

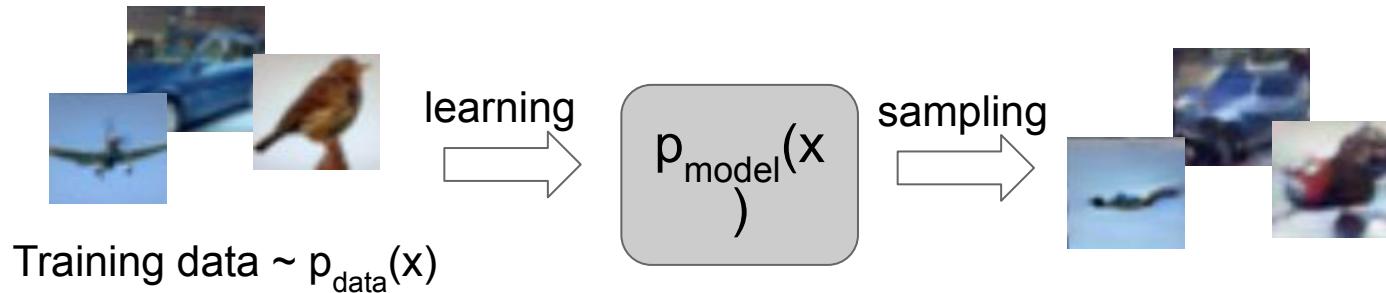
Lecture 14: Self-Supervised Learning

Administrative

- Assignment 3 due in two weeks 5/25
- Midterm grade is out
- Regrade request:
 - Gradescope regrade **only for mistakes according to the current rubric**
 - Teaching team will discuss concerns in MC & T/F next Monday

Last Lecture: Generative Modeling

Given training data, generate new samples from same distribution



Objectives:

1. Learn $p_{\text{model}}(x)$ that approximates $p_{\text{data}}(x)$
2. Sampling new x from $p_{\text{model}}(x)$

Last Lecture: Generative Modeling

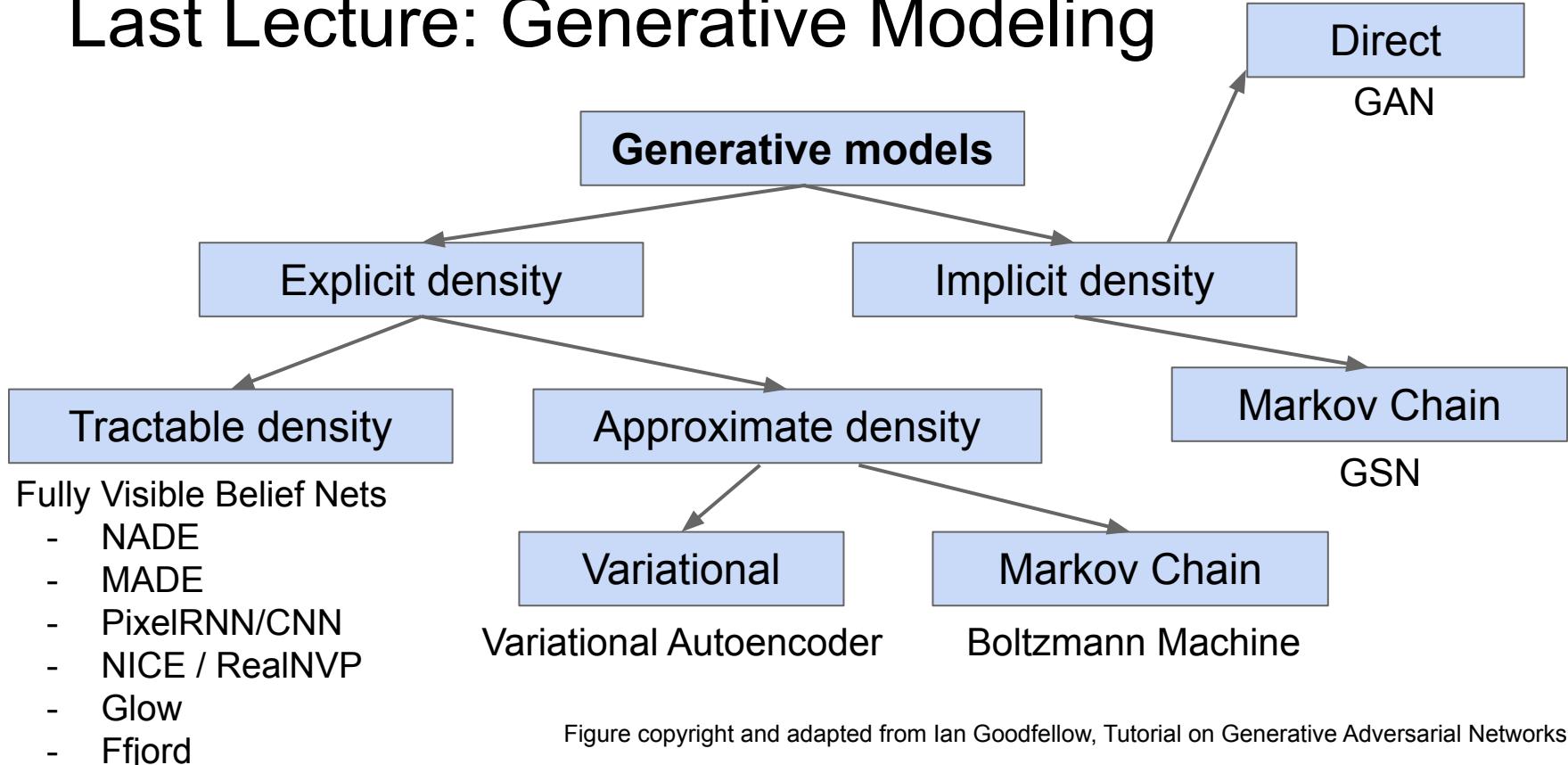


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Generative vs. Self-supervised Learning

- Both aim to learn from data without manual label annotation.
- Generative learning aims to model **data distribution** $p_{data}(x)$, e.g., generating realistic images.
- Self-supervised learning methods solve “pretext” tasks that produce **good features** for downstream tasks.
 - Learn with supervised learning objectives, e.g., classification, regression.
 - Labels of these pretext tasks are generated *automatically*

Self-supervised pretext tasks

Example: learn to predict image transformations / complete corrupted images

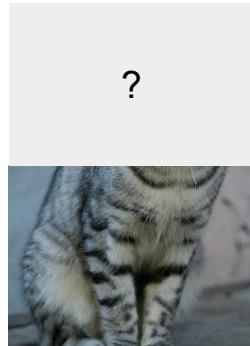
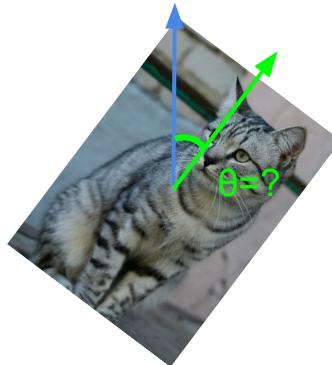


image completion



rotation prediction



"jigsaw puzzle"



colorization

1. Solving the pretext tasks allow the model to learn good features.
2. We can automatically generate labels for the pretext tasks.

Generative vs. Self-supervised Learning



Left: Drawing of a dollar bill from memory. Right: Drawing subsequently made with a dollar bill present. Image source: [Epstein, 2016](#)

Learning to generate pixel-level details is often unnecessary; learn high-level semantic features with pretext tasks instead

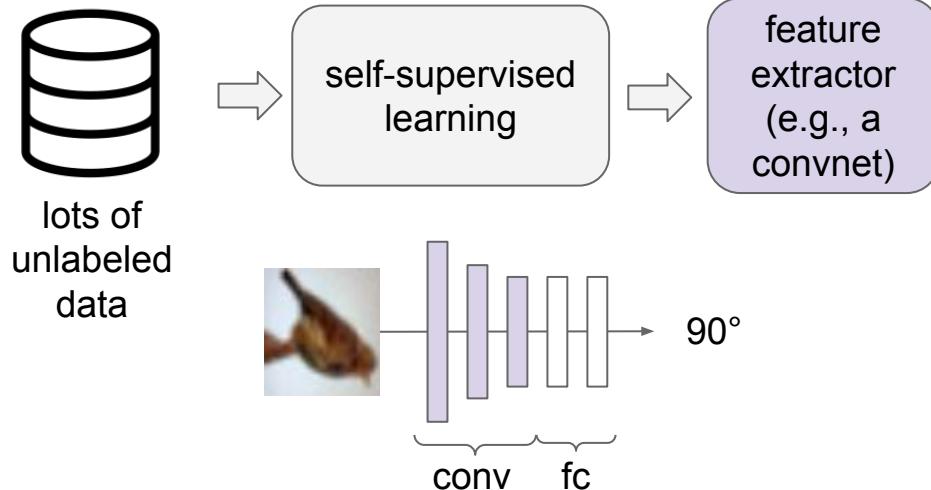
Source: [Anand, 2020](#)

How to evaluate a self-supervised learning method?

We usually don't care about the performance of the self-supervised learning task, e.g., we don't care if the model learns to predict image rotation perfectly.

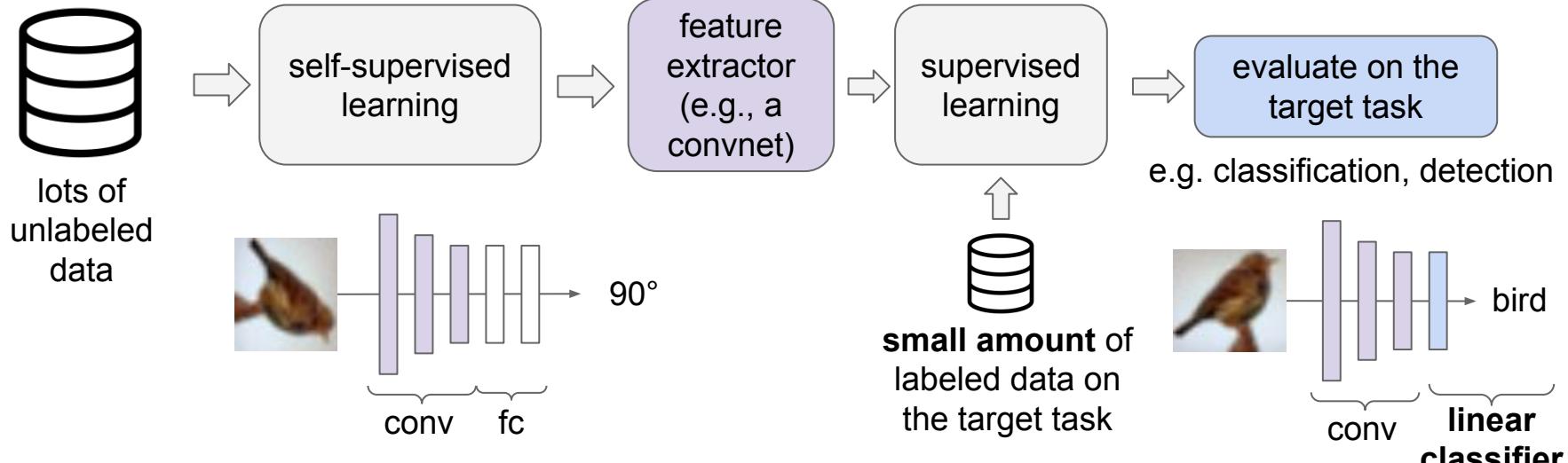
Evaluate the learned feature encoders on downstream *target tasks*

How to evaluate a self-supervised learning method?



1. Learn good feature extractors from self-supervised pretext tasks, e.g., predicting image rotations

How to evaluate a self-supervised learning method?



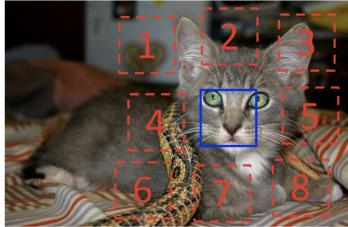
1. Learn good feature extractors from self-supervised pretext tasks, e.g., predicting image rotations

2. Attach a shallow network on the feature extractor; train the shallow network on the target task with small amount of labeled data

Broader picture

Today's lecture

computer vision



Doersch et al., 2015

robot / reinforcement learning



Dense Object Net (Florence and Manuelli et al., 2018)

language modeling

Language Models are Few-Shot Learners

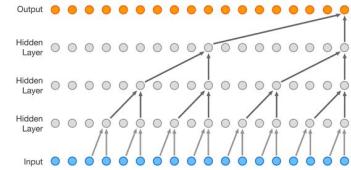
Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*
Jared Kaplan† Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry
Amanda Askell Sandhini Agarwal Ariel Herbert-Voss Gretchen Krueger Tom Henighan
Rewon Child Aditya Ramesh Daniel M. Ziegler Jeffrey Wu Clemens Winter
Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray
Benjamin Chess Jack Clark Christopher Berner
Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei
OpenAI

Abstract

Recent work has demonstrated substantial gains on many NLP tasks and benchmarks by pre-training on a large corpus of text followed by fine-tuning on a specific task. While typically task-agnostic in architecture, this method still requires task-specific fine-tuning datasets of thousands or tens of thousands of examples. By contrast, humans can generally perform a new language task from only a few examples. In this work, we show that scaling word embeddings to support few-shot learning is greatly straight-forward. Here we show that scaling a language model greatly improves task-agnostic few-shot performance, sometimes even reaching competitiveness with prior state-of-the-art fine-tuning approaches. Specifically, we train GPT-3, an autoregressive language model with 175 billion parameters, 10x larger than any previous non-sparse language model, and test its performance in the few-shot setting. For all tasks, GPT-3 is trained without gradient descent or fine-tuning, with tasks and few-shot demonstrations specified purely via text interaction with the model. GPT-3 achieves strong performance on many NLP datasets, including translation, question-answering, and cloze tasks, as well as several tasks that require on-the-fly reasoning or domain adaptation, such as understanding words, using a novel word in a sentence, or performing 3-digit arithmetic. At the same time, we also identify datasets where GPT-3 faces challenges in learning simple analogies, as well as some datasets where GPT-3 faces methodological issues related to training on large web corpora. Finally, we find that GPT-3 can generate samples of news articles which human evaluators have difficulty distinguishing from articles written by humans. We discuss broader societal impacts of this finding and of GPT-3 in general.

GPT3 (Brown, Mann, Ryder, Subbiah et al., 2020)

speech synthesis



Wavenet (van den Oord et al., 2016)

Today's Agenda

Pretext tasks from image transformations

- Rotation, inpainting, rearrangement, coloring

Contrastive representation learning

- Intuition and formulation
- Instance contrastive learning: SimCLR and MOCO
- Sequence contrastive learning: CPC

Today's Agenda

Pretext tasks from image transformations

- Rotation, inpainting, rearrangement, coloring

Contrastive representation learning

- Intuition and formulation
- Instance contrastive learning: SimCLR and MOCO
- Sequence contrastive learning: CPC

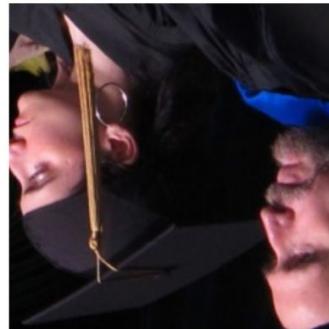
Pretext task: predict rotations



90° rotation



270° rotation



180° rotation



0° rotation

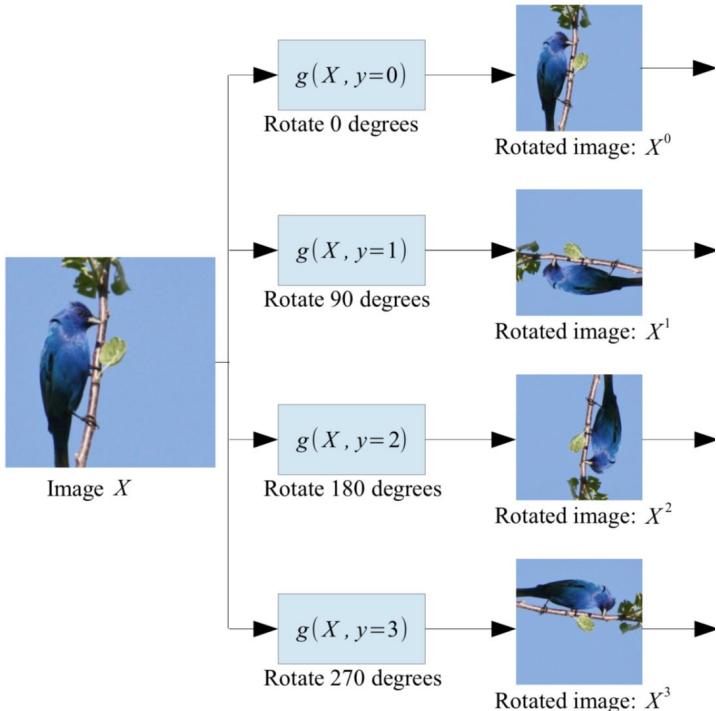


270° rotation

Hypothesis: a model could recognize the correct rotation of an object only if it has the “visual commonsense” of what the object should look like unperturbed.

(Image source: [Gidaris et al. 2018](#))

Pretext task: predict rotations

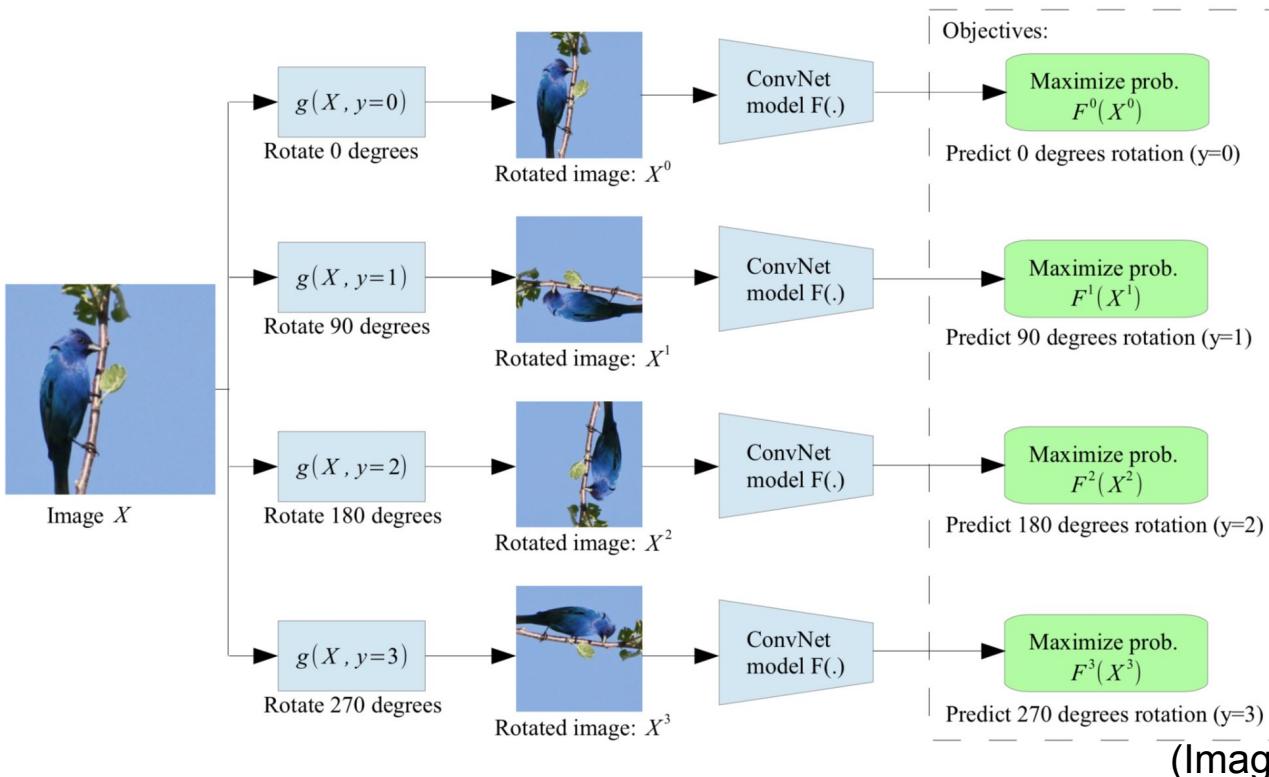


Self-supervised learning by rotating the entire input images.

The model learns to predict which rotation is applied (4-way classification)

(Image source: [Gidaris et al. 2018](#))

Pretext task: predict rotations

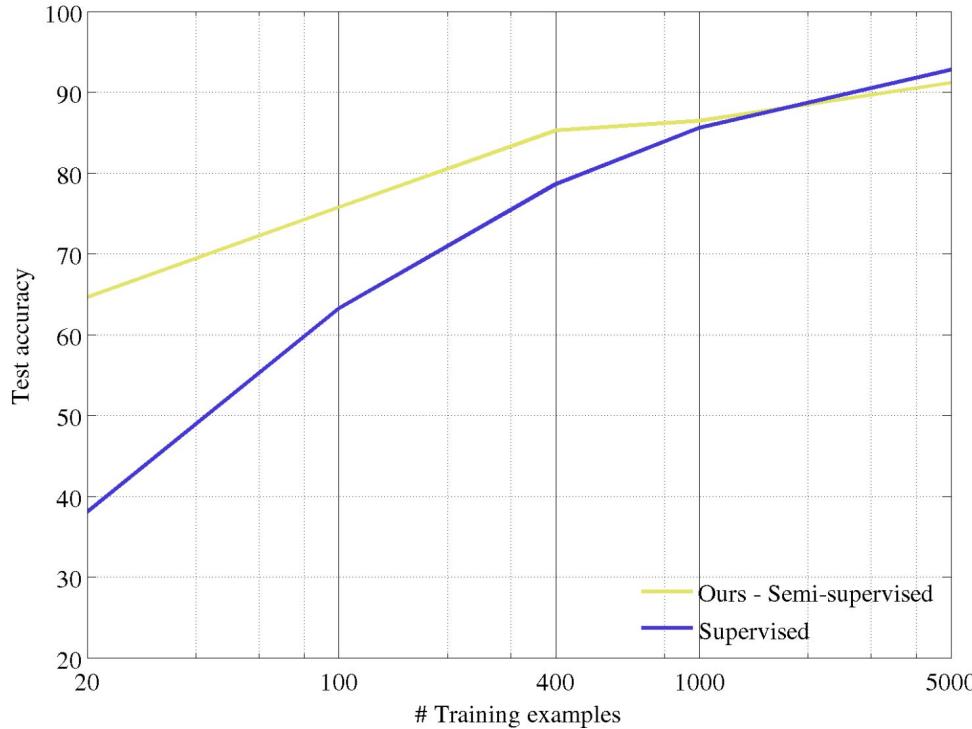


Self-supervised learning by rotating the entire input images.

The model learns to predict which rotation is applied (4-way classification)

(Image source: [Gidaris et al. 2018](#))

Evaluation on semi-supervised learning



Self-supervised learning on
CIFAR10 (entire training set).

Freeze conv1 + conv2
Learn **conv3 + linear** layers
with subset of labeled
CIFAR10 data (classification).

(Image source: [Gidaris et al. 2018](#))

Transfer learned features to supervised learning

	Classification (%mAP)	Detection (%mAP)	Segmentation (%mIoU)
Trained layers	fc6-8	all	all
ImageNet labels	78.9	79.9	56.8
Random		53.3	43.4
Random rescaled Krähenbühl et al. (2015)	39.2	56.6	45.6
Egomotion (Agrawal et al., 2015)	31.0	54.2	43.9
Context Encoders (Pathak et al., 2016b)	34.6	56.5	44.5
Tracking (Wang & Gupta, 2015)	55.6	63.1	47.4
Context (Doersch et al., 2015)	55.1	65.3	51.1
Colorization (Zhang et al., 2016a)	61.5	65.6	46.9
BIGAN (Donahue et al., 2016)	52.3	60.1	46.9
Jigsaw Puzzles (Noroozi & Favaro, 2016)	-	67.6	53.2
NAT (Bojanowski & Joulin, 2017)	56.7	65.3	49.4
Split-Brain (Zhang et al., 2016b)	63.0	67.1	46.7
ColorProxy (Larsson et al., 2017)		65.9	
Counting (Noroozi et al., 2017)	-	67.7	51.4
(Ours) RotNet	70.87	72.97	54.4
			39.1

Self-supervised learning with rotation prediction

Pretrained with full
ImageNet supervision

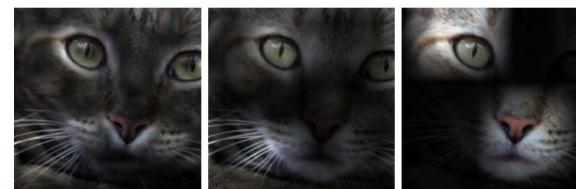
No pretraining

Self-supervised learning on
ImageNet (entire training
set) with AlexNet.

Finetune on labeled data
from **Pascal VOC 2007**.

source: [Gidaris et al. 2018](#)

Visualize learned visual attentions



Conv1 27×27 Conv3 13×13 Conv5 6×6

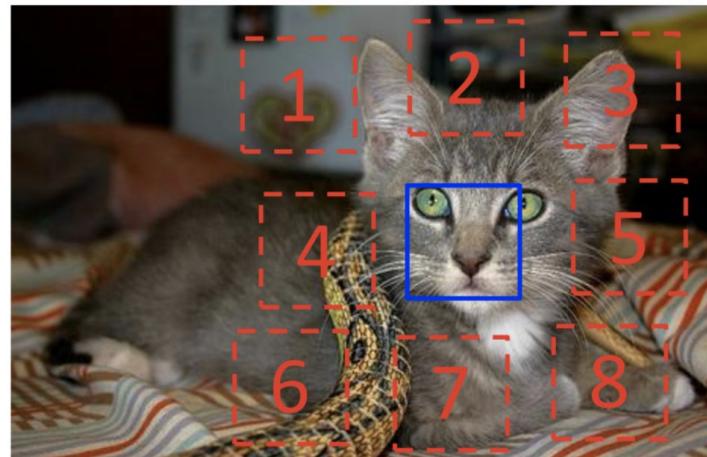
(a) Attention maps of supervised model

Conv1 27×27 Conv3 13×13 Conv5 6×6

(b) Attention maps of our self-supervised model

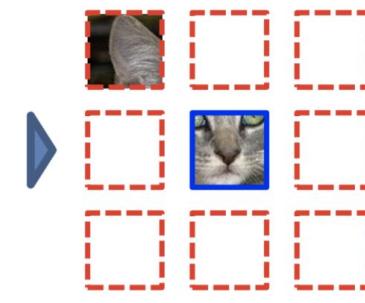
(Image source: [Gidaris et al. 2018](#))

Pretext task: predict relative patch locations



$$X = (\text{[cat eye]}, \text{[cat ear]}); Y = 3$$

Example:



Question 1:

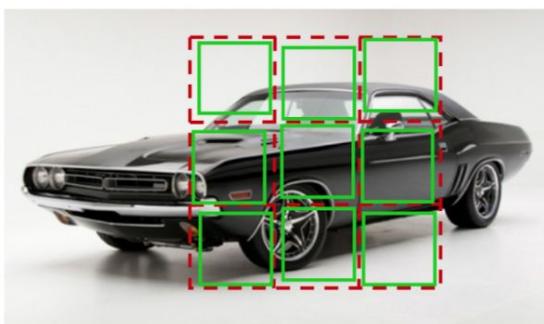


Question 2:



(Image source: [Doersch et al., 2015](#))

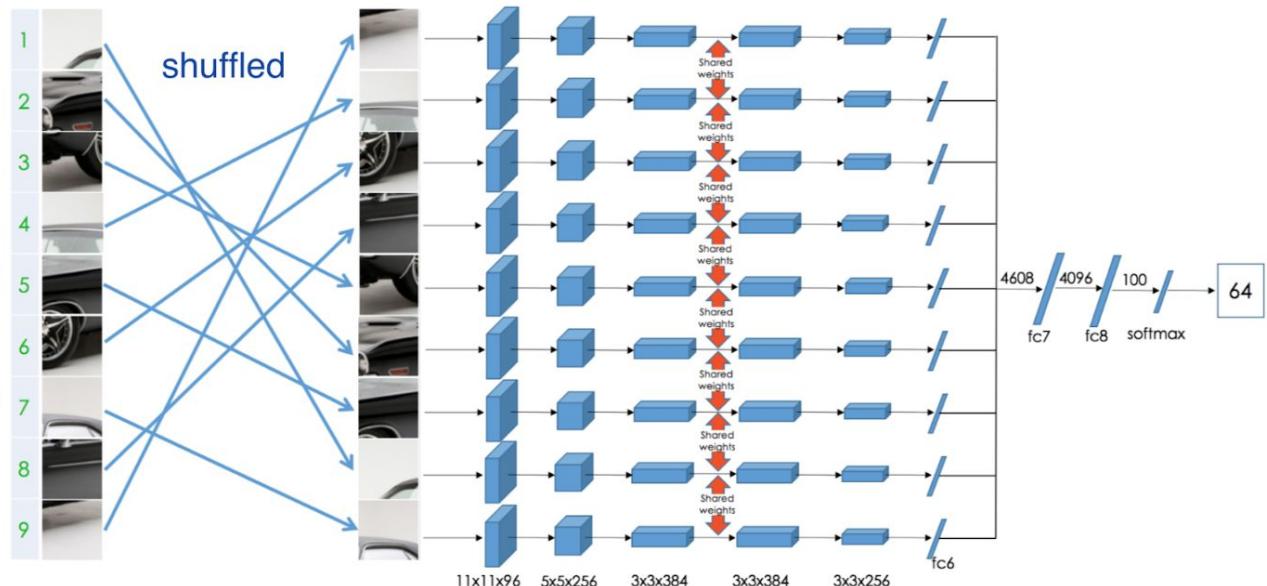
Pretext task: solving “jigsaw puzzles”



Permutation Set

index	permutation
64	9,4,6,8,3,2,5,1,7

Reorder patches according to
the selected permutation



(Image source: [Noroozi & Favaro, 2016](#))

Transfer learned features to supervised learning

Table 1: Results on PASCAL VOC 2007 Detection and Classification. The results of the other methods are taken from Pathak *et al.* [30].

Method	Pretraining time	Supervision	Classification	Detection	Segmentation
Krizhevsky <i>et al.</i> [25]	3 days	1000 class labels	78.2%	56.8%	48.0%
Wang and Gupta[39]	1 week	motion	58.4%	44.0%	-
Doersch <i>et al.</i> [10]	4 weeks	context	55.3%	46.6%	-
Pathak <i>et al.</i> [30]	14 hours	context	56.5%	44.5%	29.7%
Ours	2.5 days	context	67.6%	53.2%	37.6%

“Ours” is feature learned from solving image Jigsaw puzzles (Noroozi & Favaro, 2016). Doersch et al. is the method with relative patch location

(source: [Noroozi & Favaro, 2016](#))

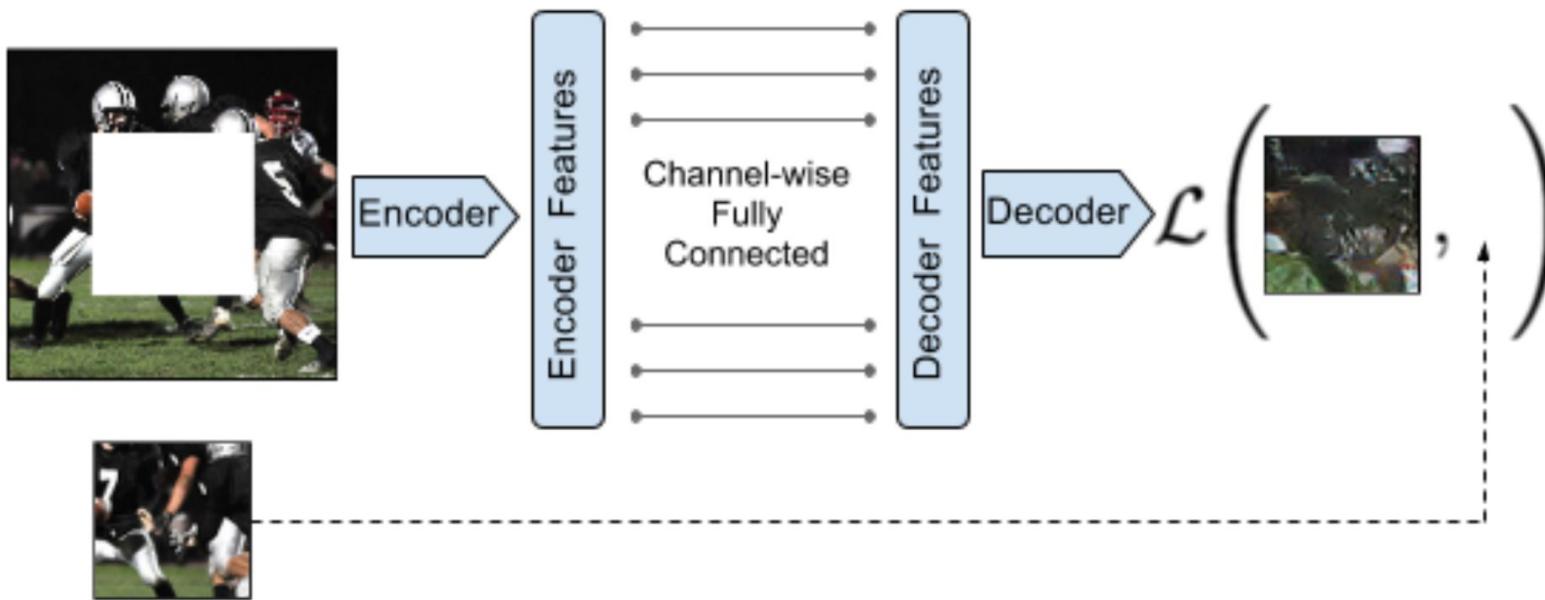
Pretext task: predict missing pixels (inpainting)



Context Encoders: Feature Learning by Inpainting (Pathak et al., 2016)

Source: [Pathak et al., 2016](#)

Learning to inpaint by reconstruction



Learning to reconstruct the missing pixels

Source: [Pathak et al., 2016](#)

Inpainting evaluation



Input (context)



reconstruction

Source: [Pathak et al., 2016](#)

Learning to inpaint by reconstruction

Loss = reconstruction + adversarial learning

$$L(x) = L_{recon}(x) + L_{adv}(x)$$

$$L_{recon}(x) = \left\| M * (x - F_\theta((1 - M) * x)) \right\|_2^2$$

$$L_{adv} = \max_D \mathbb{E}[\log(D(x))] + \log(1 - D(F((1 - M) * x)))]$$

Adversarial loss between “real” images and *inpainted images*

Source: [Pathak et al., 2016](#)

Inpainting evaluation



Input (context)

reconstruction

adversarial

recon + adv

Source: [Pathak et al., 2016](#)

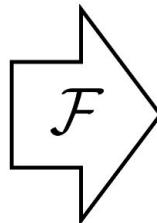
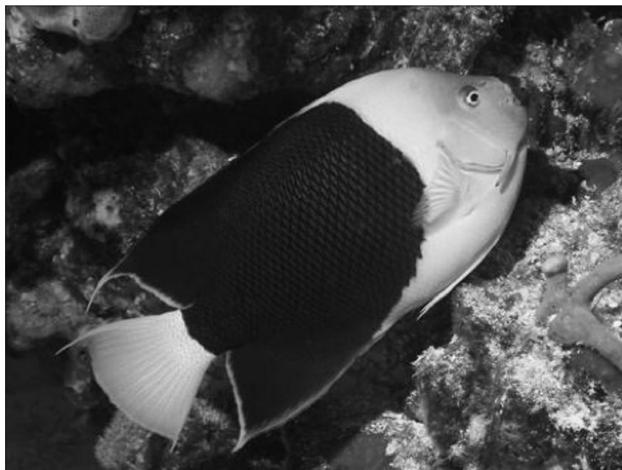
Transfer learned features to supervised learning

Pretraining Method	Supervision	Pretraining time	Classification	Detection	Segmentation
ImageNet [26]	1000 class labels	3 days	78.2%	56.8%	48.0%
Random Gaussian	initialization	< 1 minute	53.3%	43.4%	19.8%
Autoencoder	-	14 hours	53.8%	41.9%	25.2%
Agrawal <i>et al.</i> [1]	egomotion	10 hours	52.9%	41.8%	-
Wang <i>et al.</i> [39]	motion	1 week	58.7%	47.4%	-
Doersch <i>et al.</i> [7]	relative context	4 weeks	55.3%	46.6%	-
Ours	context	14 hours	56.5%	44.5%	30.0%

Self-supervised learning on ImageNet training set, transfer to classification (Pascal VOC 2007), detection (Pascal VOC 2007), and semantic segmentation (Pascal VOC 2012)

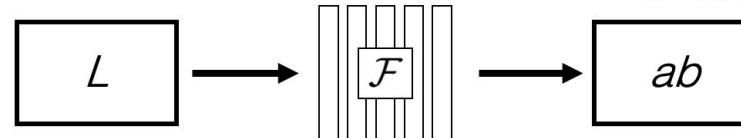
Source: [Pathak et al., 2016](#)

