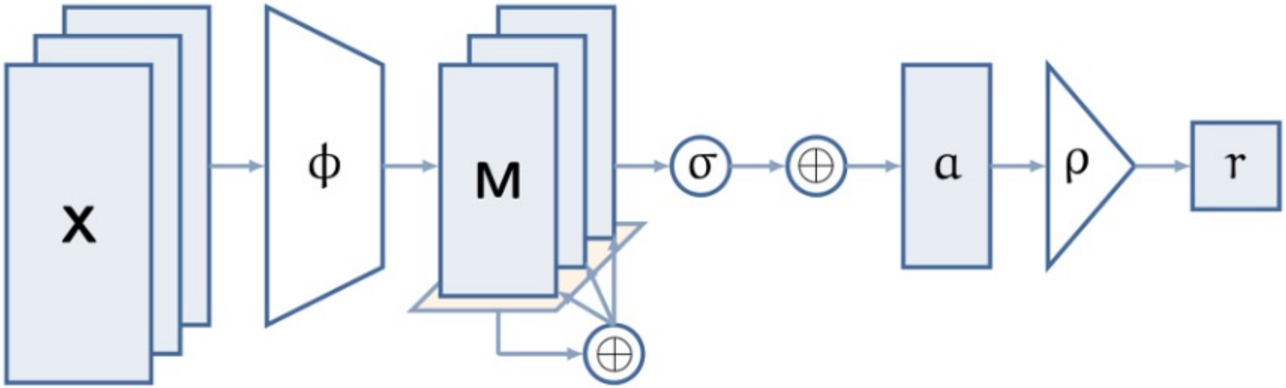
3D voxels

3D point cloud

NYU

Permutation Invariance

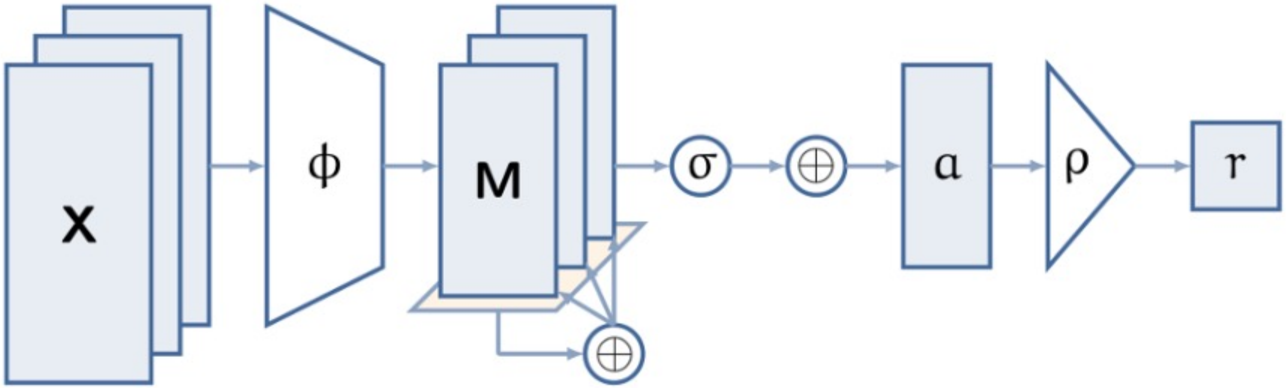
- Point cloud is a set.



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

Permutation Invariance

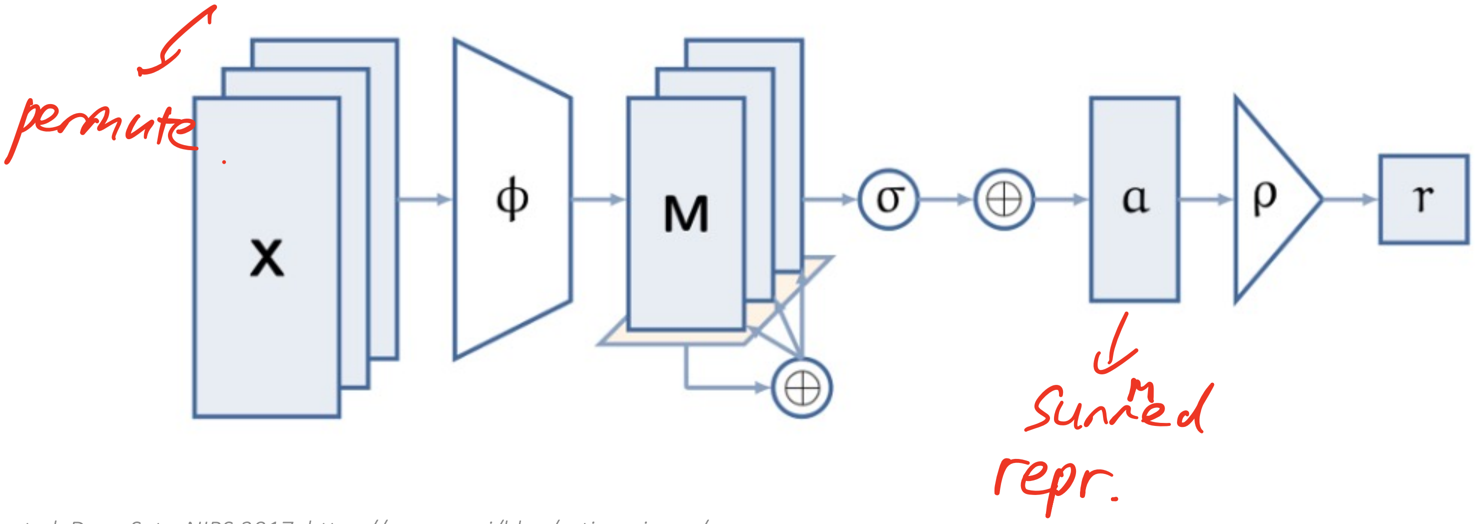
- Point cloud is a set.
- Permutation does not affect the classification in the output.



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

Permutation Invariance

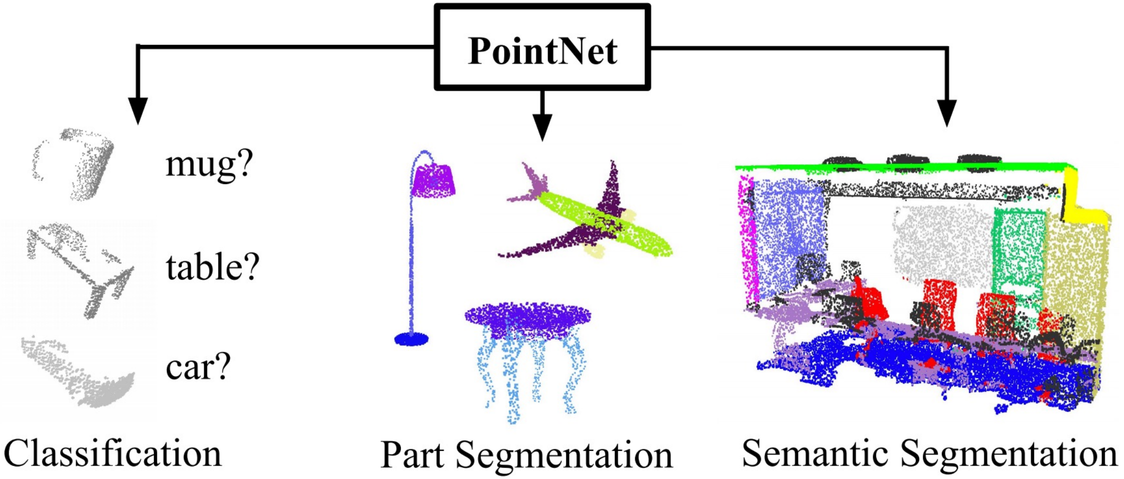
- Point cloud is a set.
- Permutation does not affect the classification in the output.
- What operations are permutation invariant?



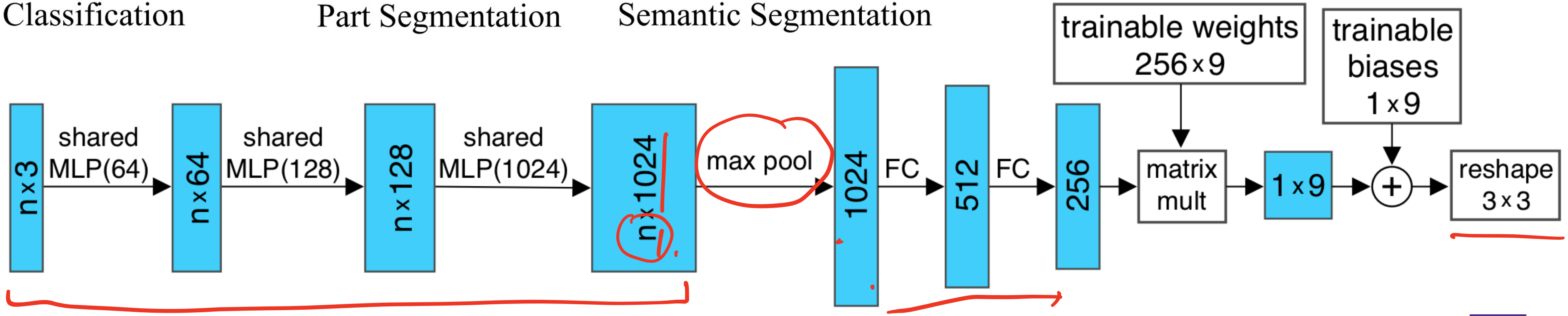
Zaheer et al. Deep Sets. NIPS 2017. https://argmax.ai/blog/setinvariance/

NYU

PointNet

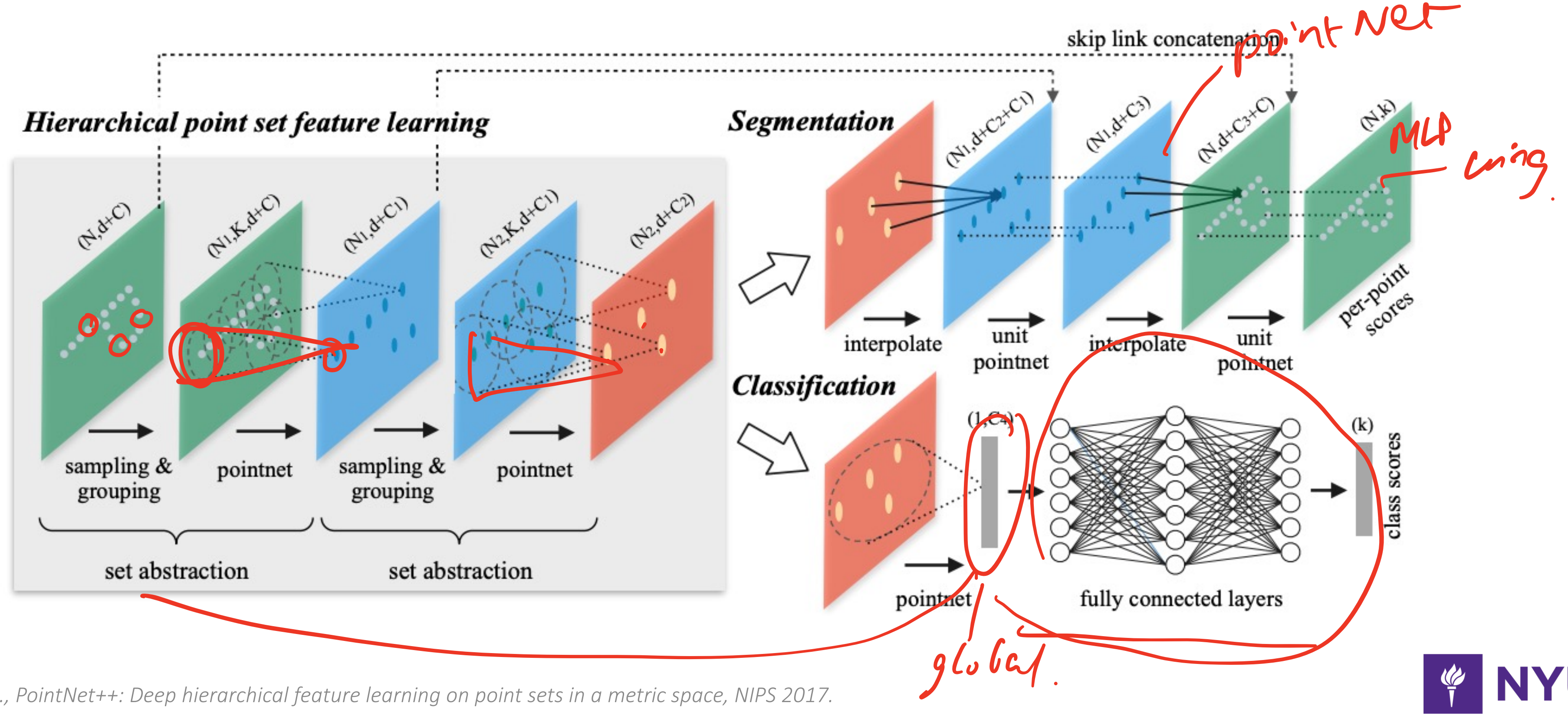


- Apply an MLP on each point.
- Max pool the features across all points.



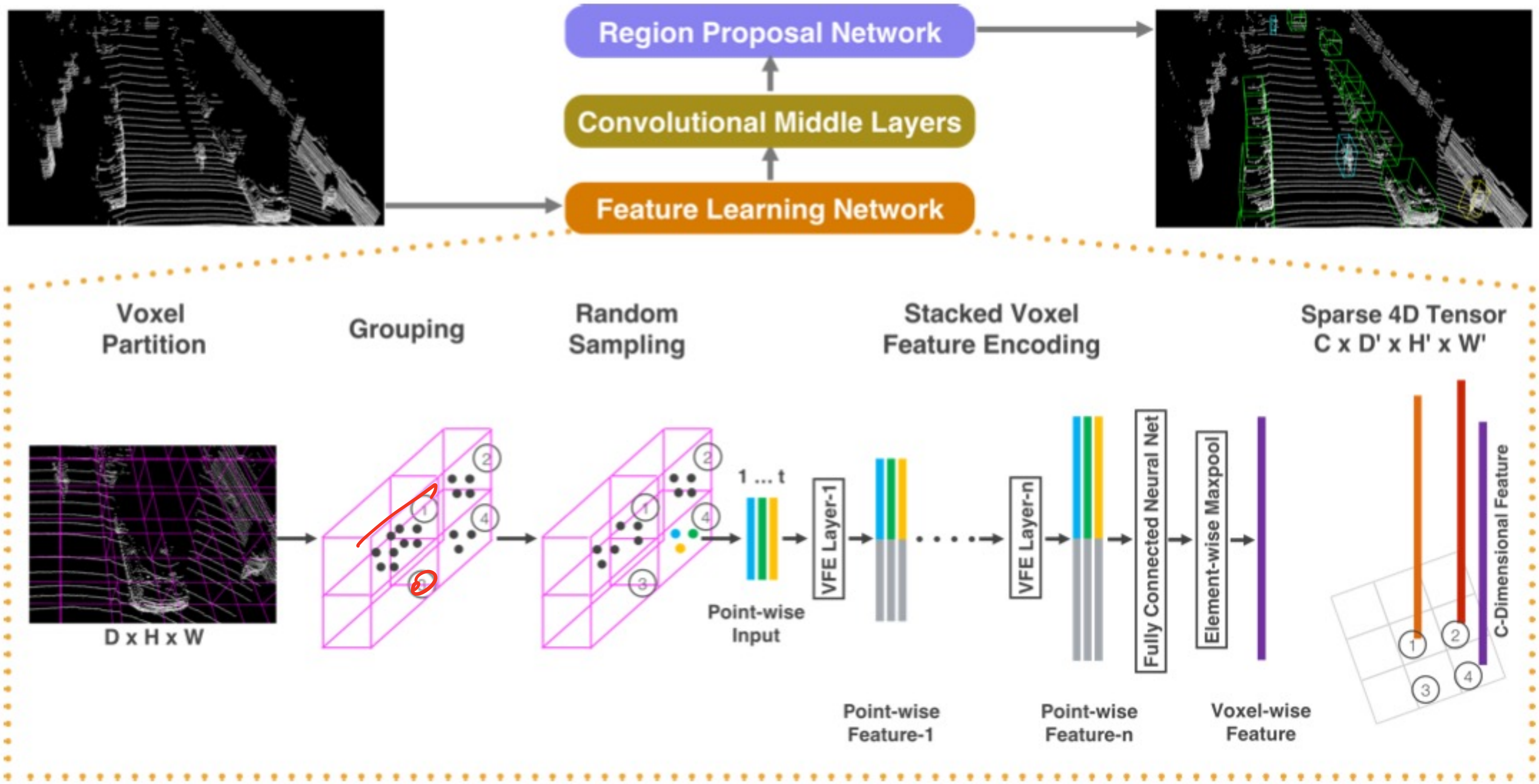
Qi et al., PointNet: Deep learning on point sets for 3D classification and segmentation, CVPR 2017.

PointNet++



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

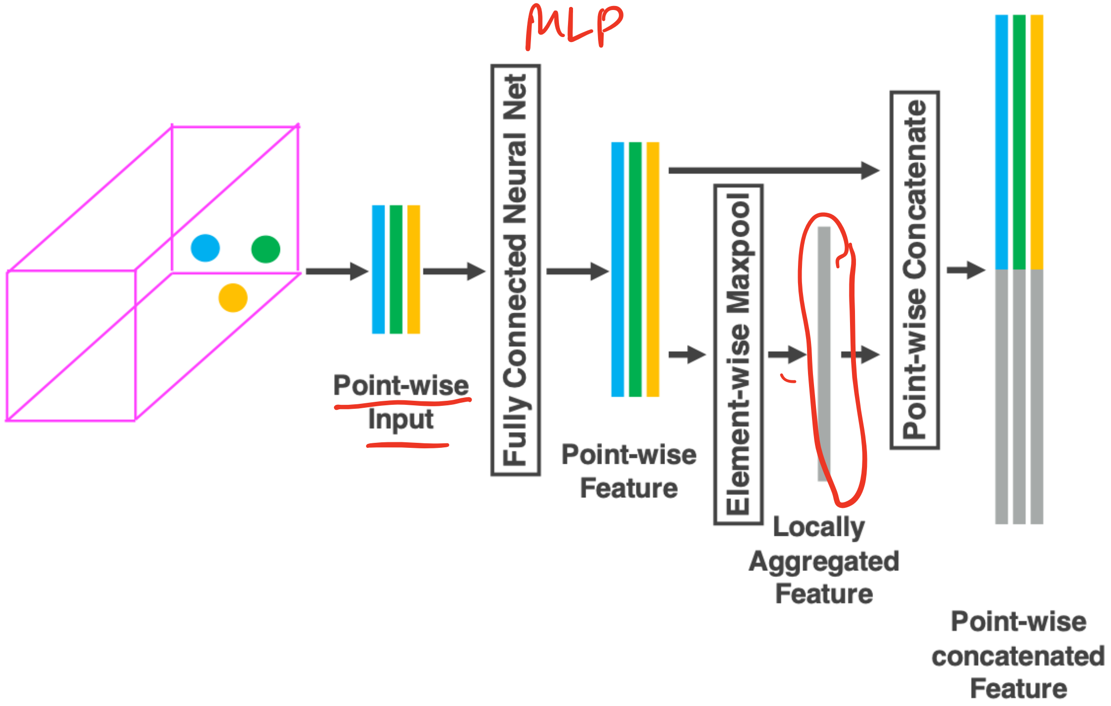
VoxelNet



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

VoxelNet

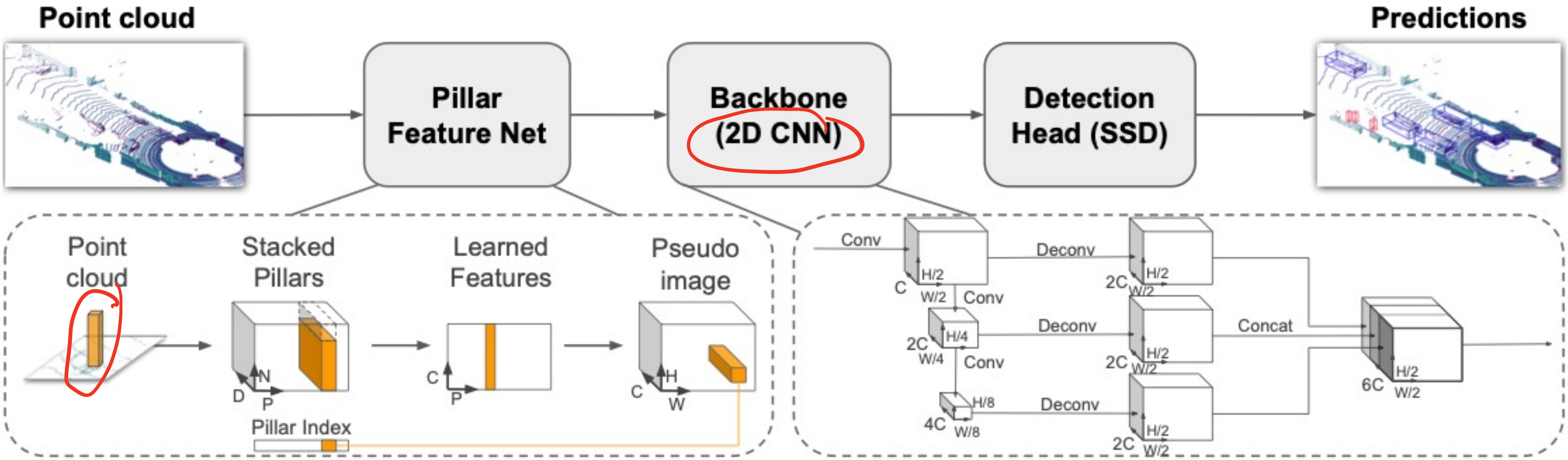
- Zooming inside voxel feature encoding (VFE)



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

NYU

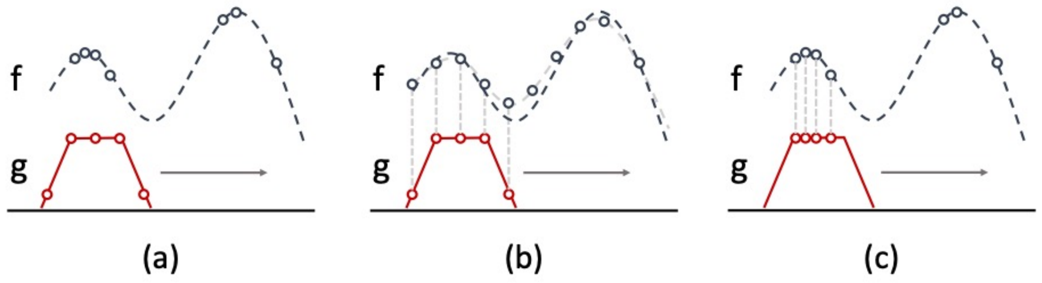
PointPillar

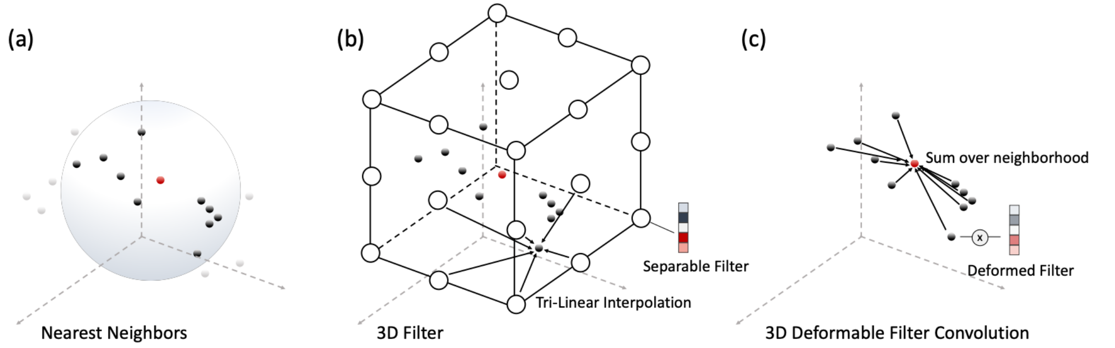


Lang et al. PointPillars: Fast Encoders for Object Detection from Point Clouds. CVPR 2019.

Deformable Convolution in Point Cloud

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

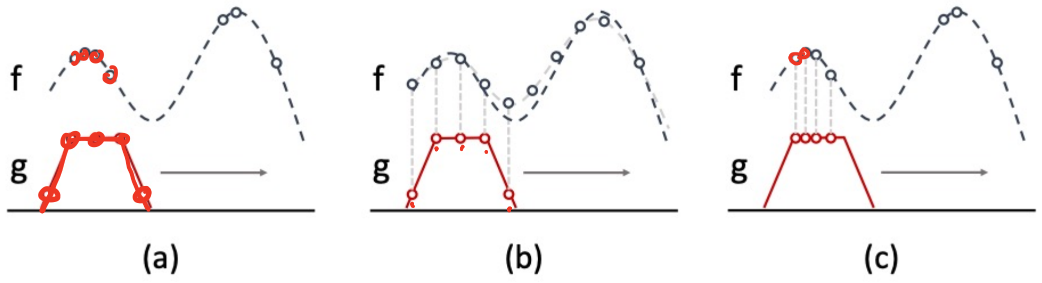


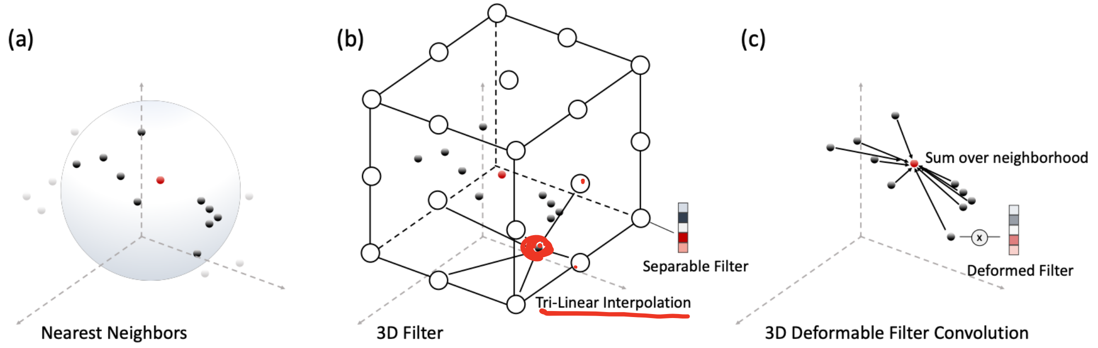


Xiong et al. Deformable Filter Convolution for Point Cloud Reasoning. 2019.