Pretext task: image coloring



Grayscale image: L channel

$$\mathbf{X} \in \mathbb{R}^{H \times W \times 1}$$

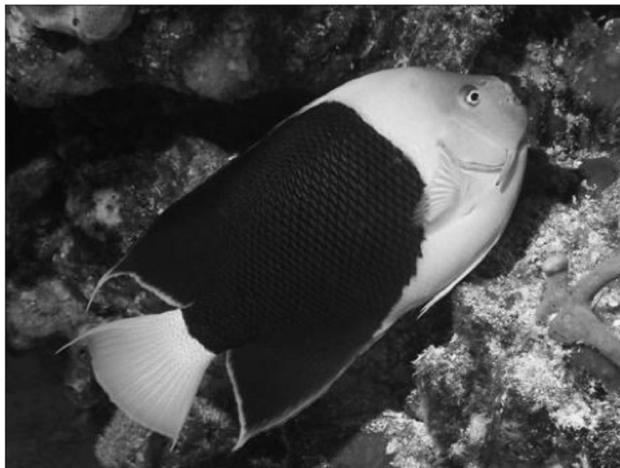


Color information: ab channels

$$\hat{\mathbf{Y}} \in \mathbb{R}^{H \times W \times 2}$$

Source: Richard Zhang / Phillip Isola ⁵

Pretext task: image coloring



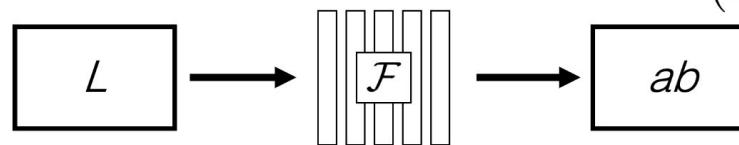
Grayscale image: L channel

$$\mathbf{X} \in \mathbb{R}^{H \times W \times 1}$$



Concatenate (L, ab) channels

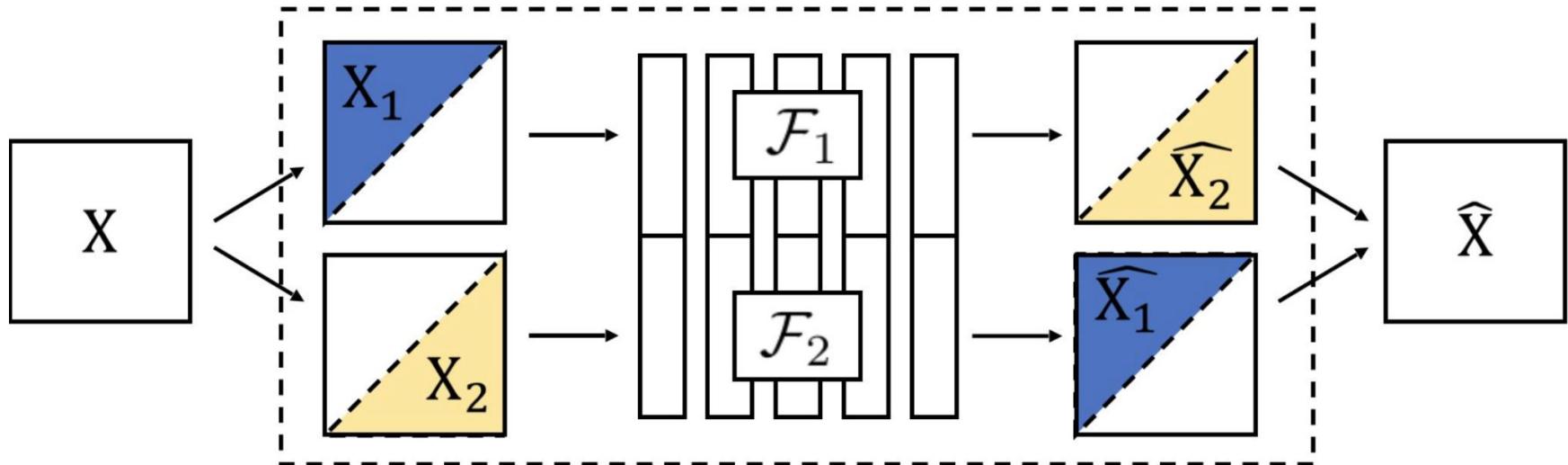
$$(\mathbf{X}, \hat{\mathbf{Y}})$$



Source: Richard Zhang / Phillip Isola

Learning features from colorization: Split-brain Autoencoder

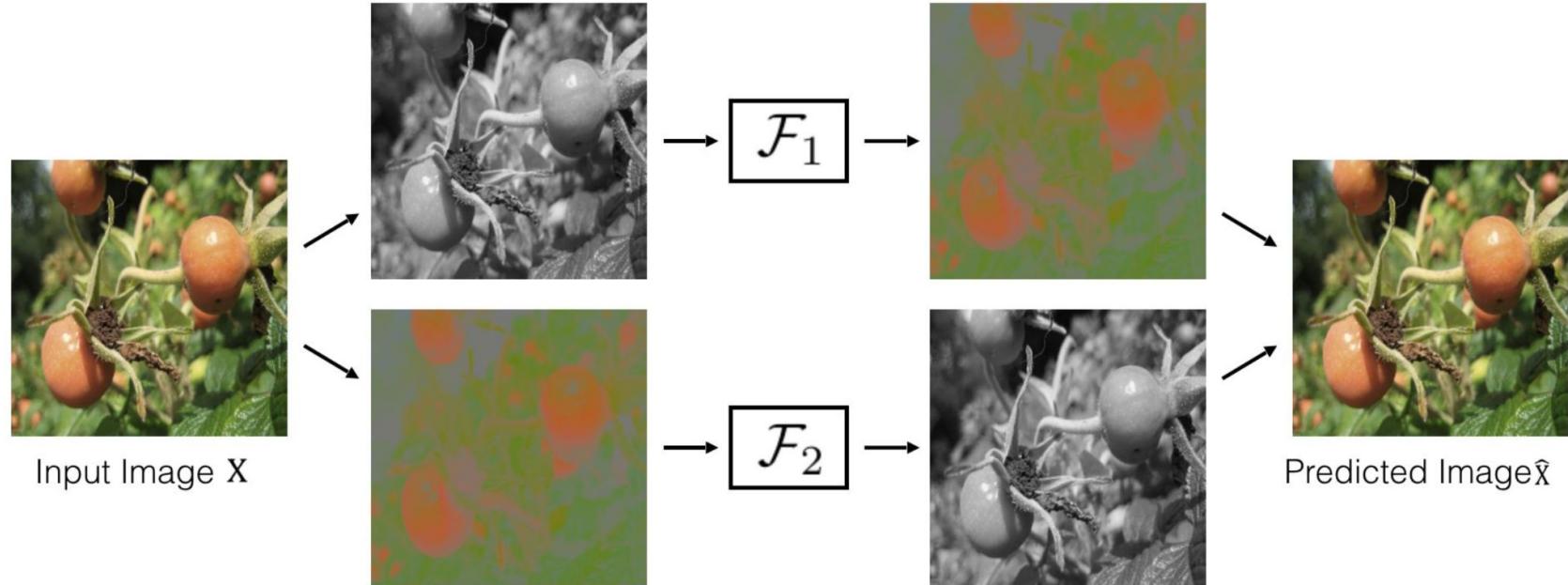
Idea: cross-channel predictions



Split-Brain Autoencoder

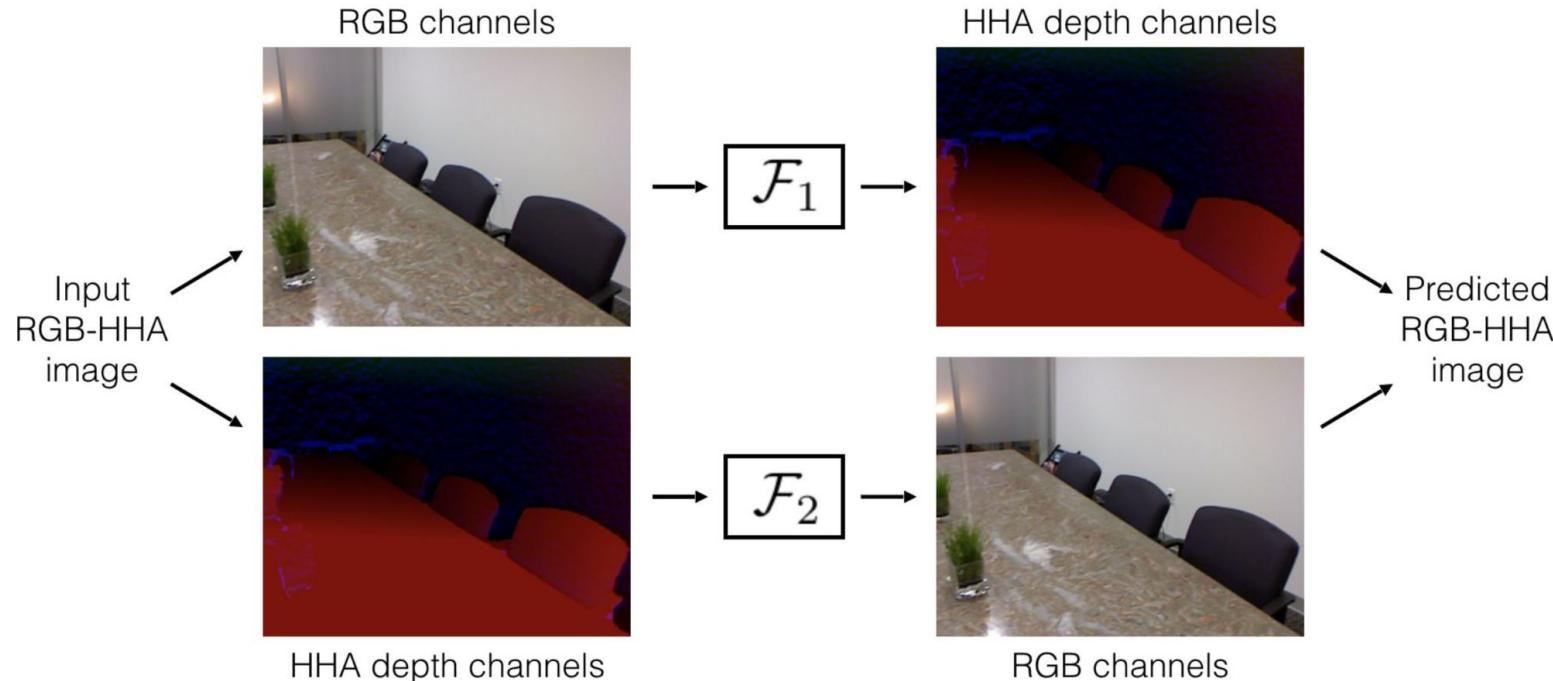
Source: Richard Zhang / Phillip Isola

Learning features from colorization: Split-brain Autoencoder



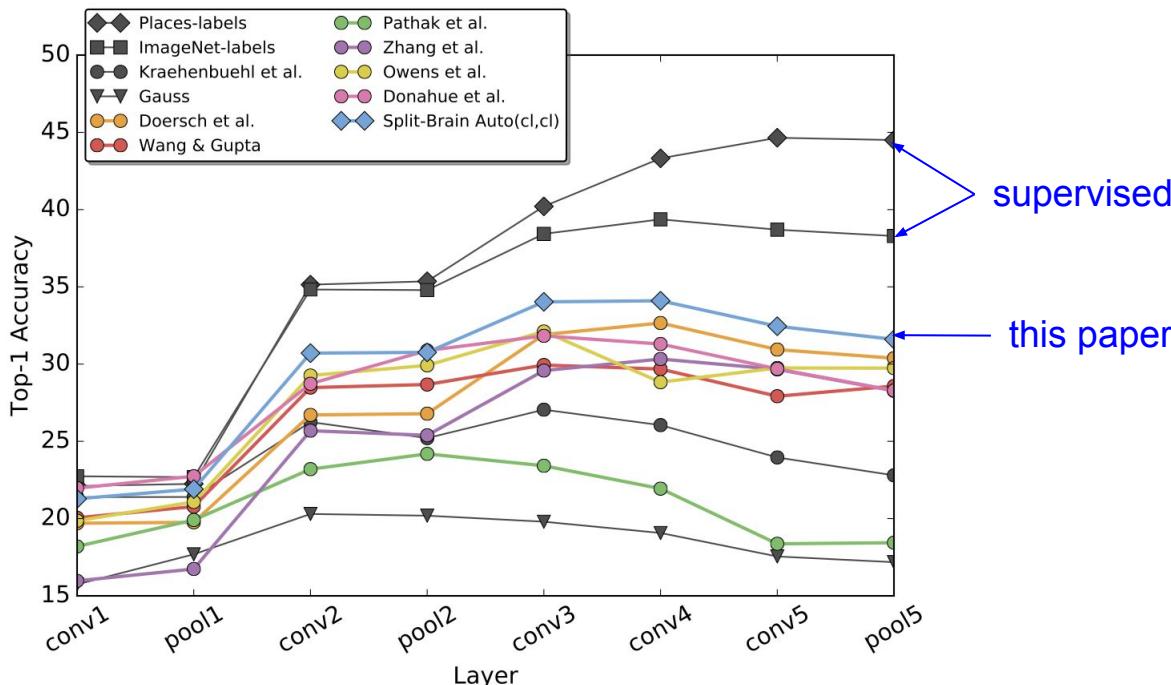
Source: Richard Zhang / Phillip Isola

Learning features from colorization: Split-brain Autoencoder



Source: Richard Zhang / Phillip Isola

Transfer learned features to supervised learning



Self-supervised learning on **ImageNet** (entire training set).

Use concatenated features from F_1 and F_2

Labeled data is from the **Places** (Zhou 2016).

Source: [Zhang et al., 2017](#)

Pretext task: image coloring



Source: Richard Zhang / Phillip Isola

Pretext task: image coloring



Source: Richard Zhang / Phillip Isola

Pretext task: video coloring

Idea: model the *temporal coherence* of colors in videos

reference frame



$t = 0$

how should I color these frames?



$t = 1$



$t = 2$



$t = 3$

...

Source: [Vondrick et al., 2018](#)

Pretext task: video coloring

Idea: model the *temporal coherence* of colors in videos

reference frame



$t = 0$

how should I color these frames?

Should be the same color!



$t = 1$



$t = 2$



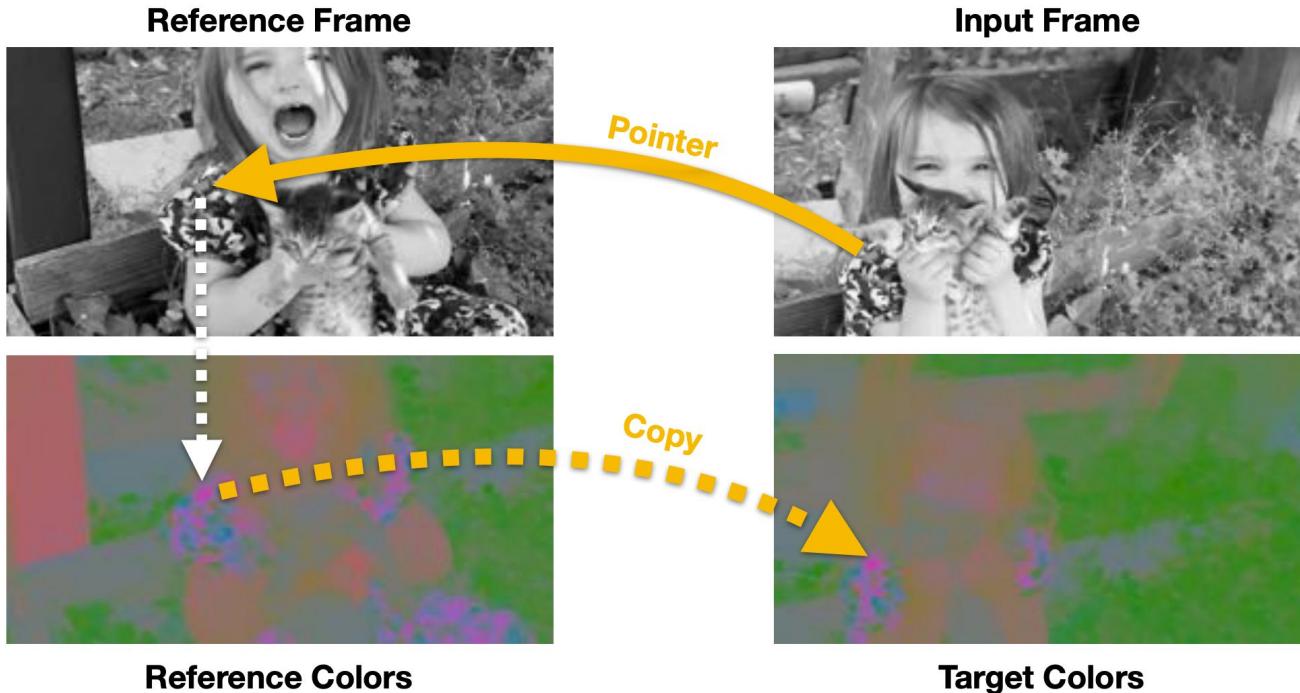
$t = 3$

...

Hypothesis: learning to color video frames should allow model to learn to track regions or objects without labels!

Source: [Vondrick et al., 2018](#)

Learning to color videos



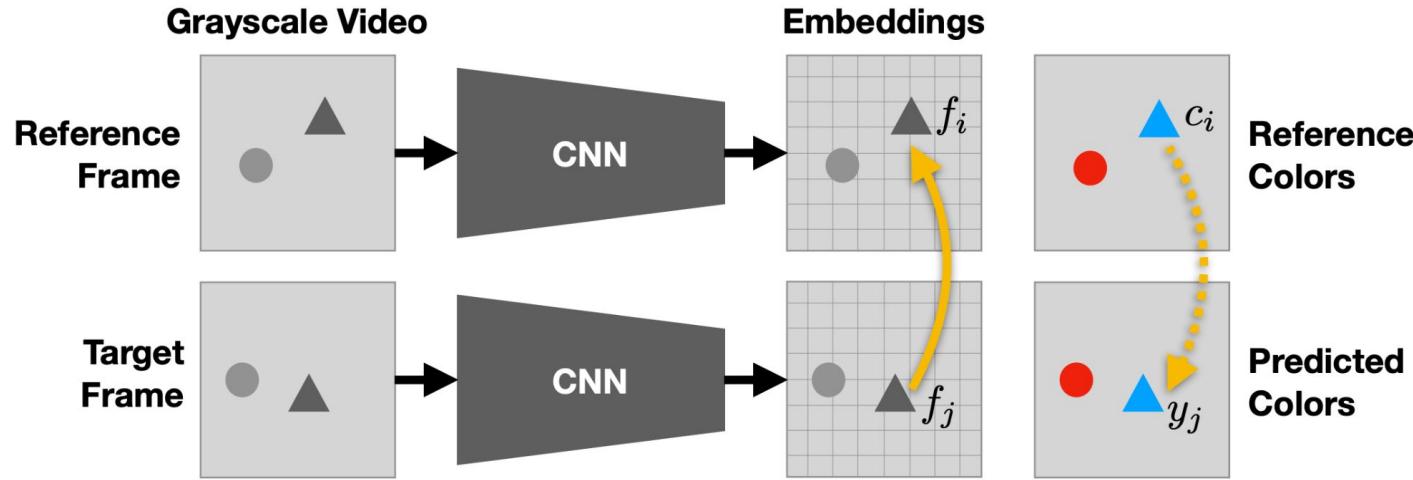
Learning objective:

Establish mappings between reference and target frames in a learned feature space.

Use the mapping as “pointers” to copy the correct color (LAB).

Source: [Vondrick et al., 2018](#)

Learning to color videos

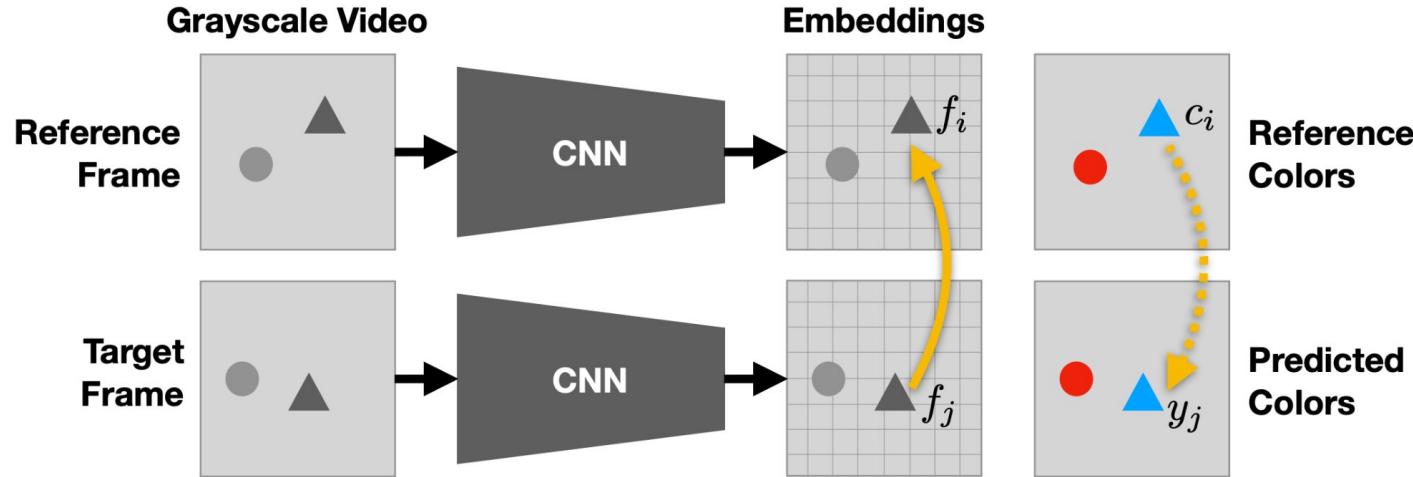


attention map on the
reference frame

$$A_{ij} = \frac{\exp(f_i^T f_j)}{\sum_k \exp(f_k^T f_j)}$$

Source: [Vondrick et al., 2018](#)

Learning to color videos



attention map on the
reference frame

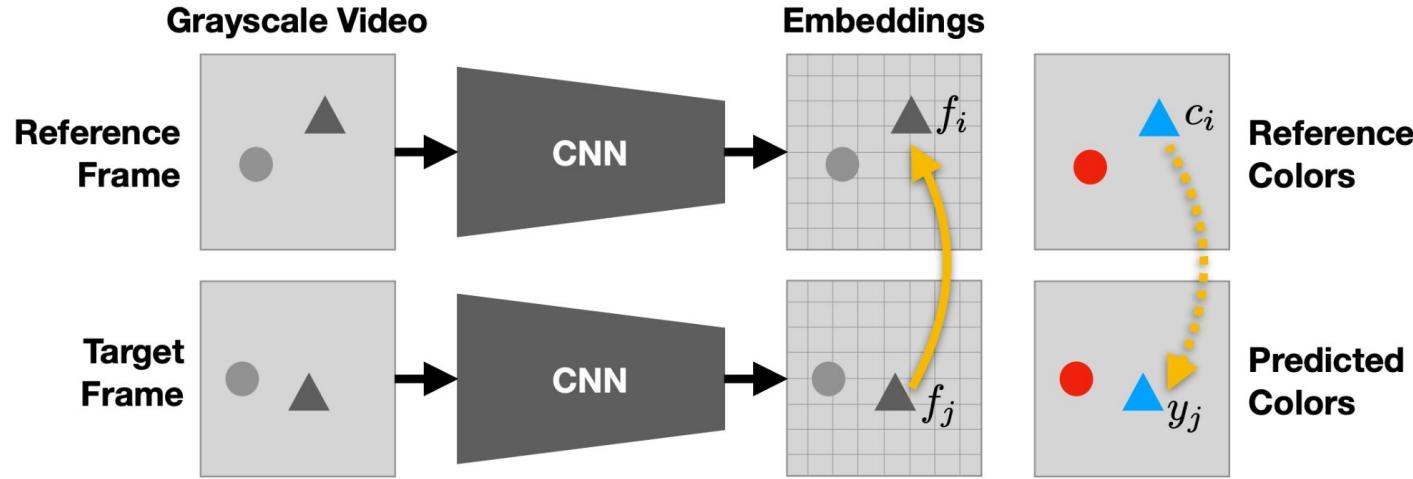
predicted color = weighted
sum of the reference color

$$A_{ij} = \frac{\exp(f_i^T f_j)}{\sum_k \exp(f_k^T f_j)}$$

$$y_j = \sum_i A_{ij} c_i$$

Source: [Vondrick et al., 2018](#)

Learning to color videos



attention map on the
reference frame

$$A_{ij} = \frac{\exp(f_i^T f_j)}{\sum_k \exp(f_k^T f_j)}$$

predicted color = weighted
sum of the reference color

$$y_j = \sum_i A_{ij} c_i$$

loss between predicted color
and ground truth color

$$\min_{\theta} \sum_j \mathcal{L}(y_j, c_j)$$

Source: [Vondrick et al., 2018](#)

Colorizing videos (qualitative)

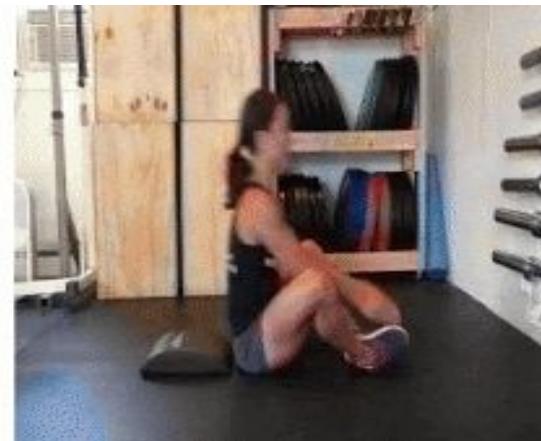
reference frame



target frames (gray)



predicted color



Source: [Google AI blog post](#)

Colorizing videos (qualitative)

reference frame



target frames (gray)



predicted color



Source: [Google AI blog post](#)

Tracking emerges from colorization

Propagate segmentation masks using learned attention



Source: [Google AI blog post](#)

Tracking emerges from colorization

Propagate pose keypoints using learned attention



Source: [Google AI blog post](#)

Summary: pretext tasks from image transformations

- Pretext tasks focus on “visual common sense”, e.g., predict rotations, inpainting, rearrangement, and colorization.
- The models are forced learn good features about natural images, e.g., semantic representation of an object category, in order to solve the pretext tasks.
- We don’t care about the performance of these pretext tasks, but rather how useful the learned features are for downstream tasks (classification, detection, segmentation).

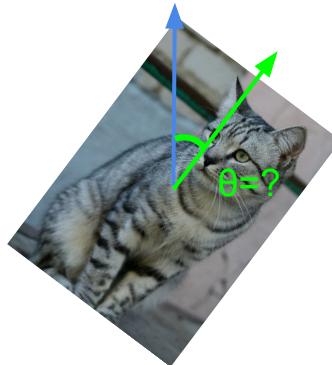
Summary: pretext tasks from image transformations

- Pretext tasks focus on “visual common sense”, e.g., predict rotations, inpainting, rearrangement, and colorization.
- The models are forced learn good features about natural images, e.g., semantic representation of an object category, in order to solve the pretext tasks.
- We don’t care about the performance of these pretext tasks, but rather how useful the learned features are for downstream tasks (classification, detection, segmentation).
- Problems: 1) coming up with individual pretext tasks is tedious, and 2) the learned representations may not be general.

Pretext tasks from image transformations



image completion



rotation prediction



“jigsaw puzzle”

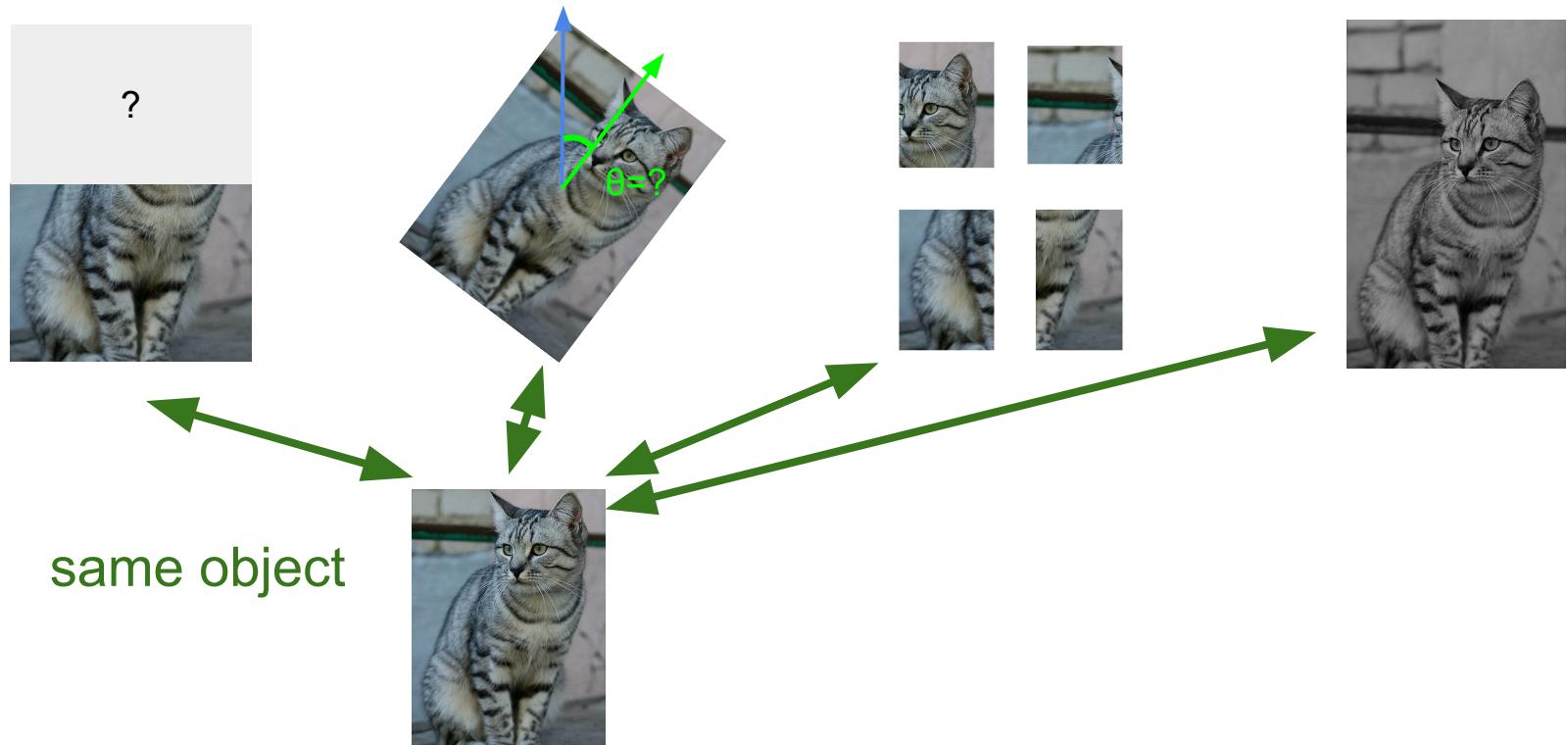


colorization

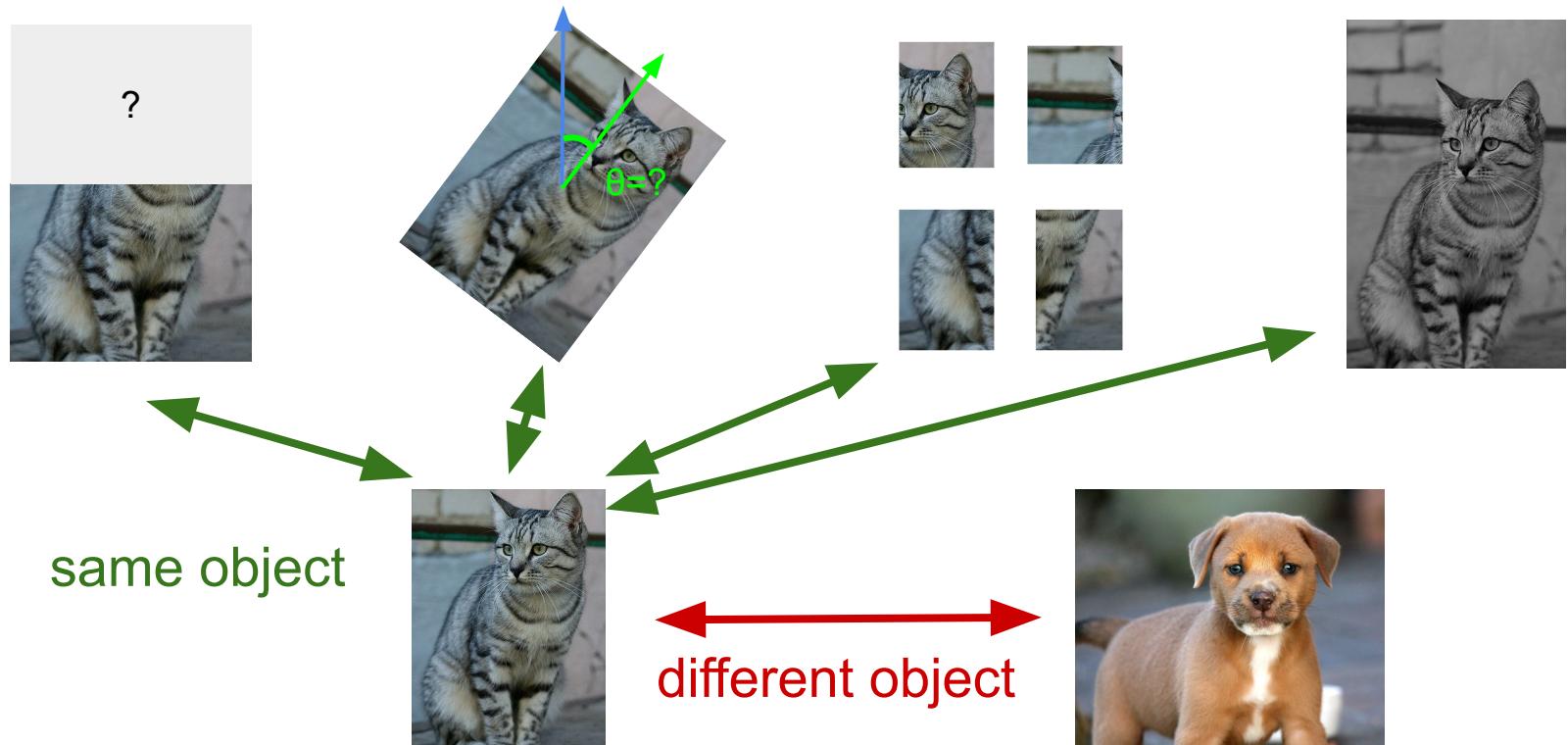
Learned representations may be tied to a specific pretext task!

Can we come up with a more general pretext task?

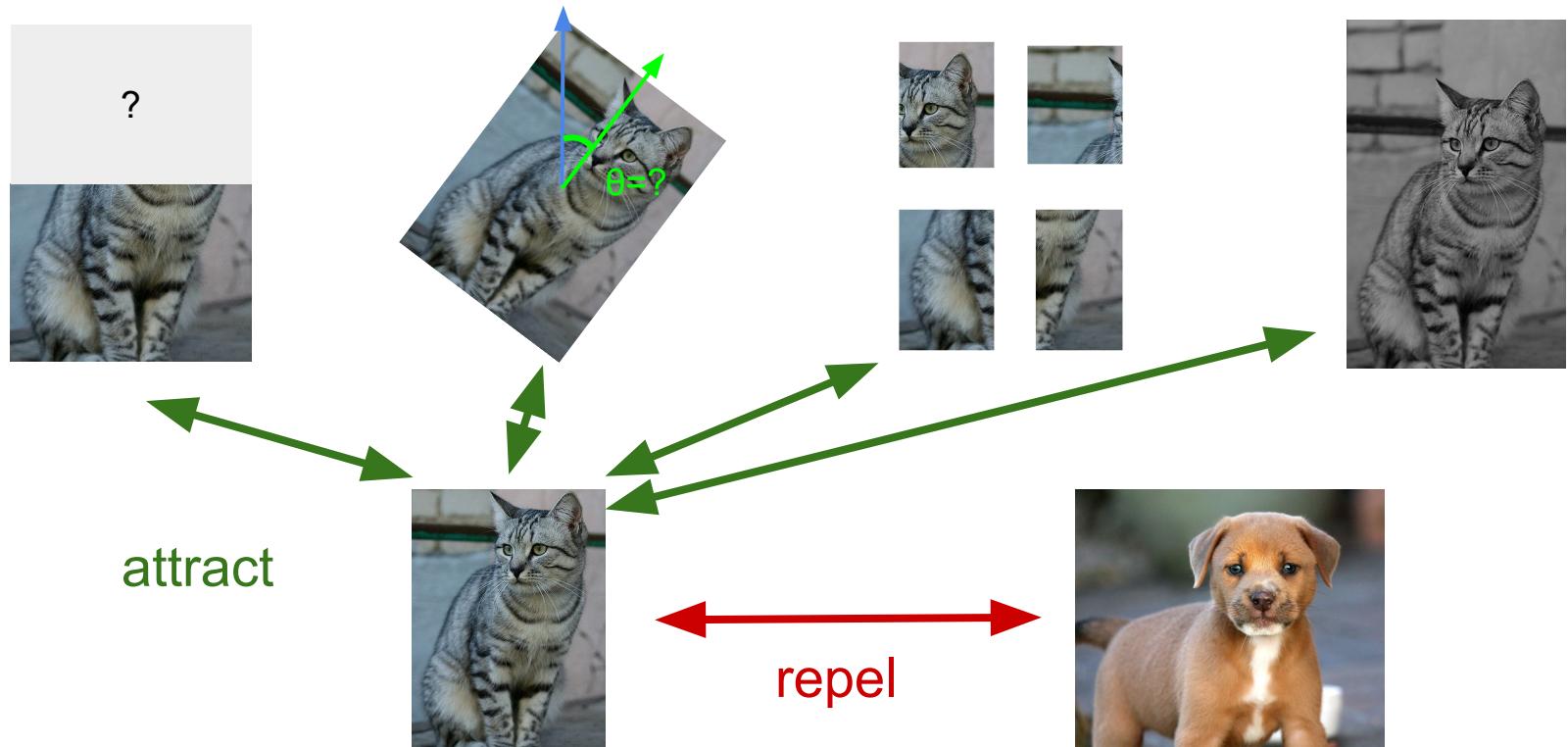
A more general pretext task?



A more general pretext task?



Contrastive Representation Learning



Today's Agenda

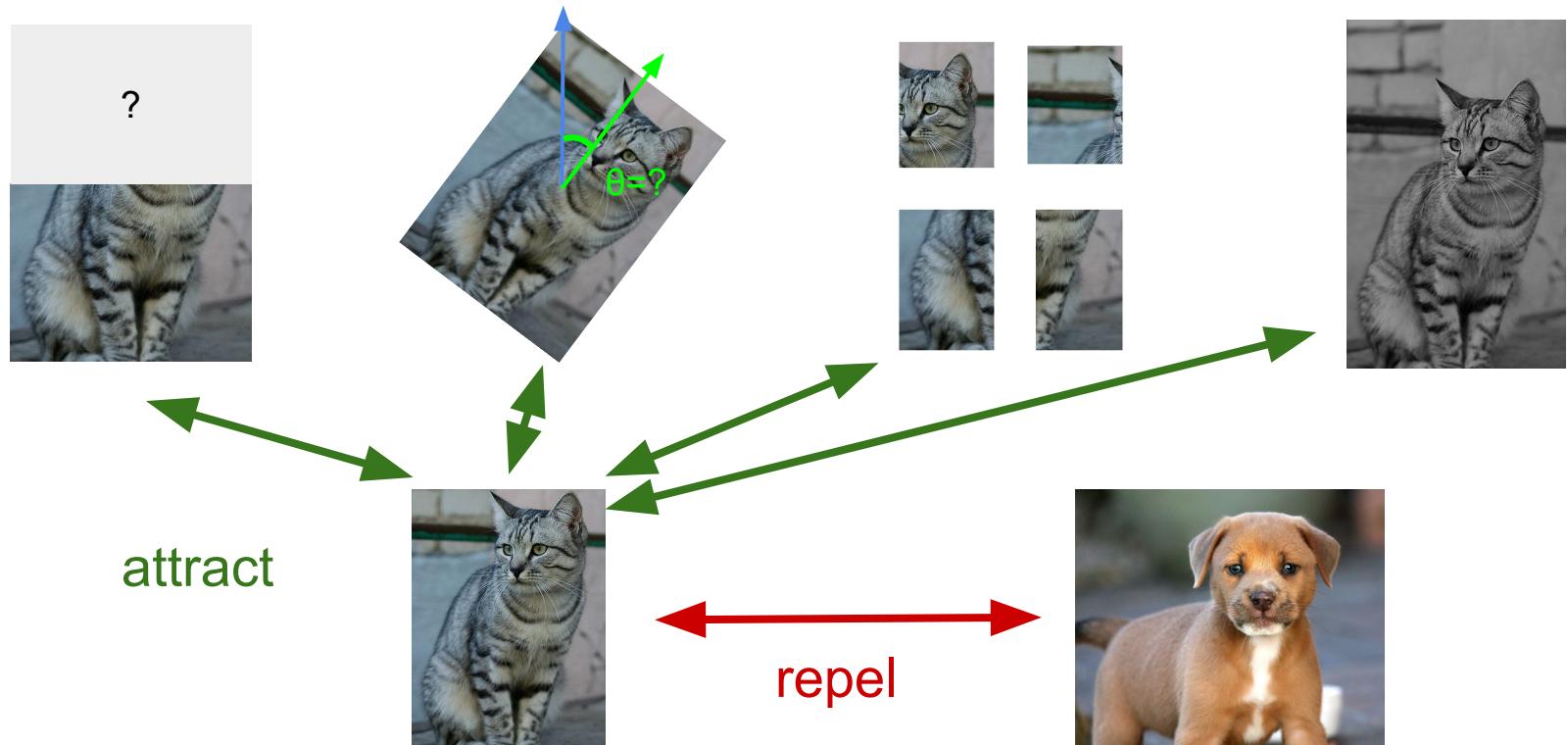
Pretext tasks from image transformations

- Rotation, inpainting, rearrangement, coloring

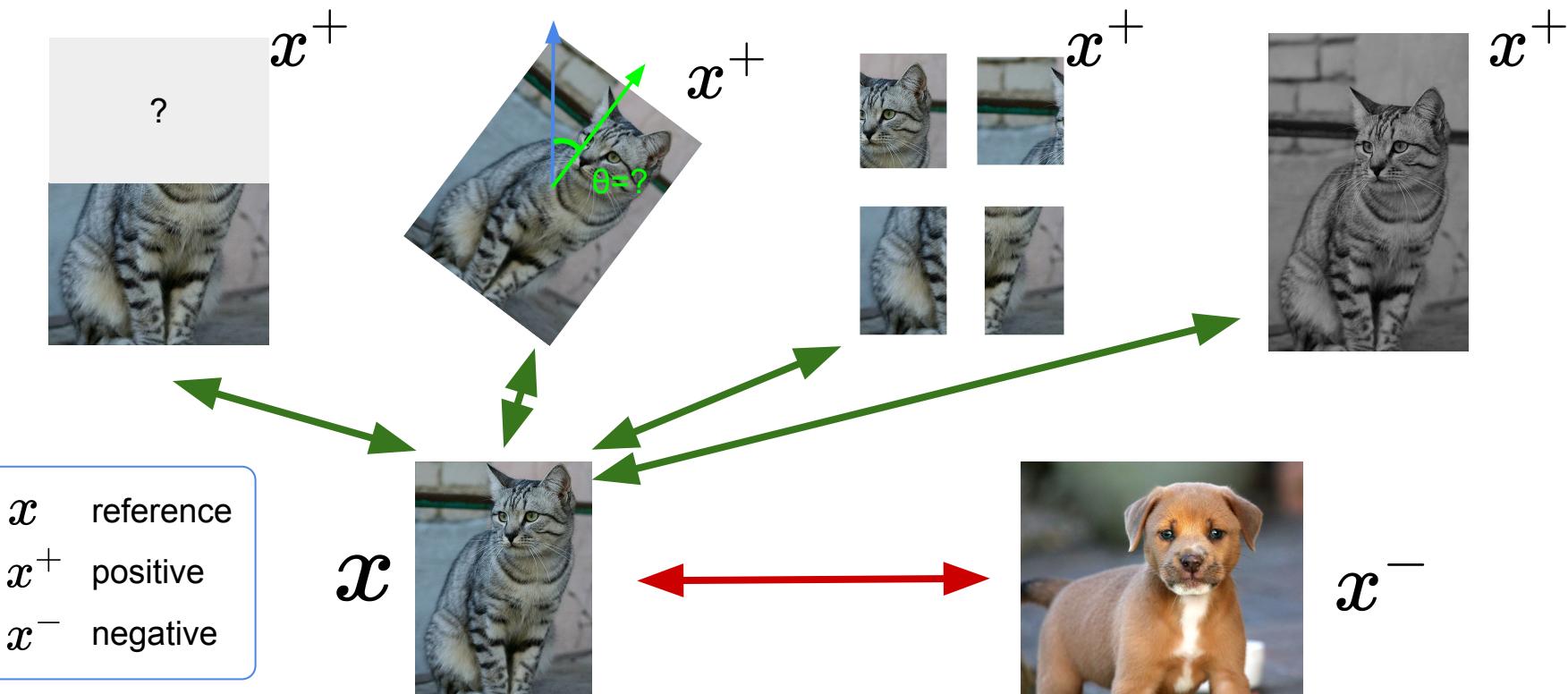
Contrastive representation learning

- Intuition and formulation
- Instance contrastive learning: SimCLR and MOCO
- Sequence contrastive learning: CPC

Contrastive Representation Learning



Contrastive Representation Learning



A formulation of contrastive learning

What we want:

$$\text{score}(f(x), f(x^+)) \gg \text{score}(f(x), f(x^-))$$

x : reference sample; x^+ positive sample; x^- negative sample

Given a chosen score function, we aim to learn an **encoder function** f that yields high score for positive pairs (x, x^+) and low scores for negative pairs (x, x^-) .

A formulation of contrastive learning

Loss function given 1 positive sample and $N - 1$ negative samples:

$$L = -\mathbb{E}_X \left[\log \frac{\exp(s(f(x), f(x^+)))}{\exp(s(f(x), f(x^+))) + \sum_{j=1}^{N-1} \exp(s(f(x), f(x_j^-)))} \right]$$

A formulation of contrastive learning

Loss function given 1 positive sample and $N - 1$ negative samples:

$$L = -\mathbb{E}_X \left[\log \frac{\exp(s(f(x), f(x^+)))}{\exp(s(f(x), f(x^+))) + \sum_{j=1}^{N-1} \exp(s(f(x), f(x_j^-)))} \right]$$

 x  x^+  x  x_1^-  x_2^-  x_3^- \dots

A formulation of contrastive learning

Loss function given 1 positive sample and $N - 1$ negative samples:

$$L = -\mathbb{E}_X \left[\log \frac{\exp(s(f(x), f(x^+)))}{\exp(s(f(x), f(x^+))) + \sum_{j=1}^{N-1} \exp(s(f(x), f(x_j^-)))} \right]$$

score for the positive pair
score for the N-1 negative pairs

This seems familiar ...

A formulation of contrastive learning

Loss function given 1 positive sample and $N - 1$ negative samples:

$$L = -\mathbb{E}_X \left[\log \frac{\exp(s(f(x), f(x^+)))}{\exp(s(f(x), f(x^+))) + \sum_{j=1}^{N-1} \exp(s(f(x), f(x_j^-)))} \right]$$

score for the positive pair
score for the N-1 negative pairs

This seems familiar ...

Cross entropy loss for a N-way softmax classifier!

I.e., learn to find the positive sample from the N samples

A formulation of contrastive learning

Loss function given 1 positive sample and $N - 1$ negative samples:

$$L = -\mathbb{E}_X \left[\log \frac{\exp(s(f(x), f(x^+)))}{\exp(s(f(x), f(x^+))) + \sum_{j=1}^{N-1} \exp(s(f(x), f(x_j^-)))} \right]$$

Commonly known as the InfoNCE loss ([van den Oord et al., 2018](#))

A *lower bound* on the mutual information between $f(x)$ and $f(x^+)$

$$MI[f(x), f(x^+)] - \log(N) \geq -L$$

The larger the negative sample size (N), the tighter the bound

Detailed derivation: [Poole et al., 2019](#)

SimCLR: A Simple Framework for Contrastive Learning

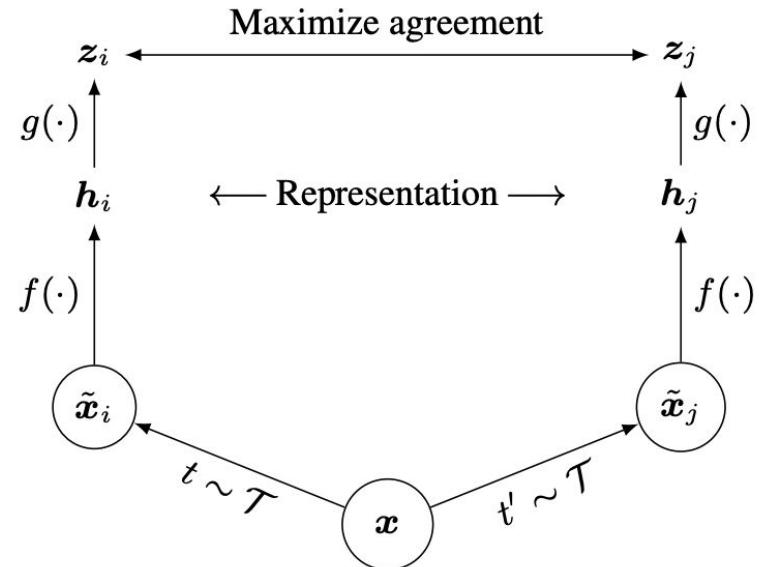
Cosine similarity as the score function:

$$s(u, v) = \frac{u^T v}{\|u\| \|v\|}$$

Use a projection network $g(\cdot)$ to project features to a space where contrastive learning is applied

Generate positive samples through data augmentation:

- random cropping, random color distortion, and random blur.