Deformable Convolution in Point Cloud

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

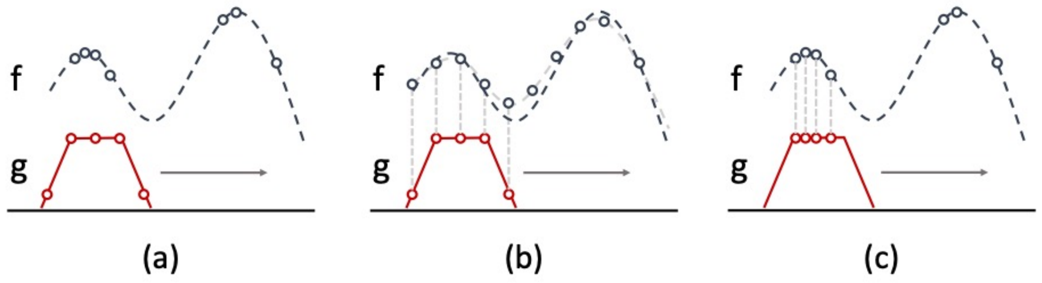


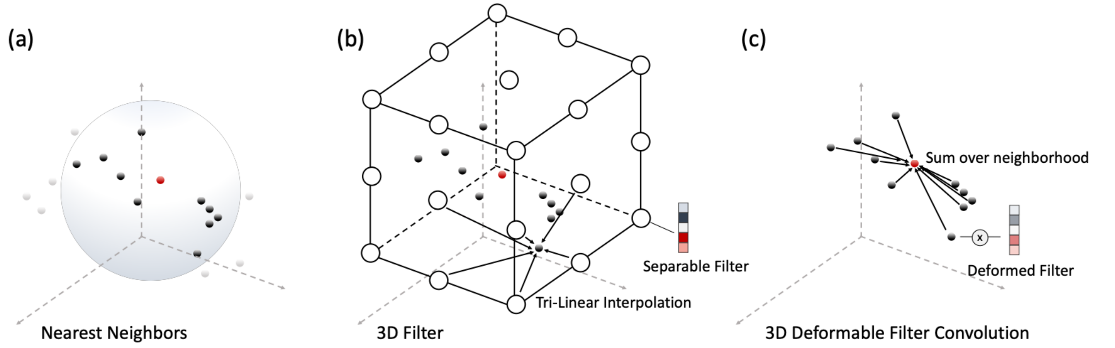


Xiong et al. Deformable Filter Convolution for Point Cloud Reasoning. 2019.

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.



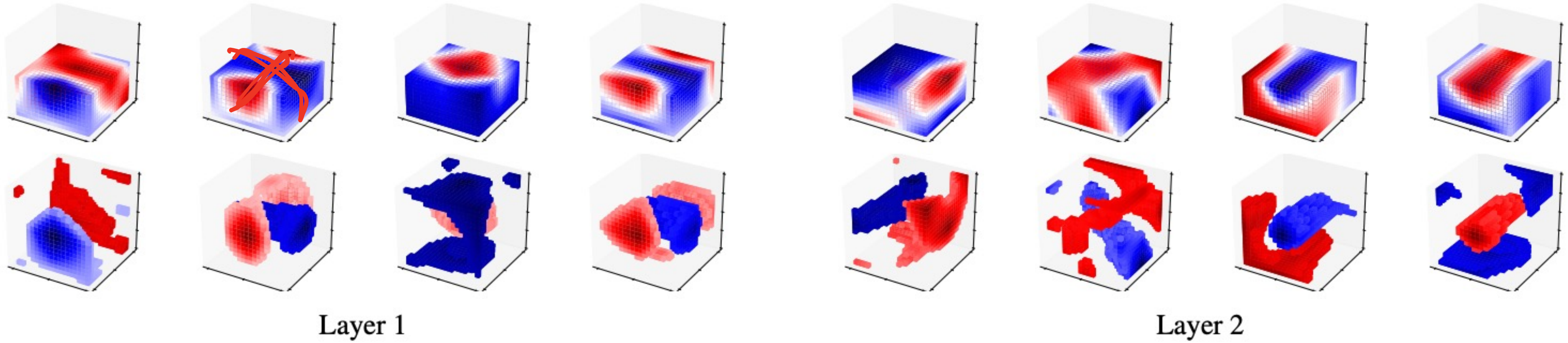


Xiong et al. Deformable Filter Convolution for Point Cloud Reasoning. 2019.

NYU

3D Filters

- Visualizing 3D convolution kernels.



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

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.)

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?

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).

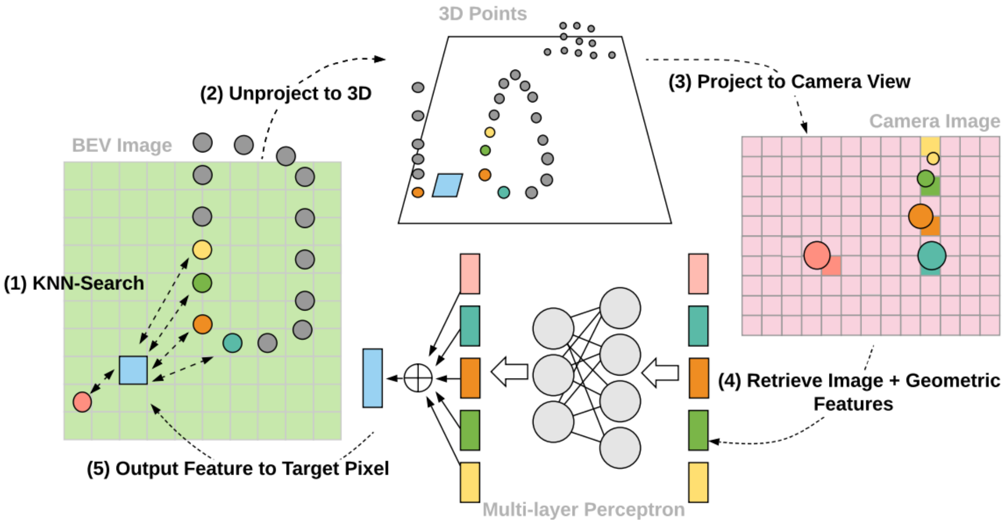
Multi-Sensor Fusion

- 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?

NYU

Camera-LiDAR Projection

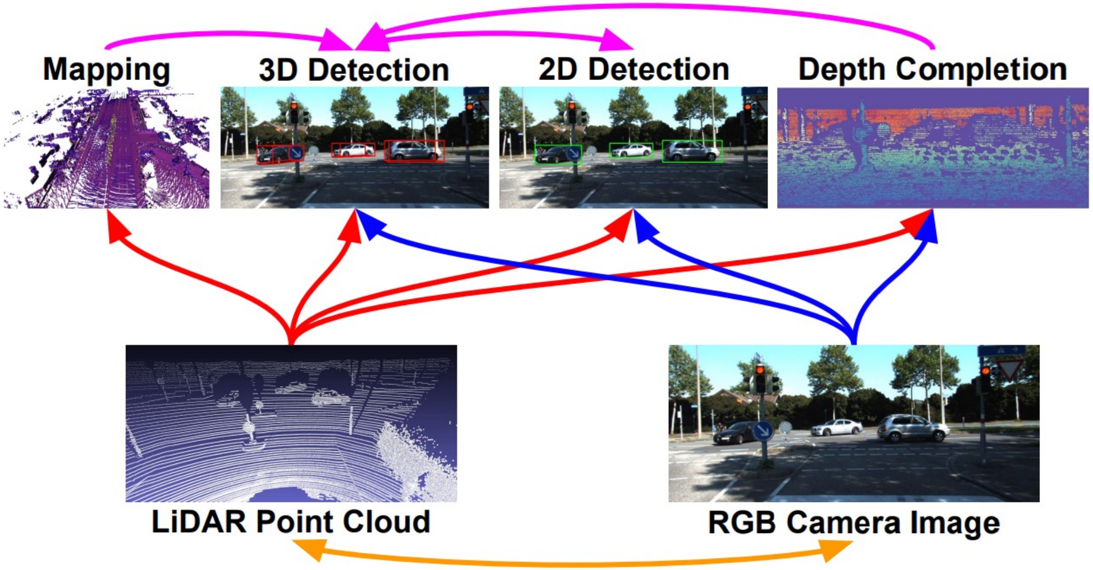
- Unproject LiDAR points to camera view (i.e. Range View)
- Query the closest camera RGB features for each LiDAR point.
- For empty space in BEV, we can interpolate from neighboring points using kNN.
- Continuous Fusion: $h_i = \sum_j MLP([f_j, x_j - x_i])$.



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

Supervised Dense Depth

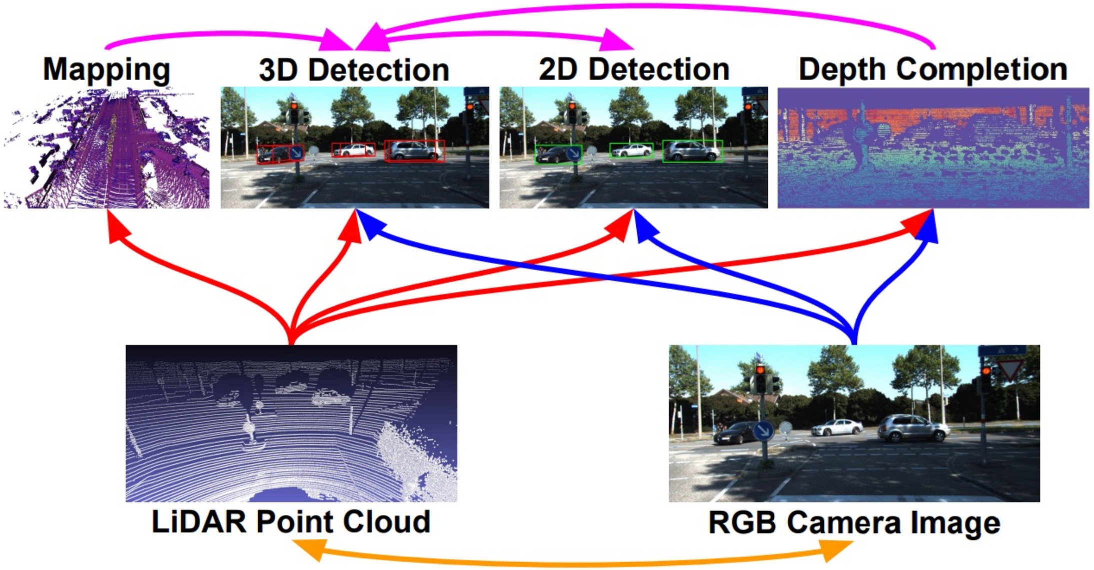
- Drawback of continuous fusion: Sparse LiDAR can cause the fusion process to be less accurate. Relies on kNN.



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

Supervised Dense Depth

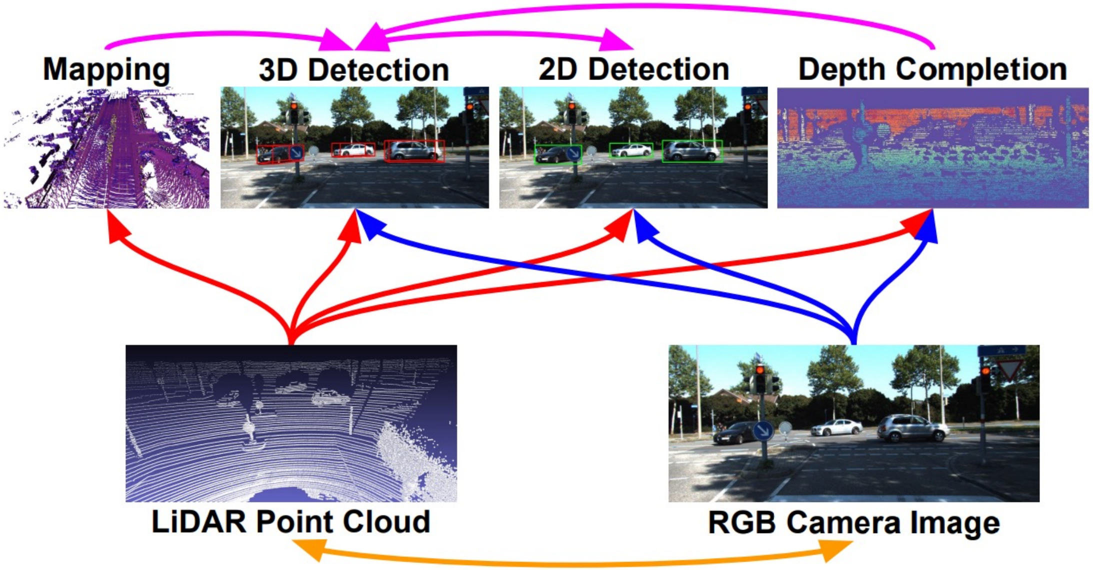
- Drawback of continuous fusion: Sparse LiDAR can cause the fusion process to be less accurate. Relies on kNN.
- Why not predict a dense depth to pair with the camera image?



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

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.