Source: [Chen et al., 2020](#)

SimCLR: generating positive samples from data augmentation



(a) Original



(b) Crop and resize



(c) Crop, resize (and flip)



(d) Color distort. (drop)



(e) Color distort. (jitter)



(f) Rotate $\{90^\circ, 180^\circ, 270^\circ\}$



(g) Cutout



(h) Gaussian noise



(i) Gaussian blur



(j) Sobel filtering

Source: [Chen et al., 2020](#)

SimCLR

Generate a positive pair
by sampling data
augmentation functions

Algorithm 1 SimCLR's main learning algorithm.

```
input: batch size  $N$ , constant  $\tau$ , structure of  $f, g, \mathcal{T}$ .
for sampled minibatch  $\{\mathbf{x}_k\}_{k=1}^N$  do
    for all  $k \in \{1, \dots, N\}$  do
        draw two augmentation functions  $t \sim \mathcal{T}, t' \sim \mathcal{T}$ 
        # the first augmentation
         $\tilde{\mathbf{x}}_{2k-1} = t(\mathbf{x}_k)$ 
         $\mathbf{h}_{2k-1} = f(\tilde{\mathbf{x}}_{2k-1})$  # representation
         $\mathbf{z}_{2k-1} = g(\mathbf{h}_{2k-1})$  # projection
        # the second augmentation
         $\tilde{\mathbf{x}}_{2k} = t'(\mathbf{x}_k)$ 
         $\mathbf{h}_{2k} = f(\tilde{\mathbf{x}}_{2k})$  # representation
         $\mathbf{z}_{2k} = g(\mathbf{h}_{2k})$  # projection
    end for
    for all  $i \in \{1, \dots, 2N\}$  and  $j \in \{1, \dots, 2N\}$  do
         $s_{i,j} = \mathbf{z}_i^\top \mathbf{z}_j / (\|\mathbf{z}_i\| \|\mathbf{z}_j\|)$  # pairwise similarity
    end for
    define  $\ell(i, j)$  as  $\ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$ 
     $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^N [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$ 
    update networks  $f$  and  $g$  to minimize  $\mathcal{L}$ 
end for
return encoder network  $f(\cdot)$ , and throw away  $g(\cdot)$ 
```

*We use a slightly different formulation in the assignment.
You should follow the assignment instructions.

Source: [Chen et al., 2020](#)

SimCLR

Generate a positive pair
by sampling data
augmentation functions

Algorithm 1 SimCLR's main learning algorithm.

```
input: batch size  $N$ , constant  $\tau$ , structure of  $f, g, \mathcal{T}$ .  
for sampled minibatch  $\{\mathbf{x}_k\}_{k=1}^N$  do  
    for all  $k \in \{1, \dots, N\}$  do  
        draw two augmentation functions  $t \sim \mathcal{T}, t' \sim \mathcal{T}$   
        # the first augmentation  
         $\tilde{\mathbf{x}}_{2k-1} = t(\mathbf{x}_k)$   
         $\mathbf{h}_{2k-1} = f(\tilde{\mathbf{x}}_{2k-1})$  # representation  
         $\mathbf{z}_{2k-1} = g(\mathbf{h}_{2k-1})$  # projection  
        # the second augmentation  
         $\tilde{\mathbf{x}}_{2k} = t'(\mathbf{x}_k)$   
         $\mathbf{h}_{2k} = f(\tilde{\mathbf{x}}_{2k})$  # representation  
         $\mathbf{z}_{2k} = g(\mathbf{h}_{2k})$  # projection  
    end for  
    for all  $i \in \{1, \dots, 2N\}$  and  $j \in \{1, \dots, 2N\}$  do  
         $s_{i,j} = \mathbf{z}_i^\top \mathbf{z}_j / (\|\mathbf{z}_i\| \|\mathbf{z}_j\|)$  # pairwise similarity  
    end for  
    define  $\ell(i, j)$  as  $\ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$   
     $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^N [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$   
    update networks  $f$  and  $g$  to minimize  $\mathcal{L}$   
end for  
return encoder network  $f(\cdot)$ , and throw away  $g(\cdot)$ 
```

*We use a slightly different formulation in the assignment.
You should follow the assignment instructions.

InfoNCE loss:
Use all non-positive samples in the batch as x^-

Source: [Chen et al., 2020](#)

SimCLR

Generate a positive pair
by sampling data
augmentation functions

Iterate through and
use each of the $2N$
sample as reference,
compute average loss

Algorithm 1 SimCLR's main learning algorithm.

```
input: batch size  $N$ , constant  $\tau$ , structure of  $f, g, \mathcal{T}$ .  
for sampled minibatch  $\{\mathbf{x}_k\}_{k=1}^N$  do  
    for all  $k \in \{1, \dots, N\}$  do  
        draw two augmentation functions  $t \sim \mathcal{T}, t' \sim \mathcal{T}$   
        # the first augmentation  
         $\tilde{\mathbf{x}}_{2k-1} = t(\mathbf{x}_k)$   
         $\mathbf{h}_{2k-1} = f(\tilde{\mathbf{x}}_{2k-1})$  # representation  
         $\mathbf{z}_{2k-1} = g(\mathbf{h}_{2k-1})$  # projection  
        # the second augmentation  
         $\tilde{\mathbf{x}}_{2k} = t'(\mathbf{x}_k)$   
         $\mathbf{h}_{2k} = f(\tilde{\mathbf{x}}_{2k})$  # representation  
         $\mathbf{z}_{2k} = g(\mathbf{h}_{2k})$  # projection  
    end for  
    for all  $i \in \{1, \dots, 2N\}$  and  $j \in \{1, \dots, 2N\}$  do  
         $s_{i,j} = \mathbf{z}_i^\top \mathbf{z}_j / (\|\mathbf{z}_i\| \|\mathbf{z}_j\|)$  # pairwise similarity  
    end for  
    define  $\ell(i, j)$  as  $\ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$   
     $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^N [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$   
    update networks  $f$  and  $g$  to minimize  $\mathcal{L}$   
end for  
return encoder network  $f(\cdot)$ , and throw away  $g(\cdot)$ 
```

*We use a slightly different formulation in the assignment.
You should follow the assignment instructions.

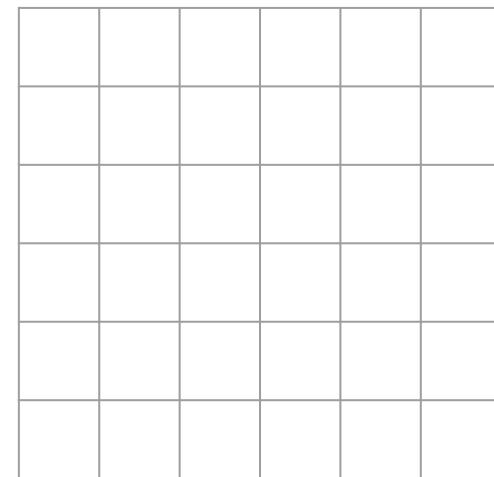
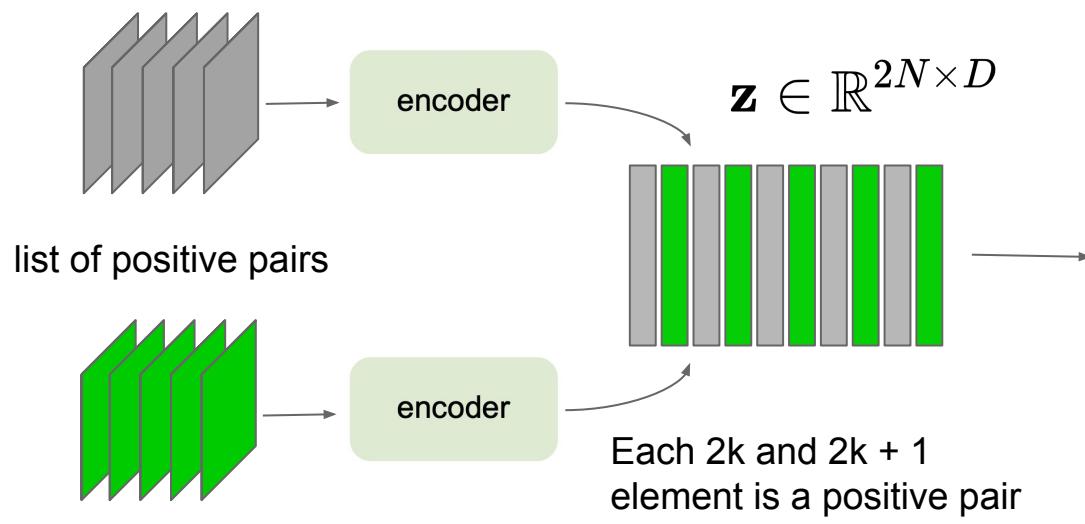
InfoNCE loss:
Use all non-positive samples in the batch as x^-

Source: [Chen et al., 2020](#)

SimCLR: mini-batch training

$$s_{i,j} = \frac{z_i^T z_j}{\|z_i\| \|z_j\|}$$

“Affinity matrix”

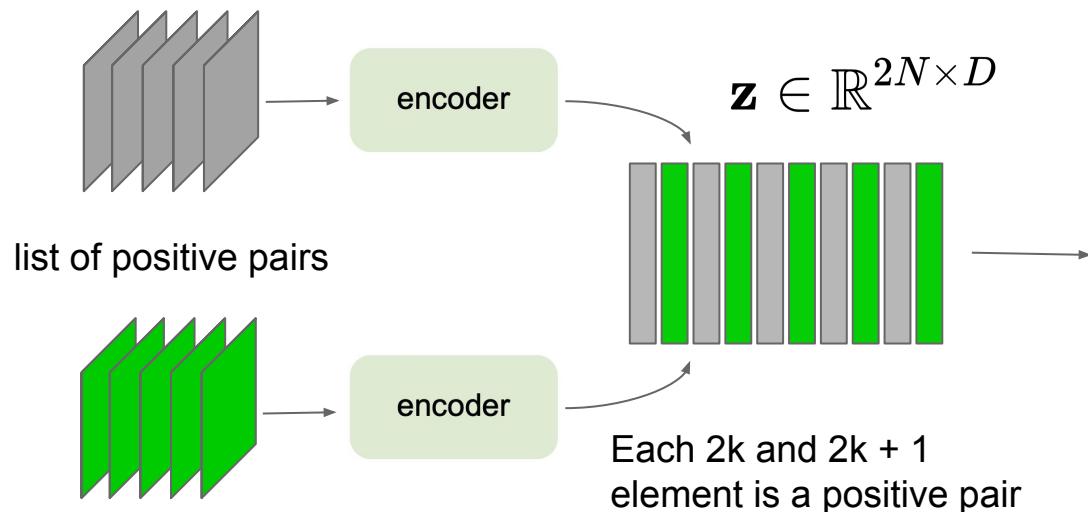


*We use a slightly different formulation in the assignment.
You should follow the assignment instructions.

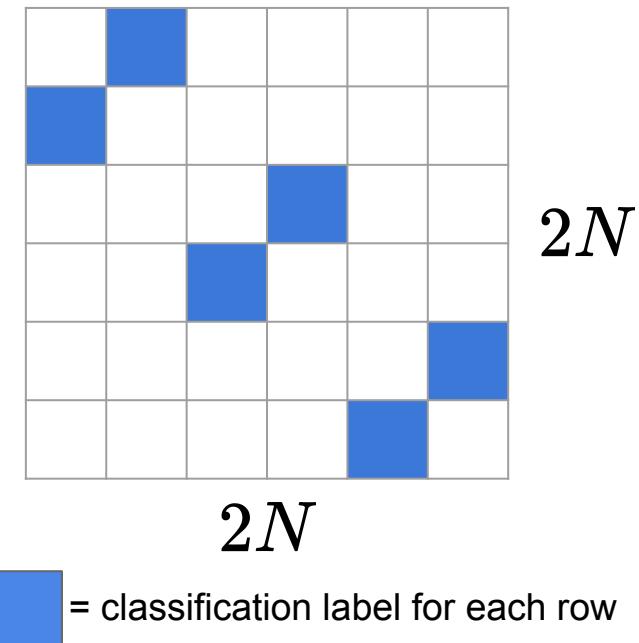
SimCLR: mini-batch training

$$S_{i,j} = \frac{z_i^T z_j}{||z_i|| ||z_j||}$$

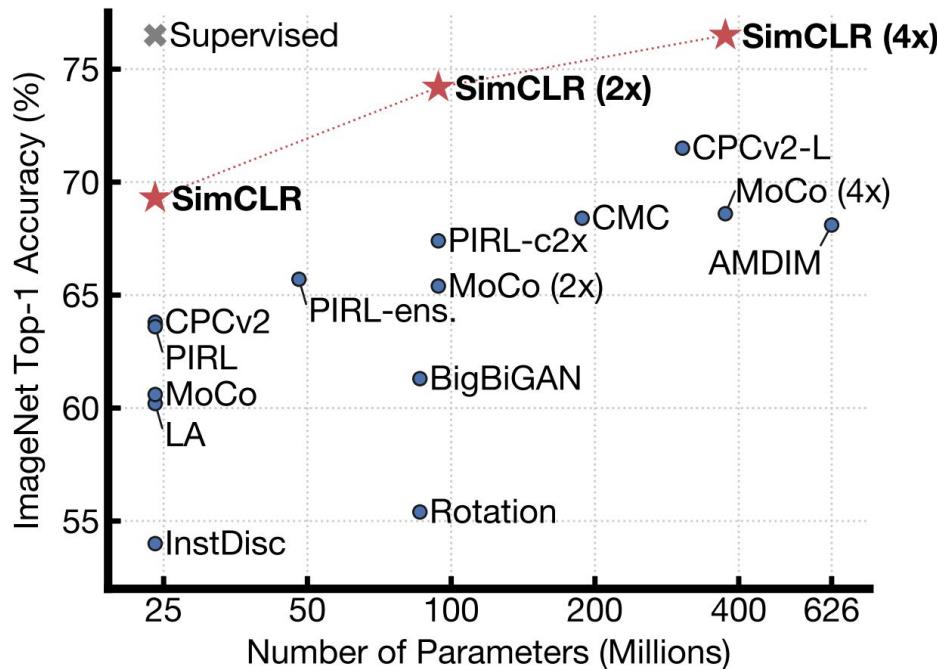
“Affinity matrix”



*We use a slightly different formulation in the assignment.
You should follow the assignment instructions.



Training linear classifier on SimCLR features



Train feature encoder on **ImageNet** (entire training set) using SimCLR.

Freeze feature encoder, train a linear classifier on top with labeled data.

Source: [Chen et al., 2020](#)

Semi-supervised learning on SimCLR features

Method	Architecture	Label fraction		
		1%	10%	Top 5
Supervised baseline	ResNet-50	48.4	80.4	
<i>Methods using other label-propagation:</i>				
Pseudo-label	ResNet-50	51.6	82.4	
VAT+Entropy Min.	ResNet-50	47.0	83.4	
UDA (w. RandAug)	ResNet-50	-	88.5	
FixMatch (w. RandAug)	ResNet-50	-	89.1	
S4L (Rot+VAT+En. M.)	ResNet-50 (4×)	-	91.2	
<i>Methods using representation learning only:</i>				
InstDisc	ResNet-50	39.2	77.4	
BigBiGAN	RevNet-50 (4×)	55.2	78.8	
PIRL	ResNet-50	57.2	83.8	
CPC v2	ResNet-161(*)	77.9	91.2	
SimCLR (ours)	ResNet-50	75.5	87.8	
SimCLR (ours)	ResNet-50 (2×)	83.0	91.2	
SimCLR (ours)	ResNet-50 (4×)	85.8	92.6	

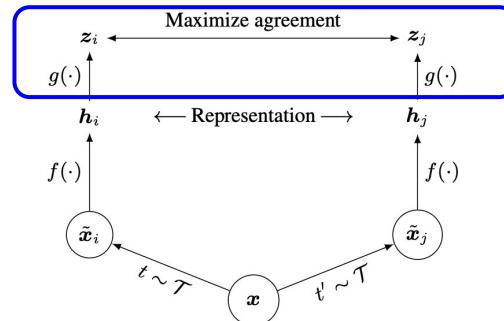
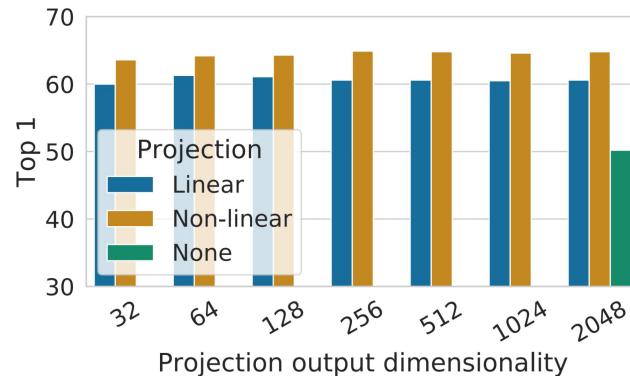
Table 7. ImageNet accuracy of models trained with few labels.

Train feature encoder on **ImageNet** (entire training set) using SimCLR.

Finetune the encoder with 1% / 10% of labeled data on ImageNet.

Source: [Chen et al., 2020](#)

SimCLR design choices: projection head



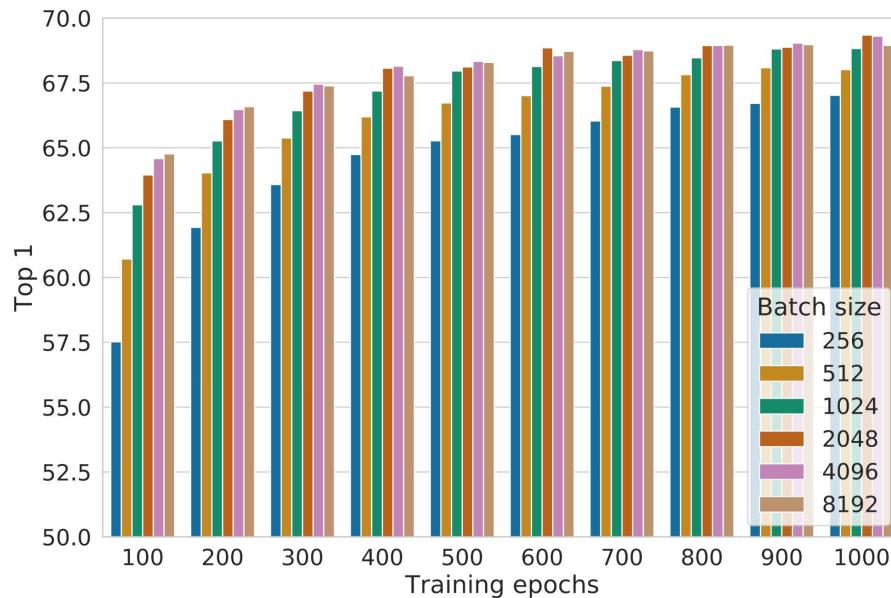
Linear / non-linear projection heads improve representation learning.

A possible explanation:

- contrastive learning objective may discard useful information for downstream tasks
- representation space \mathbf{z} is trained to be invariant to data transformation.
- by leveraging the projection head $\mathbf{g}(\cdot)$, more information can be preserved in the \mathbf{h} representation space

Source: [Chen et al., 2020](#)

SimCLR design choices: large batch size



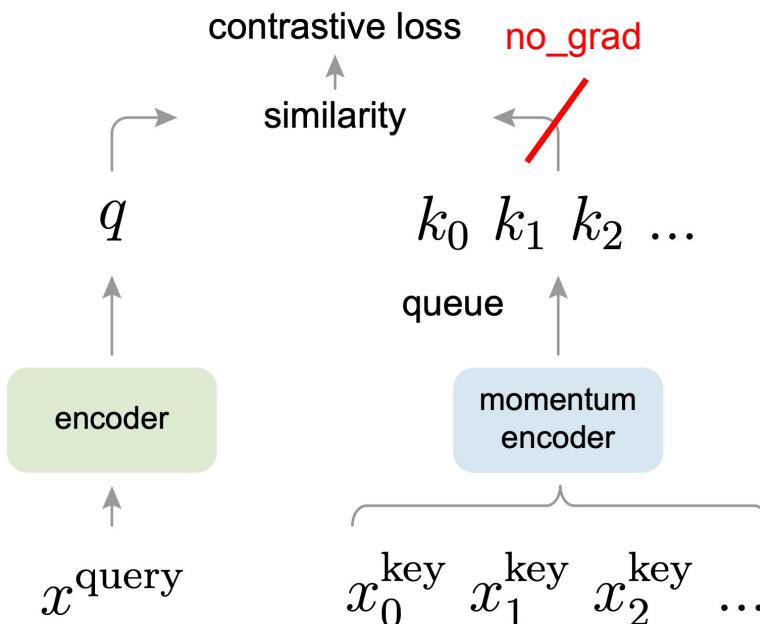
Large training batch size is crucial for SimCLR!

Large batch size causes large memory footprint during backpropagation:
requires distributed training on TPUs
(ImageNet experiments)

Figure 9. Linear evaluation models (ResNet-50) trained with different batch size and epochs. Each bar is a single run from scratch.¹⁰

Source: [Chen et al., 2020](#)

Momentum Contrastive Learning (MoCo)

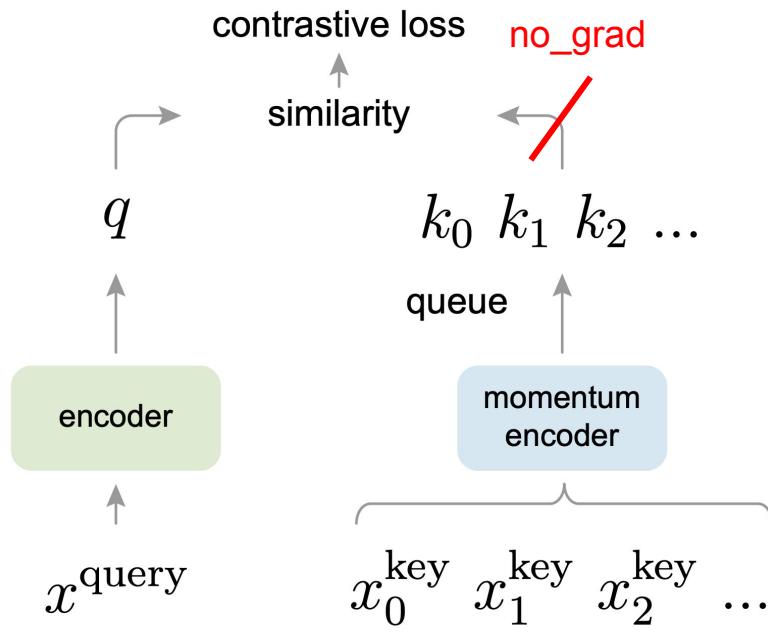


Key differences to SimCLR:

- Keep a running **queue** of keys (negative samples).
- Compute gradients and update the encoder **only through the queries**.
- Decouple min-batch size with the number of keys: can support **a large number of negative samples**.

Source: [He et al., 2020](#)

Momentum Contrastive Learning (MoCo)



Key differences to SimCLR:

- Keep a running **queue** of keys (negative samples).
- Compute gradients and update the encoder **only through the queries**.
- Decouple min-batch size with the number of keys: can support **a large number of negative samples**.
- The key encoder is **slowly progressing** through the momentum update rules:

$$\theta_k \leftarrow m\theta_k + (1 - m)\theta_q$$

Source: [He et al., 2020](#)

MoCo

Generate a positive pair
by sampling data
augmentation functions

No gradient through
the positive sample

Update the FIFO
negative sample queue

Algorithm 1 Pseudocode of MoCo in a PyTorch-like style.

```
# f_q, f_k: encoder networks for query and key
# queue: dictionary as a queue of K keys (CxK)
# m: momentum
# t: temperature

f_k.params = f_q.params # initialize
for x in loader: # load a minibatch x with N samples
    x_q = aug(x) # a randomly augmented version
    x_k = aug(x) # another randomly augmented version

    q = f_q.forward(x_q) # queries: NxC
    k = f_k.forward(x_k) # keys: NxC
    k = k.detach() # no gradient to keys

    # positive logits: Nx1
    l_pos = bmm(q.view(N, 1, C), k.view(N, C, 1))

    # negative logits: NxK
    l_neg = mm(q.view(N, C), queue.view(C, K))

    # logits: Nx(1+K)
    logits = cat([l_pos, l_neg], dim=1)

    # contrastive loss, Eqn.(1)
    labels = zeros(N) # positives are the 0-th
    loss = CrossEntropyLoss(logits/t, labels)

    # SGD update: query network
    loss.backward()
    update(f_q.params)

    # momentum update: key network
    f_k.params = m*f_k.params+(1-m)*f_q.params

    # update dictionary
    enqueue(queue, k) # enqueue the current minibatch
    dequeue(queue) # dequeue the earliest minibatch
```

bmm: batch matrix multiplication; mm: matrix multiplication; cat: concatenation.

Use the running
queue of keys as the
negative samples

InfoNCE loss

Update f_k through
momentum

Source: [He et al., 2020](#)

“MoCo V2”

Improved Baselines with Momentum Contrastive Learning

Xinlei Chen Haoqi Fan Ross Girshick Kaiming He
Facebook AI Research (FAIR)

A hybrid of ideas from SimCLR and MoCo:

- **From SimCLR:** non-linear projection head and strong data augmentation.
- **From MoCo:** momentum-updated queues that allow training on a large number of negative samples (no TPU required!).

Source: [Chen et al., 2020](#)

MoCo vs. SimCLR vs. MoCo V2

Key takeaways:

- Non-linear projection head and strong data augmentation are crucial for contrastive learning.

case	unsup. pre-train				ImageNet acc.	VOC detection		
	MLP	aug+	cos	epochs		AP ₅₀	AP	AP ₇₅
supervised					76.5	81.3	53.5	58.8
MoCo v1				200	60.6	81.5	55.9	62.6
(a)	✓			200	66.2	82.0	56.4	62.6
(b)		✓		200	63.4	82.2	56.8	63.2
(c)	✓	✓		200	67.3	82.5	57.2	63.9
(d)	✓	✓	✓	200	67.5	82.4	57.0	63.6
(e)	✓	✓	✓	800	71.1	82.5	57.4	64.0

Table 1. **Ablation of MoCo baselines**, evaluated by ResNet-50 for (i) ImageNet linear classification, and (ii) fine-tuning VOC object detection (mean of 5 trials). “MLP”: with an MLP head; “aug+”: with extra blur augmentation; “cos”: cosine learning rate schedule.

Source: [Chen et al., 2020](#)

MoCo vs. SimCLR vs. MoCo V2

case	MLP	aug+	cos	unsup. pre-train epochs	batch	ImageNet acc.
MoCo v1 [6]				200	256	60.6
SimCLR [2]	✓	✓	✓	200	256	61.9
SimCLR [2]	✓	✓	✓	200	8192	66.6
MoCo v2	✓	✓	✓	200	256	67.5
<i>results of longer unsupervised training follow:</i>						
SimCLR [2]	✓	✓	✓	1000	4096	69.3
MoCo v2	✓	✓	✓	800	256	71.1

Table 2. **MoCo vs. SimCLR**: ImageNet linear classifier accuracy (**ResNet-50, 1-crop 224×224**), trained on features from unsupervised pre-training. “aug+” in SimCLR includes blur and stronger color distortion. SimCLR ablations are from Fig. 9 in [2] (we thank the authors for providing the numerical results).

Key takeaways:

- Non-linear projection head and strong data augmentation are crucial for contrastive learning.
- Decoupling mini-batch size with negative sample size allows MoCo-V2 to outperform SimCLR with smaller batch size (256 vs. 8192).

Source: [Chen et al., 2020](#)

MoCo vs. SimCLR vs. MoCo V2

Key takeaways:

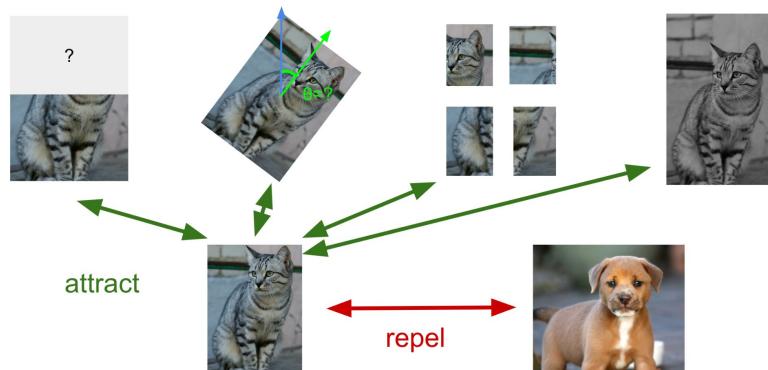
- Non-linear projection head and strong data augmentation are crucial for contrastive learning.
- Decoupling mini-batch size with negative sample size allows MoCo-V2 to outperform SimCLR with smaller batch size (256 vs. 8192).
- ... all with much smaller memory footprint! (“end-to-end” means SimCLR here)

mechanism	batch	memory / GPU	time / 200-ep.
MoCo	256	5.0G	53 hrs
end-to-end	256	7.4G	65 hrs
end-to-end	4096	93.0G [†]	n/a

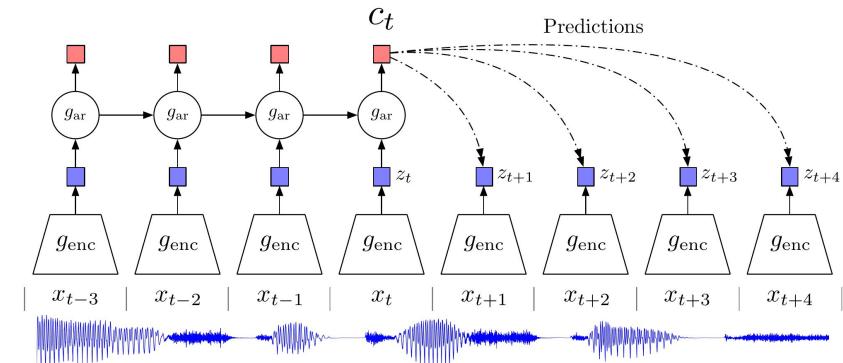
Table 3. **Memory and time cost** in 8 V100 16G GPUs, implemented in PyTorch. [†]: based on our estimation.

Source: [Chen et al., 2020](#)

Instance vs. Sequence Contrastive Learning



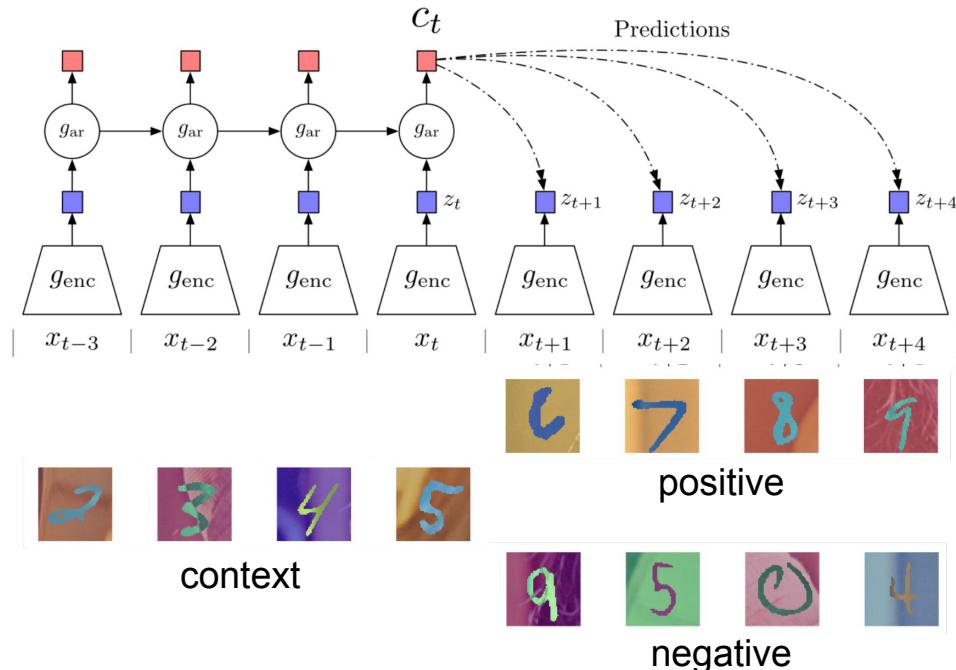
Instance-level contrastive learning:
contrastive learning based on
positive & negative instances.
Examples: SimCLR, MoCo



Source: [van den Oord et al., 2018](#)

Sequence-level contrastive learning:
contrastive learning based on
sequential / temporal orders.
Example: **Contrastive Predictive Coding (CPC)**

Contrastive Predictive Coding (CPC)



Contrastive: contrast between “right” and “wrong” sequences using contrastive learning.

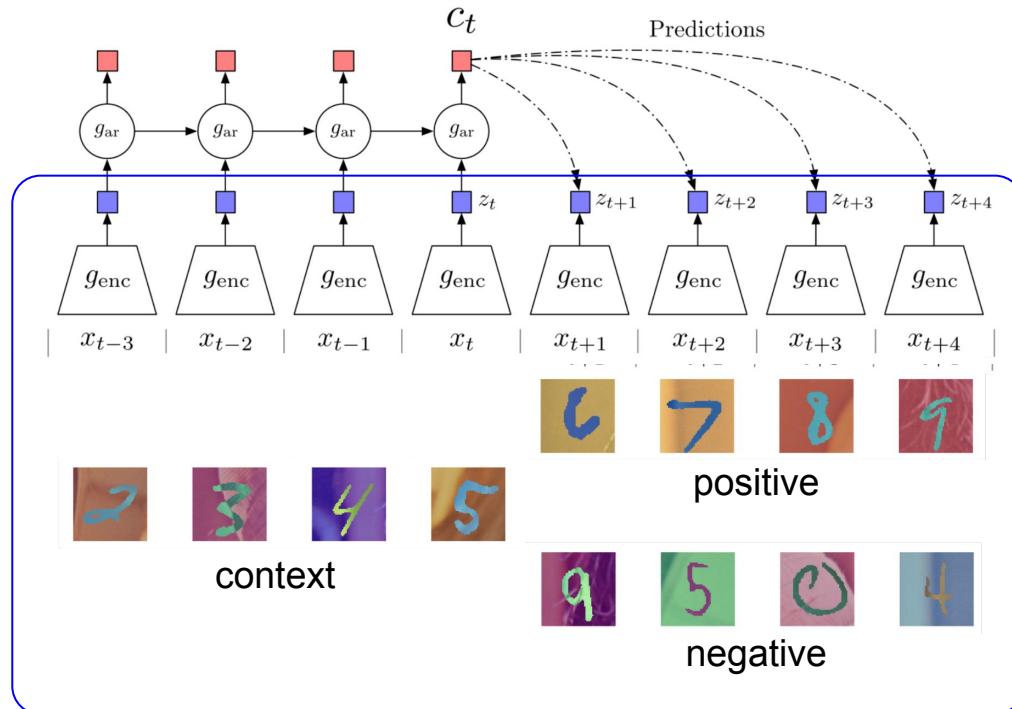
Predictive: the model has to predict future patterns given the current context.

Coding: the model learns useful feature vectors, or “code”, for downstream tasks, similar to other self-supervised methods.

Figure [source](#)

Source: [van den Oord et al., 2018](#),

Contrastive Predictive Coding (CPC)

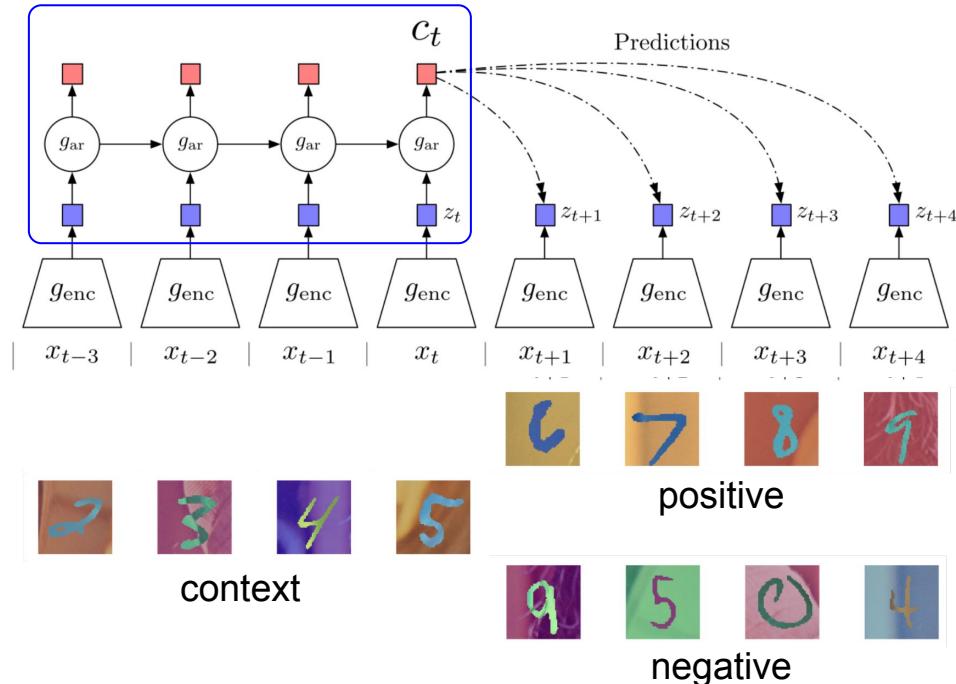


1. Encode all samples in a sequence into vectors $\mathbf{z}_t = \mathbf{g}_{enc}(\mathbf{x}_t)$

Figure [source](#)

Source: [van den Oord et al., 2018](#),

Contrastive Predictive Coding (CPC)

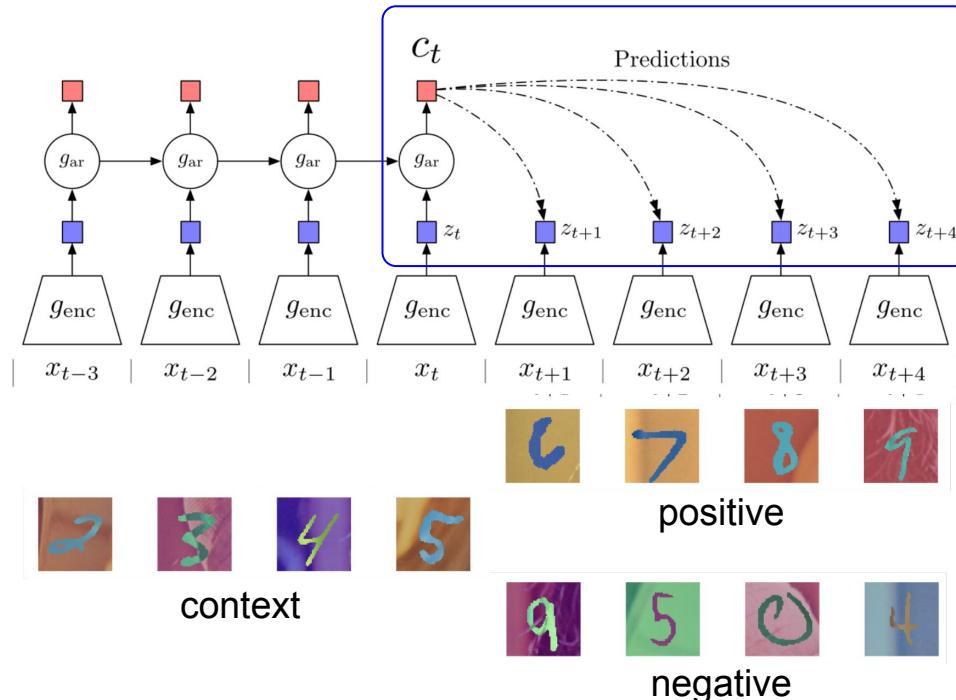


1. Encode all samples in a sequence into vectors $\mathbf{z}_t = \mathbf{g}_{enc}(\mathbf{x}_t)$
2. Summarize context (e.g., half of a sequence) into a context code \mathbf{c}_t using an auto-regressive model (\mathbf{g}_{ar}). The original paper uses GRU-RNN here.

Figure [source](#)

Source: [van den Oord et al., 2018](#),

Contrastive Predictive Coding (CPC)



1. Encode all samples in a sequence into vectors $\mathbf{z}_t = \mathbf{g}_{enc}(\mathbf{x}_t)$
2. Summarize context (e.g., half of a sequence) into a context code \mathbf{c}_t using an auto-regressive model (\mathbf{g}_{ar})
3. Compute InfoNCE loss between the context \mathbf{c}_t and future code \mathbf{z}_{t+k} using the following time-dependent score function:

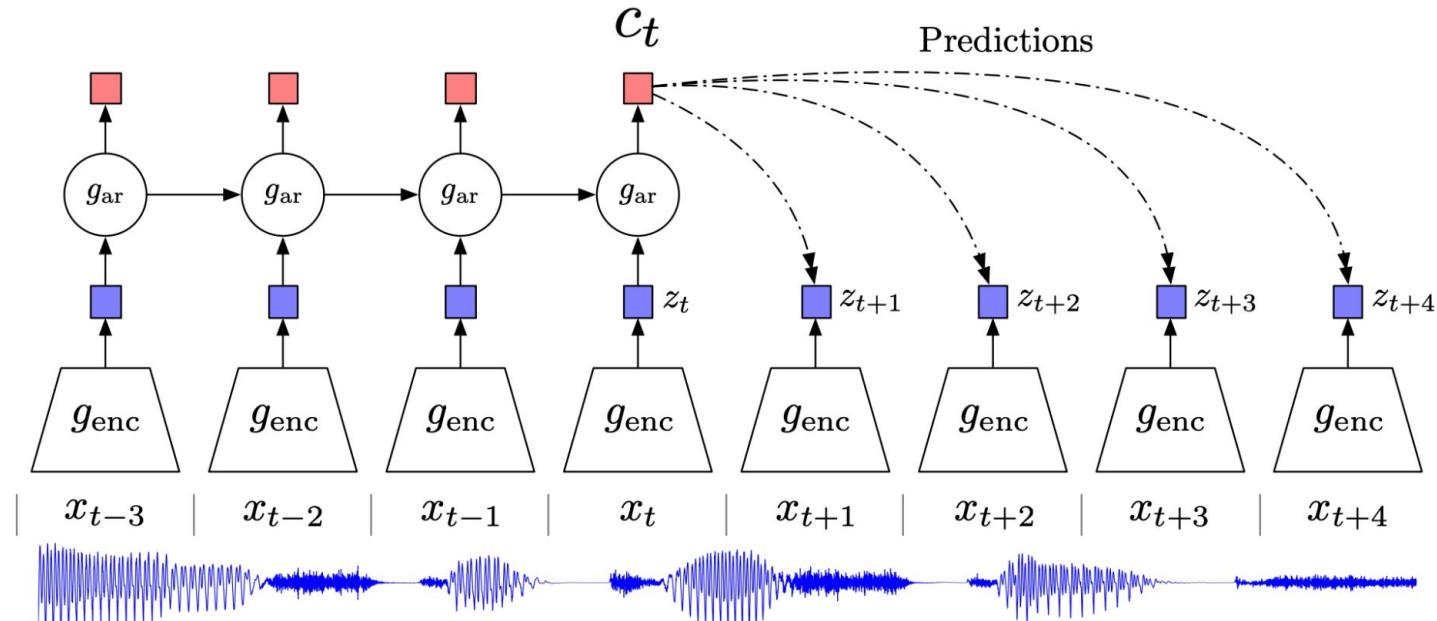
$$s_k(z_{t+k}, c_t) = z_{t+k}^T W_k c_t$$

, where W_k is a trainable matrix.

Figure [source](#)

Source: [van den Oord et al., 2018](#),

CPC example: modeling audio sequences



Source: [van den Oord et al., 2018](#),

CPC example: modeling audio sequences

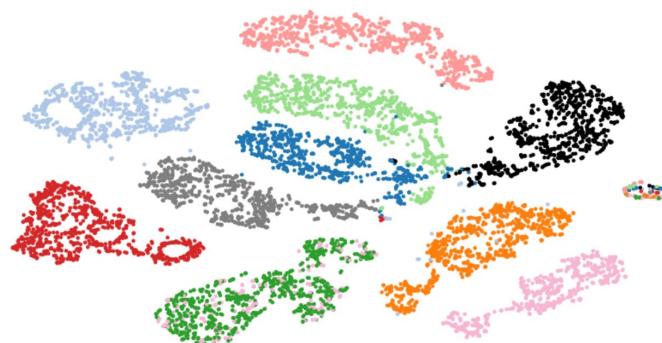


Figure 2: t-SNE visualization of audio (speech) representations for a subset of 10 speakers (out of 251). Every color represents a different speaker.

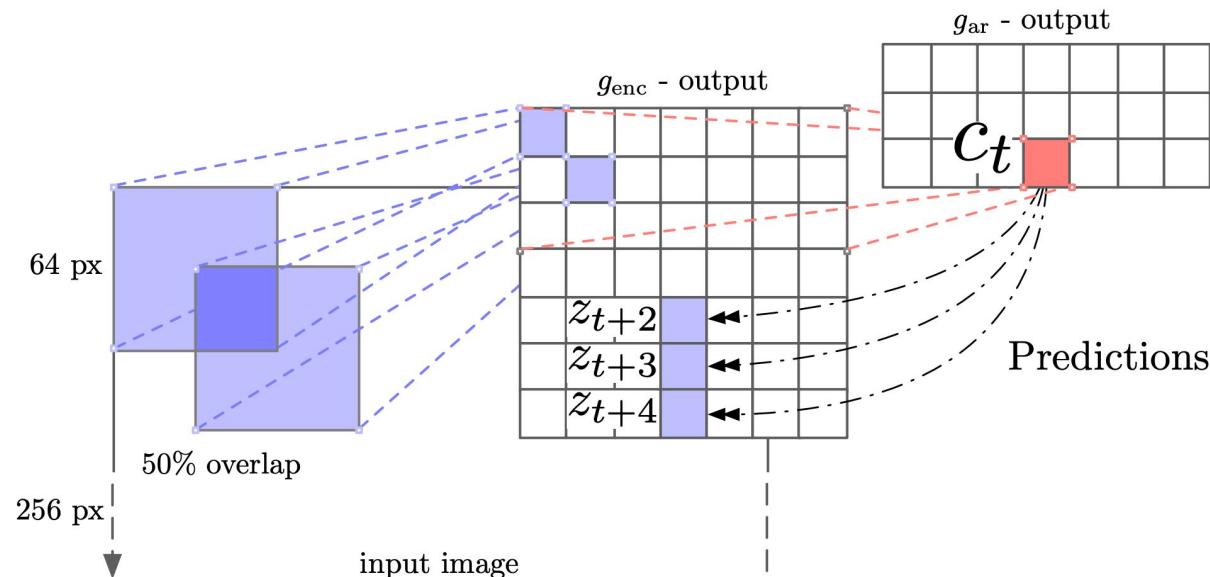
Method	ACC
Phone classification	
Random initialization	27.6
MFCC features	39.7
CPC	64.6
Supervised	74.6
Speaker classification	
Random initialization	1.87
MFCC features	17.6
CPC	97.4
Supervised	98.5

Linear classification on trained representations (LibriSpeech dataset)

Source: [van den Oord et al., 2018](#),

CPC example: modeling visual context

Idea: split image into patches, model rows of patches from top to bottom as a sequence. I.e., use top rows as context to predict bottom rows.



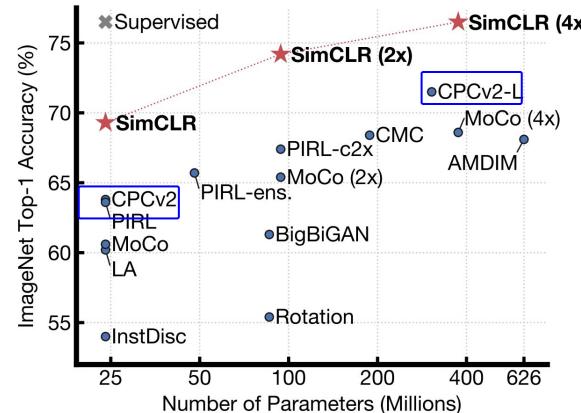
Source: [van den Oord et al., 2018](#),

CPC example: modeling visual context

Method	Top-1 ACC
Using AlexNet conv5	
Video [28]	29.8
Relative Position [11]	30.4
BiGan [35]	34.8
Colorization [10]	35.2
Jigsaw [29] *	38.1
Using ResNet-V2	
Motion Segmentation [36]	27.6
Exemplar [36]	31.5
Relative Position [36]	36.2
Colorization [36]	39.6
CPC	48.7

Table 3: ImageNet top-1 unsupervised classification results. *Jigsaw is not directly comparable to the other AlexNet results because of architectural differences.

- Compares favorably with other pretext task-based self-supervised learning method.
- Doesn't do as well compared to newer instance-based contrastive learning methods on image feature learning.



Source: [van den Oord et al., 2018](#),

Summary: Contrastive Representation Learning

A general formulation for contrastive learning:

$$\text{score}(f(x), f(x^+)) \gg \text{score}(f(x), f(x^-))$$

InfoNCE loss: N-way classification among positive and negative samples

$$L = -\mathbb{E}_X \left[\log \frac{\exp(s(f(x), f(x^+)))}{\exp(s(f(x), f(x^+))) + \sum_{j=1}^{N-1} \exp(s(f(x), f(x_j^-)))} \right]$$

Commonly known as the InfoNCE loss ([van den Oord et al., 2018](#))

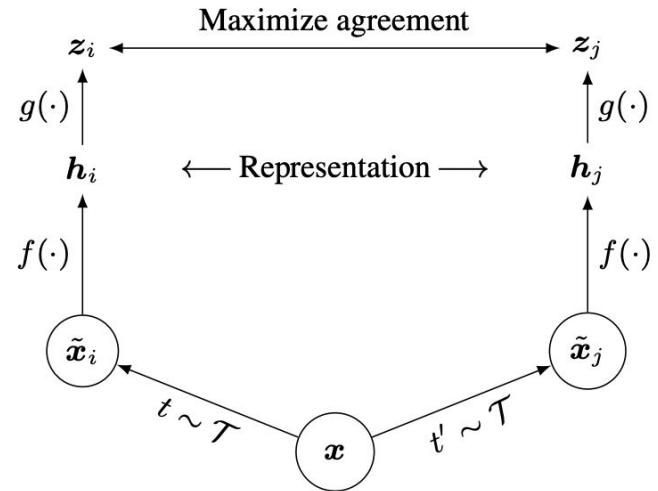
A *lower bound* on the mutual information between $f(x)$ and $f(x^+)$

$$MI[f(x), f(x^+)] - \log(N) \geq -L$$

Summary: Contrastive Representation Learning

SimCLR: a simple framework for contrastive representation learning

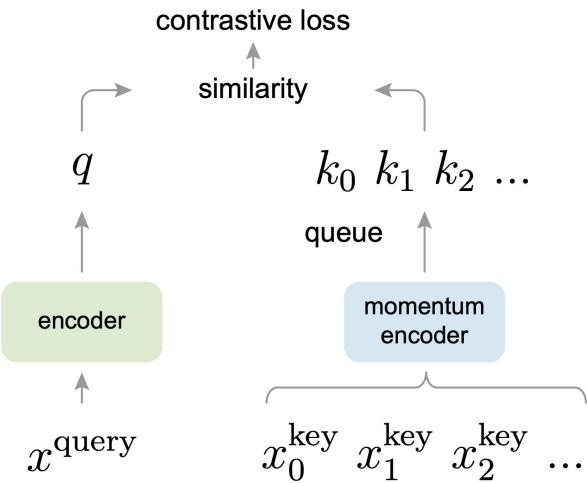
- **Key ideas**: non-linear projection head to allow flexible representation learning
- Simple to implement, effective in learning visual representation
- Requires large training batch size to be effective; large memory footprint



Summary: Contrastive Representation Learning

MoCo (v1, v2): contrastive learning using momentum sample encoder

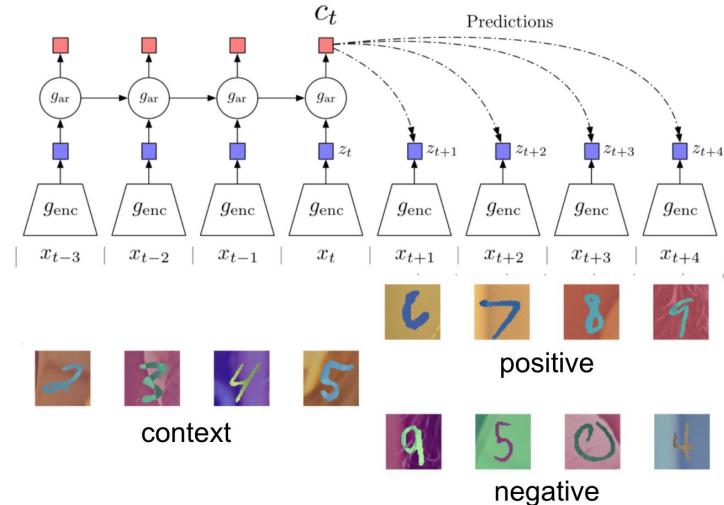
- Decouples negative sample size from minibatch size; allows large batch training without TPU
- MoCo-v2 combines the key ideas from SimCLR, i.e., nonlinear projection head, strong data augmentation, with momentum contrastive learning



Summary: Contrastive Representation Learning

CPC: sequence-level contrastive learning

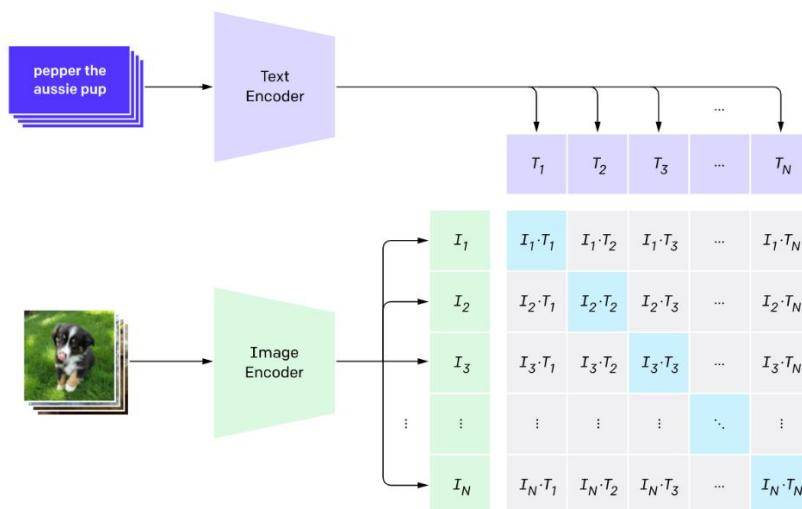
- Contrast “right” sequence with “wrong” sequence.
- InfoNCE loss with a time-dependent score function.
- Can be applied to a variety of learning problems, but not as effective in learning image representations compared to instance-level methods.



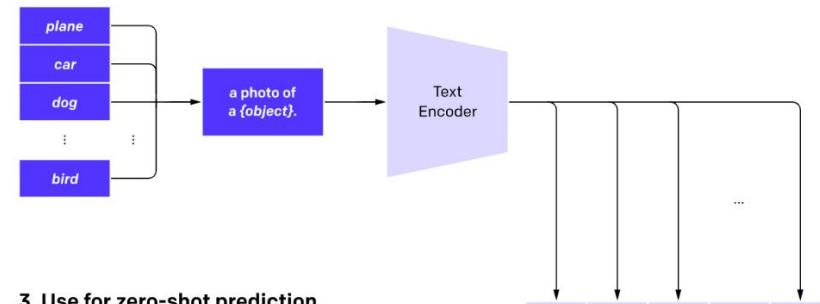
Other examples

Contrastive learning between image and natural language sentences

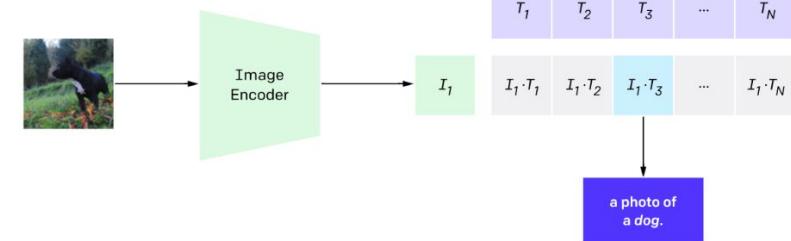
1. Contrastive pre-training



2. Create dataset classifier from label text



3. Use for zero-shot prediction

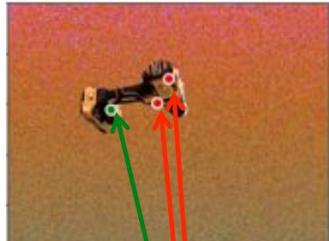


CLIP (*Contrastive Language–Image Pre-training*) Radford *et al.*, 2021

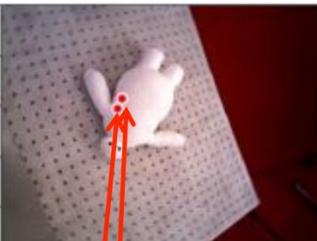
Other examples

Contrastive learning on pixel-wise feature descriptors

(c) Background Randomization



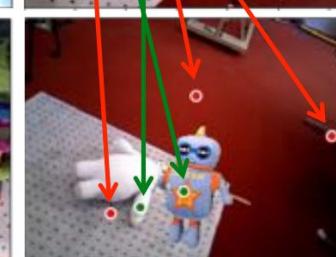
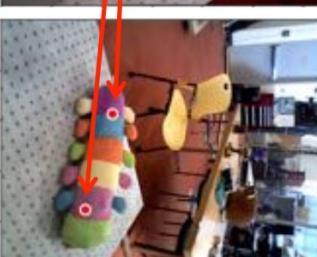
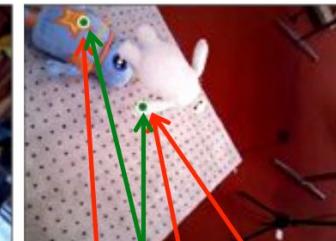
(d) Cross Object Loss



(e) Direct Multi Object

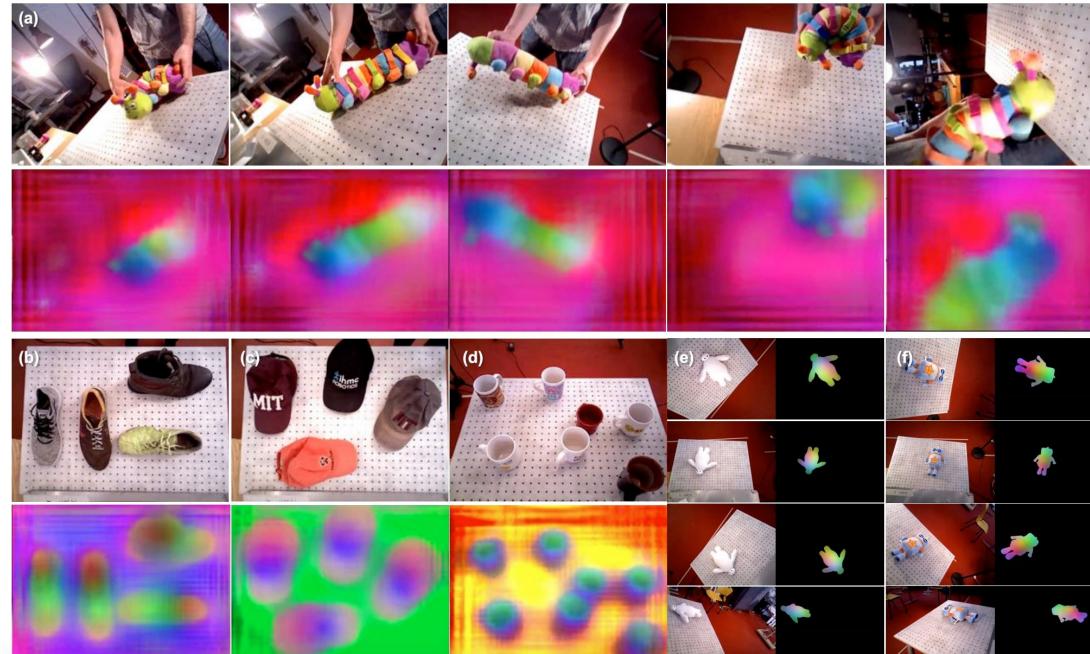


(f) Synthetic Multi Object



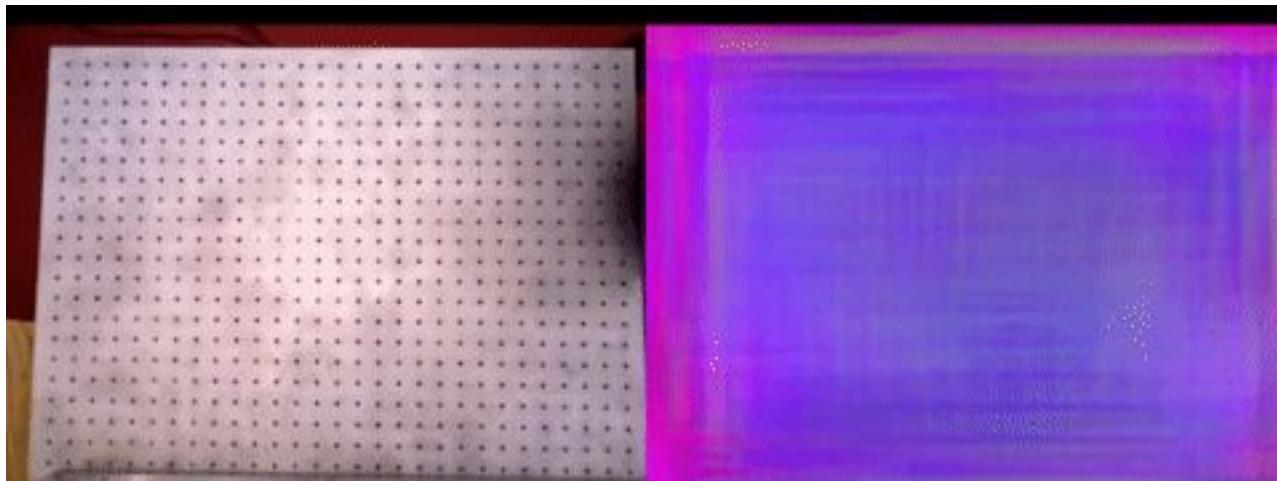
Dense Object Net, Florence et al., 2018

Other examples



Dense Object Net, Florence et al., 2018

Other examples



Dense Object Net, Florence et al., 2018

Next time: **Low-Level Vision**

Today's Agenda

Pretext tasks from image transformations

- Rotation, inpainting, rearrangement, coloring

Contrastive representation learning

- Intuition and formulation
- Instance contrastive learning: SimCLR and MOCO
- Sequence contrastive learning: CPC

Frontier:

- Contrastive Language Image Pre-training (CLIP)

Frontier: Contrastive Language–Image Pre-training (CLIP)

Self-Supervised Learning

General idea: pretend there is a part of the data you don't know and train the neural network to predict that.

Y. LeCun

Self-Supervised Learning

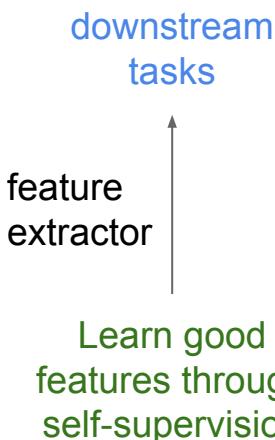
- ▶ Predict any part of the input from any other part.
- ▶ Predict the **future** from the **past**.
- ▶ Predict the **future** from the **recent past**.
- ▶ Predict the **past** from the **present**.
- ▶ Predict the **top** from the **bottom**.
- ▶ Predict the **occluded** from the **visible**
- ▶ **Pretend there is a part of the input you don't know and predict that.**

Time → ← Past Present Future →

“The Cake of Learning”

How Much Information is the Machine Given during Learning?

Y. LeCun



- ▶ “Pure” Reinforcement Learning (**cherry**)
 - ▶ The machine predicts a scalar reward given once in a while.
 - ▶ **A few bits for some samples**

- ▶ Supervised Learning (**icing**)
 - ▶ The machine predicts a category or a few numbers for each input
 - ▶ Predicting human-supplied data
 - ▶ **10→10,000 bits per sample**



- ▶ Self-Supervised Learning (**cake génoise**)
 - ▶ The machine predicts any part of its input for any observed part.
 - ▶ Predicts future frames in videos
 - ▶ **Millions of bits per sample**

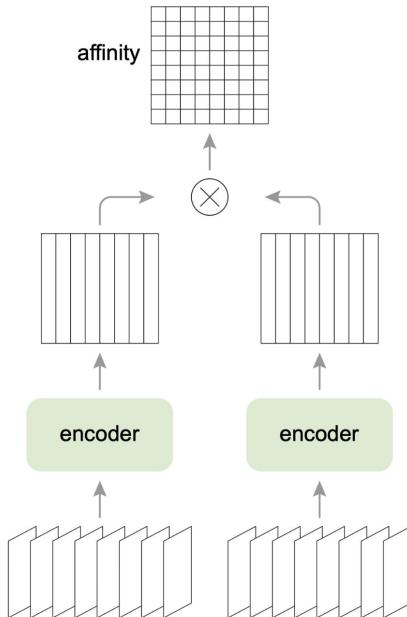
© 2019 IEEE International Solid-State Circuits Conference

1.1: Deep Learning Hardware: Past, Present, & Future

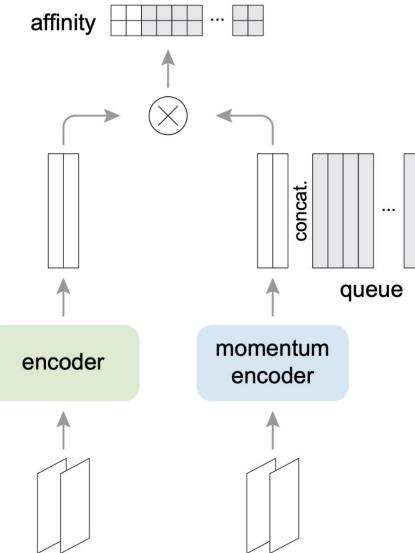
59

Source: Lecun 2019 Keynote at ISSCC

Can we do better?



SimCLR



**Momentum Contrast
(MoCo)**

Source: [Chen et al., 2020b](#)

CS231N: Low-Level Vision

Jia Deng

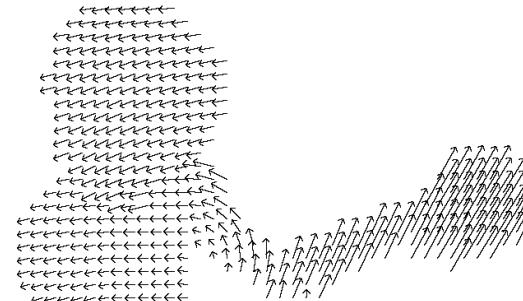
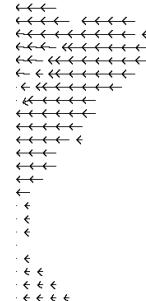
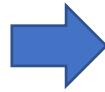
Optical Flow

- Predict per-pixel 2D motion between a pair of frames

Frame 1



Frame 2



Applications

Robotics



Self-driving cars (Waymo)



Everydayrobots.com

AR/VR



Project starline (Google)



Hololens (Microsoft)

Robotics

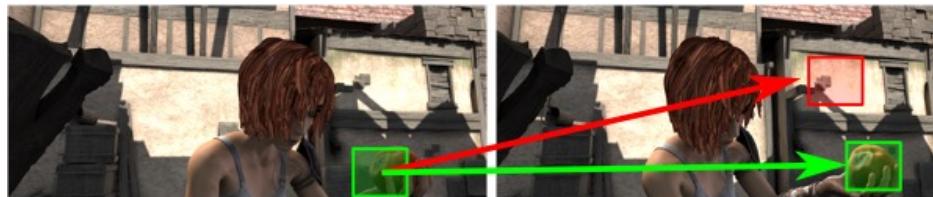
3D Vision

Graphics

Optical Flow as Optimization

- Objective: appearance constancy + plausibility of flow field

$$E(\Delta x) = \text{Distance}(I(x_i), I(x_i + \Delta x_i)) + \text{Regularization}(I, \Delta x)$$



[Horn and Schunck, 1981]
[Black and Anandan, 1993]
[Zach et al. 2007]

[Brox et al. 2004]
[Brox and Malik, 2010]
[Weinzaepfel et al, 2013]

[Liu et al. 2009]
[Roth et al. 2009]
[Menze et al, 2015]
[Sun et al, 2010]

[Bailer et al. 2015]
[Chen and Koltun, 2016]
[Xu et al, 2017]

Optical Flow

- Classical approaches:

The Model of Horn and Schunck [1]

$$\min_{u,v} \left\{ E = \underbrace{\int_{\Omega} |\nabla u|^2 + |\nabla v|^2 d^2x}_{\text{Regularization Term}} + \lambda \underbrace{\int_{\Omega} \rho(u,v)^2 d^2x}_{\text{Data Term (OFC)}} \right\}$$

- + Convex $\rho(u,v) = I_t + (u,v) \cdot \nabla I \approx 0$
- + Easy to solve
- Does not allow for sharp edges in the solution
- Sensitive to outliers violating the OFC

[1] Horn and Schunck. Determining Optical Flow. Artificial Intelligence, 1981

Optical Flow

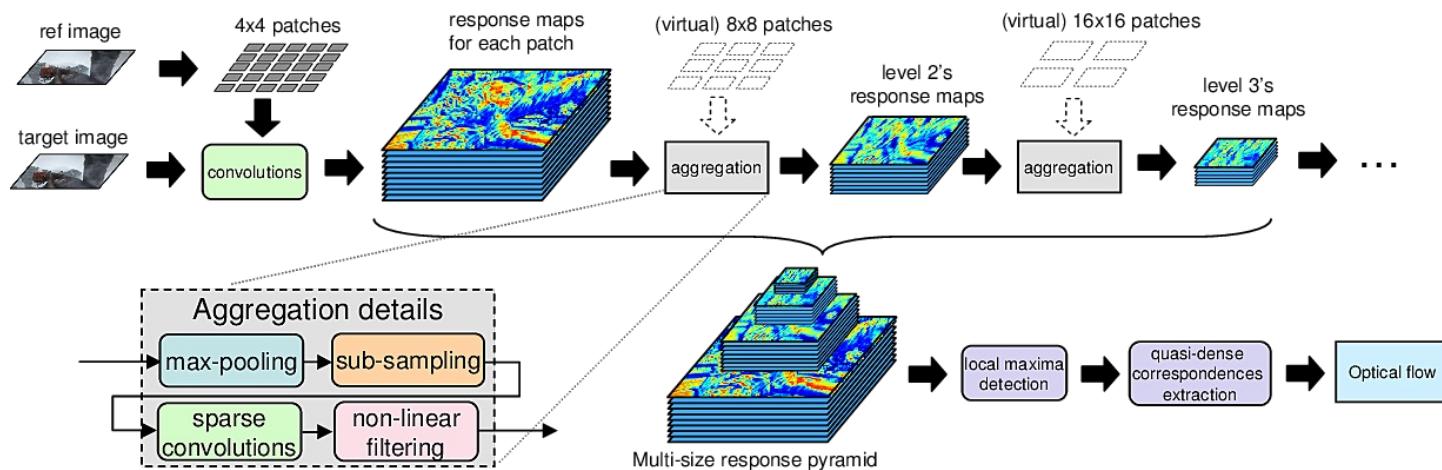
- Classical approaches: TV-L1 Flow (TV: total variation)
 - Replace quadratic functions by L_1 – norms
 - Done by Cohen, Aubert, Brox, Bruhn, ...

$$\min_{u,v} \left\{ E = \int_{\Omega} |\nabla u| + |\nabla v| \, d^2x + \lambda \int_{\Omega} |\rho(u,v)| \, d^2x \right\}$$

- +Allows for discontinuities in the flow field
- +Robust to some extent to outliers in the OFC
- +Still convex
- Much harder to solve

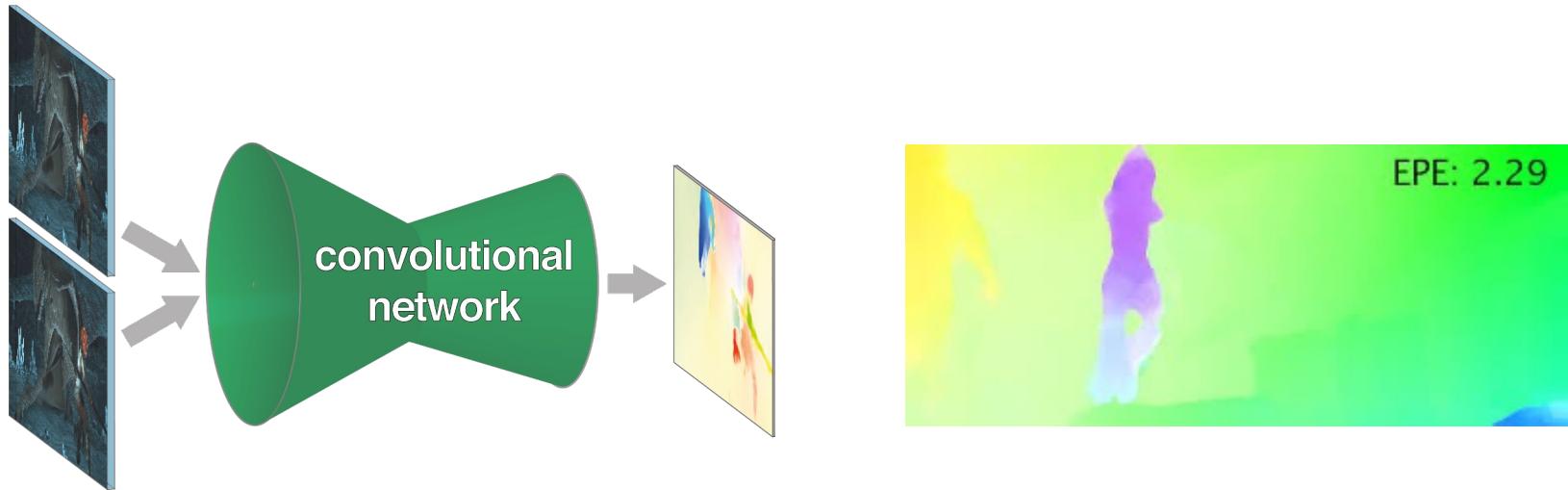
Optical Flow

- Classical approaches: DeepFlow



FlowNet [Dosovitskiy et al. 2015]

- First optical flow network
- U-Net on concatenated frames
- Simple and Fast -- but underperforming the best optimization approaches

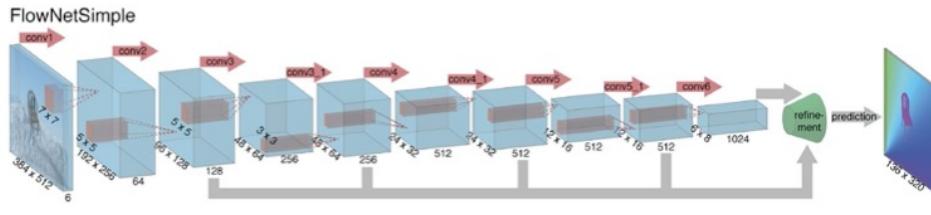


Optical Flow

- Deep Learning: FlowNet

FlowNet S (Simple) architecture

- Input: two **stacked** images (**[image(t), image(t-1)]**)
- **Encode**: 9 Convolutional layers (strides: 2)
 - conv 7*7: 1 layers
 - conv 5*5: 2 layers
 - conv 3*3: 6 layers
- **Decode**: Refinement layers (described later)



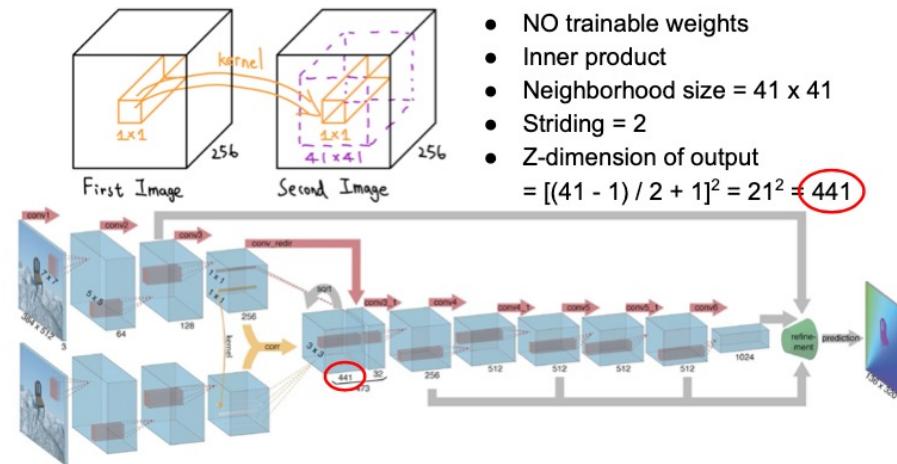
Slide credit: K-Inoue @ki42 & Oscar @wang

Optical Flow

- Deep Learning: FlowNet

FlowNet C (Correlation) architecture

- Input: two images ($[image(t), image(t-1)]$) Kernel-like processing
 - Correlation layer calculating “correlation” of two images



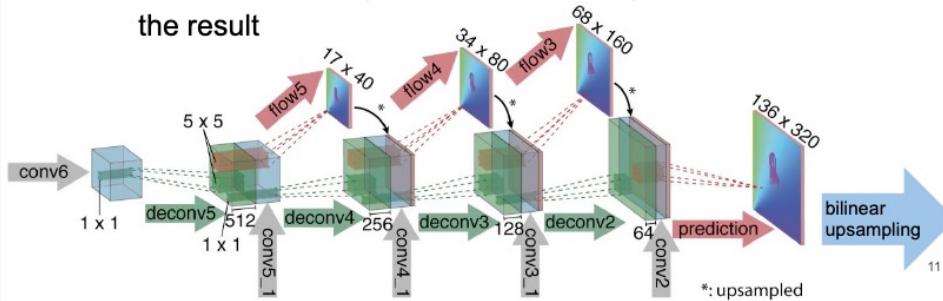
Slide credit: K-Inoue @ki42 &
Oscar @wang

Optical Flow

- Deep Learning: FlowNet

Refinement layers in FlowNet S/C

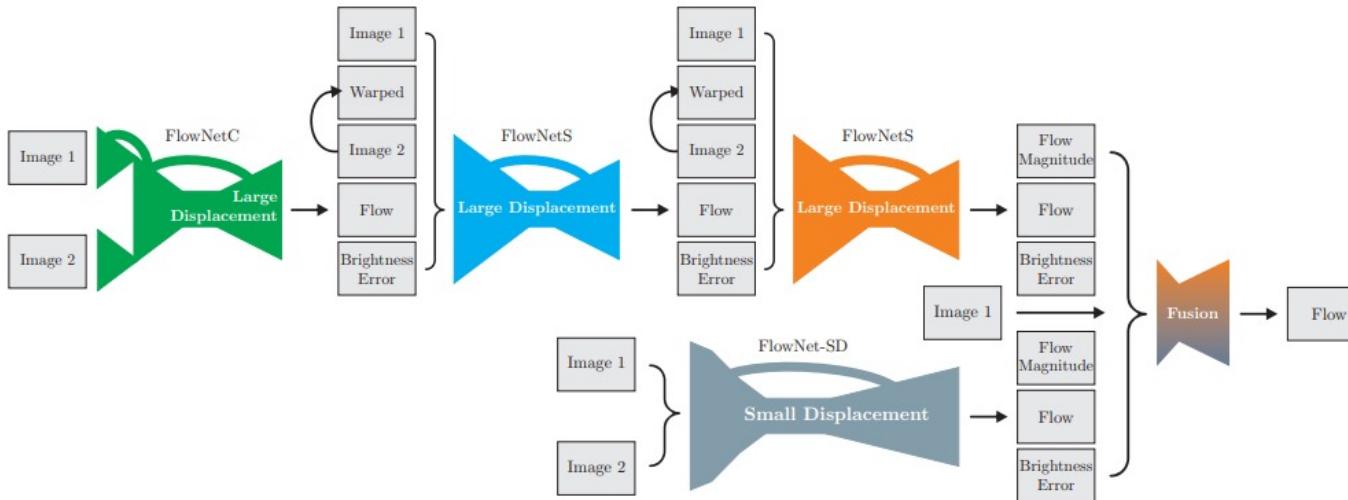
1. **4 De-convolution layers & 4 Upsampled prediction layers**
 - De-convolution: Transposed convolution + LeakyReLU
 - Upsampled prediction: Transposed convolution (evaluated)
 - **De-conv** + Previous feature map + **Upsampled prediction**
2. **Bilinear upsampling (4x)**
 - Cheaper & Adding more refinement layers did not improve the result



Slide credit: K-Inoue @ki42 &
Oscar @wang

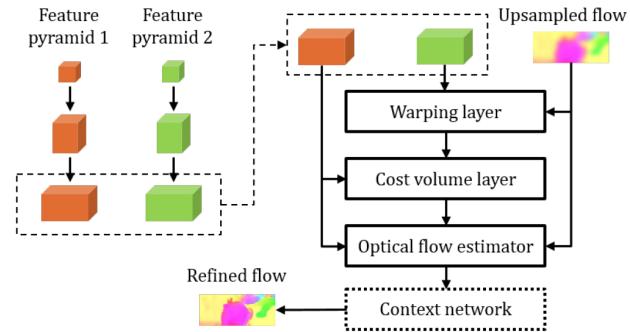
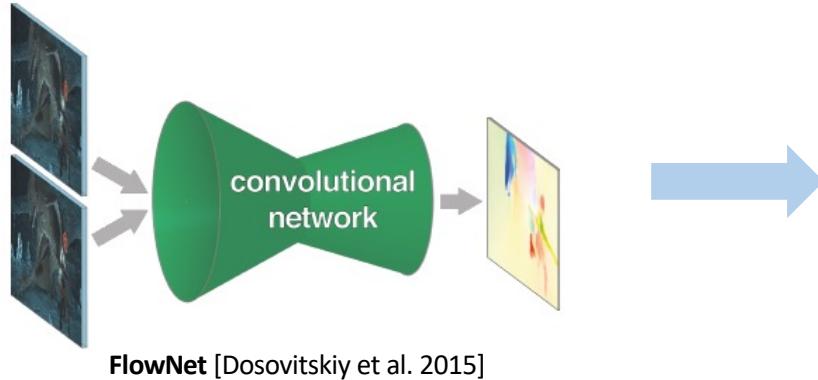
Optical Flow

- Deep Learning: FlowNet 2.0



Ilg E, Mayer N, Saikia T, Keuper M, Dosovitskiy A, Brox T. Flownet 2.0: Evolution of optical flow estimation with deep networks. In Proceedings of the IEEE conference on computer vision and pattern recognition 2017 (pp. 2462-2470).

Deep Learning and Optical Flow



- Inductive bias: warping, cost volume
- Iterative refinement limited to pyramid levels

[Ranjan and Black, 2017]
[Ilg et al., 2017]
[Hui et al, 2018]

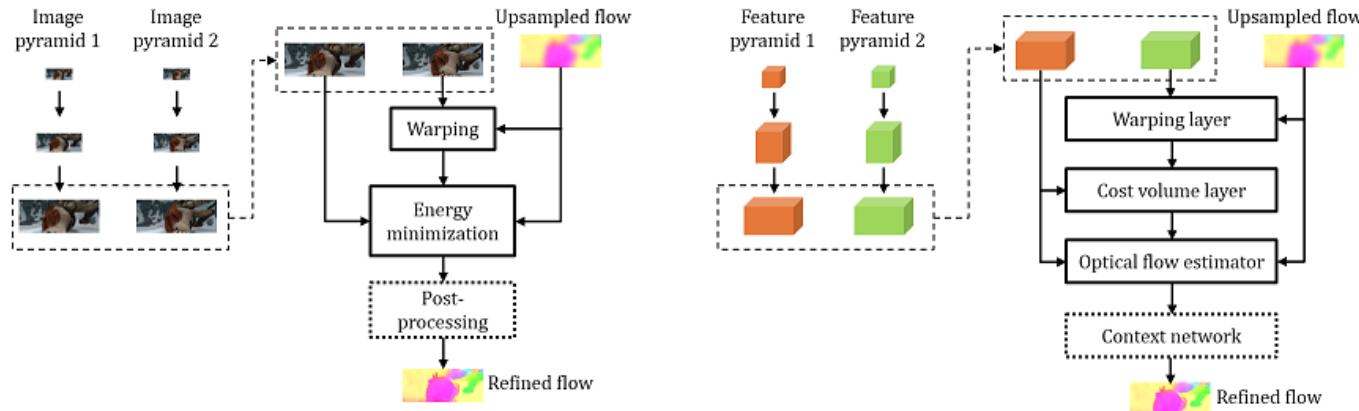
[Maurer and Bruhn, 2018]
[Ilg et al., 2017]
[Neoral et al, 2018]

[Bar-Haim and Wolf, 2020]
[Zhao et al, 2020]
[Z Yin et al, 2019]

[Yang and Ramanan, 2018]
[Lu et al, 2020]

Optical Flow

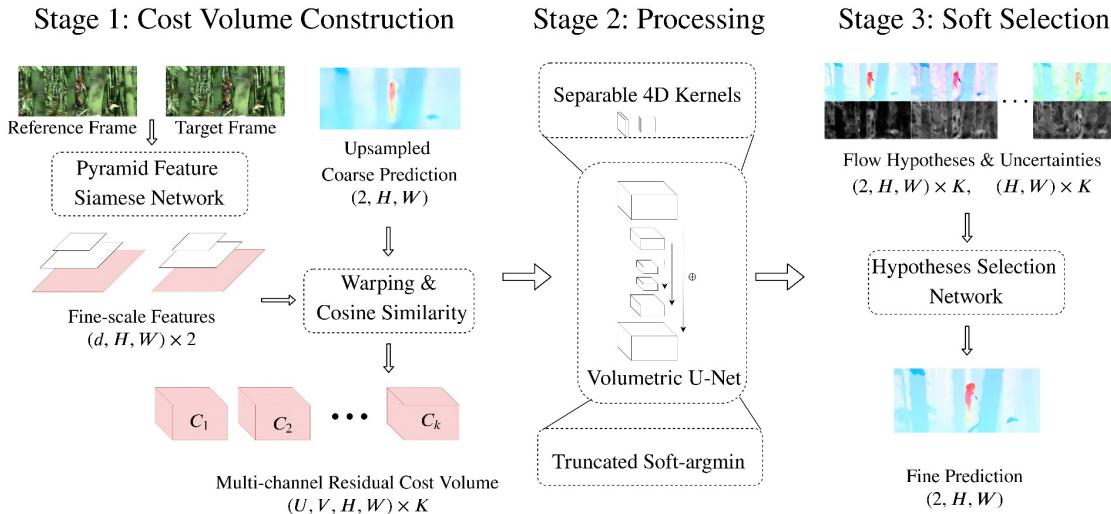
- Deep Learning: PWC-Net



Sun D, Yang X, Liu MY, Kautz J. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2018 (pp. 8934-8943).