3D Perception

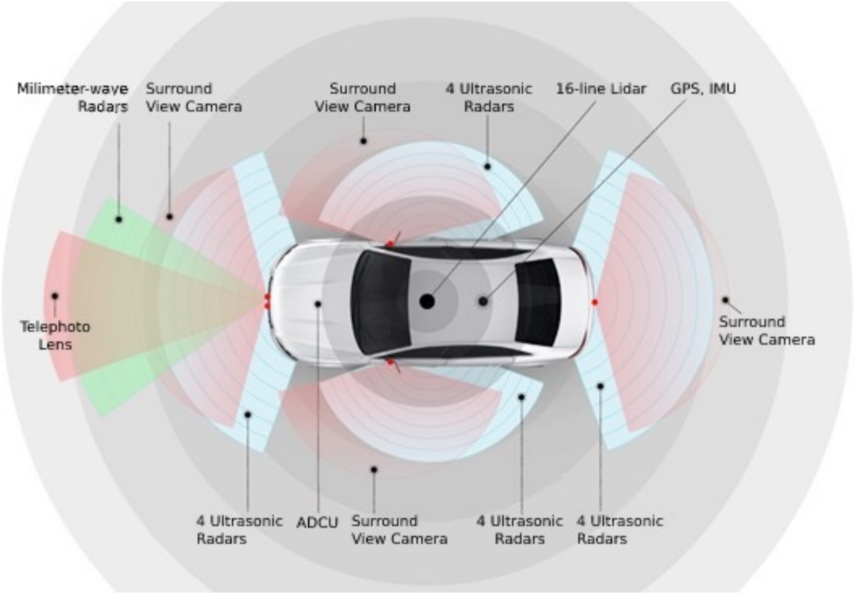
- With the ease of use of automatic differentiation libraries, we can compose a computation graph in millions of ways.
- We can design layers and operators to accomodate different types of inputs and outputs. 3D, point cloud, sparse data, etc.

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.

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?



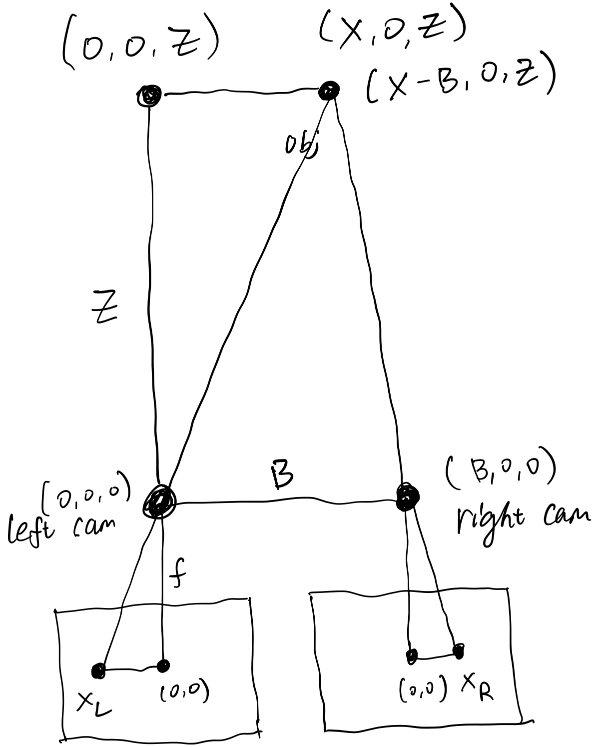
NYU

Classic Vision on Depth and Disparity

- One source of depth is from the displacement of pixels in a stereo setup.

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.



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

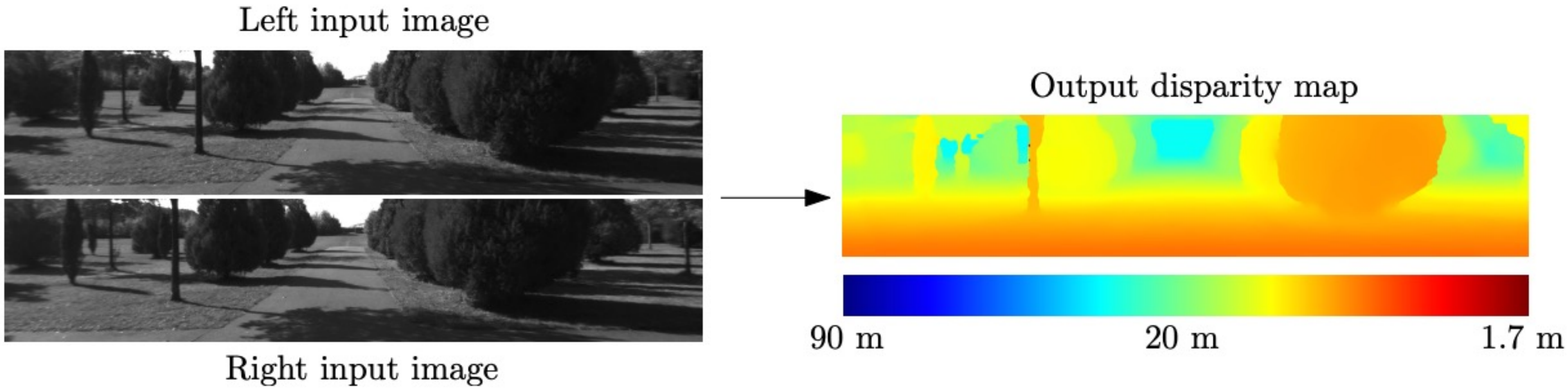
$$x_R = \frac{(X-B)f}{Z}$$

$$\begin{aligned}\delta &= x_L - x_R \\ &= \frac{[X-(X-B)]f}{Z} \\ &= \frac{Bf}{Z}\end{aligned}$$

$$\boxed{Z = \frac{Bf}{\delta}}$$

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.

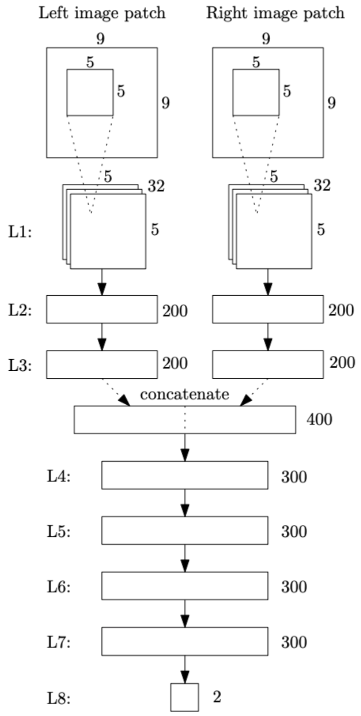
The Energy-Based Approach

- 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

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

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

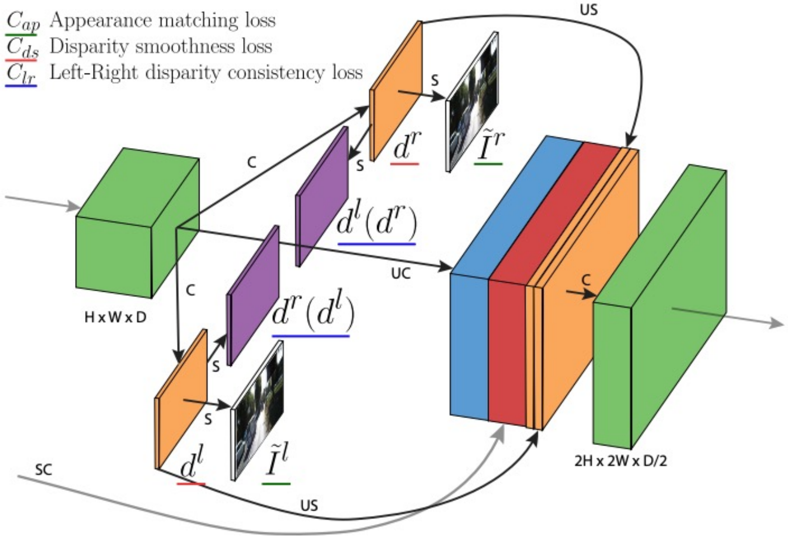


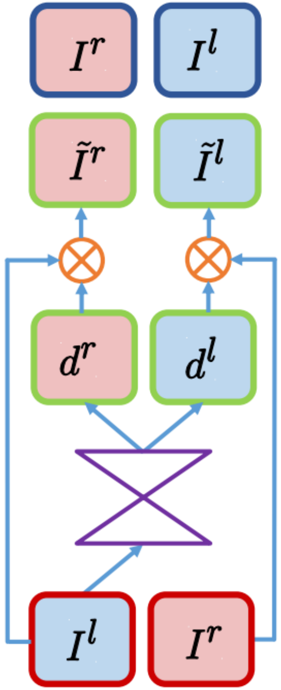
Zbontar & LeCun. Computing the Stereo Matching Cost with a Convolutional Neural Network. CVPR 2015.

Self-Supervised Depth

- Appearance matching loss

$$C^l_{ap} = \frac{1}{N}\sum_{i,j}\alpha\frac{1-\mathrm{SSIM}(I^l_{ij},\tilde{I}^l_{ij})}{2} + (1-\alpha)\left\| I^l_{ij} - \tilde{I}^l_{ij} \right\|.$$





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

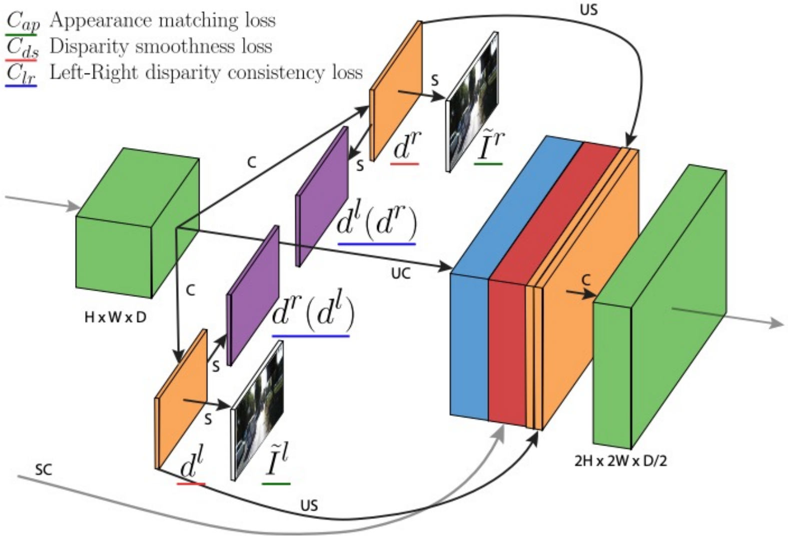
Self-Supervised Depth

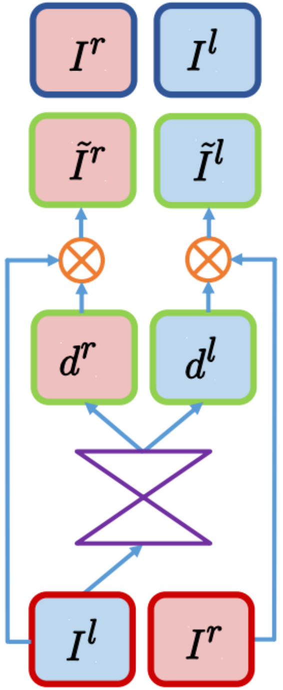
- Appearance matching loss

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

- 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.

Self-Supervised Depth

- Appearance matching loss

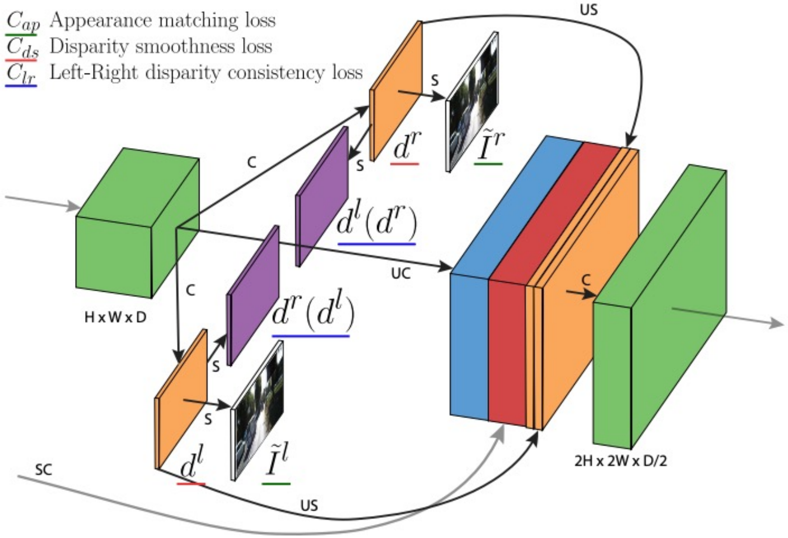
$$C_{ap}^{l} = \frac{1}{N}\sum_{i,j} \alpha \frac{1-\mathrm{SSIM}(I_{ij}^{l}, \tilde{I}_{ij}^{l})}{2} + (1-\alpha)\left\| I_{ij}^{l} - \tilde{I}_{ij}^{l} \right\|.$$

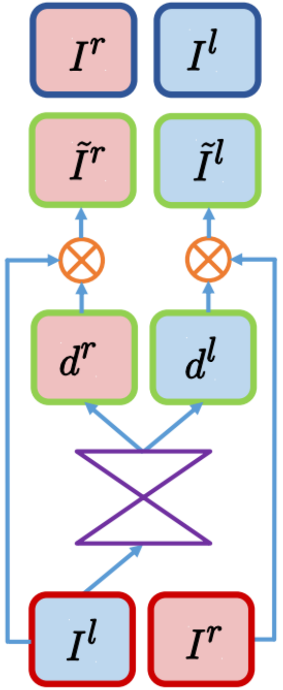
- Disparity smoothness loss

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

- 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.

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.

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

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.

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

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

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

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.