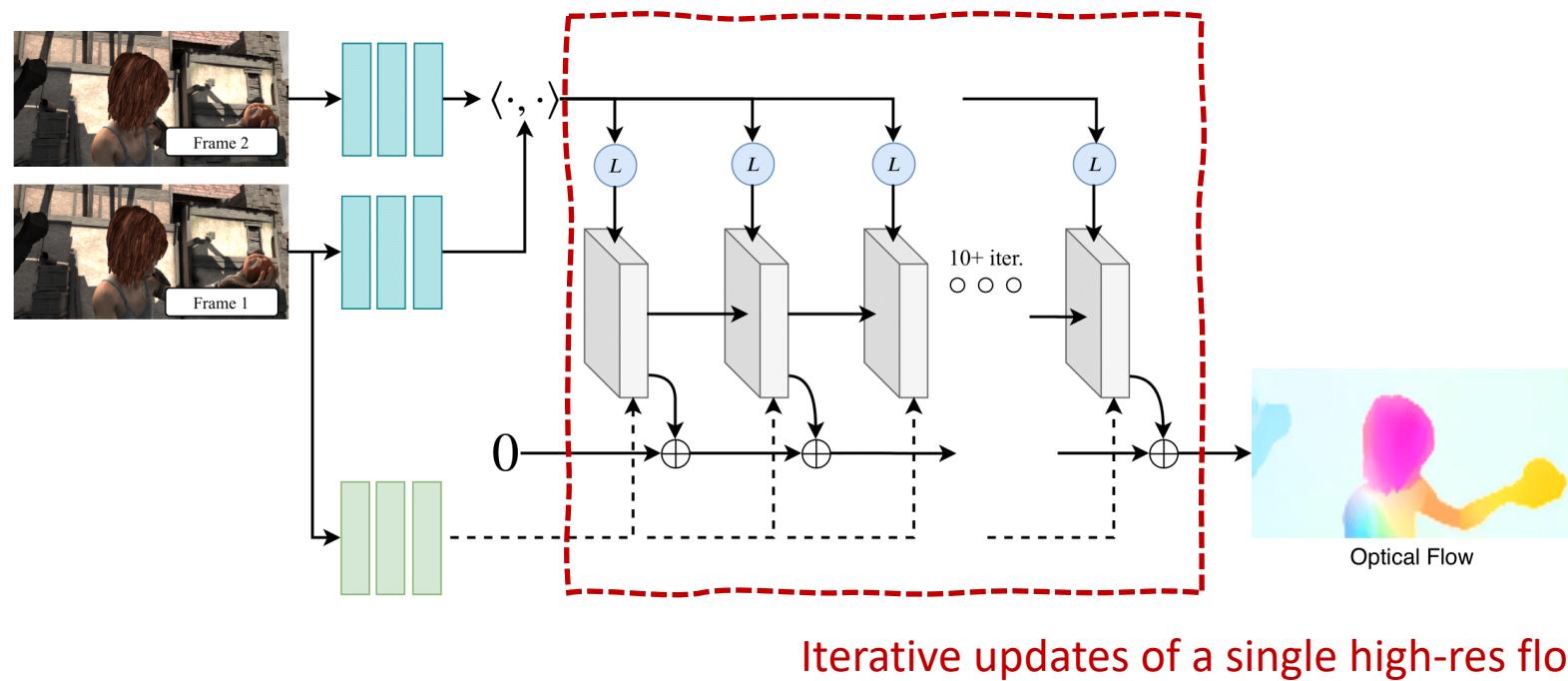
Optical Flow

- Deep Learning: VCN



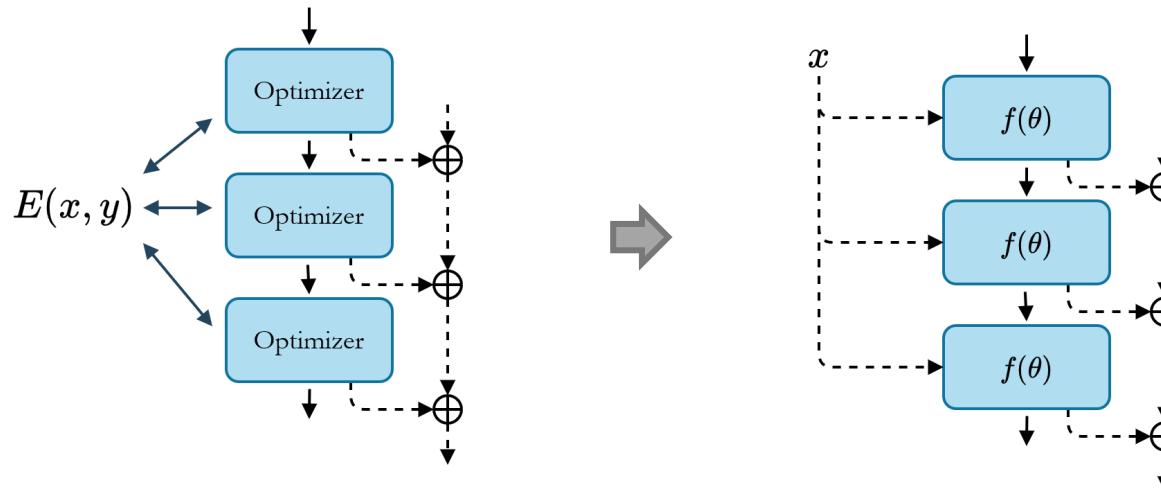
Yang G, Ramanan D. Volumetric Correspondence Networks for Optical Flow. In Advances in Neural Information Processing Systems 2019 (pp. 793-803).

RAFT: Recurrent All-Pairs Field Transforms



Strategy: Optimization-Inspired Neural Architectures

Design neural networks to behave like classical optimization algorithms



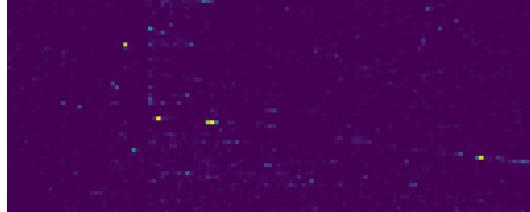
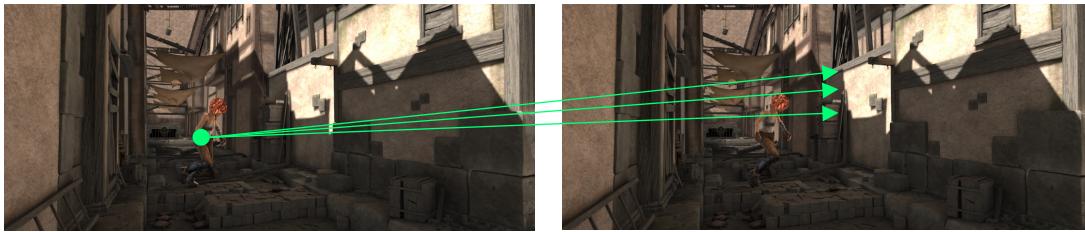
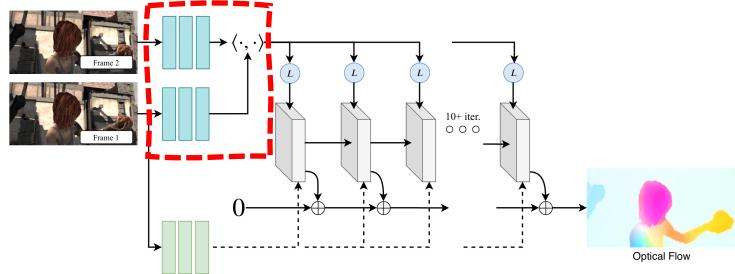
+ Recurrent iterative updates

RAFT: Recurrent All-Pairs Field Transforms

- *State-of-the-art accuracy:* **16%** better on KITTI, **30%** better on Sintel
- *High efficiency:* **10x** faster training, **10fps** on 436x1088 video
- *Strong Generalization:* **40%** better synthetic to real generalization

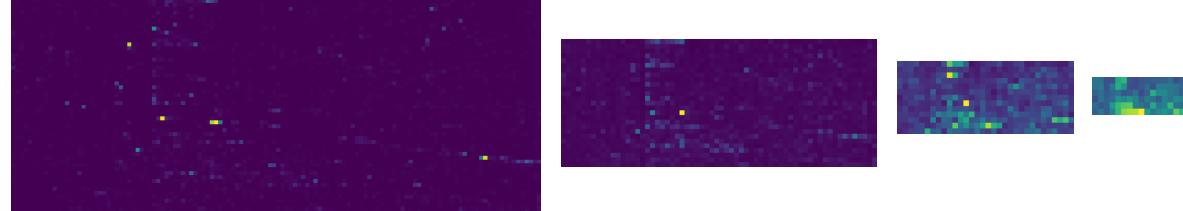
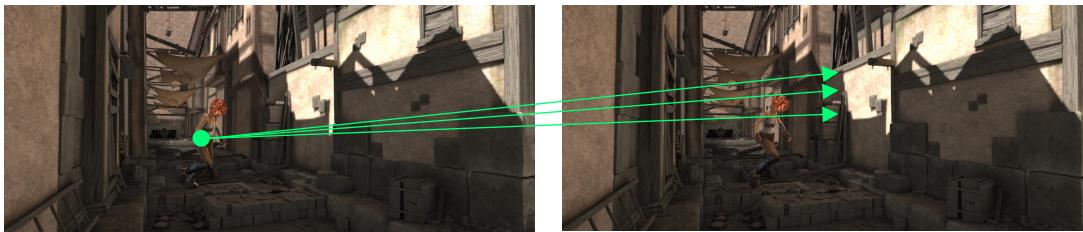
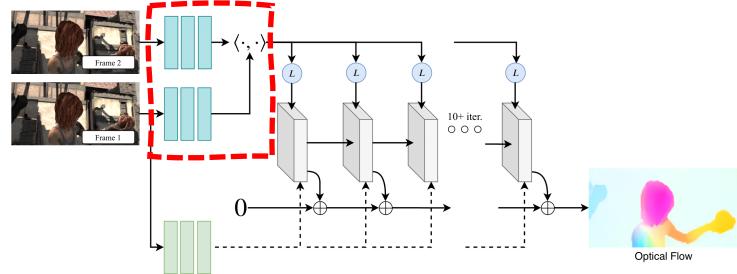
All-Pairs Visual Similarities

- Dot product between all pairs



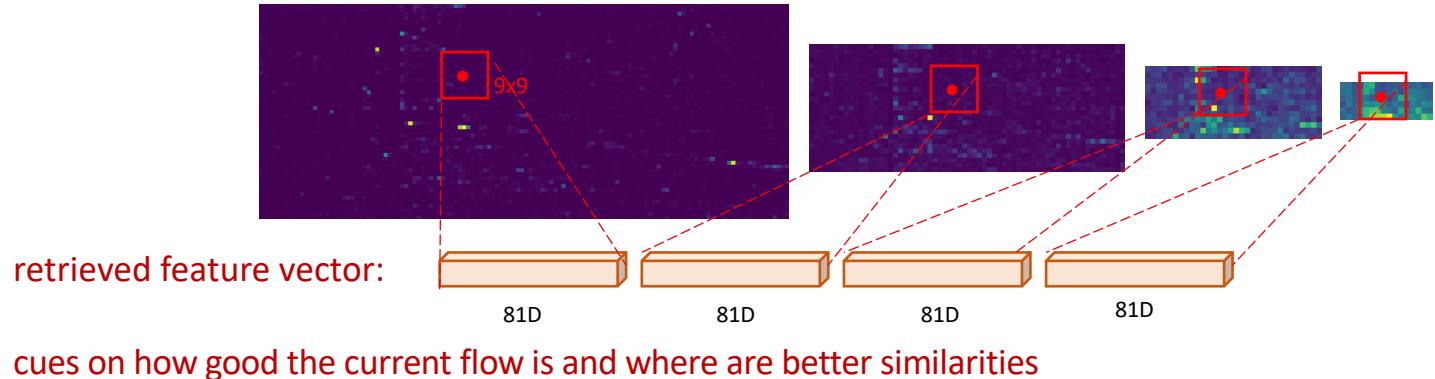
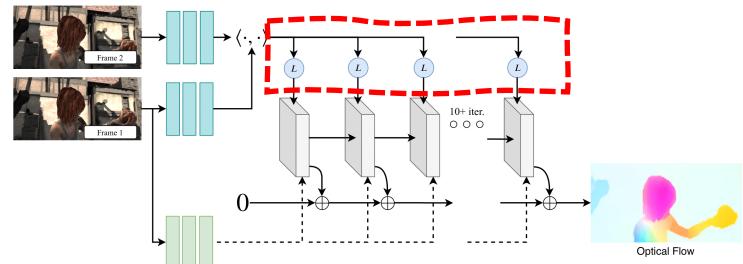
All-Pairs Visual Similarities

- Dot product between all pairs
- Repeated pooling of last two dimensions



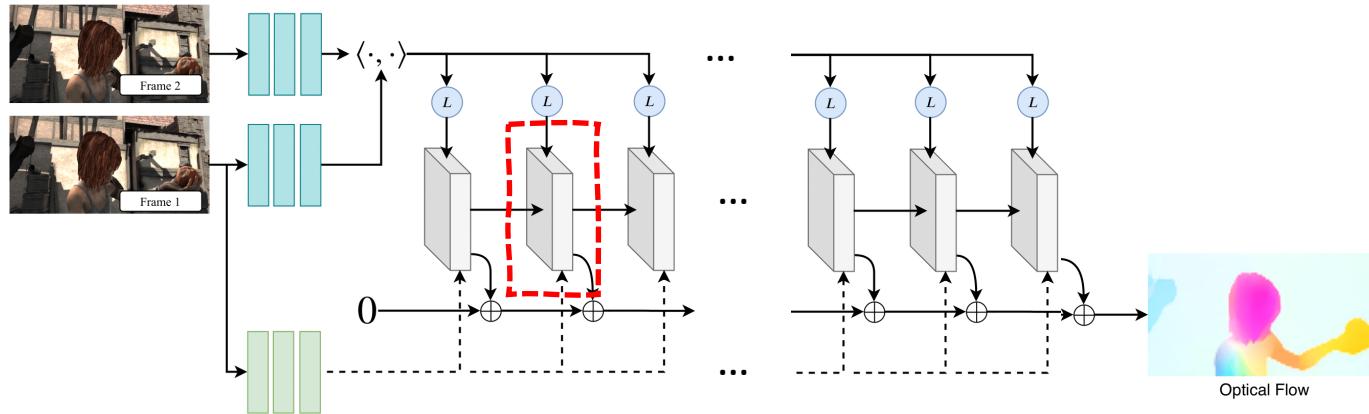
All-Pairs Visual Similarities

- Dot product between all pairs
- Repeated pooling of last two dimensions
- Use current flow estimate to retrieve a feature vector



Update Operator

- GRU-Based recurrent update operator

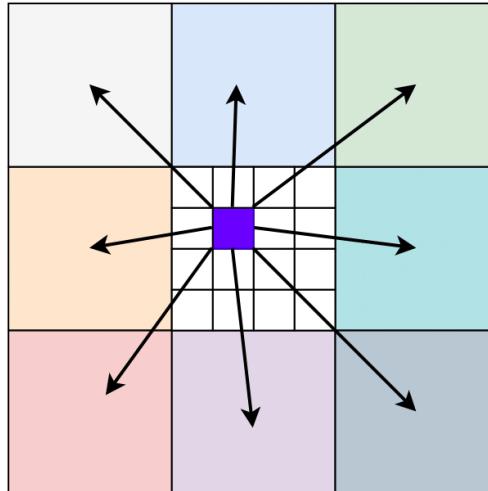


- Designed to mimic updates of first order optimization algorithm [1]
- But no explicit objective or gradient

[1] Adler, Jonas, and Ozan Öktem. "Learned primal-dual reconstruction." 2018

Convex Upsampling

- Upsamples flow to **full resolution**
- Convex combination of 3x3 coarse resolution neighbors



$$\text{purple square} = w_1 \text{ (light gray square)} \oplus w_2 \text{ (light blue square)} \oplus w_3 \text{ (light green square)} \oplus$$

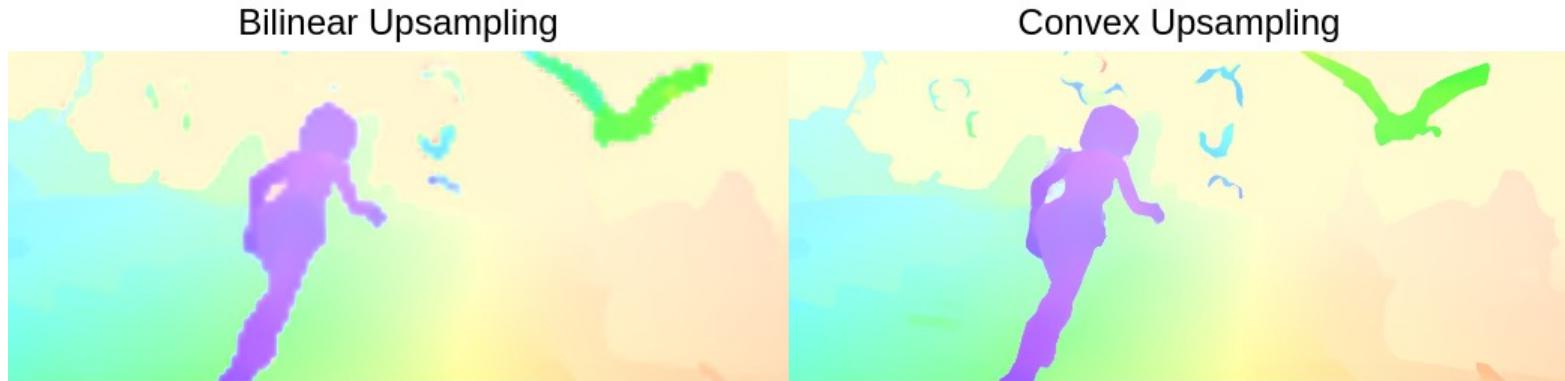
$$w_4 \text{ (orange square)} \oplus w_5 \text{ (light blue square)} \oplus w_6 \text{ (cyan square)} \oplus$$

$$w_7 \text{ (pink square)} \oplus w_8 \text{ (purple square)} \oplus w_9 \text{ (gray square)}$$

Coefficients Predicted by Network (w_1, \dots, w_9)

Convex Upsampling

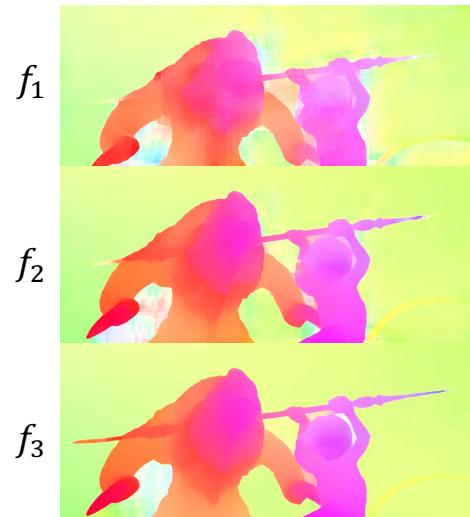
- Upsamples flow to **full resolution**
- Convex combination of 3x3 coarse resolution neighbors



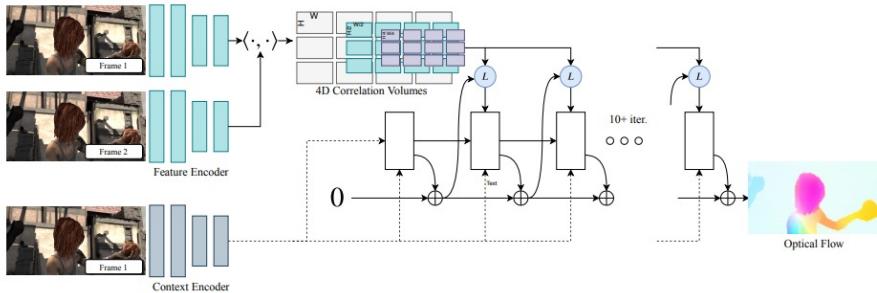
Training

- Supervised directly on sequence of full resolution flow fields

$$Loss = \sum_i^N \frac{1.25^i}{1.25^N} \left\| f_{gt} - f_i \right\|_1$$

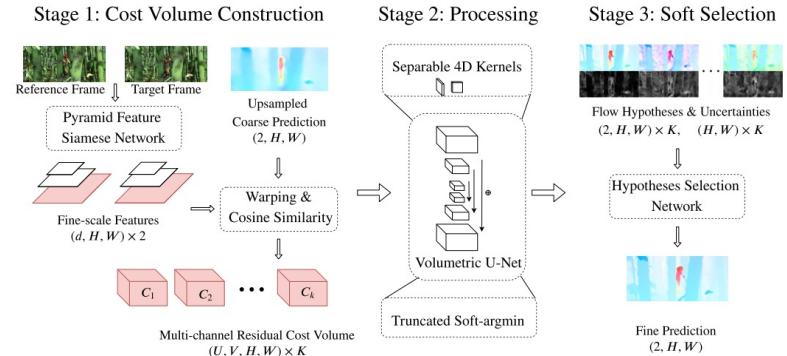


RAFT versus VCN



RAFT [Teed & Deng, 2020]

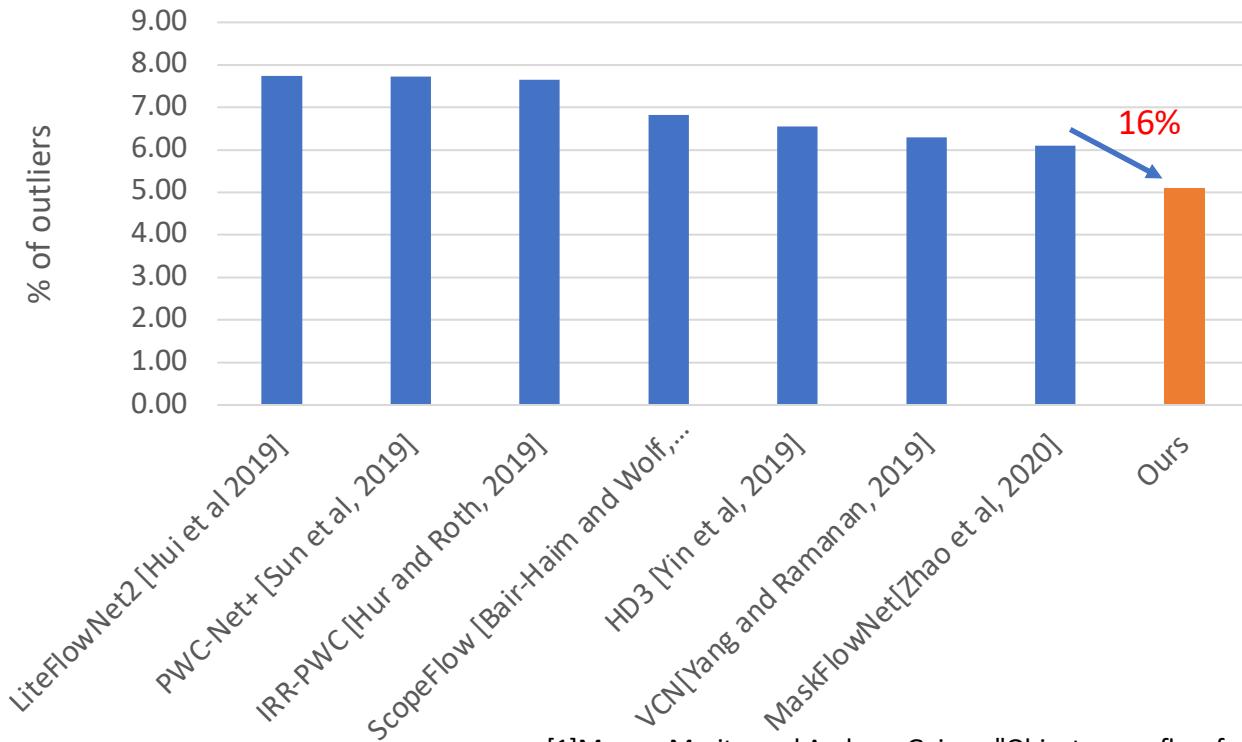
- Construct 4D cost volume
- **2D convolution on slices of cost volume**



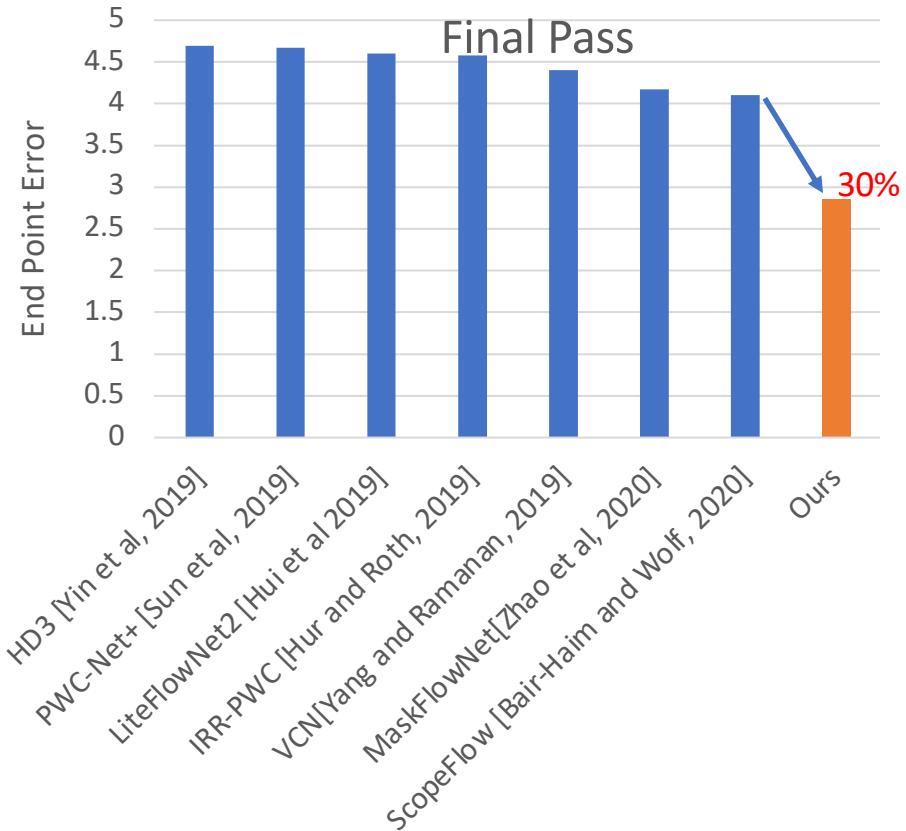
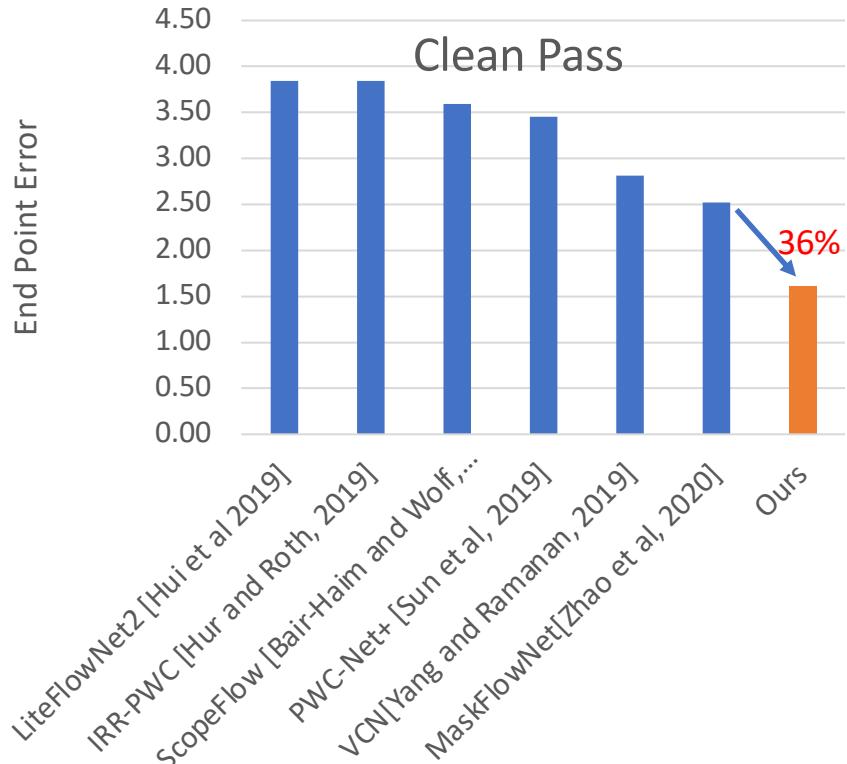
VCN [Yang & Ramanan, 2019]

- Construct 4D cost volume
- **4D convolution on entire cost volume**

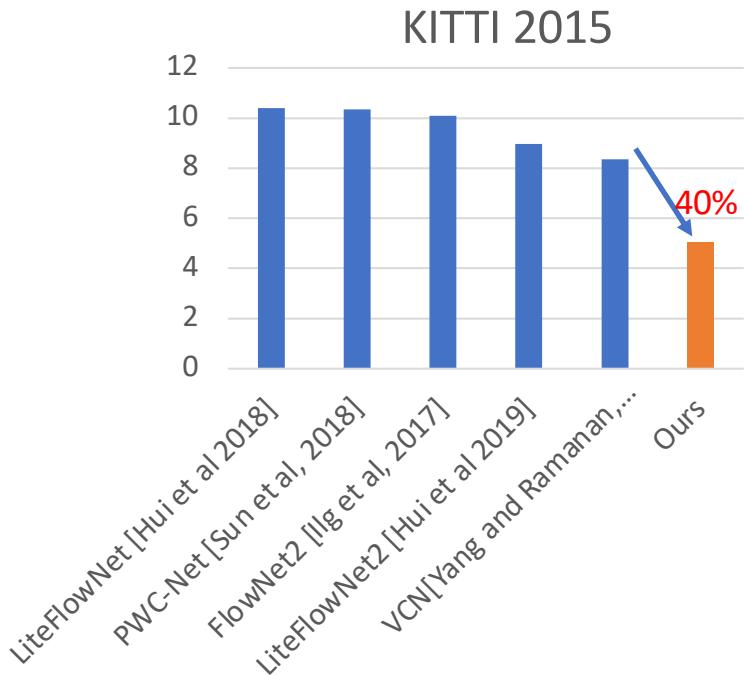
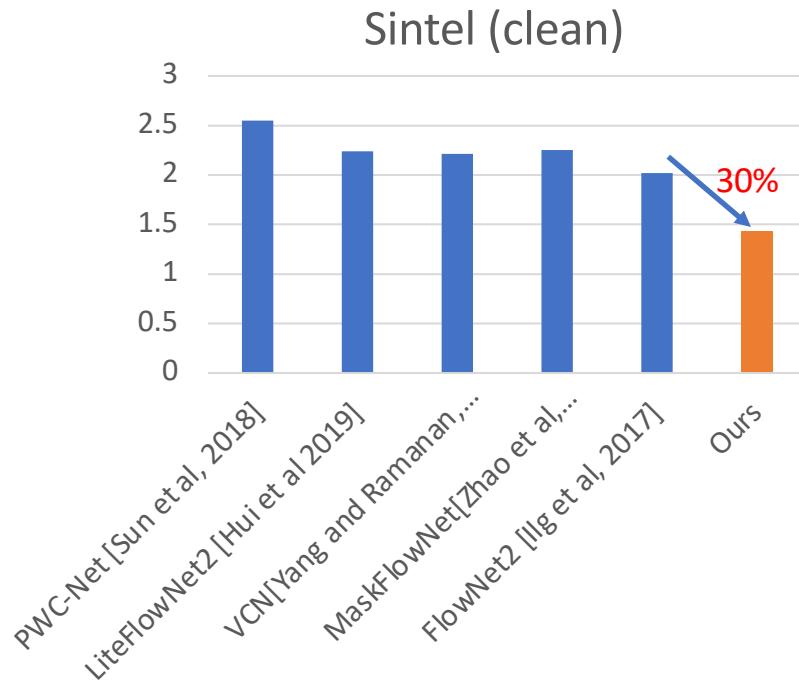
KITTI-2015[1] Results



Sintel Results

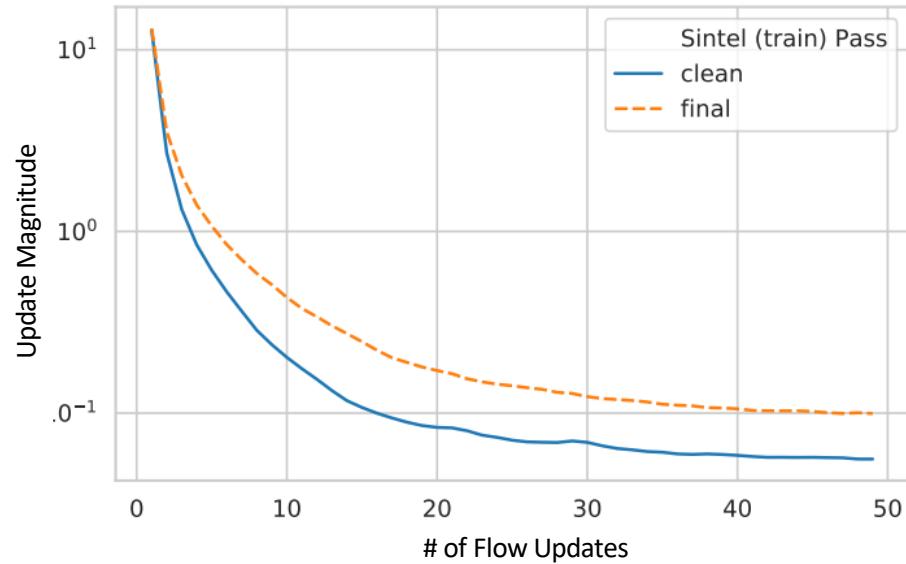
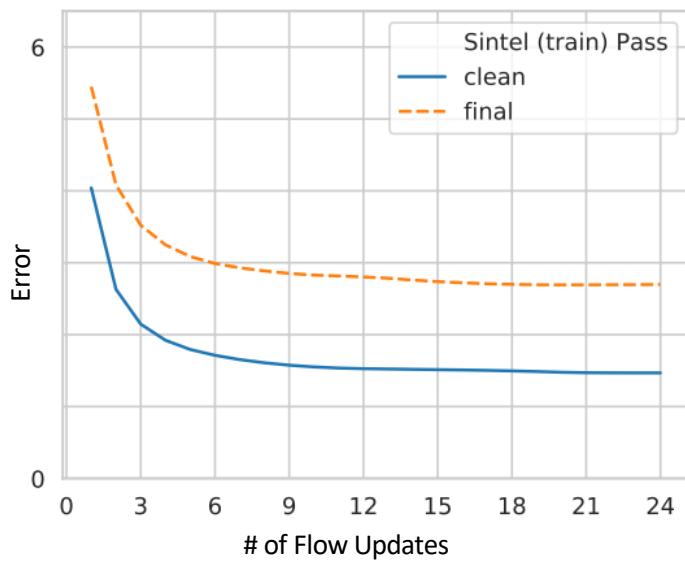


Cross-Dataset Generalization



Models trained on **FlyingChairs** (Fischer et al. 2015) and **FlyingThings3D** (Mayer et al, 2016)

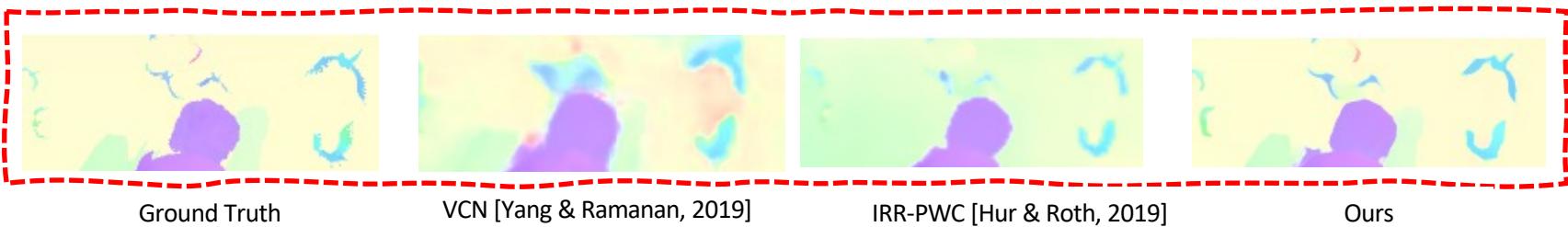
Convergence



Convergence Visualized



RAFT can recover the motion of small, fast moving objects

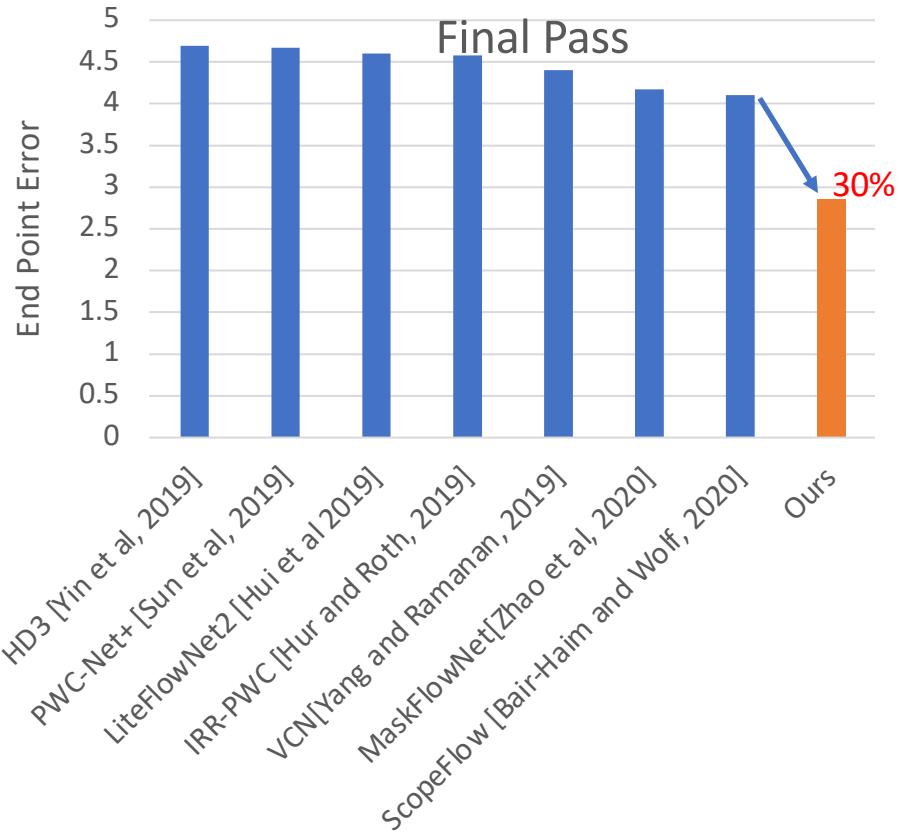
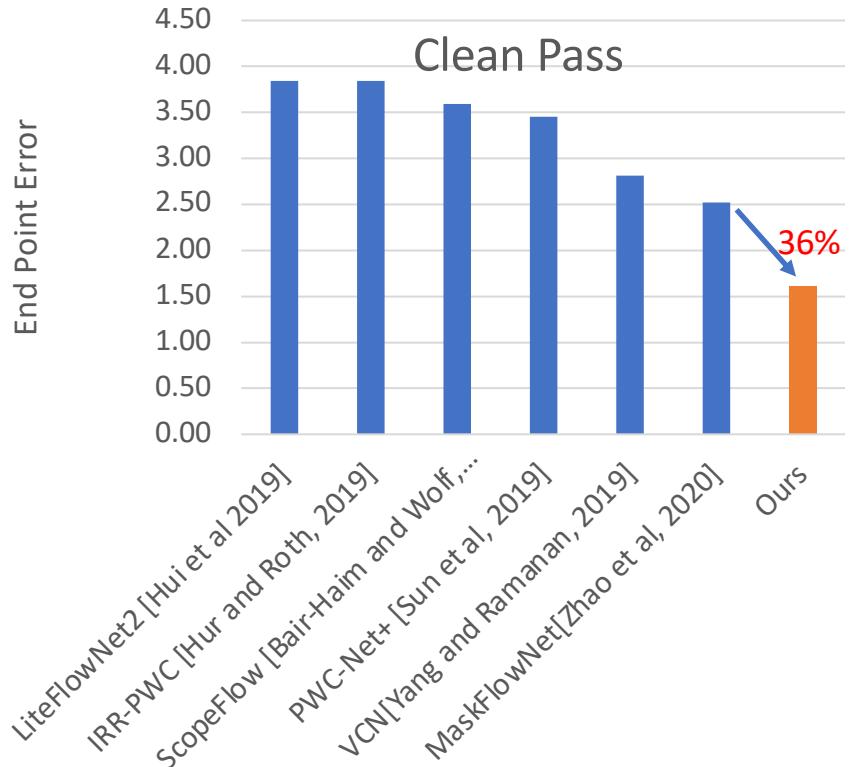


KITTI-2015: <http://www.cvlibs.net/datasets/kitti/index.php>



DAVIS (1080p) <https://davischallenge.org/>

Sintel Results



Robust Vision Challenge ECCV 2020

Method	Middlebury (Detailed subrankings)	KITTI (Detailed subrankings)	MPI Sintel (Detailed subrankings)	VIPER (Detailed subrankings)
1 RAFT-TF_RVC	1	2	1	1
2 PRAFlow_RVC	2	1	2	3
3 C-RAFT_RVC	5	3	3	4
4 VCN_RVC	3	6	5	5
	Volumetric Correspondence Networks for Optical Flow, NeurIPS 2019, [Project page] - Submitted by Gengshan Yang (CMU)			
5 IRR-PWC_RVC	7	5	7	2
	Iterative Residual Refinement for Joint Optical Flow and Occlusion Estimation [Project page] - Submitted by Junhwa Hur (TU Darmstadt)			
5 LSM_FLOW_RVC	6	4	4	7
	LSM: Learning Subspace Minimization for Low-Level Vision [Project page] - Submitted by Chengzhou Tang (Simon Fraser University)			
7 PWC-Net_RVC	4	7	6	6
	PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume, CVPR 2018, [Project page] - Submitted by Deqing Sun (Google)			
8 TVL1_RVC	8	8	8	8
	Baseline - Submitted by Toby Weid (Middlebury College)			
9 H+S_RVC	9	9	9	9
	Baseline - Submitted by Toby Weid (Middlebury College)			

All top 3 submissions used RAFT

Winner

A TensorFlow Implementation of RAFT

Deqing Sun, Charles Herrmann, Varun Jampani, Mike Krainin, Forrester Cole, Austin Stone, Rico Jonschkowski, Ramin Zabih, William Freeman, and Ce Liu

Google Research



Stereo



Many slides adapted from Steve Seitz and Svetlana Lazebnik



Binocular stereo

- Given a calibrated binocular stereo pair, fuse it to produce a depth image

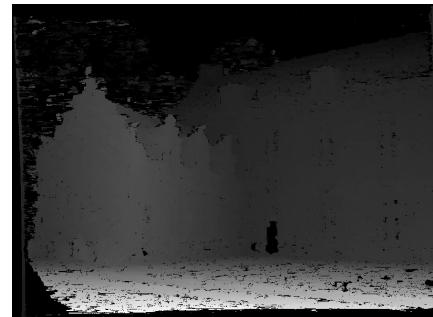
image 1



image 2



Dense depth map



Binocular stereo

- Given a calibrated binocular stereo pair, fuse it to produce a depth image



Where does the depth information come from?