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

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

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

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

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

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.

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

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

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

$$I_y = \frac{\partial I}{\partial y} \qquad \boxed{I_x u + I_y v + I_t = 0.}$$

NYU

Classical Approach

- Under-constrained system

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

Classical Approach

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

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

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

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

Classical Approach

- Under-constrained system

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

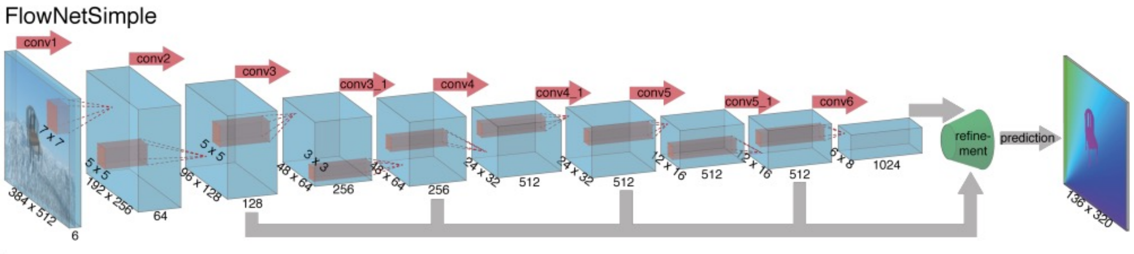
- Use a local patch and assume smooth motion
- Rigid, contains many assumptions

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

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

Correlation Volume Approach

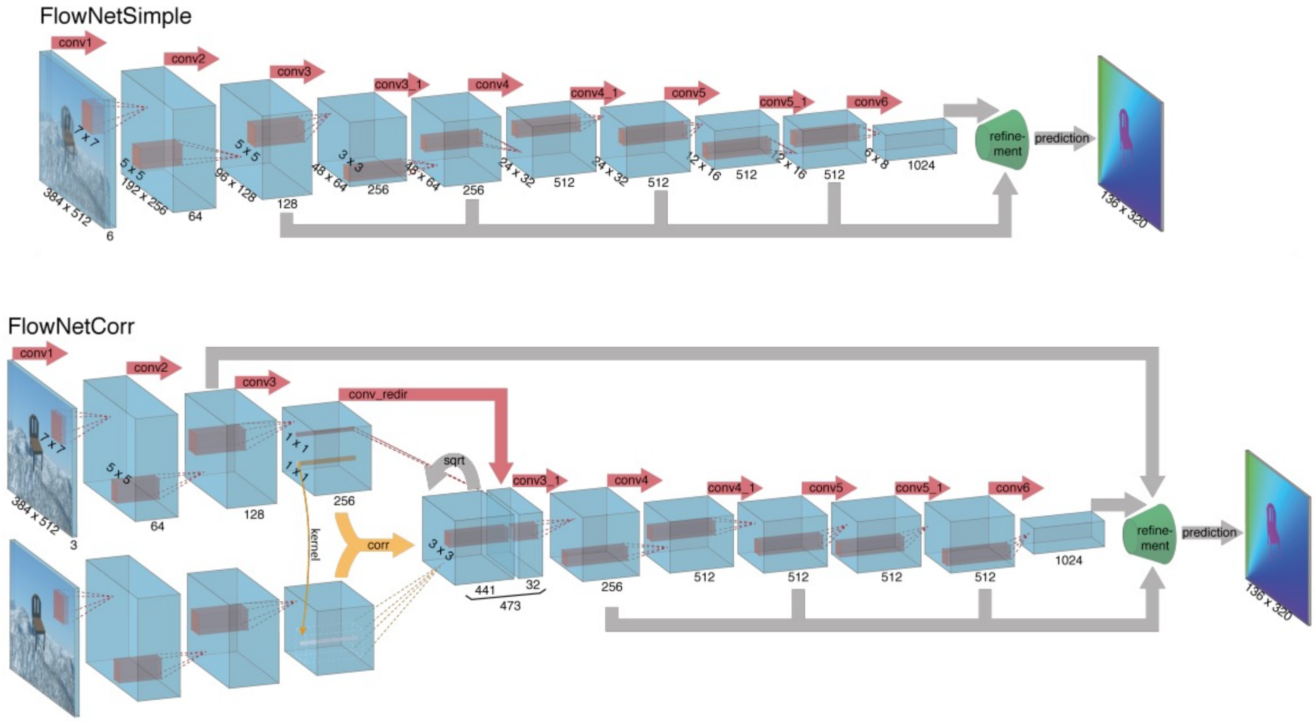
- Simple Approach: Concatenate the two images together.



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

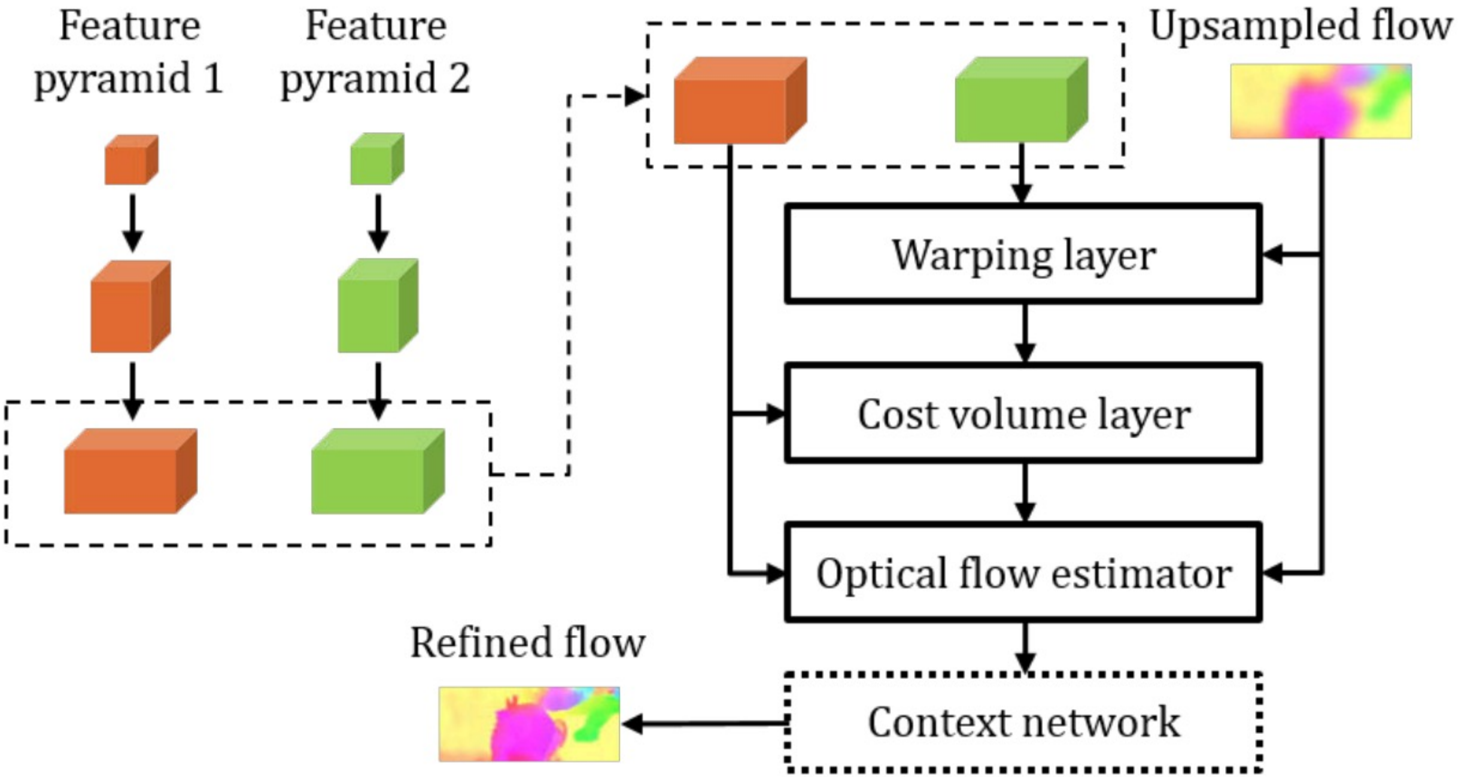
Correlation Volume Approach

- 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.

Iterative Refining through Feature Pyramid



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

Unsupervised Flow

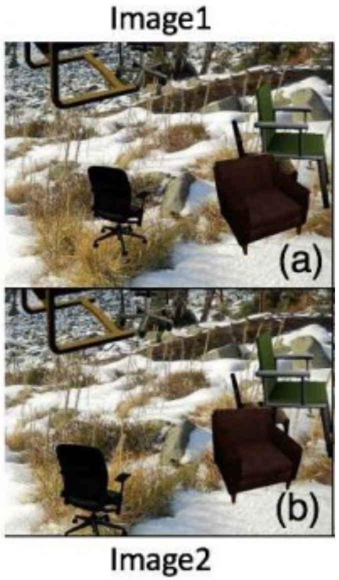
- Photometric Consistency (Appearance)

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

Unsupervised Flow

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

Image1

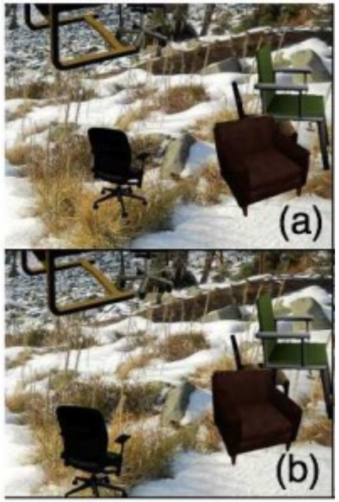
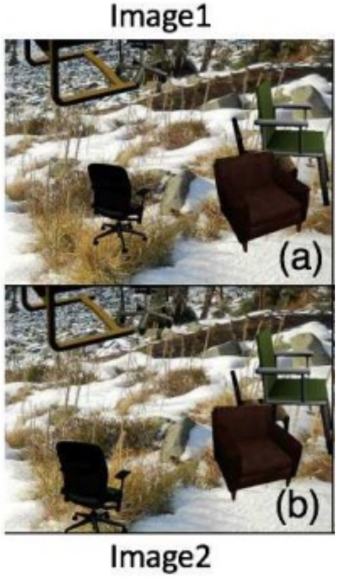


Image2

Wang et al., 2018

Unsupervised Flow

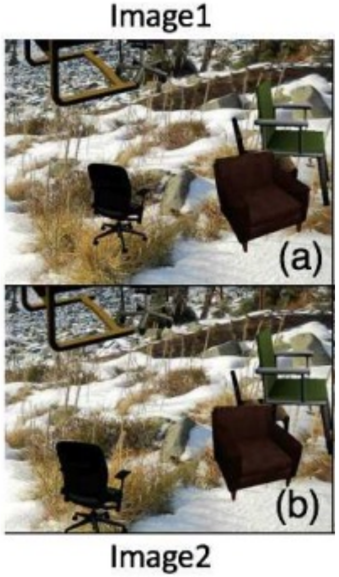
- Photometric Consistency (Appearance)
- Occlusion Estimation
  - Forward-backward consistency
- Smoothness
- Self-supervision: Ensure consistent flow at different augmentation (e.g. crops)



Wang et al., 2018

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

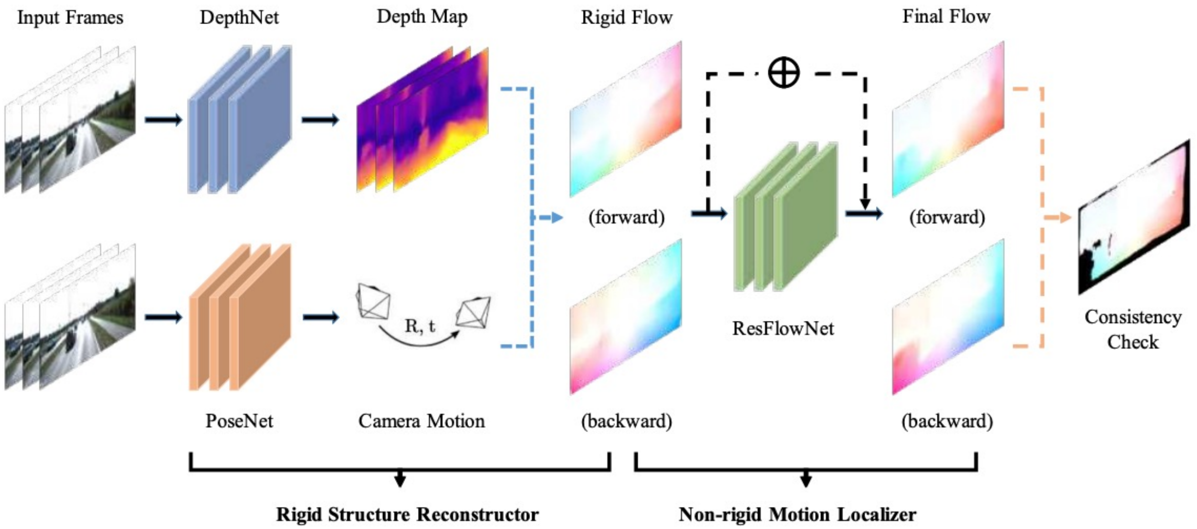
Jonschkowski et al. What Matters in Unsupervised Optical Flow. ECCV 2020

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.