Binocular stereo

- Given a calibrated binocular stereo pair, fuse it to produce a depth image
 - Humans can do it



Stereograms: Invented by Sir Charles Wheatstone, 1838



Binocular stereo

- Given a calibrated binocular stereo pair, fuse it to produce a depth image
 - Humans can do it



Autostereograms: www.magiceye.com



Binocular stereo

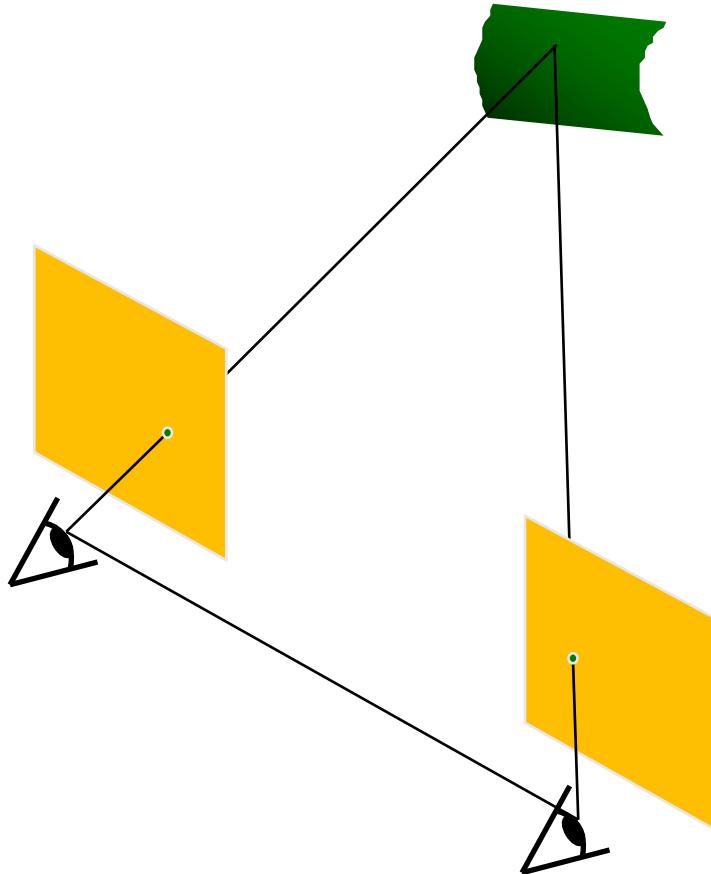
- Given a calibrated binocular stereo pair, fuse it to produce a depth image
 - Humans can do it



Autostereograms: www.magiceye.com



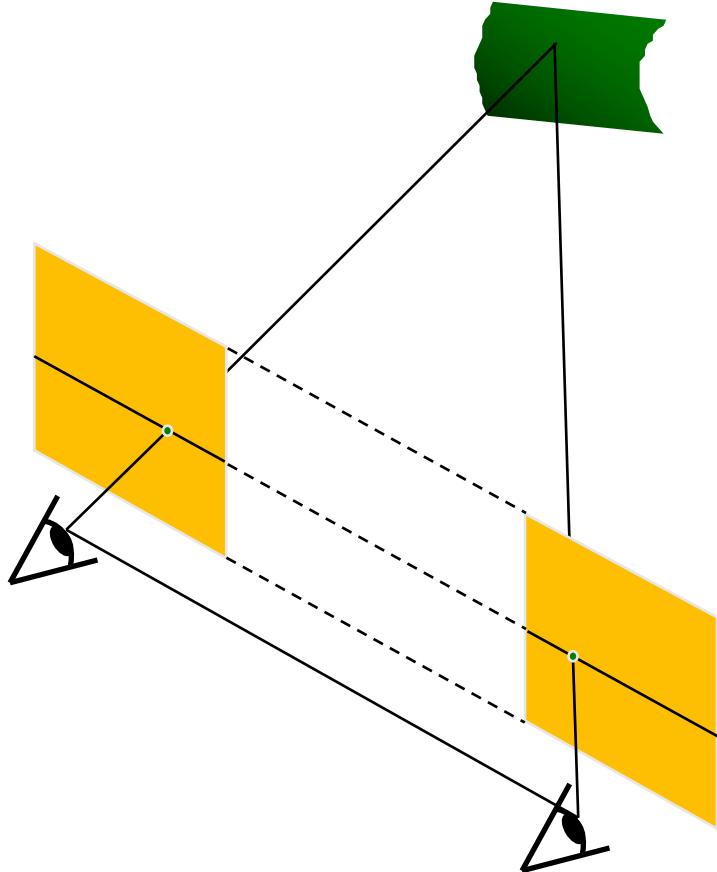
Simplest Case: Parallel images



- Image planes of cameras are parallel to each other and to the baseline
- Camera centers are at same height
- Focal lengths are the same



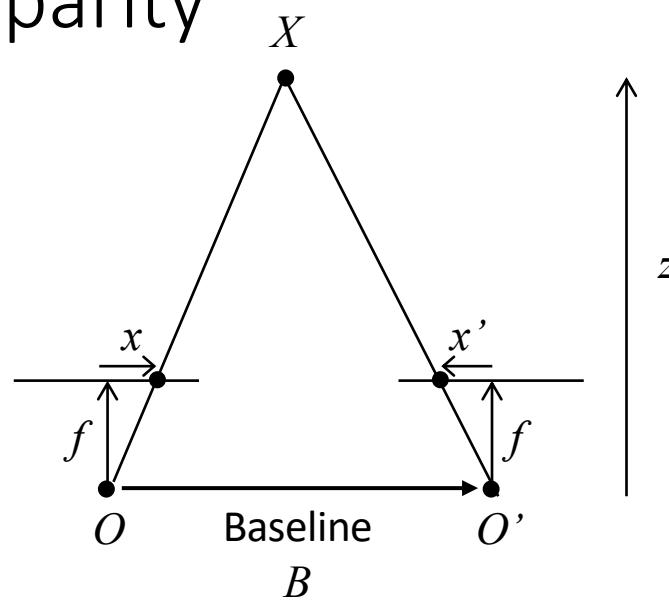
Simplest Case: Parallel images



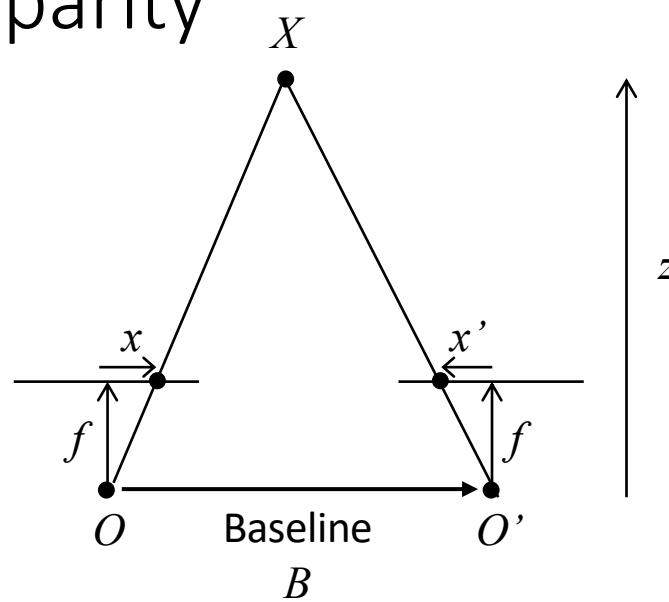
- Image planes of cameras are parallel to each other and to the baseline
- Camera centers are at same height
- Focal lengths are the same
- Then epipolar lines fall along the horizontal scan lines of the images



Depth from disparity



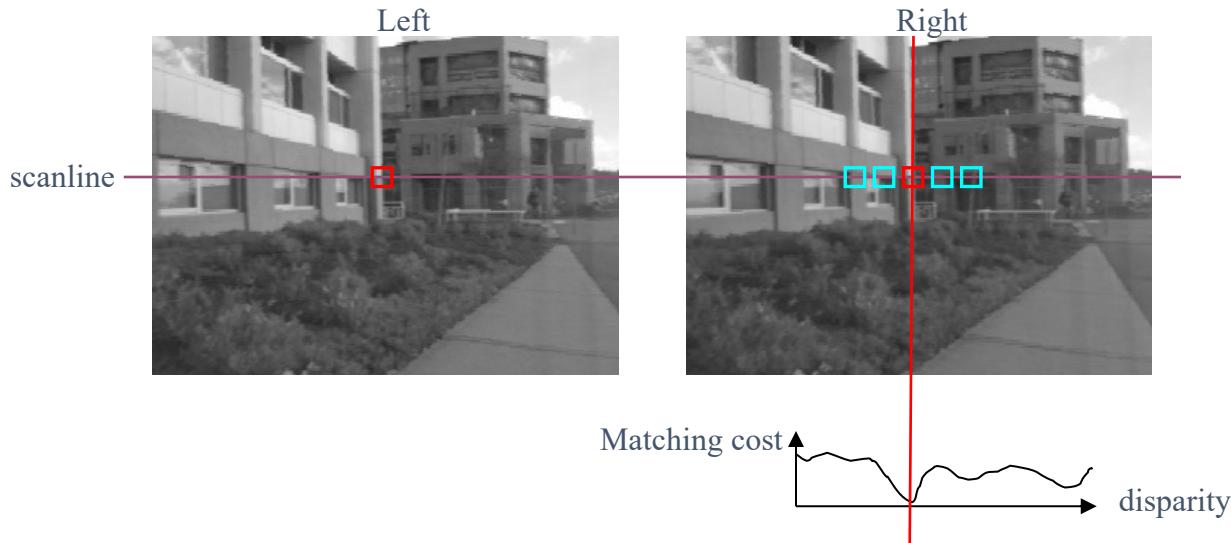
Depth from disparity



$$disparity = x - x' = \frac{B \cdot f}{z}$$

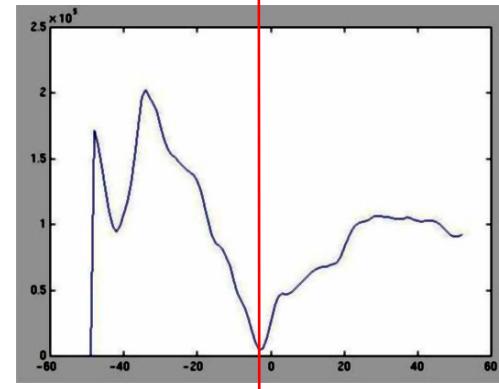


Correspondence search

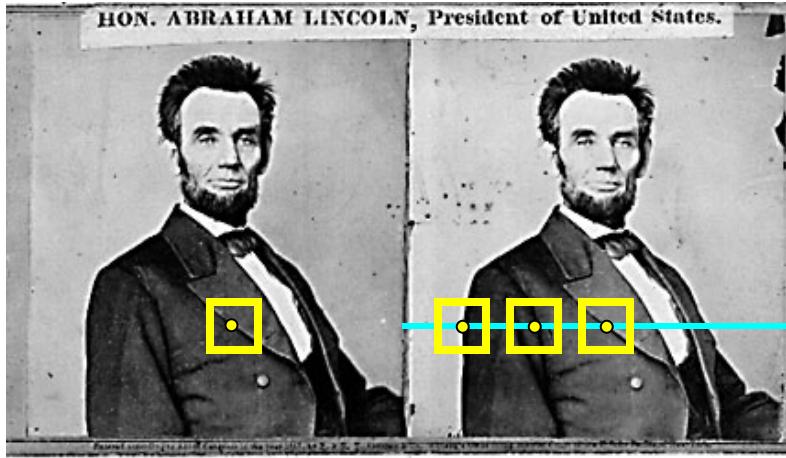


- Slide a window along the right scanline and compare contents of that window with the reference window in the left image
- Matching cost: SSD or normalized correlation

Correspondence search

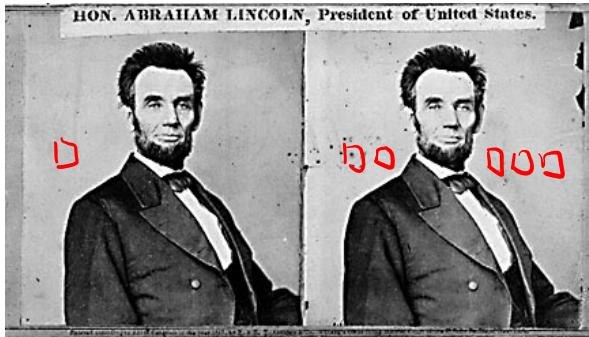


Basic stereo algorithm



- For each pixel x in the first image
 - Find corresponding epipolar scanline in the right image
 - Examine all pixels on the scanline and pick the best match x'
 - Compute disparity $x-x'$ and set $\text{depth}(x) = B*f/(x-x')$

Failures of correspondence search



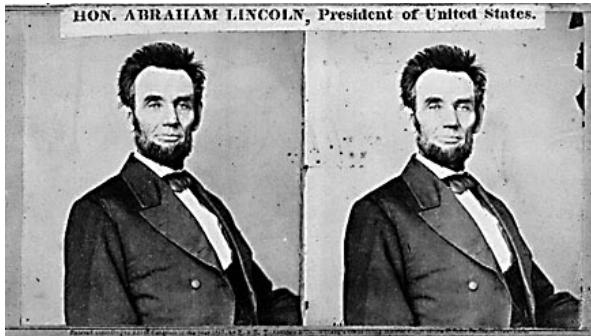
Textureless surfaces



Occlusions, repetition



Failures of correspondence search



Textureless surfaces



Occlusions, repetition



Non-Lambertian surfaces, specularities

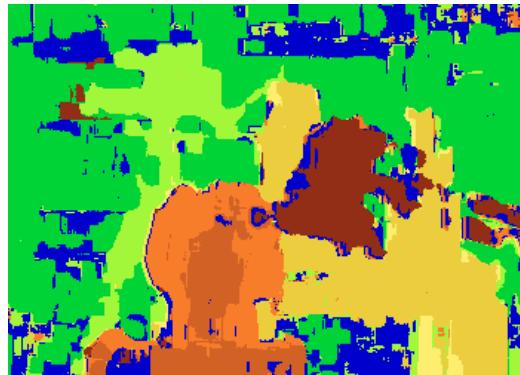


Results with window search

Data



Window-based matching



Ground truth



Better methods exist...



Graph cuts



Ground truth

Y. Boykov, O. Veksler, and R. Zabih, [Fast Approximate Energy Minimization via Graph Cuts](#), PAMI 2001

For the latest and greatest: <http://www.middlebury.edu/stereo/>



How can we improve window-based matching?

- The similarity constraint is **local** (each reference window is matched independently)
- Need to enforce **non-local** correspondence constraints



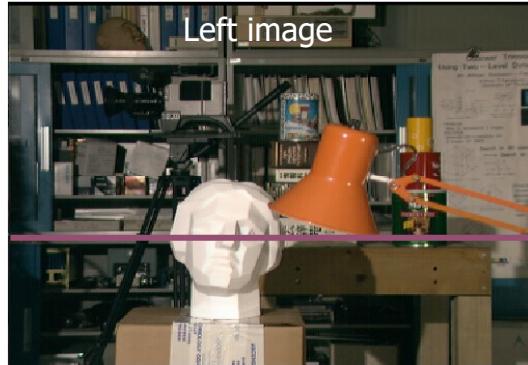
Non-local constraints

- Uniqueness
 - For any point in one image, there should be at most one matching point in the other image
- Ordering
 - Corresponding points should be in the same order in both views
- Smoothness
 - We expect disparity values to change slowly (for the most part)

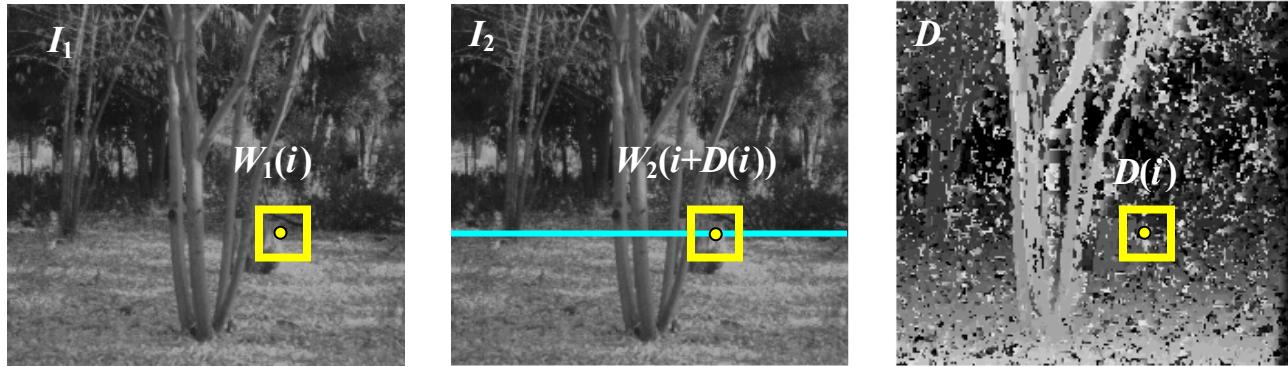


Scanline stereo

- Try to coherently match pixels on the entire scanline
- Different scanlines are still optimized independently

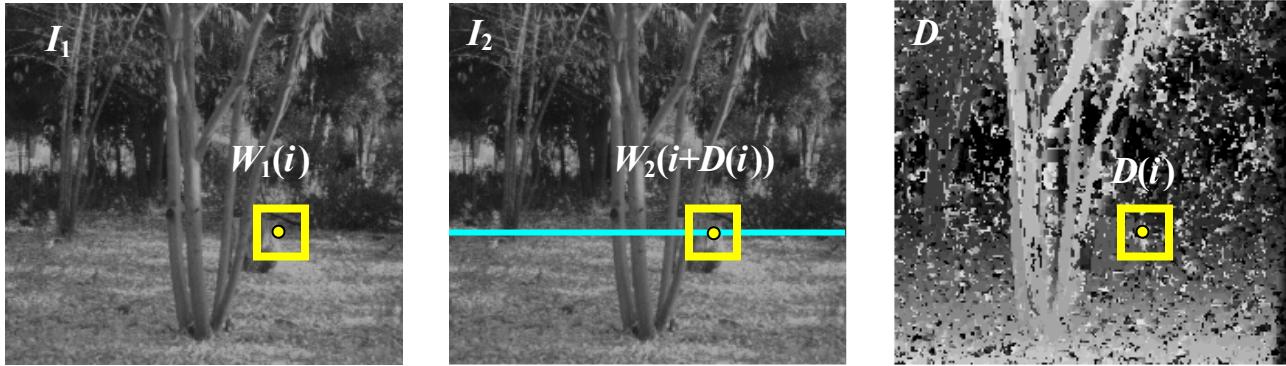


Stereo matching as energy minimization



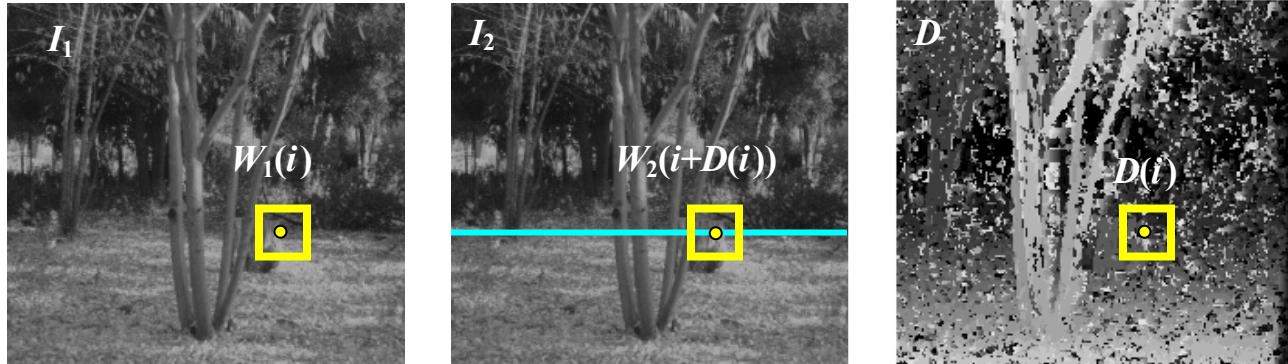
$$E(D) = \sum_i (W_1(i) - W_2(i + D(i)))^2 + \lambda \sum_{\text{neighbors } i, j} \rho(D(i) - D(j))$$

Stereo matching as energy minimization



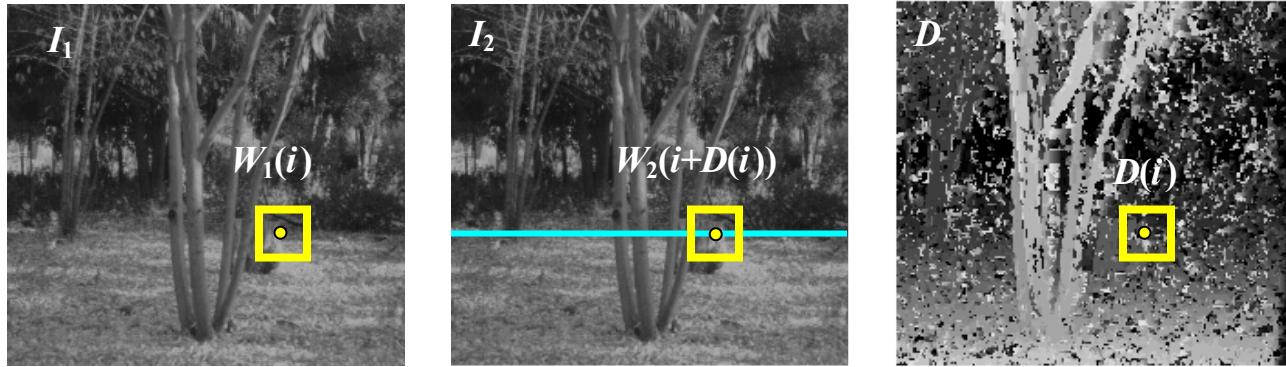
$$E(D) = \underbrace{\sum_i (W_1(i) - W_2(i + D(i)))^2}_{\text{data term}} + \lambda \underbrace{\sum_{\text{neighbors } i, j} \rho(D(i) - D(j))}_{\text{smoothness term}}$$

Stereo matching as energy minimization



$$E(D) = \underbrace{\sum_i (W_1(i) - W_2(i + D(i)))^2}_{\text{data term}} + \lambda \underbrace{\sum_{\text{neighbors } i,j} \rho(D(i) - D(j))}_{\text{smoothness term}}$$

Stereo matching as energy minimization



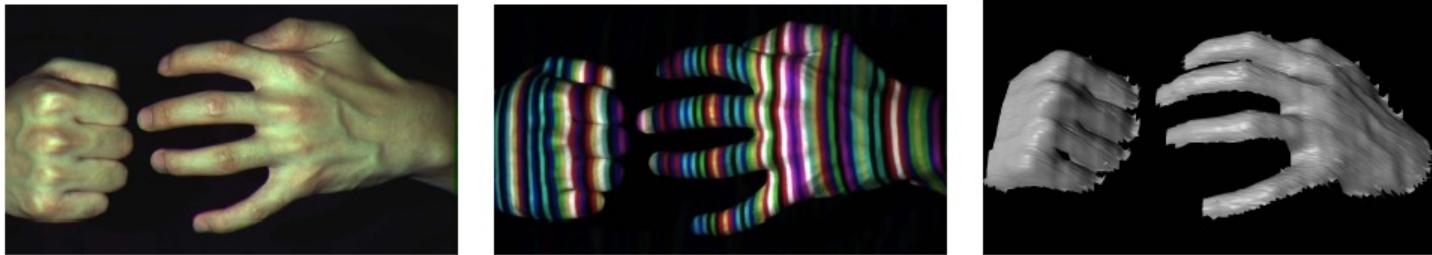
$$E(D) = \underbrace{\sum_i (W_1(i) - W_2(i + D(i)))^2}_{\text{data term}} + \lambda \underbrace{\sum_{\text{neighbors } i, j} \rho(D(i) - D(j))}_{\text{smoothness term}}$$

- Energy functions of this form can be minimized using *graph cuts*

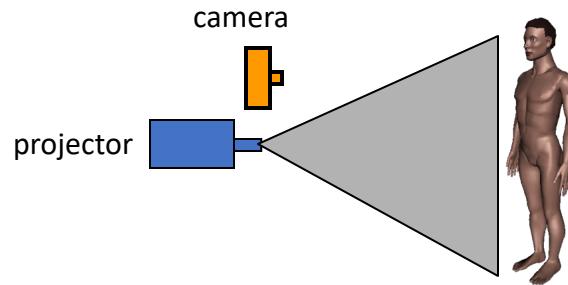
Y. Boykov, O. Veksler, and R. Zabih, [Fast Approximate Energy Minimization via Graph Cuts](#), PAMI 2001



Active stereo with structured light



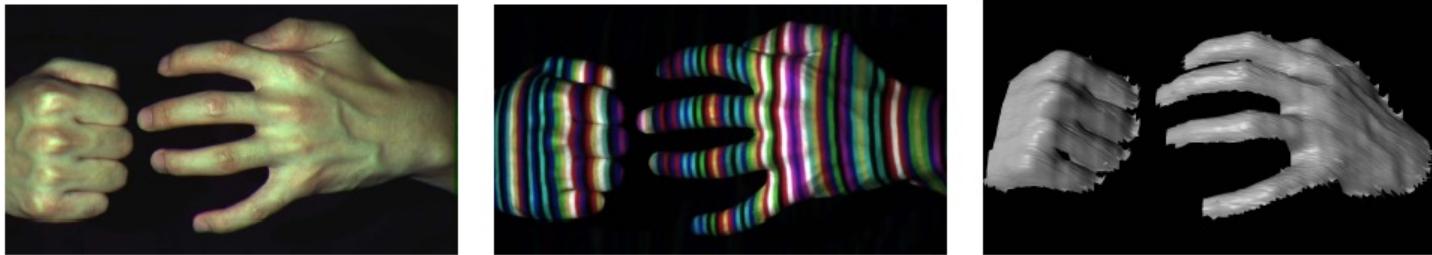
- Project “structured” light patterns onto the object
 - Simplifies the correspondence problem
 - Allows us to use only one camera



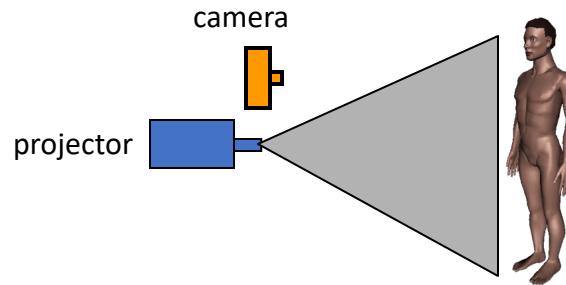
L. Zhang, B. Curless, and S. M. Seitz. [Rapid Shape Acquisition Using Color Structured Light and Multi-pass Dynamic Programming](#). 3DPVT 2002



Active stereo with structured light



- Project “structured” light patterns onto the object
 - Simplifies the correspondence problem
 - Allows us to use only one camera



L. Zhang, B. Curless, and S. M. Seitz. [Rapid Shape Acquisition Using Color Structured Light and Multi-pass Dynamic Programming](#). 3DPVT 2002



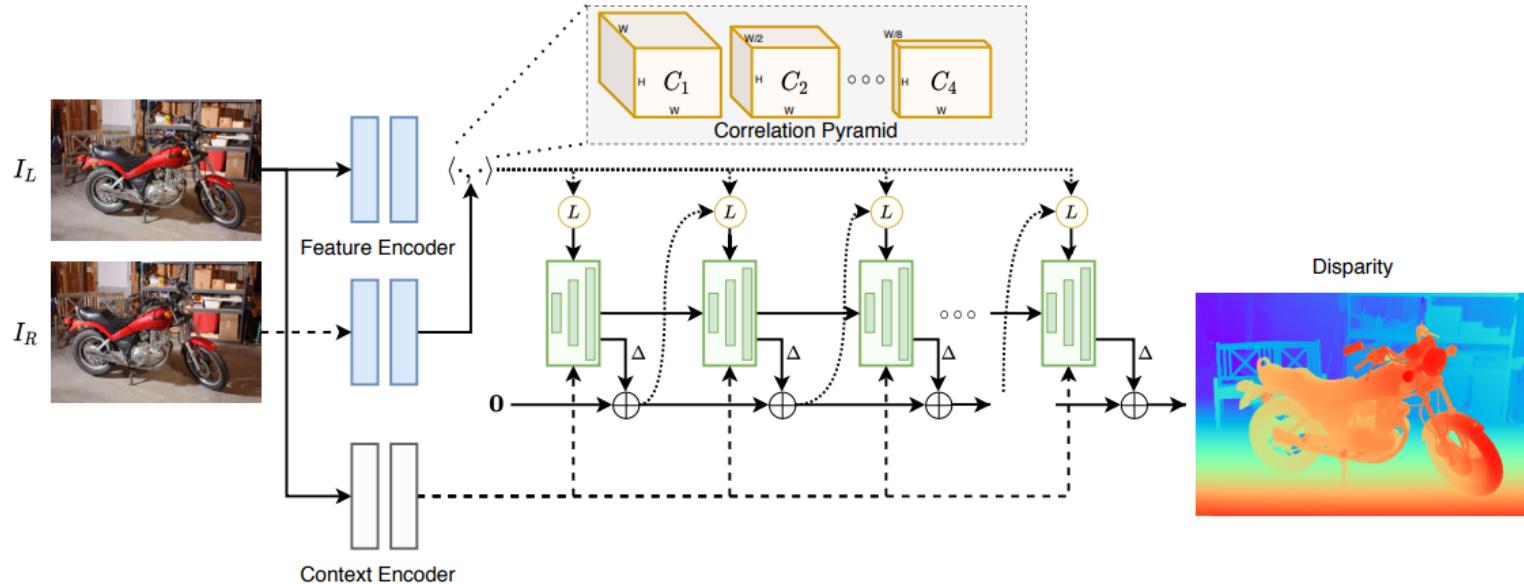
Kinect: Structured infrared light



<http://bbzippo.wordpress.com/2010/11/28/kinect-in-infrared/>



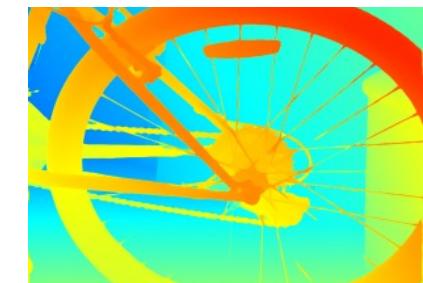
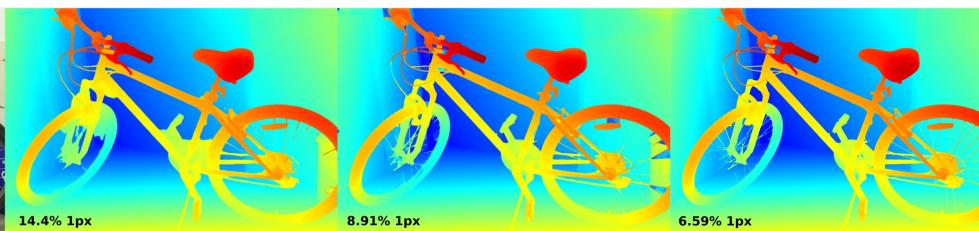
RAFT-Stereo: RAFT for rectified two-view stereo



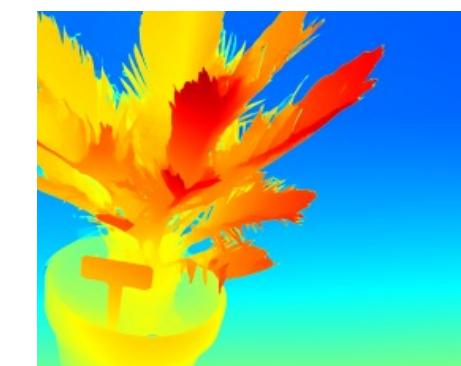
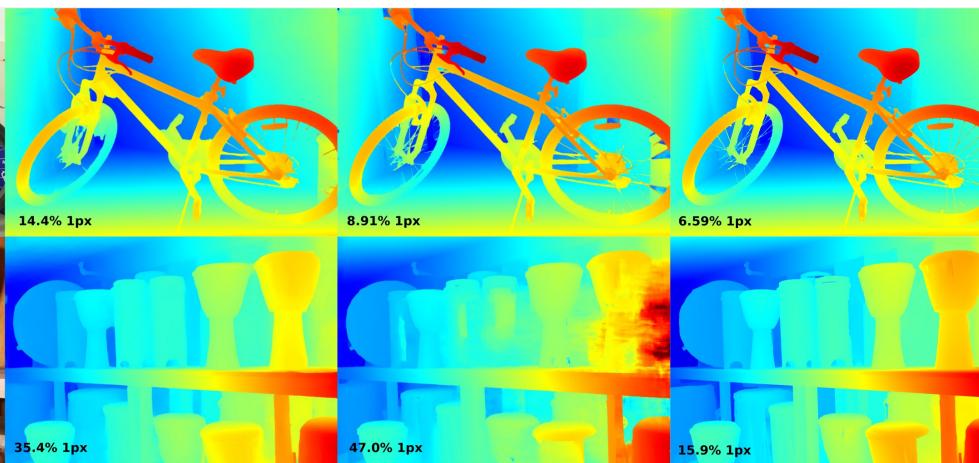
[Teed, Lipson, Deng, 2020]

RAFT-Stereo: 1st on Middlebury [Scharstein et al, 2014]

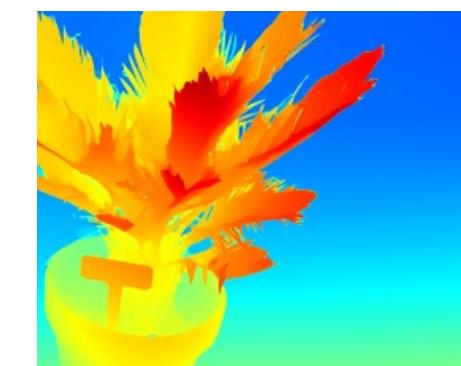
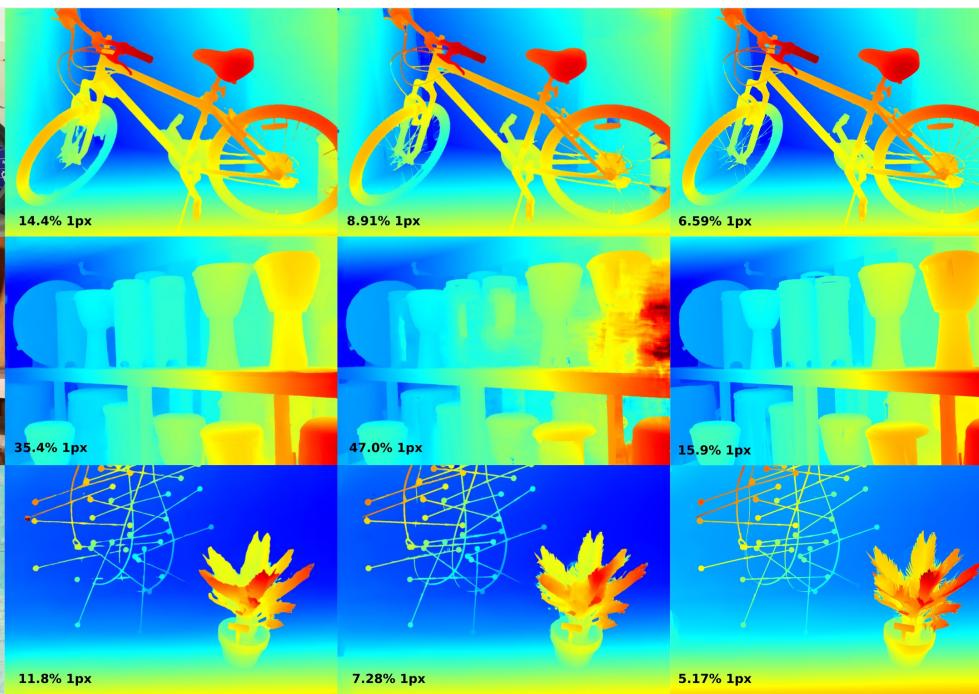
Bicycle2



DjembeL



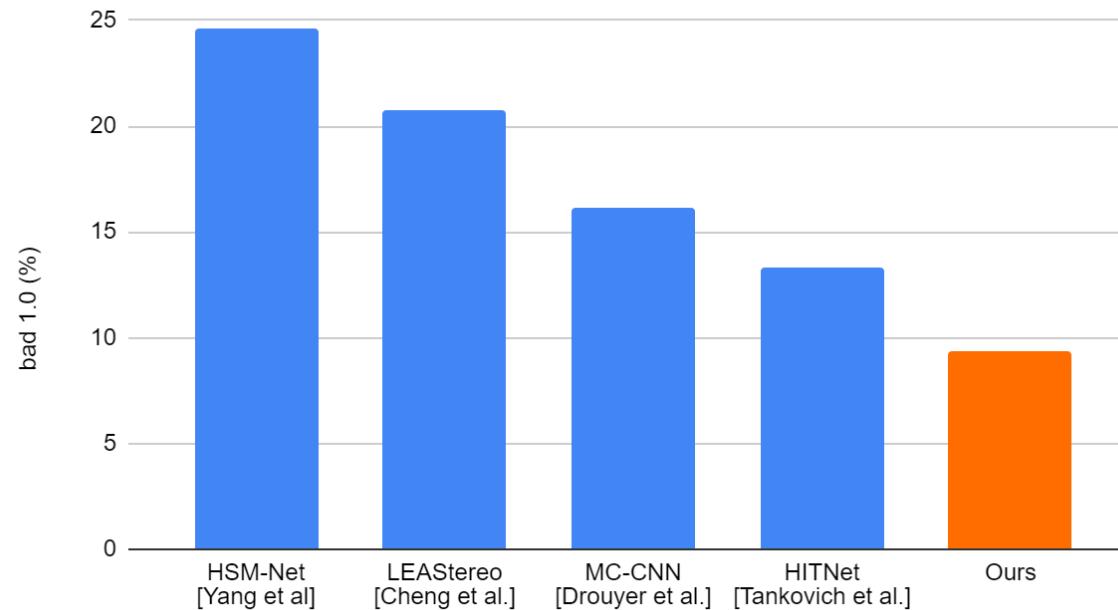
AustraliaP



[Lipson, Teed, Deng, 3DV 2021] Best Student Paper Award

Middlebury Stereo Benchmark

Middleburry: bad 1.0 (%)







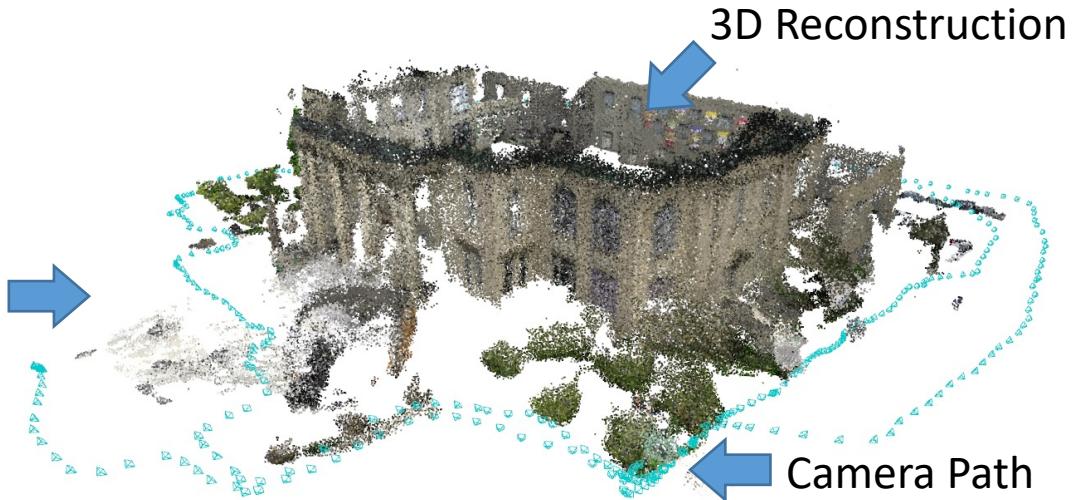




Visual SLAM:

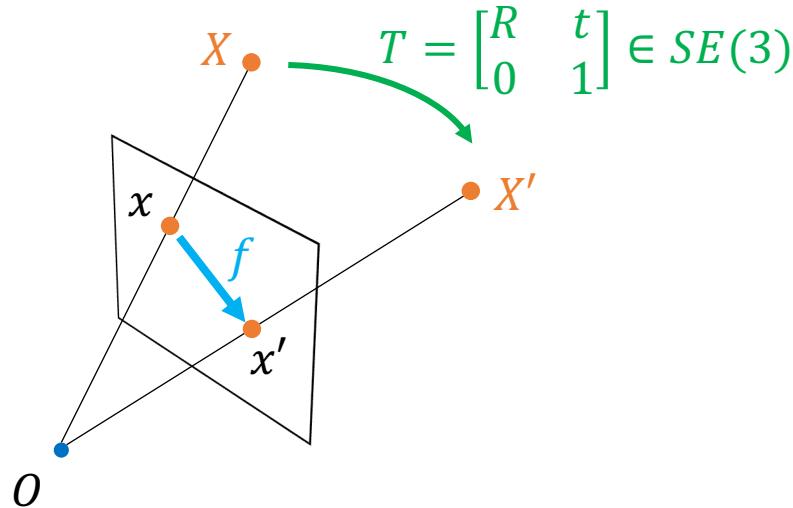
Simultaneous Localization and Mapping

- Input: video of (largely) static scene
- Output: 3D map and camera trajectory



Classical Approach: Optimization with Multiview Geometry

2D motion (optical flow) is a known analytical function of 3D points and 3D motion



$$f = F(X, T)$$

Step 1. Estimate 2D flow f

→ Match pixels by manual features

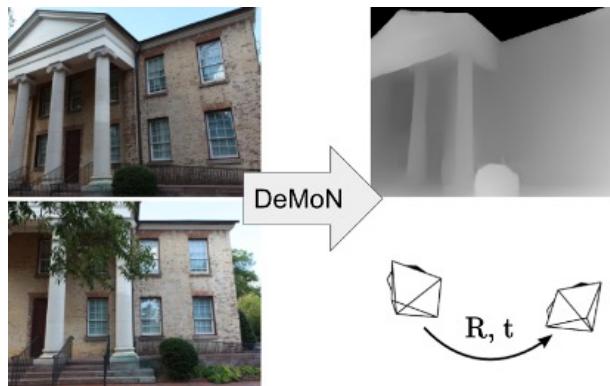
Step 2. Solve for 3D given flow

$$\min_{X, T} \|f - F(X, T)\|^2$$

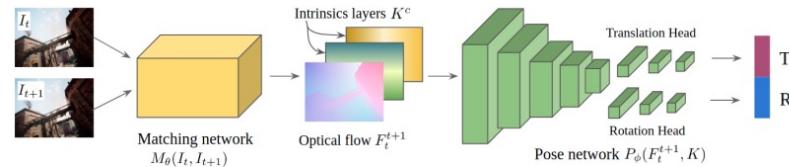
Insufficient Robustness: Failures are frequent and catastrophic

Deep Visual SLAM

Train a network to directly regress ***3D points*** (depth) and ***3D motion***



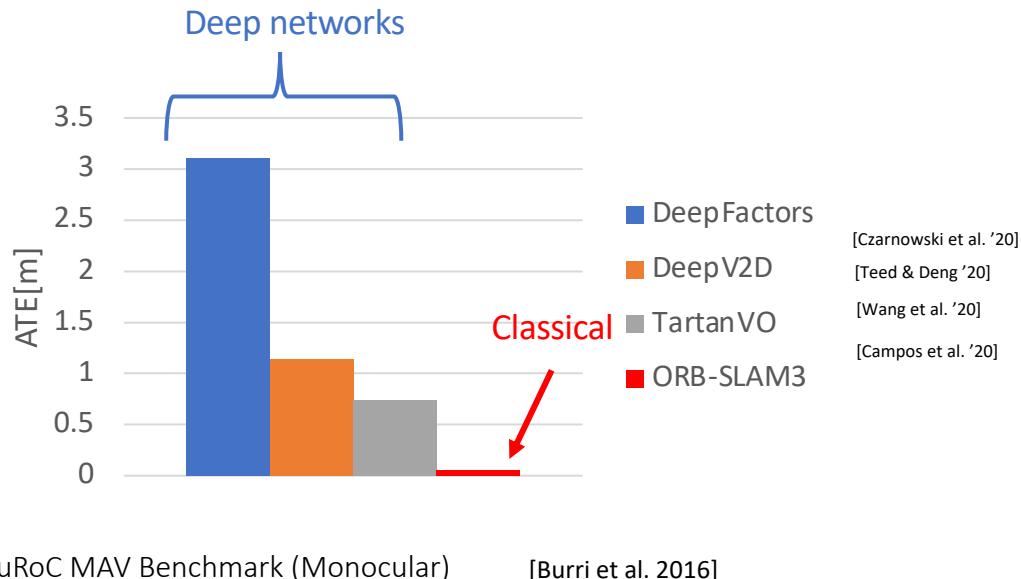
DeMoN [Ummenhofer et al., 2017]



TartanVO [Wang et al., 2021]

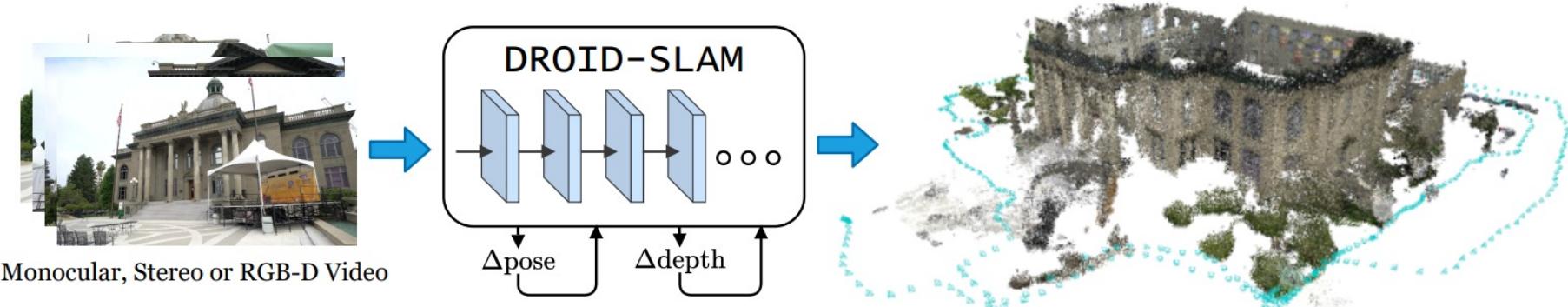
Problems with Deep Visual SLAM

- ***Lower Accuracy:*** large amounts of drift, global inconsistency
- ***Weaker Generalization:*** doesn't generalize to new datasets or cameras



DROID-SLAM

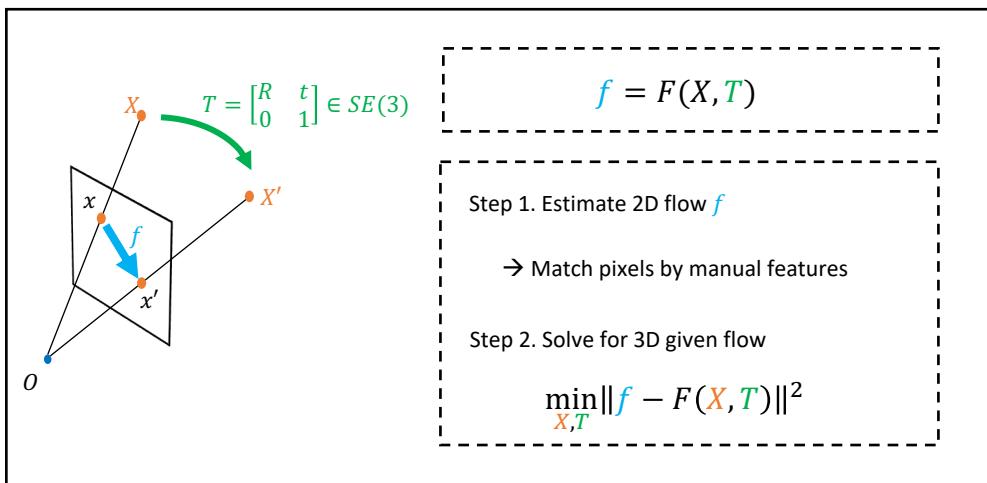
DROID: Differentiable Recurrent Optimization-Inspired Design



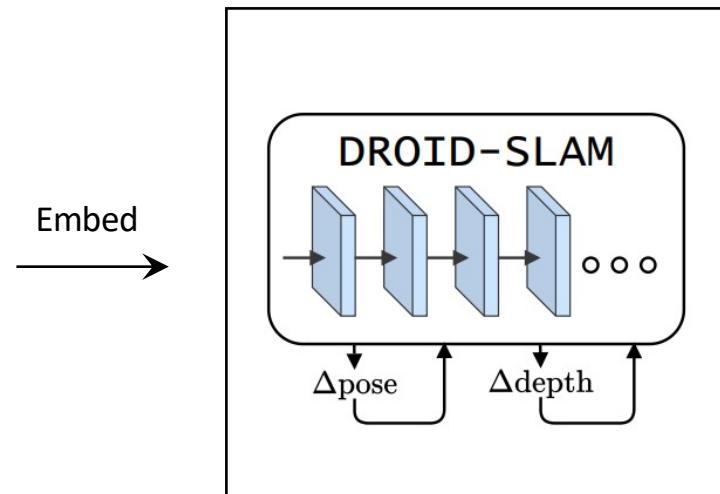
- **Accurate** – reduce error by **60%-80%** over prior systems
- **Robust** – **6X** fewer catastrophic failures
- **Generalizable** – trained only on synthetic data

DROID-SLAM

DROID: Differentiable Recurrent Optimization-Inspired Design



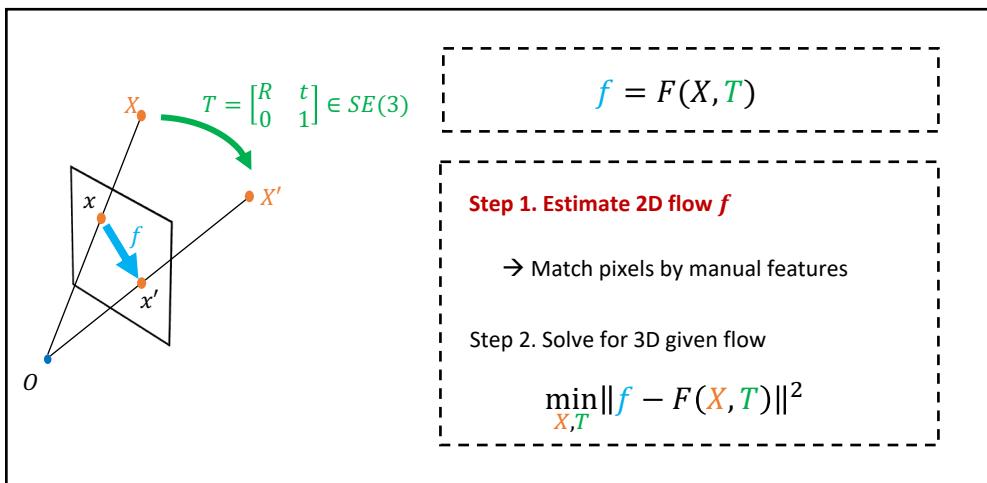
Symbolic knowledge from classical approaches



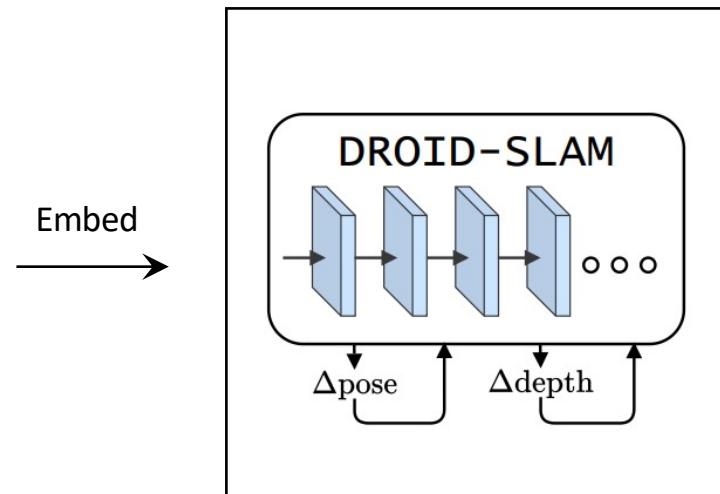
End-to-end neural architecture

DROID-SLAM

DROID: Differentiable Recurrent Optimization-Inspired Design



Symbolic knowledge from classical approaches



End-to-end neural architecture

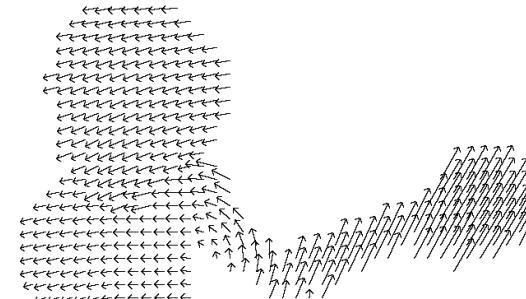
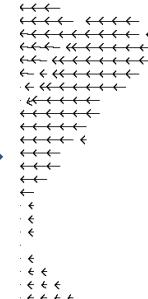
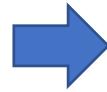
Estimate 2D motion (optical flow)

- Predict per-pixel 2D motion between a pair of frames

Frame 1

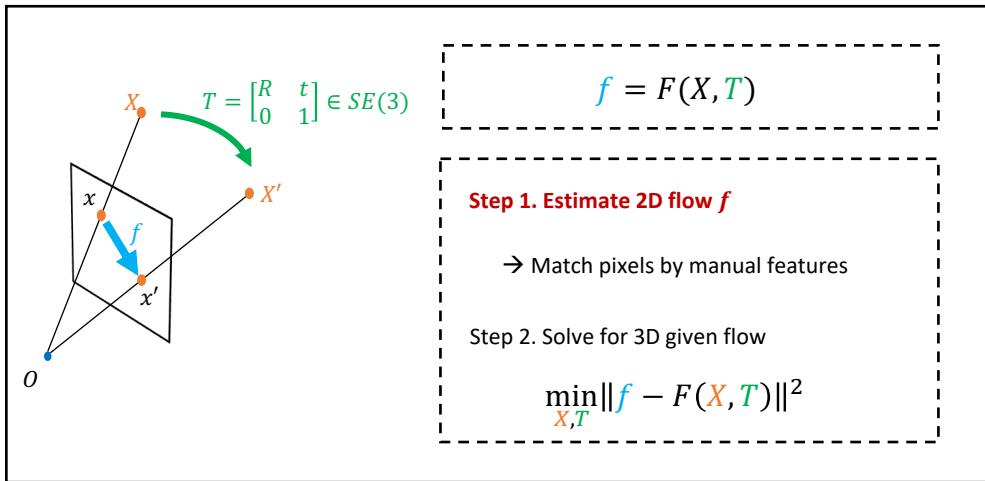


Frame 2

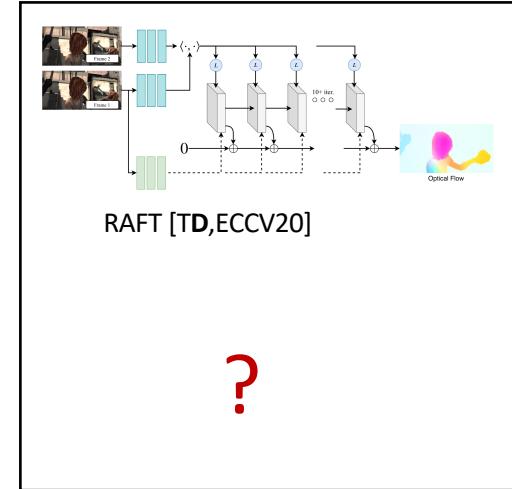


DROID-SLAM

DROID: Differentiable Recurrent Optimization-Inspired Design



Embed →

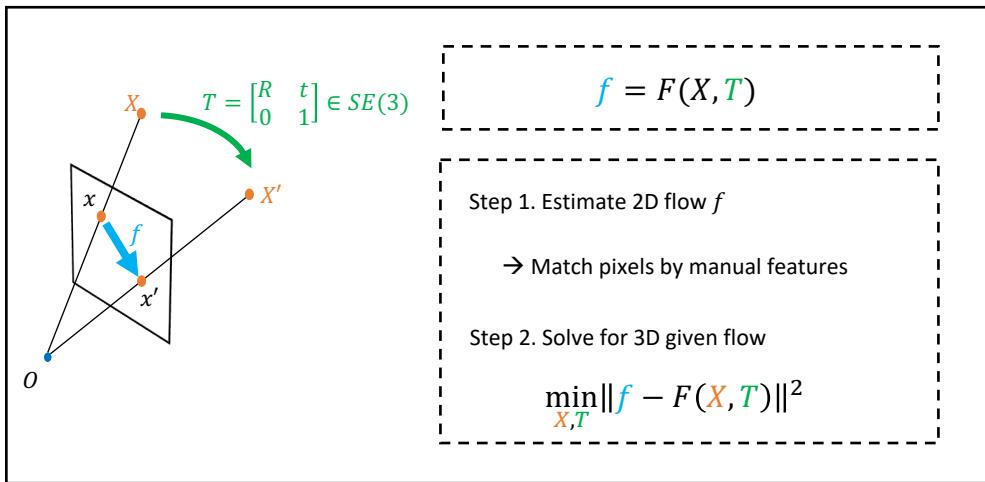


Symbolic knowledge from classical approaches

End-to-end neural architecture

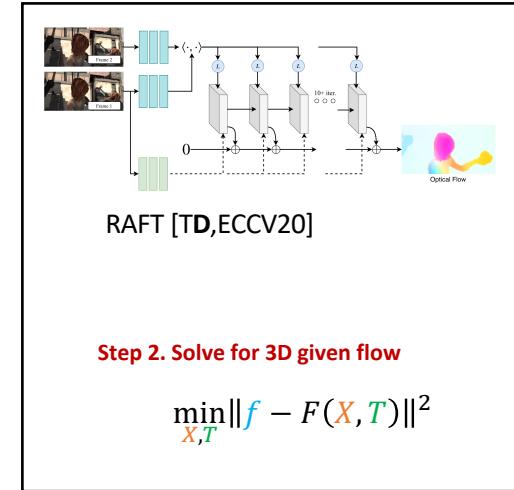
DROID-SLAM

DROID: Differentiable Recurrent Optimization-Inspired Design



Symbolic knowledge from classical approaches

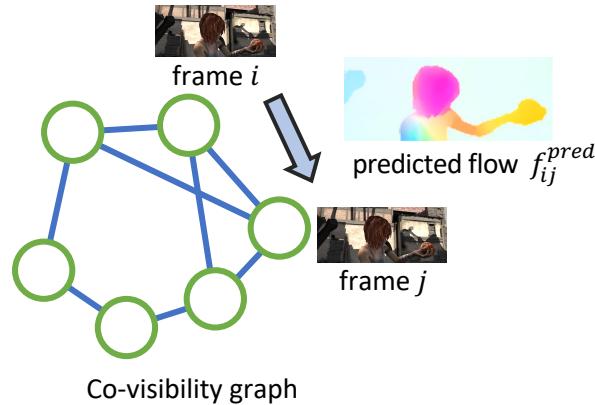
Embed →



End-to-end neural architecture

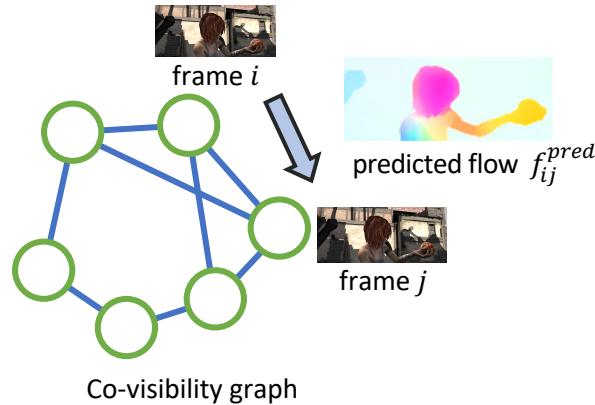
Dense Bundle Adjustment (DBA)

- **Given:** co-visibility graph $(\mathcal{V}, \mathcal{E})$, predicted flow f_{ij}^{pred}



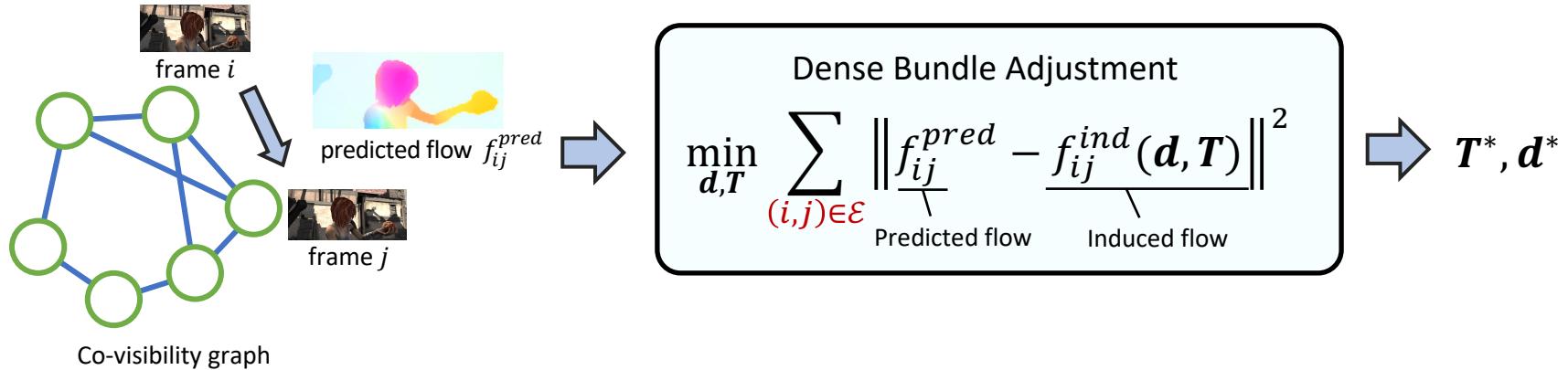
Dense Bundle Adjustment (DBA)

- **Given:** co-visibility graph $(\mathcal{V}, \mathcal{E})$, predicted flow f_{ij}^{pred}
- **Want:** depth maps $\mathbf{d} = (d_1, \dots, d_i, \dots)$, camera poses $\mathbf{T} = (T_1, \dots, T_i, \dots)$



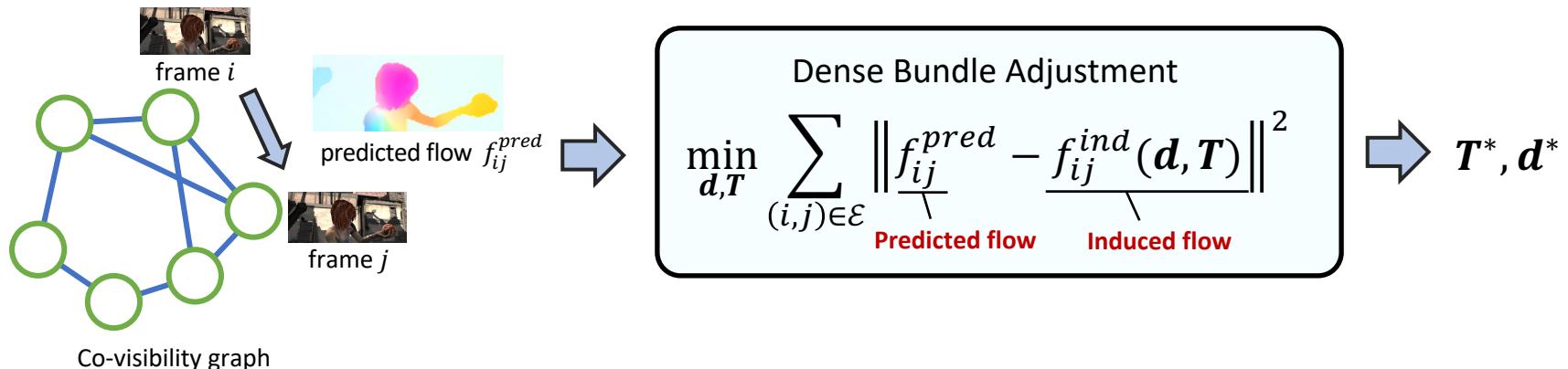
Dense Bundle Adjustment (DBA)

- **Given:** co-visibility graph $(\mathcal{V}, \mathcal{E})$, predicted flow f_{ij}^{pred}
- **Want:** depth maps $\mathbf{d} = (d_1, \dots, d_i, \dots)$, camera poses $\mathbf{T} = (T_1, \dots, T_i, \dots)$



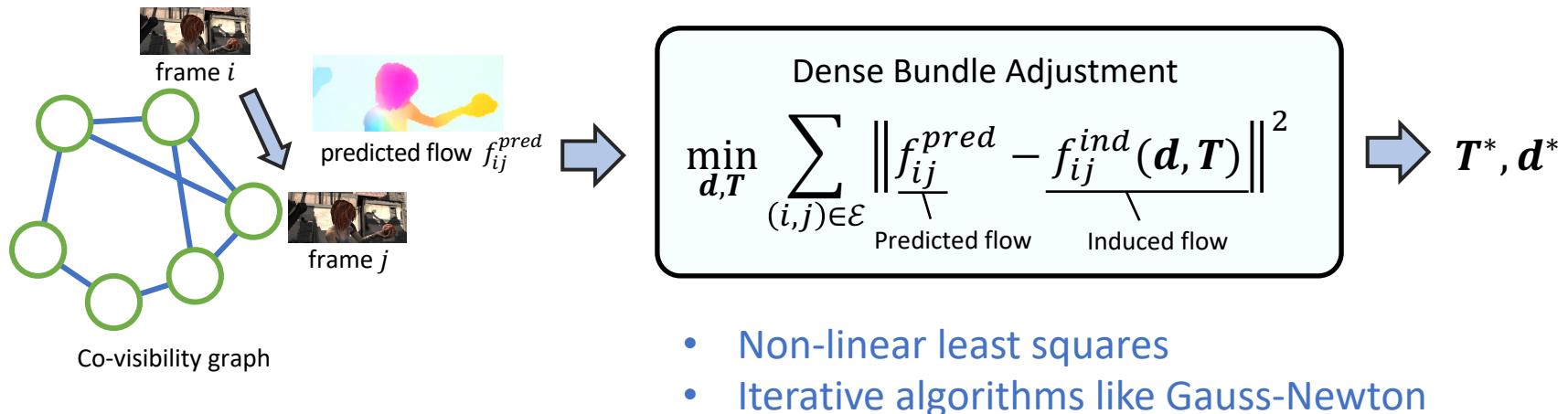
Dense Bundle Adjustment (DBA)

- **Given:** co-visibility graph $(\mathcal{V}, \mathcal{E})$, predicted flow f_{ij}^{pred}
- **Want:** depth maps $\mathbf{d} = (d_1, \dots, d_i, \dots)$, camera poses $\mathbf{T} = (T_1, \dots, T_i, \dots)$

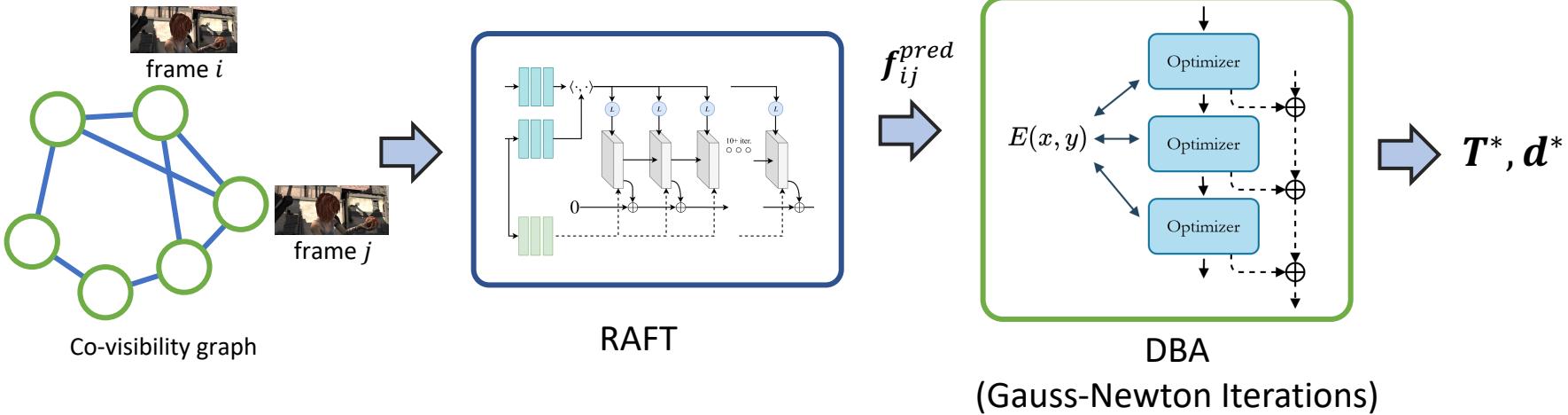


Dense Bundle Adjustment (DBA)

- **Given:** co-visibility graph $(\mathcal{V}, \mathcal{E})$, predicted flow f_{ij}^{pred}
- **Want:** depth maps $\mathbf{d} = (d_1, \dots, d_i, \dots)$, camera poses $\mathbf{T} = (T_1, \dots, T_i, \dots)$

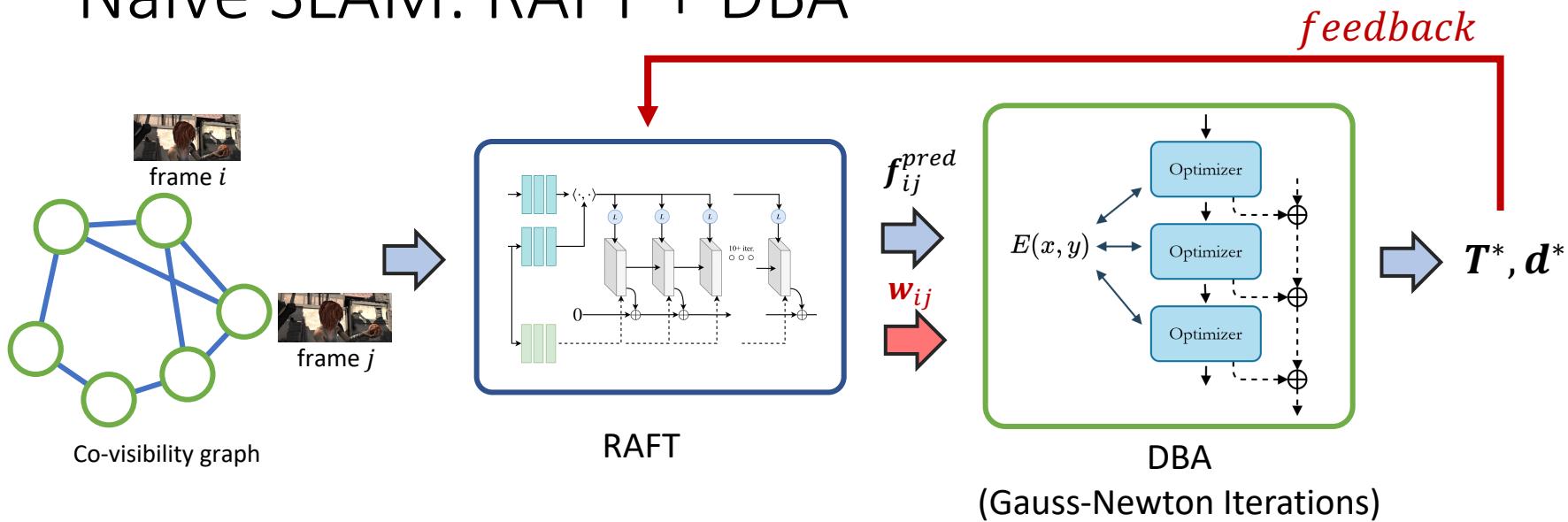


Naïve SLAM: RAFT + DBA



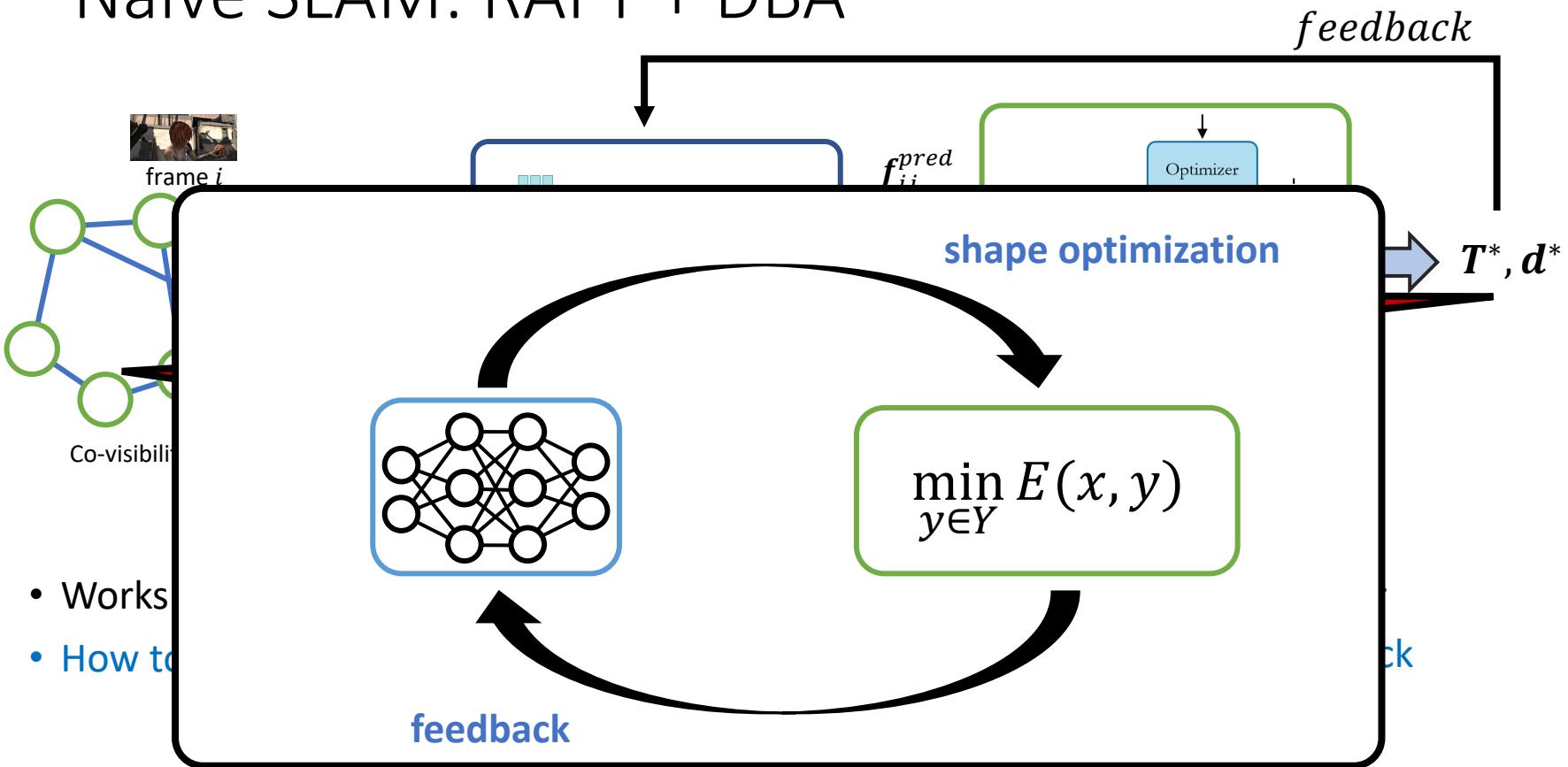
- Works poorly, because of outliers: visibility, dynamic objects, prediction error

Naïve SLAM: RAFT + DBA



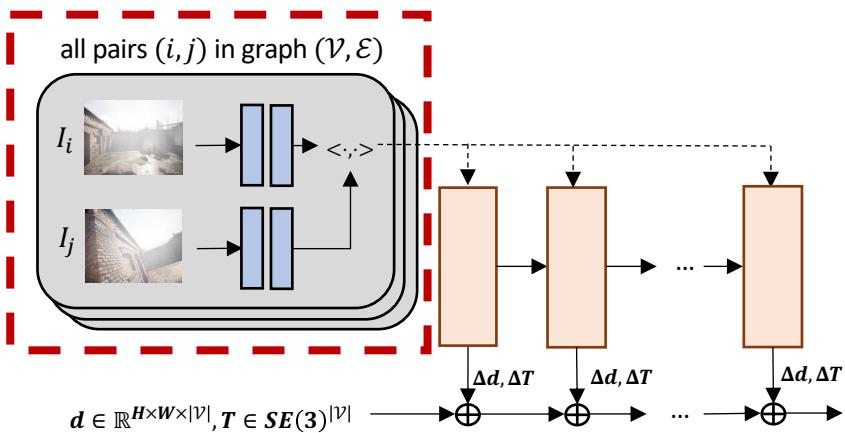
- Works poorly, because of outliers: visibility, dynamic objects, prediction error
- How to exclude outliers?
 - (1) Predicted Confidence Map
 - (2) Feedback

Naïve SLAM: RAFT + DBA



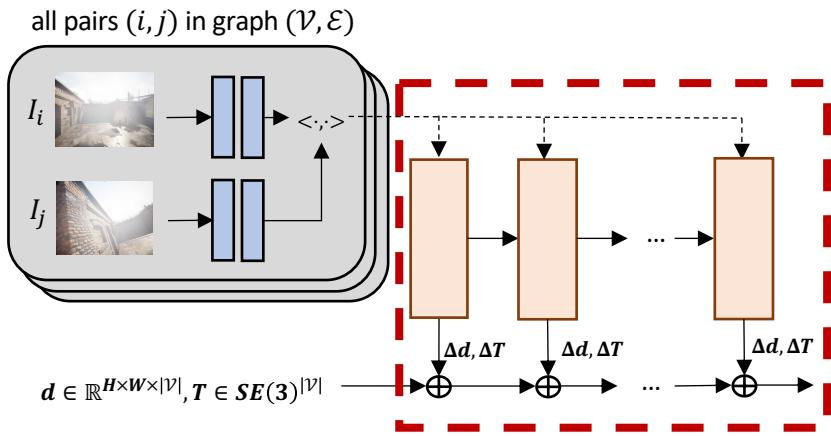
DROID-SLAM: Architecture

- Recurrent Updates + Analytical Layer



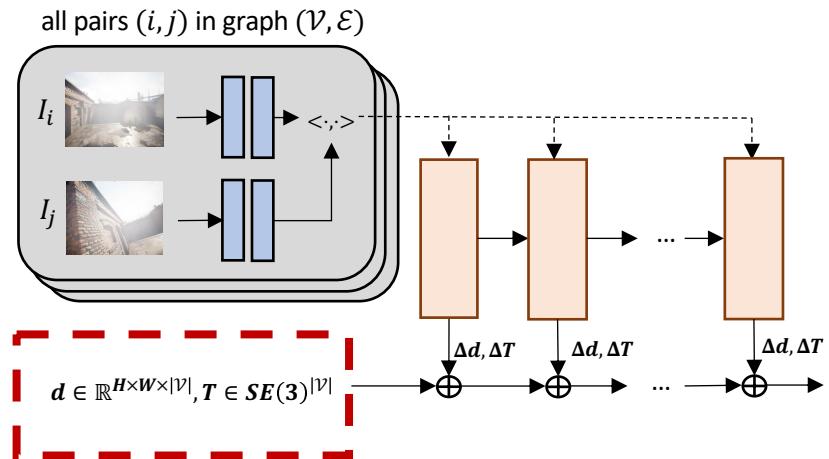
DROID-SLAM: Architecture

- Recurrent Updates + Analytical Layer



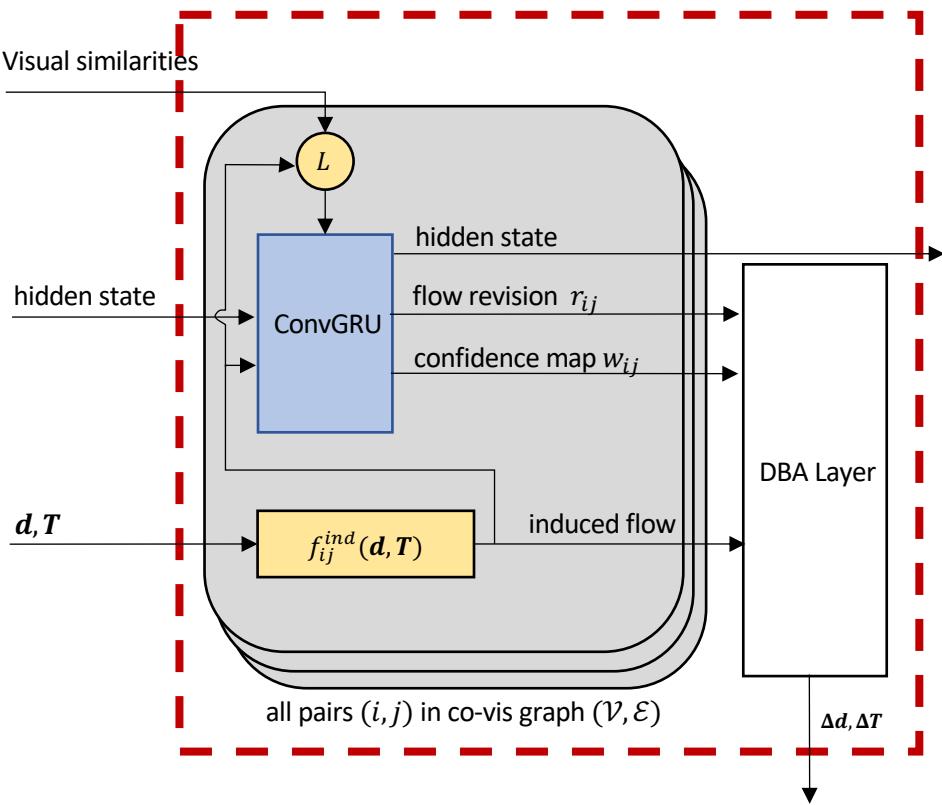
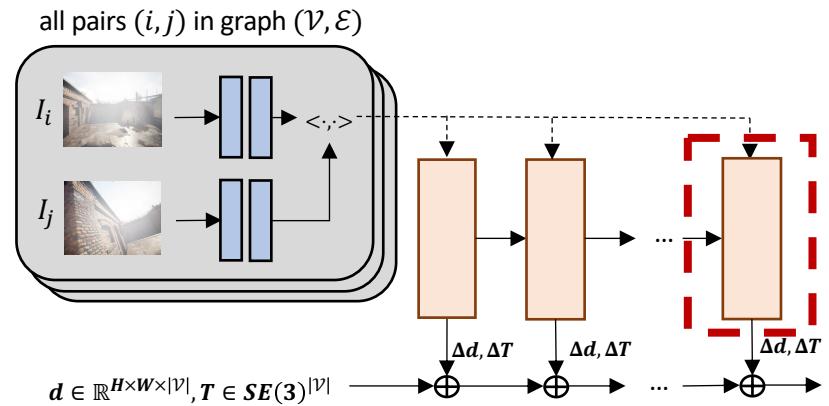
DROID-SLAM: Architecture