Training Losses

- Appearance Loss (Warping):

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

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

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:

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

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:

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

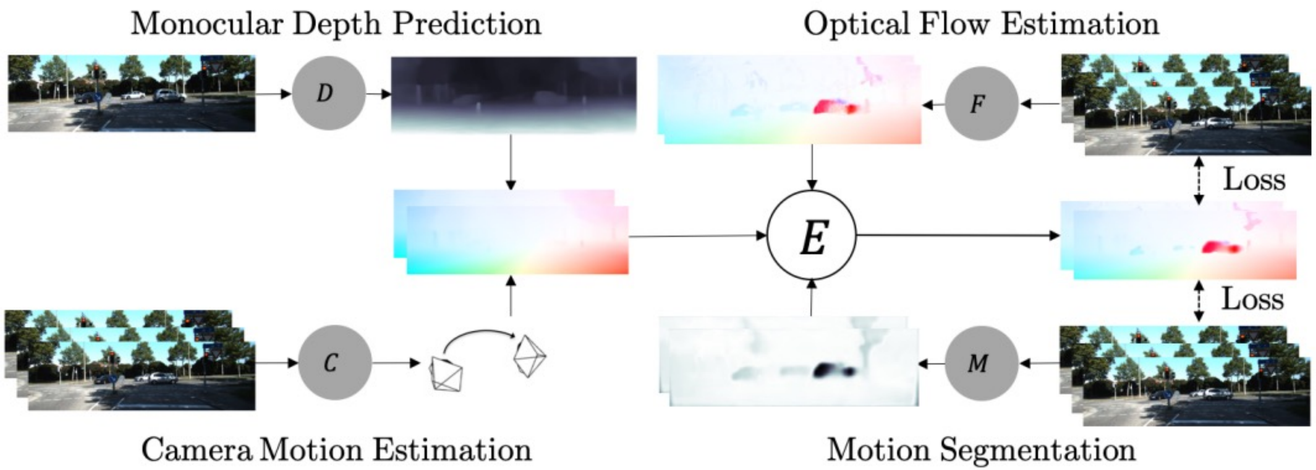
- Forward-Backward Consistency:

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

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

Summary

- Leverage cross correlation structure for spatial similarity matching.

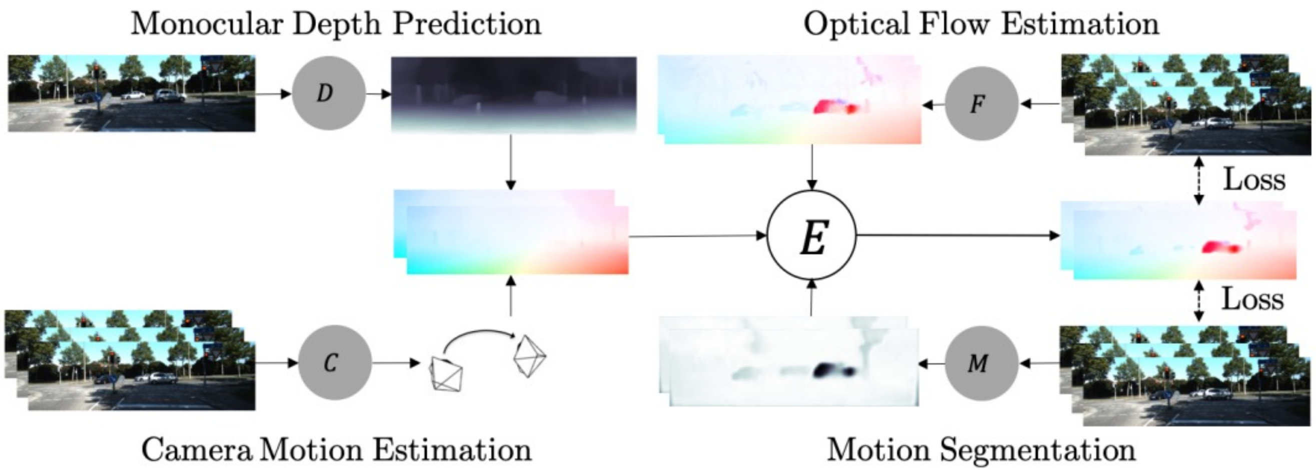


Ranjan et al. Competitive Collaboration: Joint Unsupervised Learning of Depth, Camera, Motion, Optical Flow and Motion Segmentation. CVPR 2019

NYU

Summary

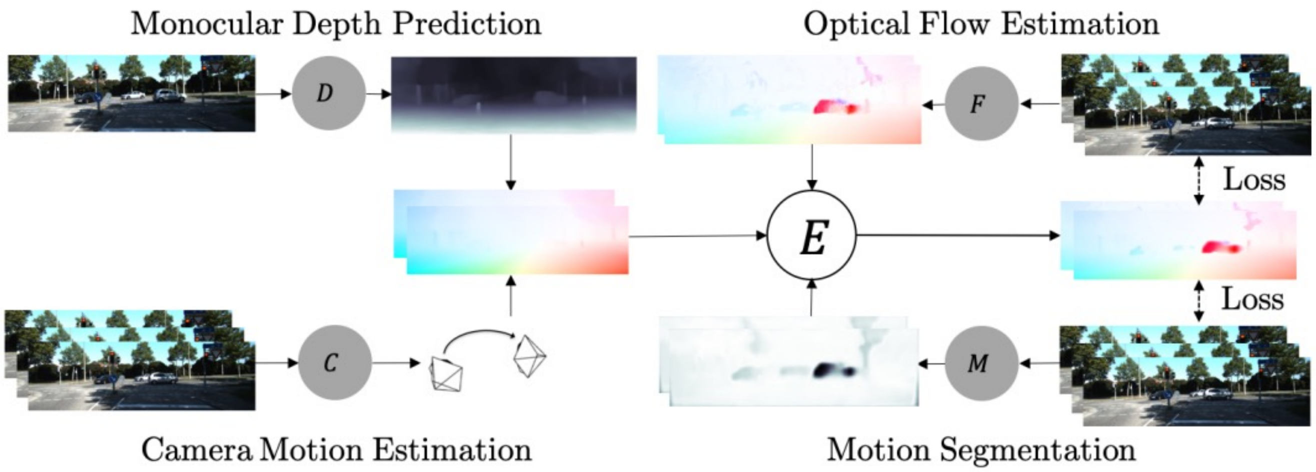
- Leverage cross correlation structure for spatial similarity matching.
- Can be used towards: depth, flow, and pose prediction.



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

Summary

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



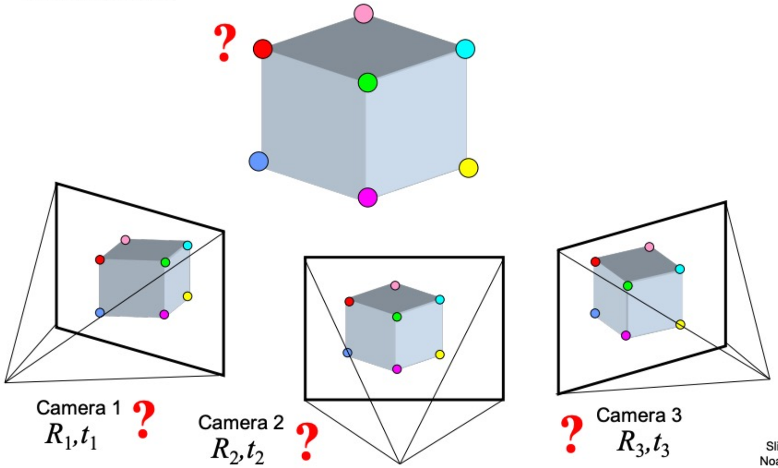
Ranjan et al. Competitive Collaboration: Joint Unsupervised Learning of Depth, Camera, Motion, Optical Flow and Motion Segmentation. CVPR 2019

NYU

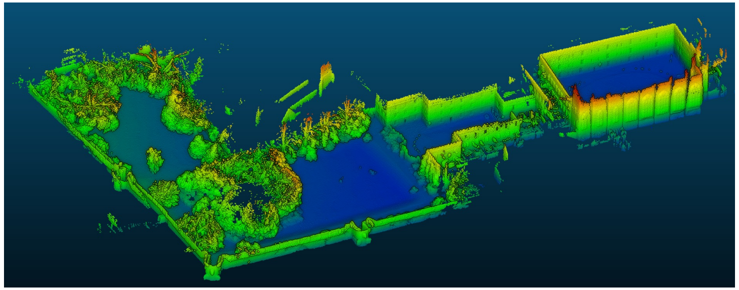
Classical Mapping

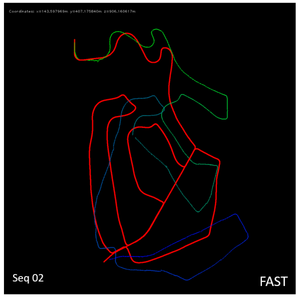
- Estimating 3D structure and location from 2D observations.
- Simultaneous Localization and Mapping.
- Common Techniques: Extended Kalman Filter, GraphSLAM

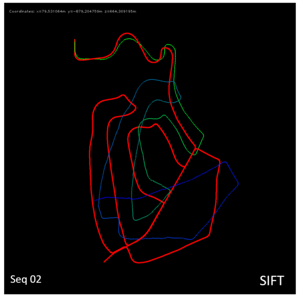
- Given a set of corresponding points in two or more images, compute the camera parameters and the 3D point coordinates



Slide credit:
Noah Snavely







Garg & Jain

Common Drawbacks

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

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.

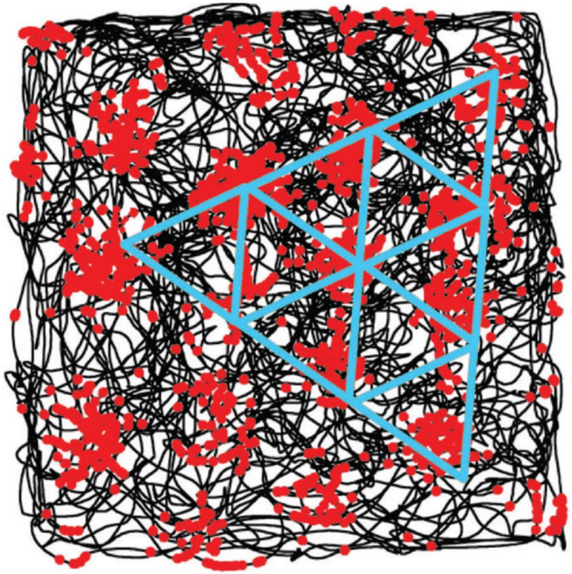
Common Drawbacks

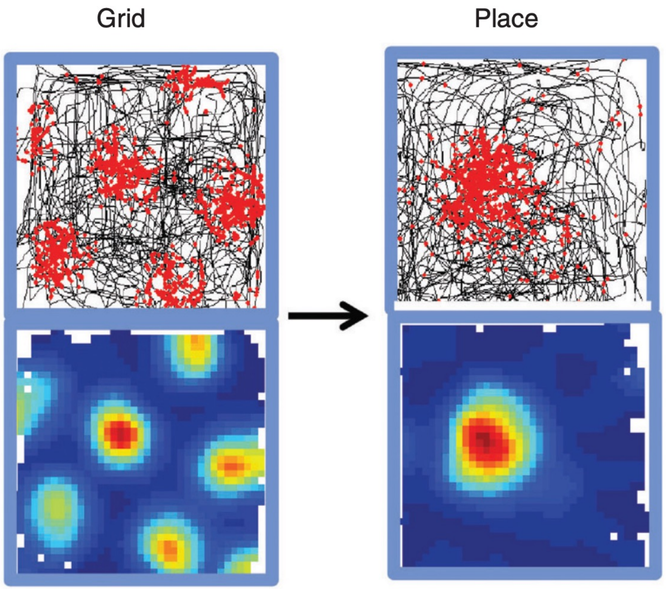
- Probabilistic inference can take long to compute, and mapping takes a large memory storage.
- Great for 3D reconstruction but downstream tasks may not need a full precision explicit map.
- May not fully understand dynamic objects (averaging across multiple scans).

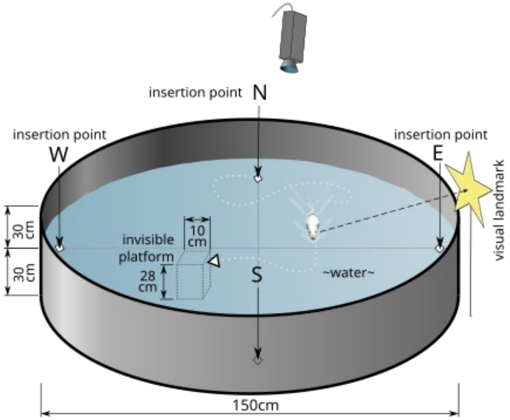
Common Drawbacks

- 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?

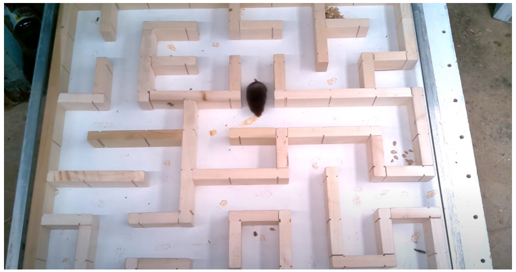
Mapping in the Brain: Grid and Place Cells







Morris water maze



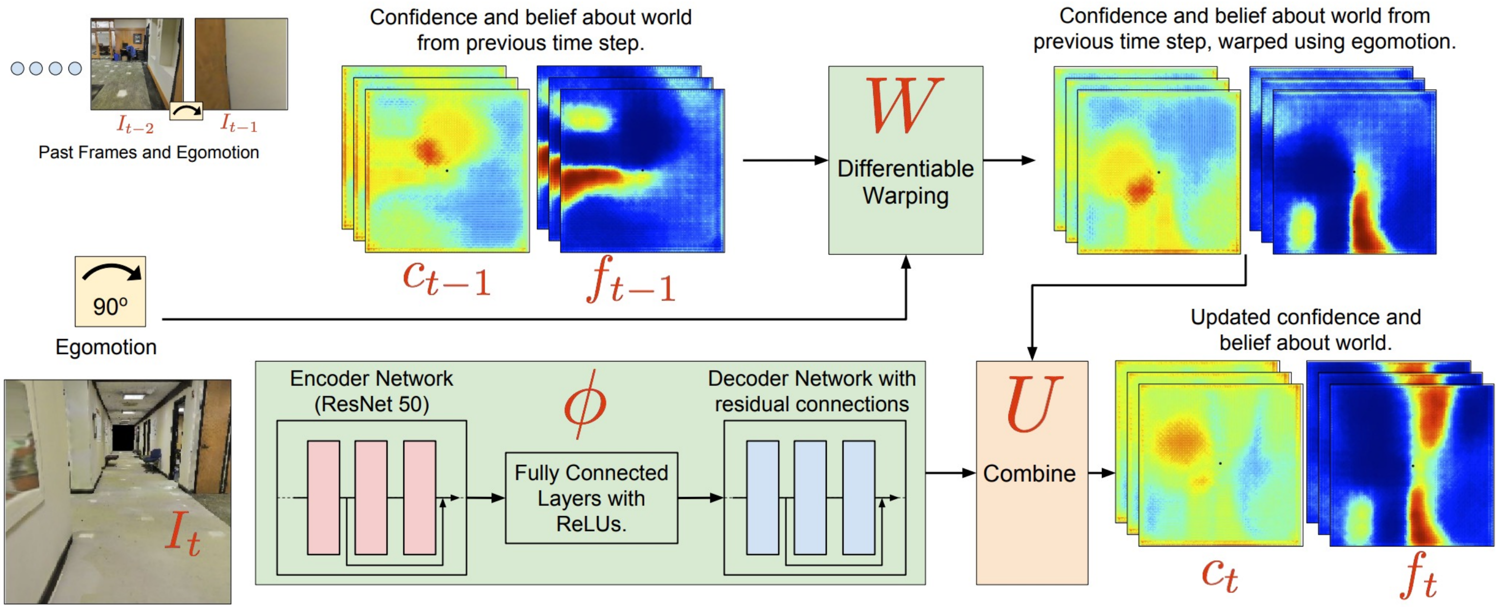
Matthias Wandel, 2018

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

NYU

Neural Mapping

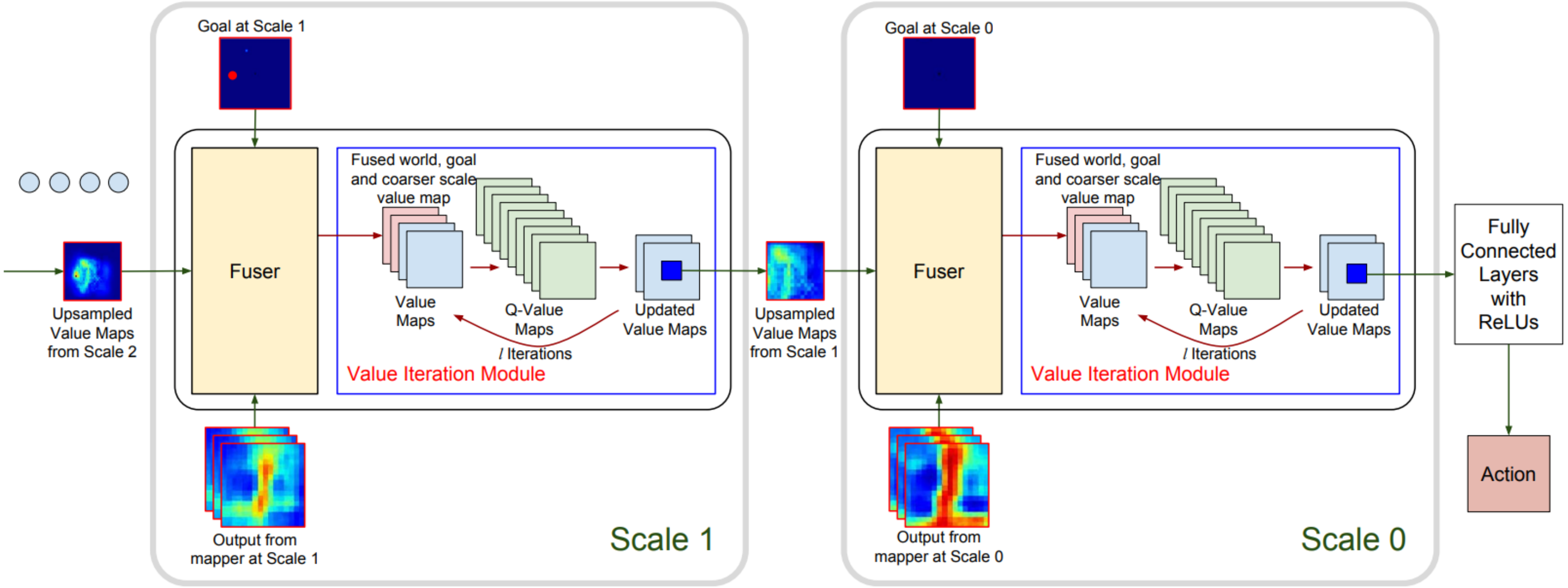
- Can we learn a mapping representation?
- Metric space, top-down warping (known egomotion).



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

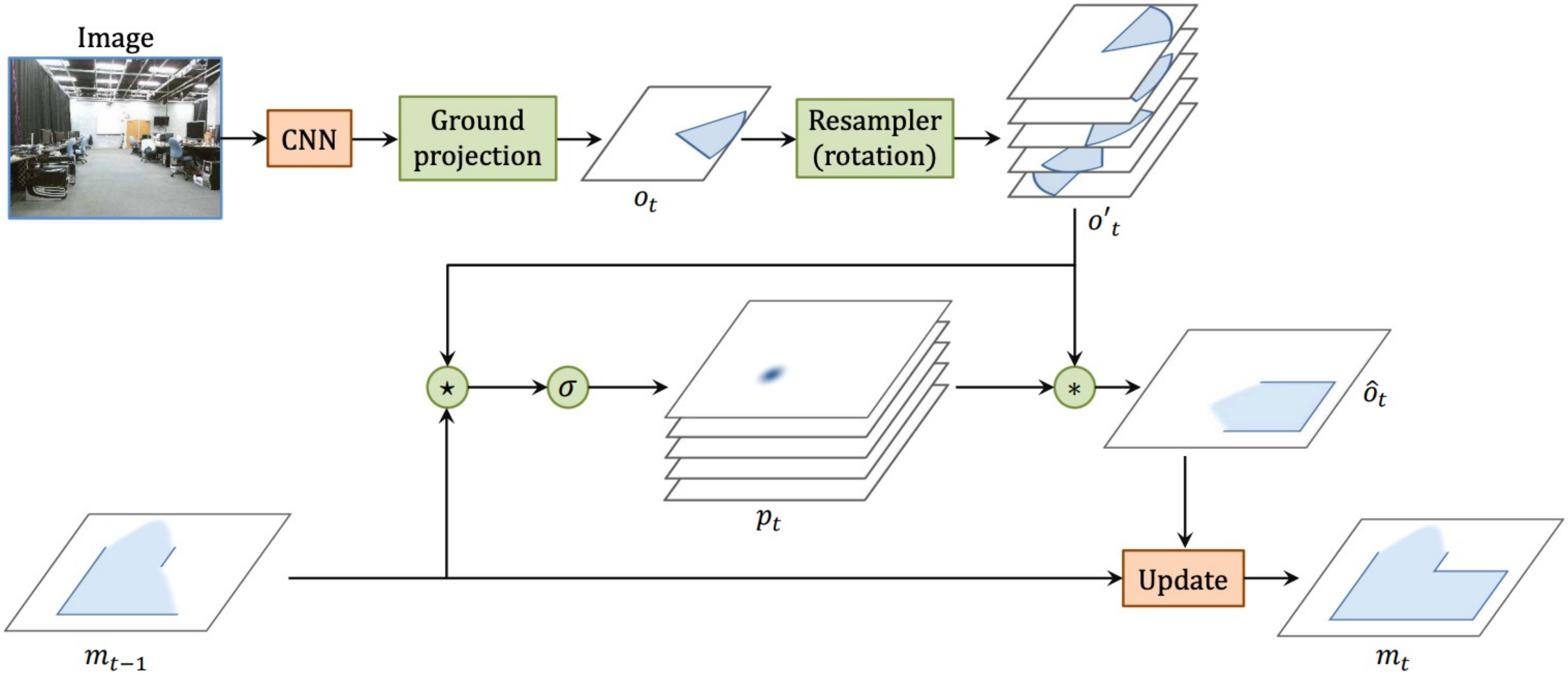
Hierarchical Planning

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



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

Simultaneous Localization and Registration

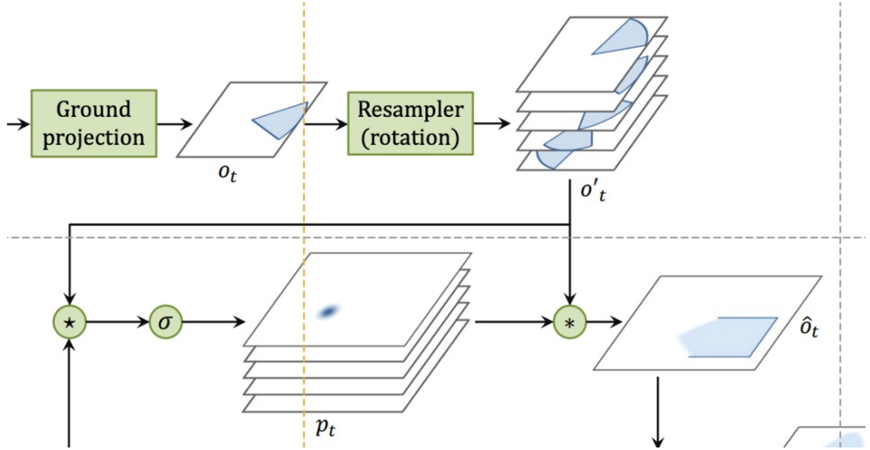


Henriques & Vedaldi. MapNet: An Allocentric Spatial Memory for Mapping Environments. CVPR 2018.

Simultaneous Localization and Registration

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

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



NYU

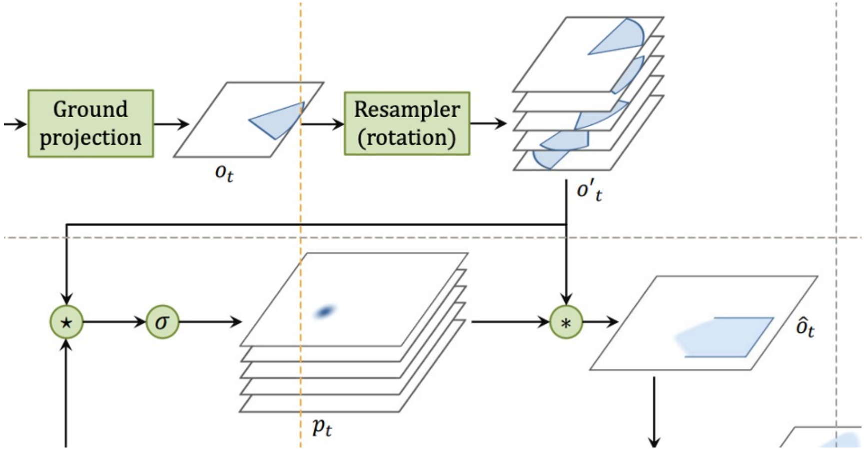
Simultaneous Localization and Registration

- 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.

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



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

Simultaneous Localization and Registration

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

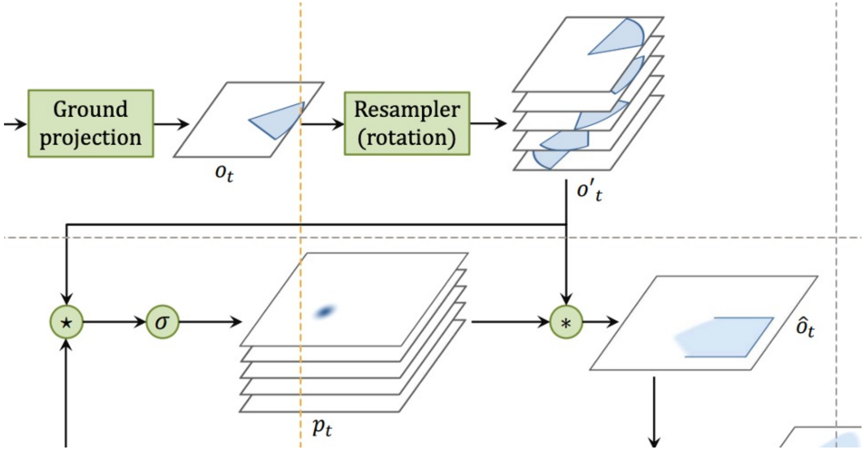
$$o'_{ijkl} = [R(o, 2\pi l/r)]_{ijk}.$$

- o'_t convolve with the base feature.

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

- Transform observations into allocentric

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



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

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.

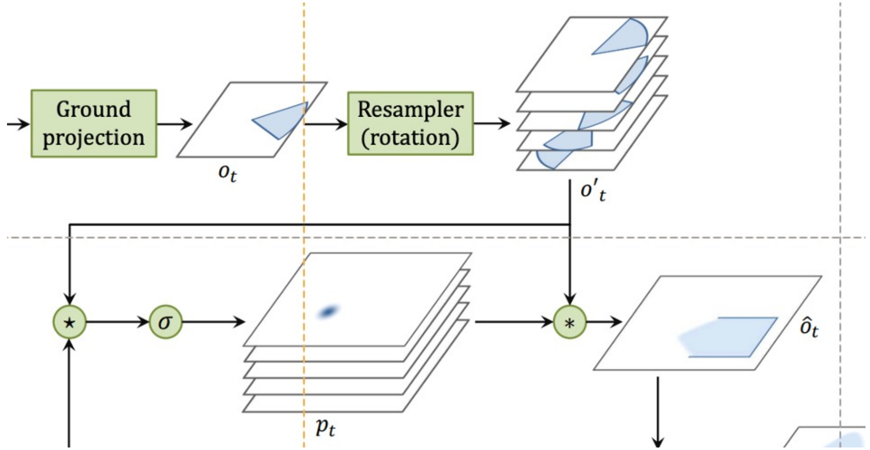
$$p_t = \mathrm{Softmax}(m_{t-1} * o'_t).$$

- Transform observations into allocentric

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

- Update belief:

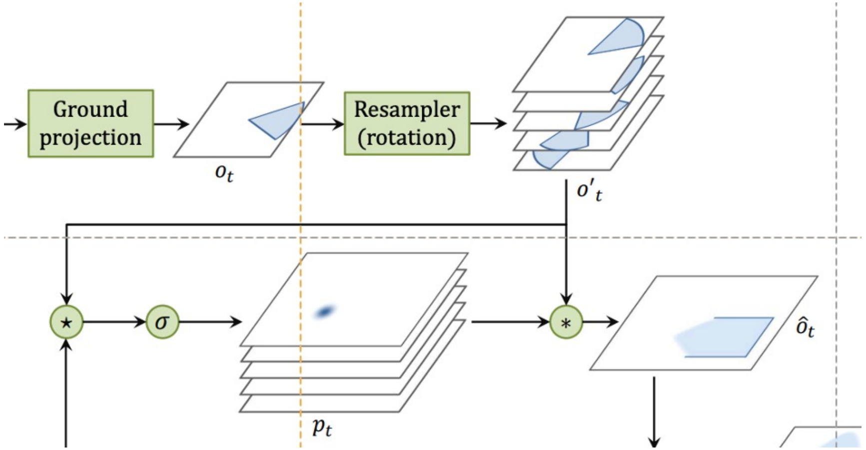
$$m_{i,j,t+1} = \mathrm{LSTM}(m_{i,j,t}, \hat{o}_{i,j,t}).$$



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

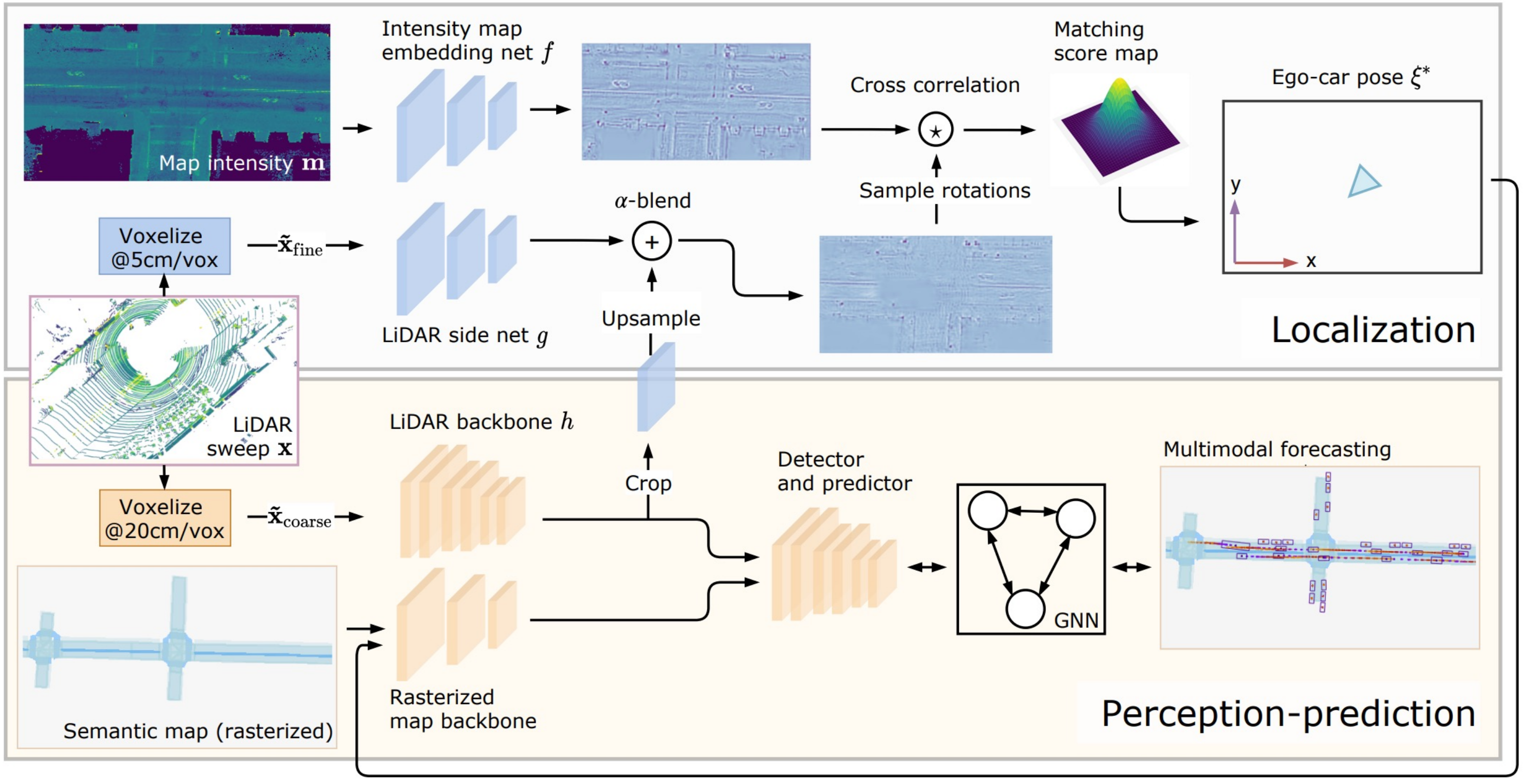


Loss:
$$\mathcal{L}(p) = -\log \sum_t p_{H_t W_t R_t t}.$$

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

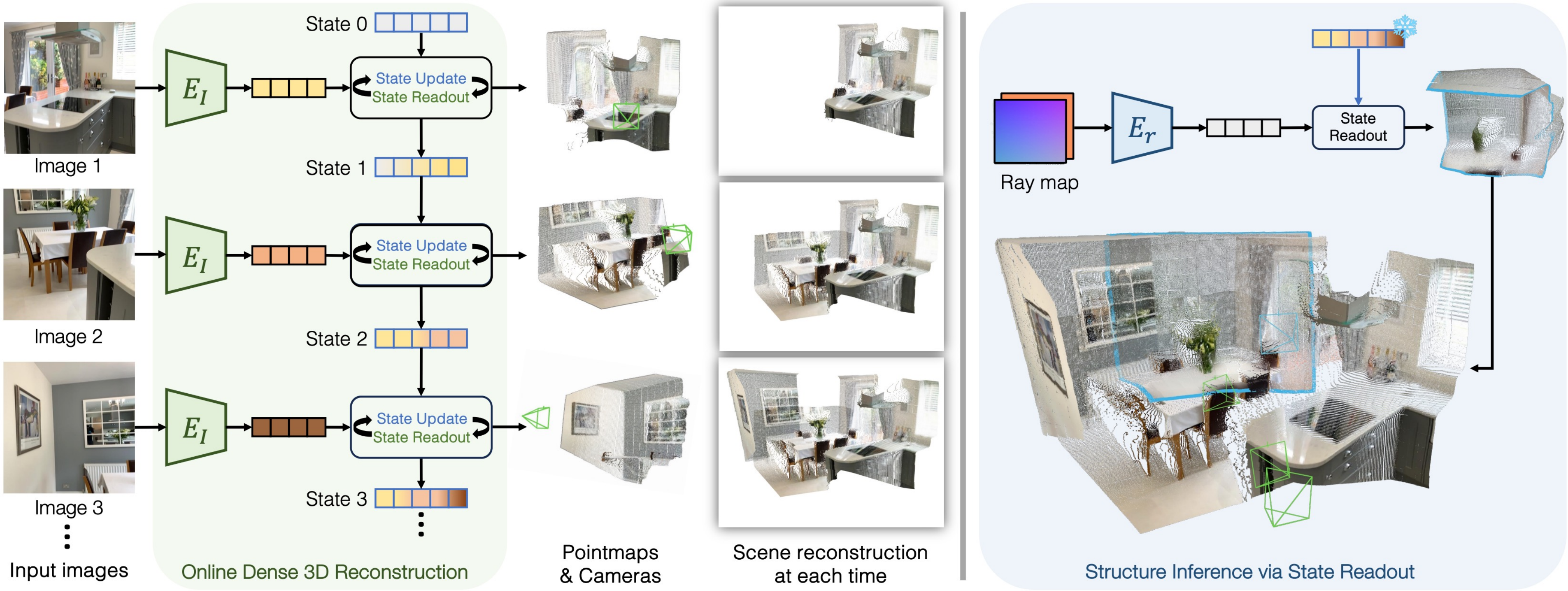
NYU

Joint Localization, Perception and Prediction



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

Continuous 3D Perception and Mapping

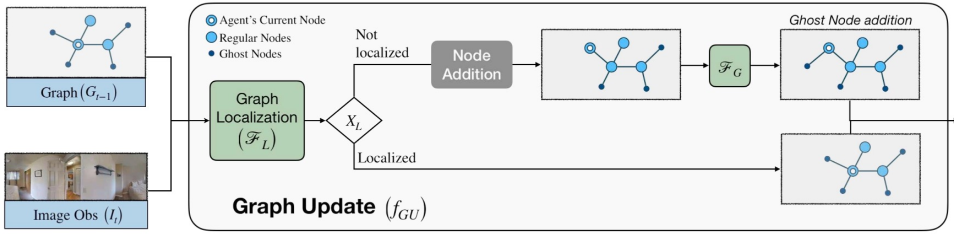


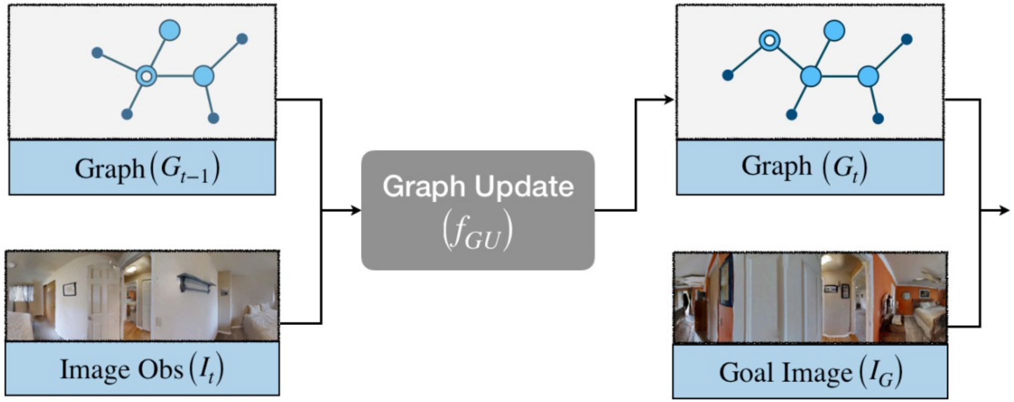
Wang et al. Continuous 3D Perception Model with Persistent State. arXiv 2025.

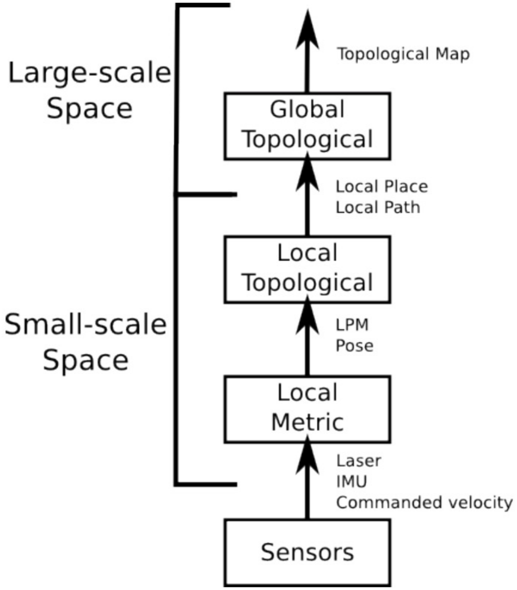
NYU

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.

Summary

Summary

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

Summary

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

Summary

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

Summary

- Covers 3D, 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.

Summary

- Covers 3D, 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.

Summary

- Covers 3D, 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.

Summary

- Covers 3D, 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).