- Recurrent Updates + Analytical Layer



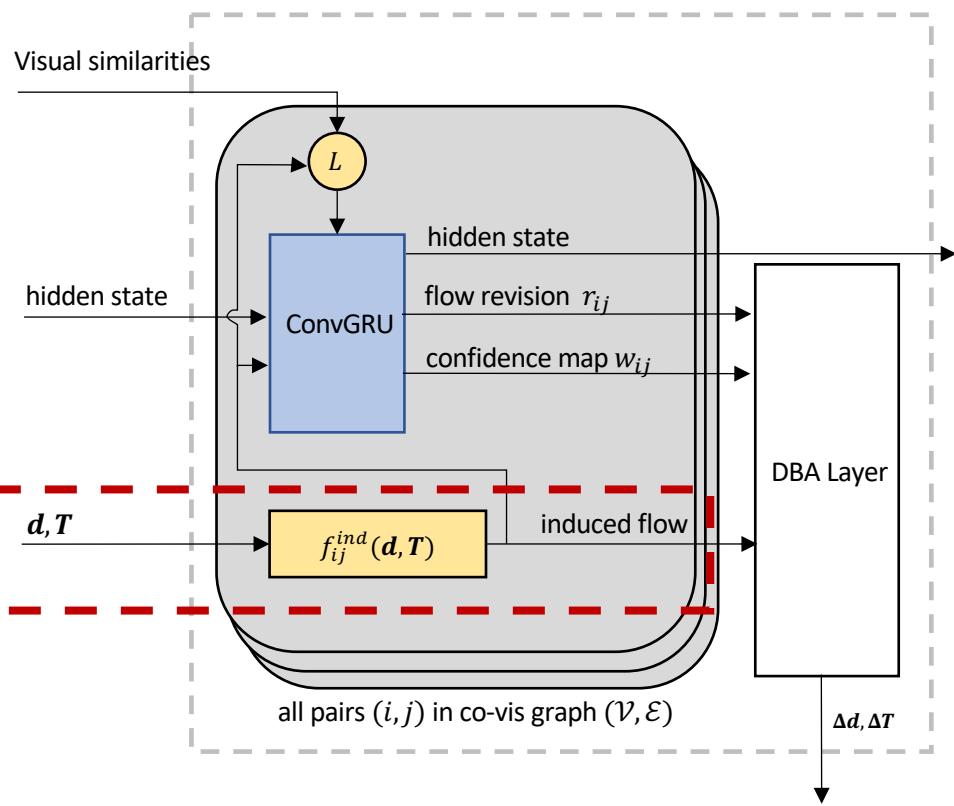
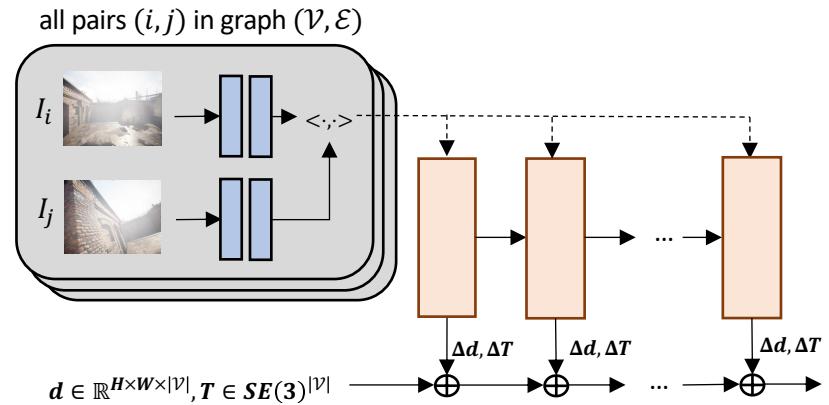
DROID-SLAM: Architecture

- Recurrent Updates + Analytical Layer



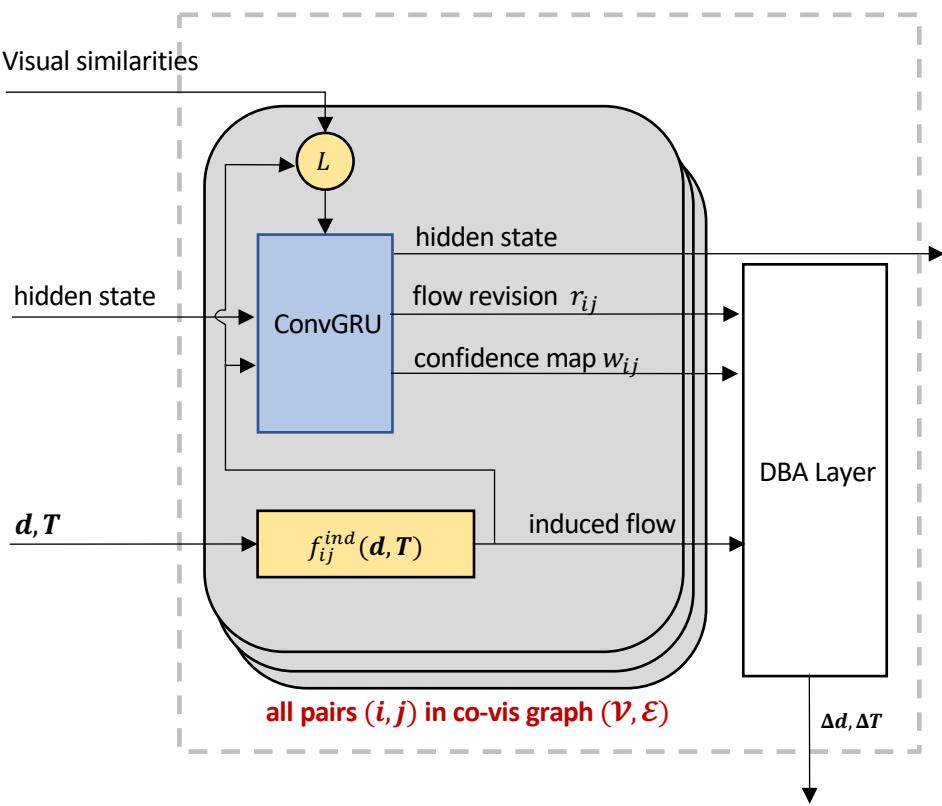
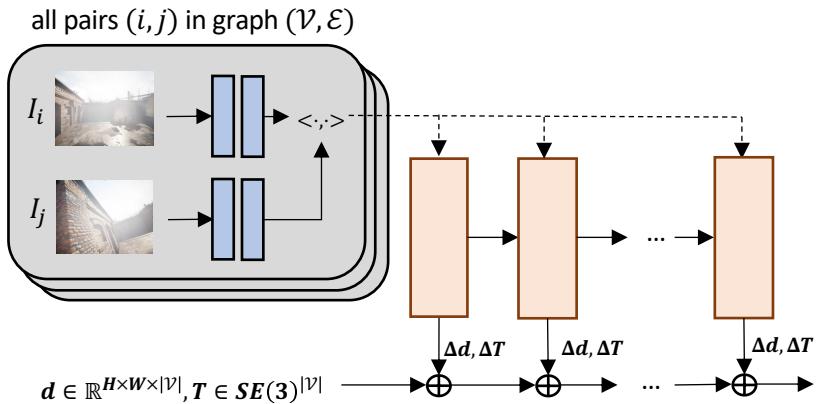
DROID-SLAM: Architecture

- Recurrent Updates + Analytical Layer



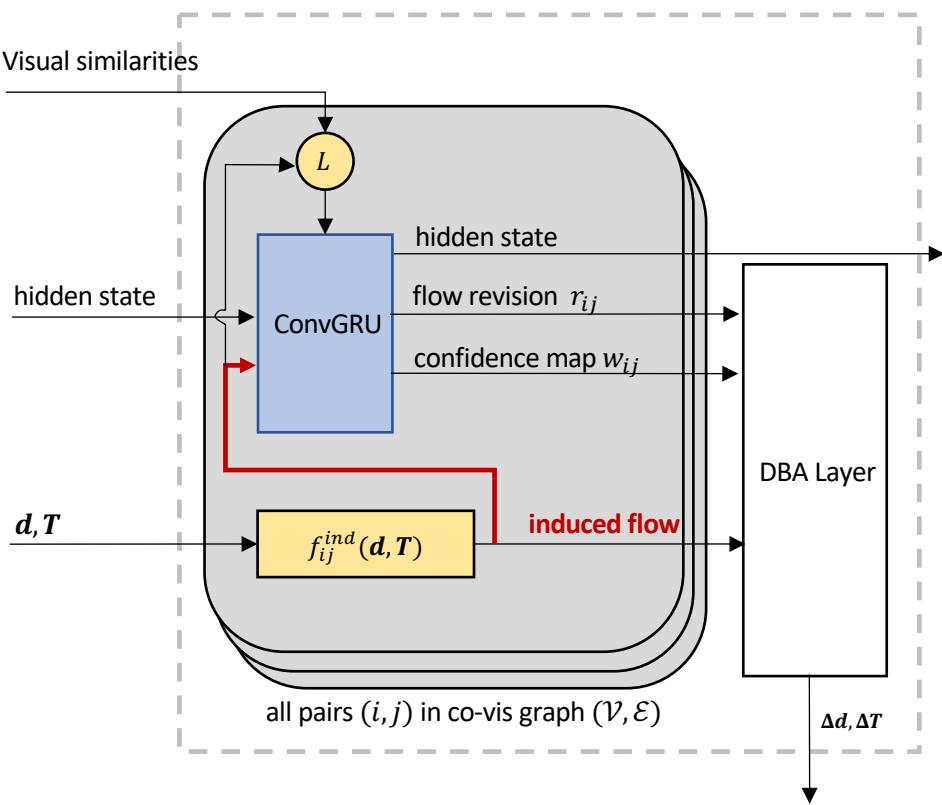
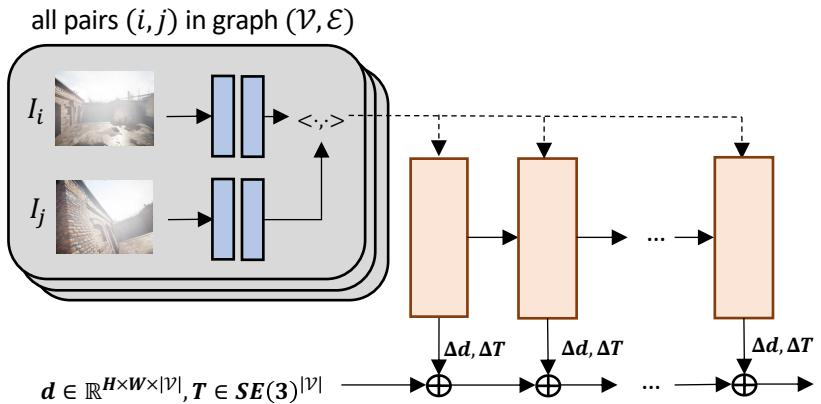
DROID-SLAM: Architecture

- Recurrent Updates + Analytical Layer



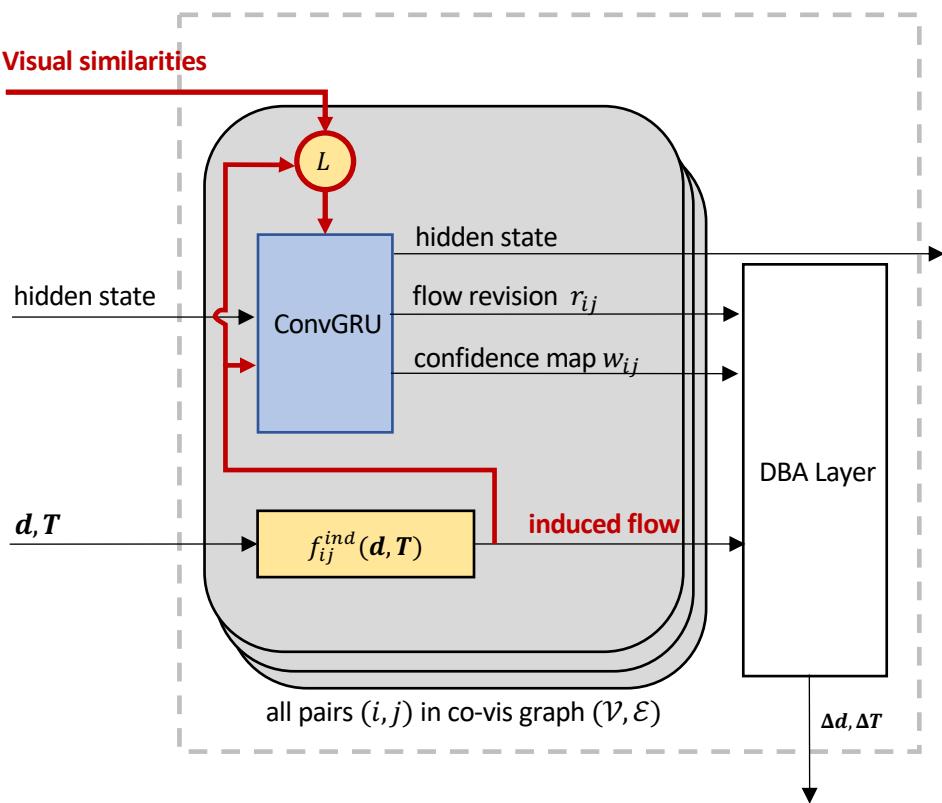
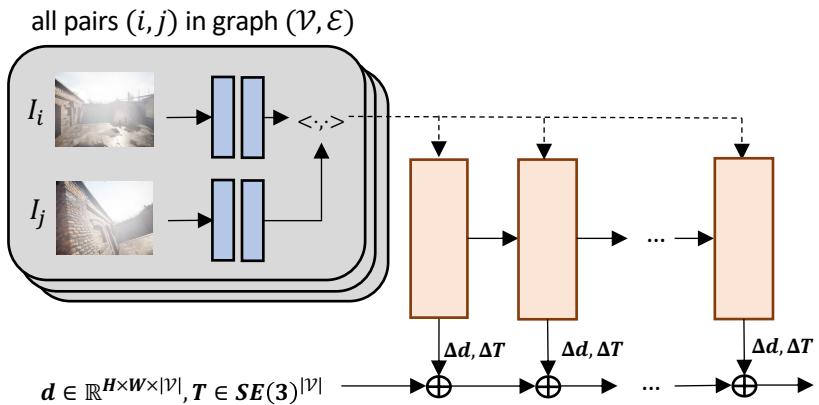
DROID-SLAM: Architecture

- Recurrent Updates + Analytical Layer



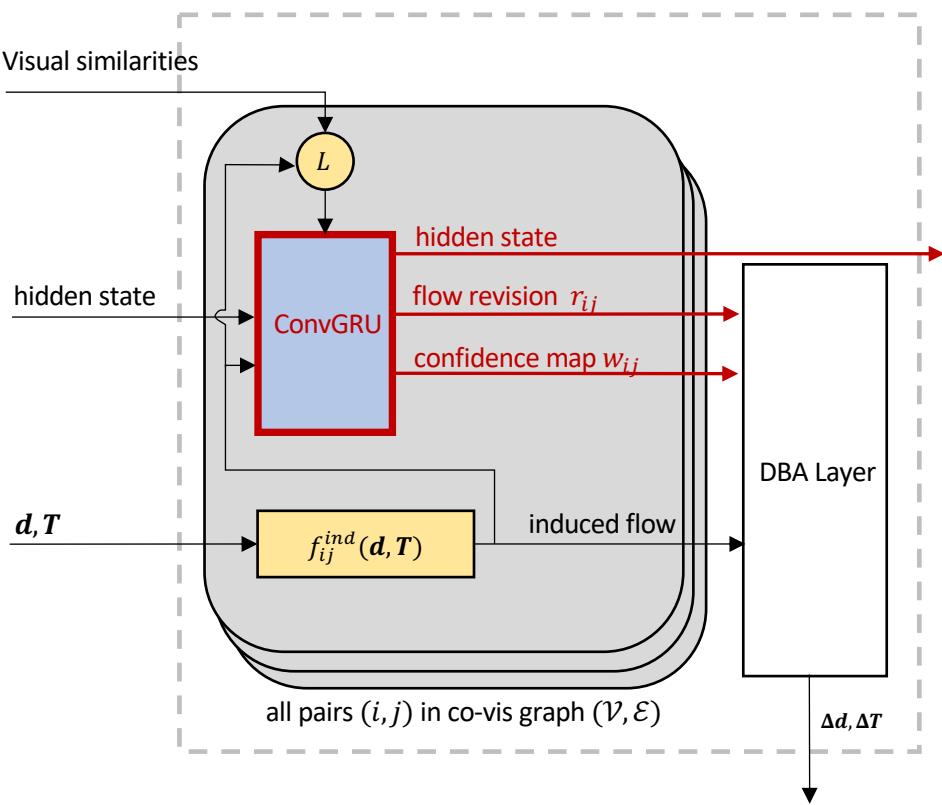
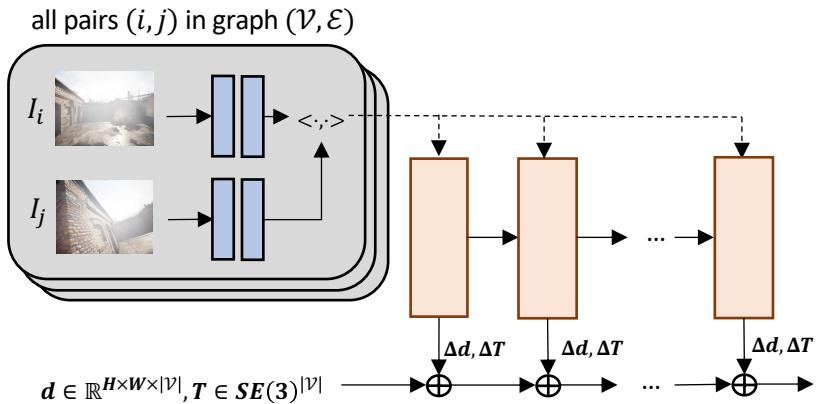
DROID-SLAM: Architecture

- Recurrent Updates + Analytical Layer



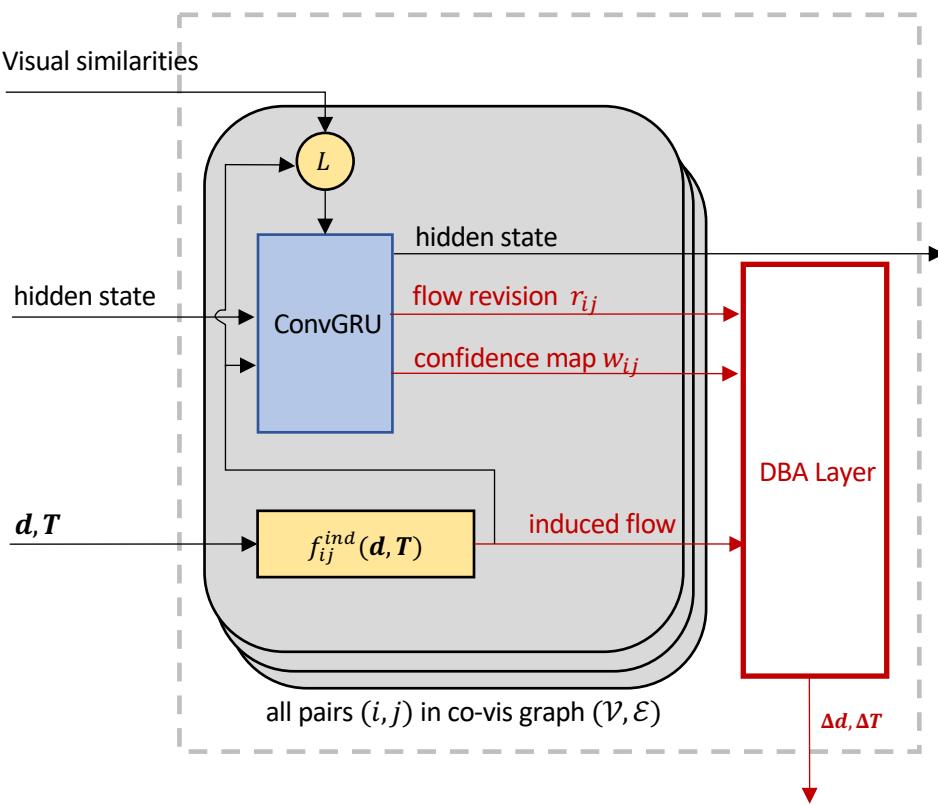
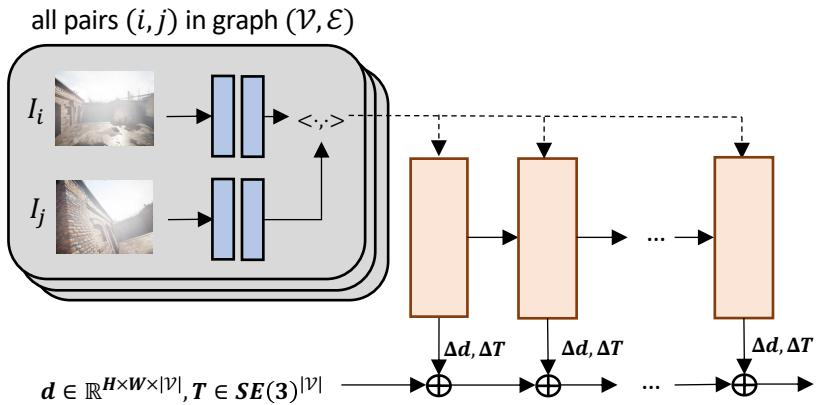
DROID-SLAM: Architecture

- Recurrent Updates + Analytical Layer



DROID-SLAM: Architecture

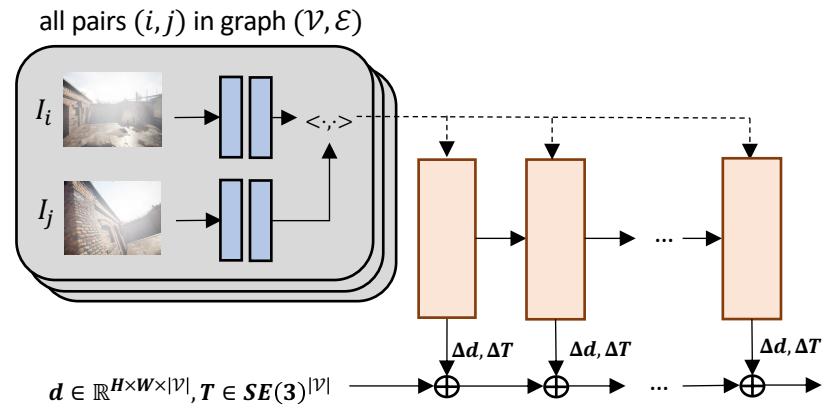
- Recurrent Updates + Analytical Layer



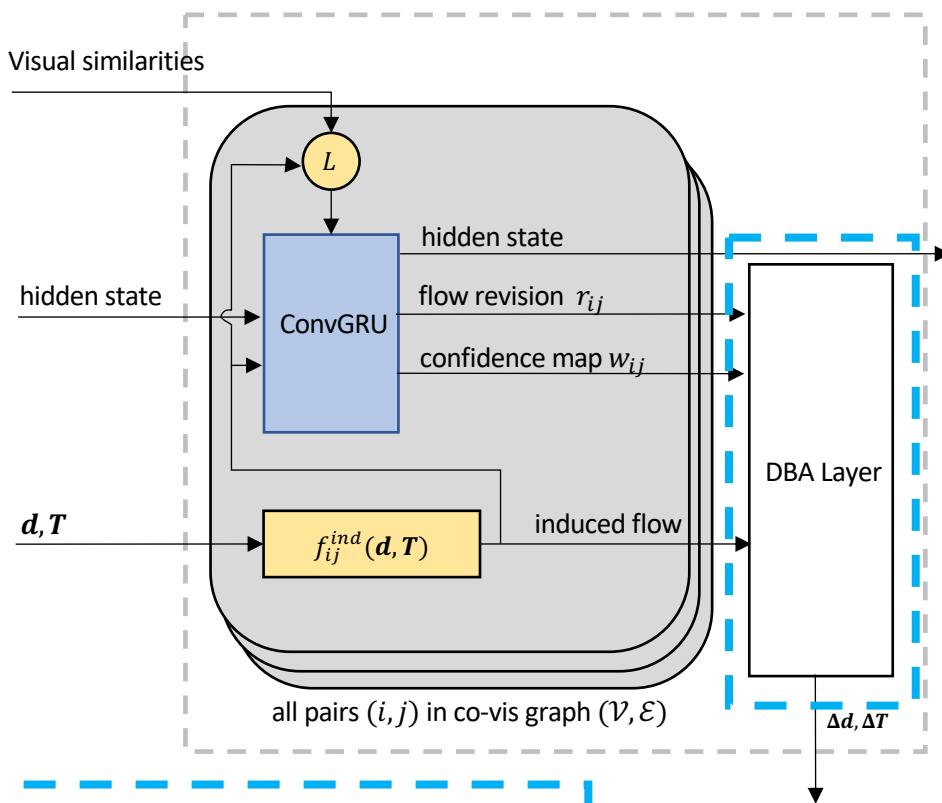
DBA Layer: how to update depth and poses to make induced flow better?

DROID-SLAM: Architecture

- Recurrent Updates + Analytical Layer

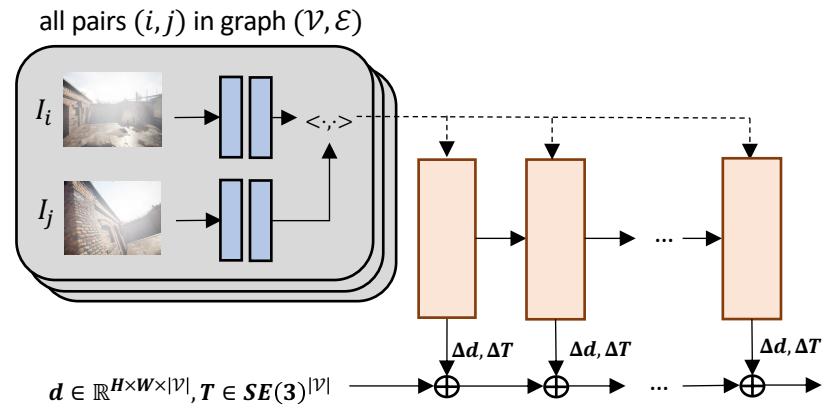


$$\boxed{\min_{\Delta d, \Delta T} \sum_{(i,j) \in \mathcal{E}} \| f_{ij}^{ind}(\mathbf{d}, \mathbf{T}) + r_{ij} - f_{ij}^{ind}(\mathbf{d} + \Delta \mathbf{d}, \mathbf{T} + \Delta \mathbf{T}) \|_{diag(w_{ij})}^2}$$

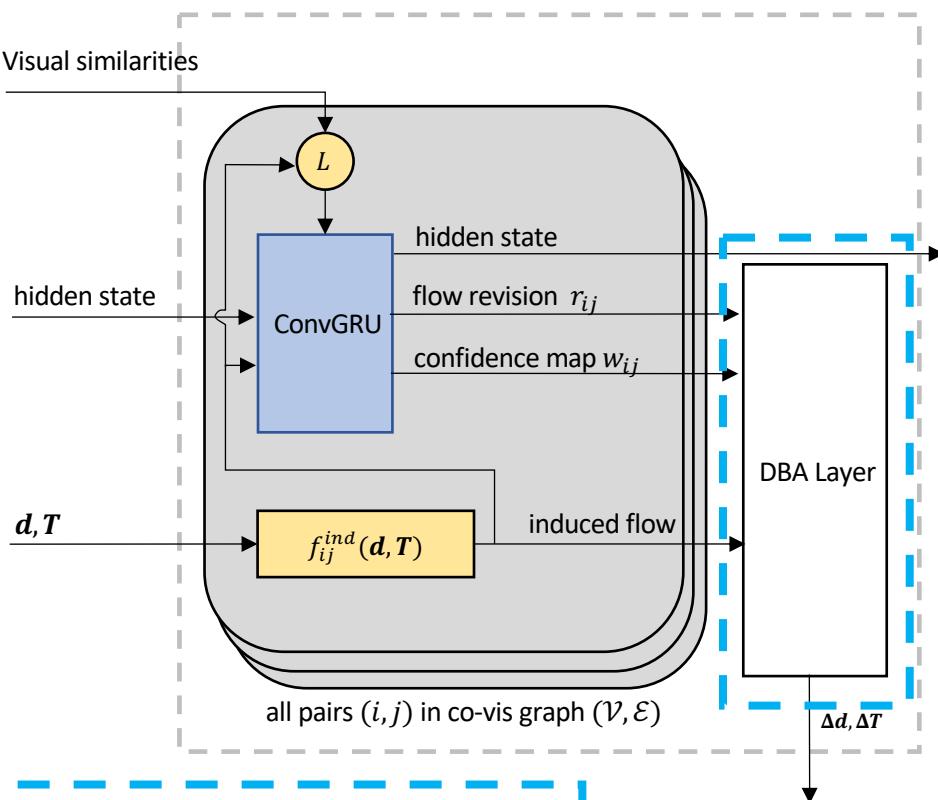


DROID-SLAM: Architecture

- Recurrent Updates + Analytical Layer

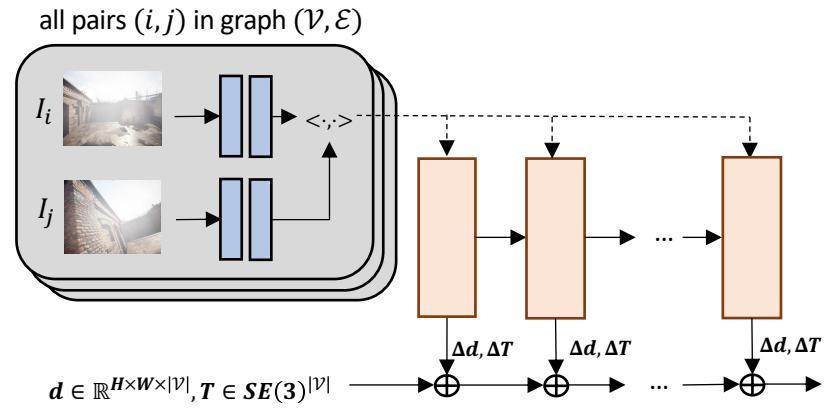


$$\min_{\Delta d, \Delta T} \sum_{(i,j) \in \mathcal{E}} \| f_{ij}^{ind}(d, T) + r_{ij} - f_{ij}^{ind}(d + \Delta d, T + \Delta T) \|_{diag(w_{ij})}^2$$



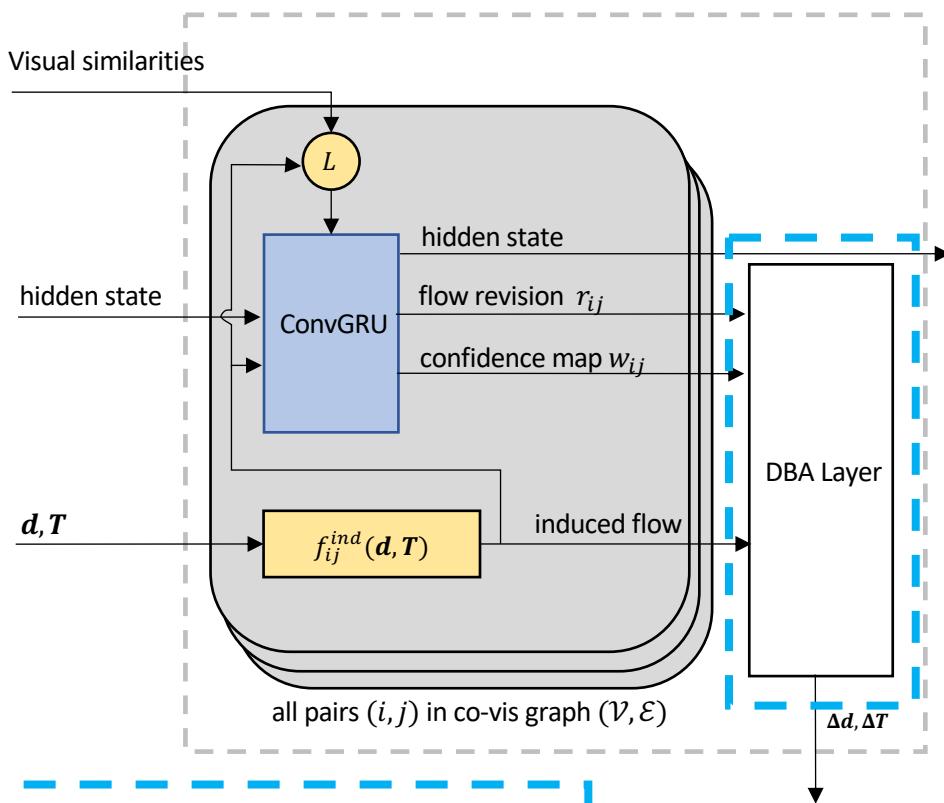
DROID-SLAM: Architecture

- Recurrent Updates + Analytical Layer



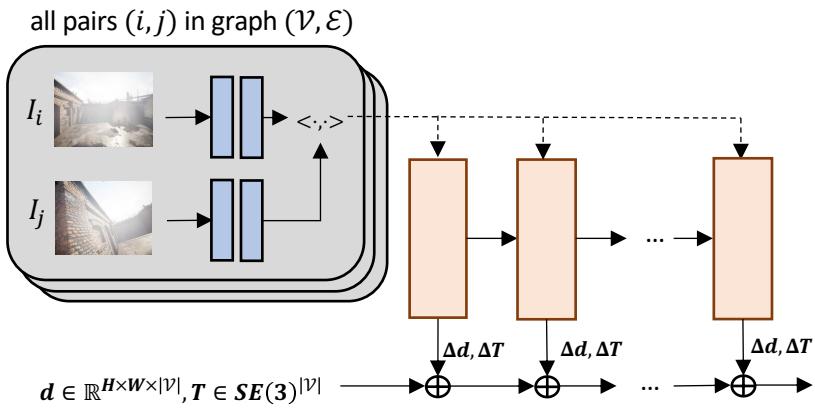
$$\min_{\Delta d, \Delta T} \sum_{(i,j) \in \mathcal{E}} \| f_{ij}^{ind}(\mathbf{d}, \mathbf{T}) + r_{ij} - f_{ij}^{ind}(\mathbf{d} + \Delta d, \mathbf{T} + \Delta T) \|_{diag(w_{ij})}^2$$

Current induced flow between frame i, j



DROID-SLAM: Architecture

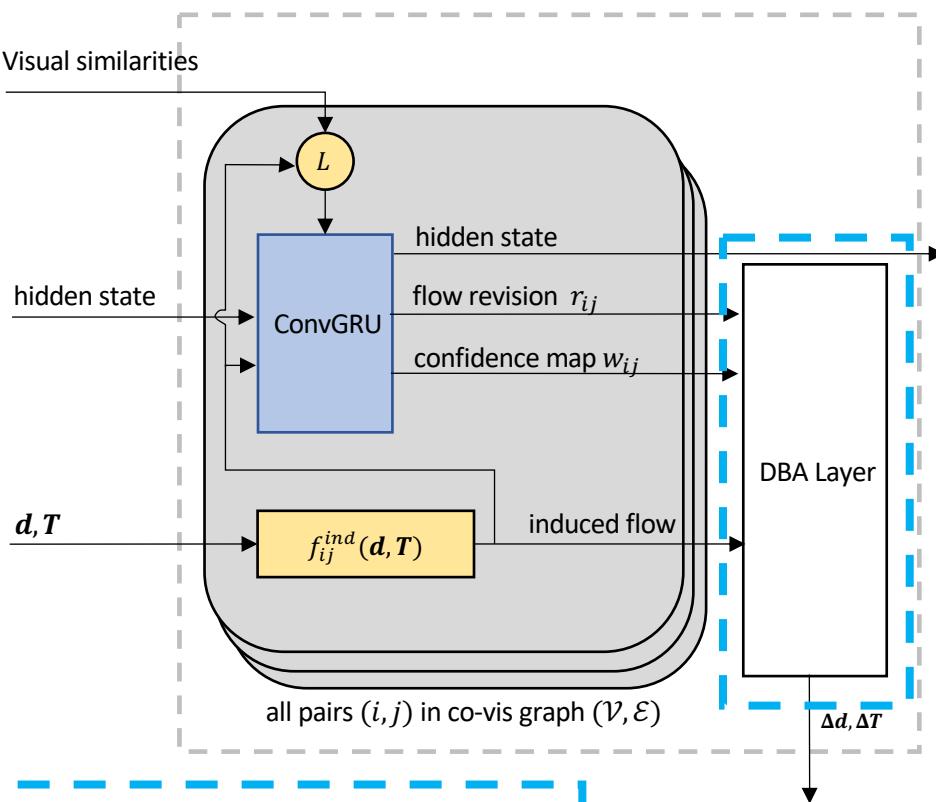
- Recurrent Updates + Analytical Layer



$$\min_{\Delta d, \Delta T} \sum_{(i,j) \in \mathcal{E}} \| f_{ij}^{ind}(\mathbf{d}, \mathbf{T}) + r_{ij} - f_{ij}^{ind}(\mathbf{d} + \Delta d, \mathbf{T} + \Delta T) \|_{diag(w_{ij})}^2$$

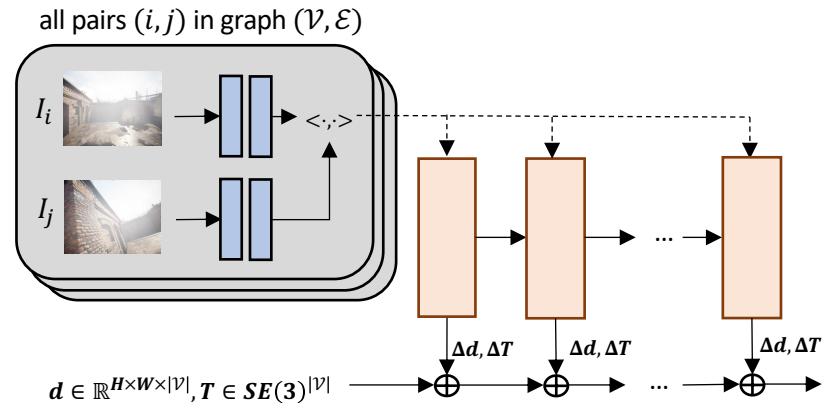
Current induced flow between frame i, j

flow revision



DROID-SLAM: Architecture

- Recurrent Updates + Analytical Layer

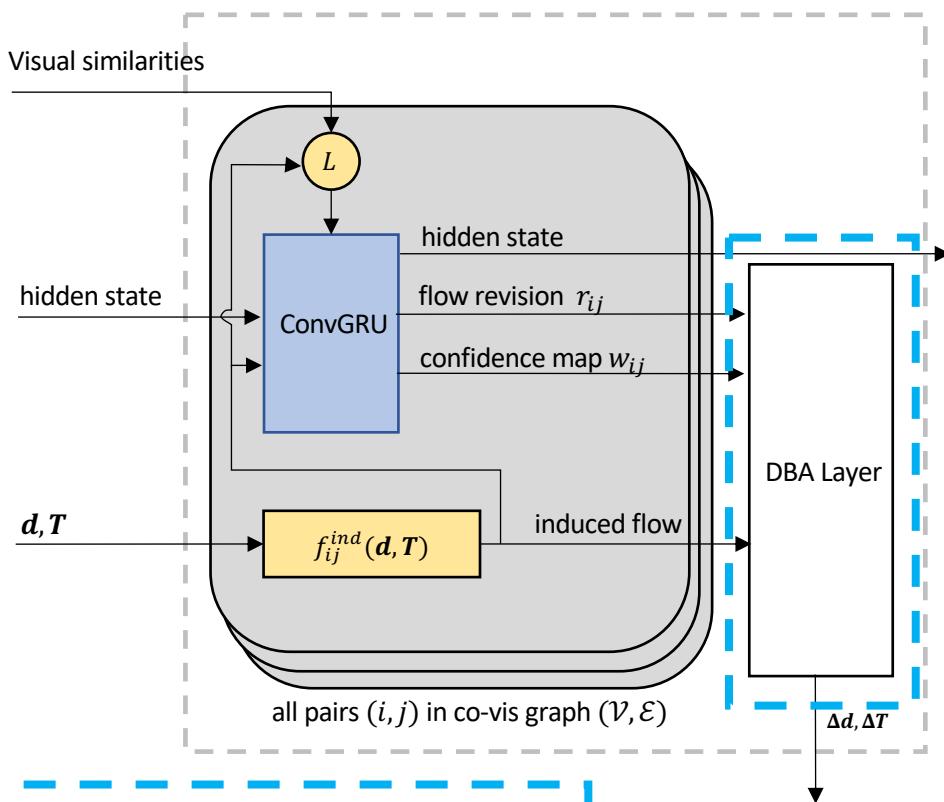


$$\min_{\Delta d, \Delta T} \sum_{(i,j) \in \mathcal{E}} \| f_{ij}^{ind}(\mathbf{d}, \mathbf{T}) + r_{ij} - f_{ij}^{ind}(\mathbf{d} + \Delta \mathbf{d}, \mathbf{T} + \Delta \mathbf{T}) \|_{diag(w_{ij})}^2$$

Current induced flow between frame i, j

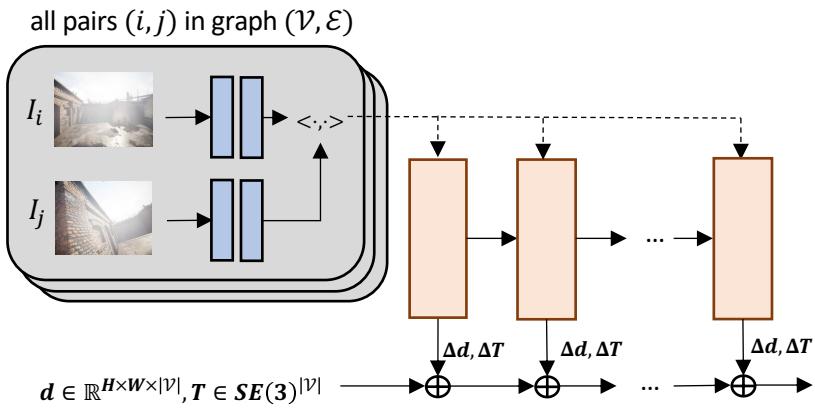
flow revision

new induced flow between frame i, j



DROID-SLAM: Architecture

- Recurrent Updates + Analytical Layer



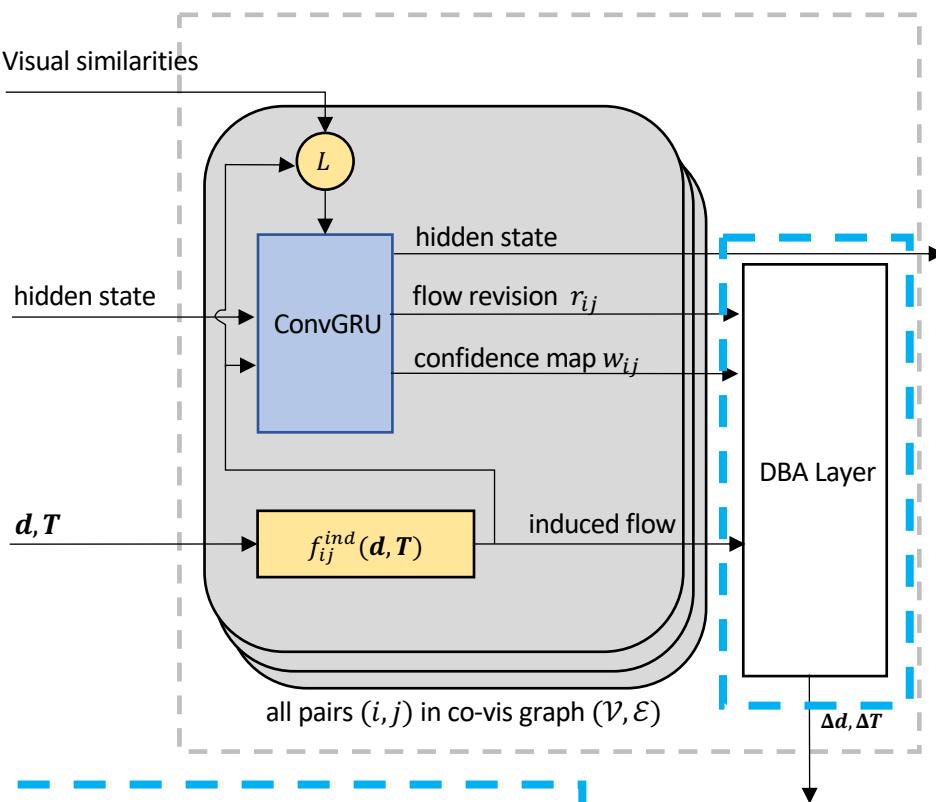
$$\min_{\Delta d, \Delta T} \sum_{(i,j) \in \mathcal{E}} \|f_{ij}^{ind}(d, T) + r_{ij} - f_{ij}^{ind}(d + \Delta d, T + \Delta T)\|_{diag(w_{ij})}^2$$

Current induced flow between frame i, j

flow revision

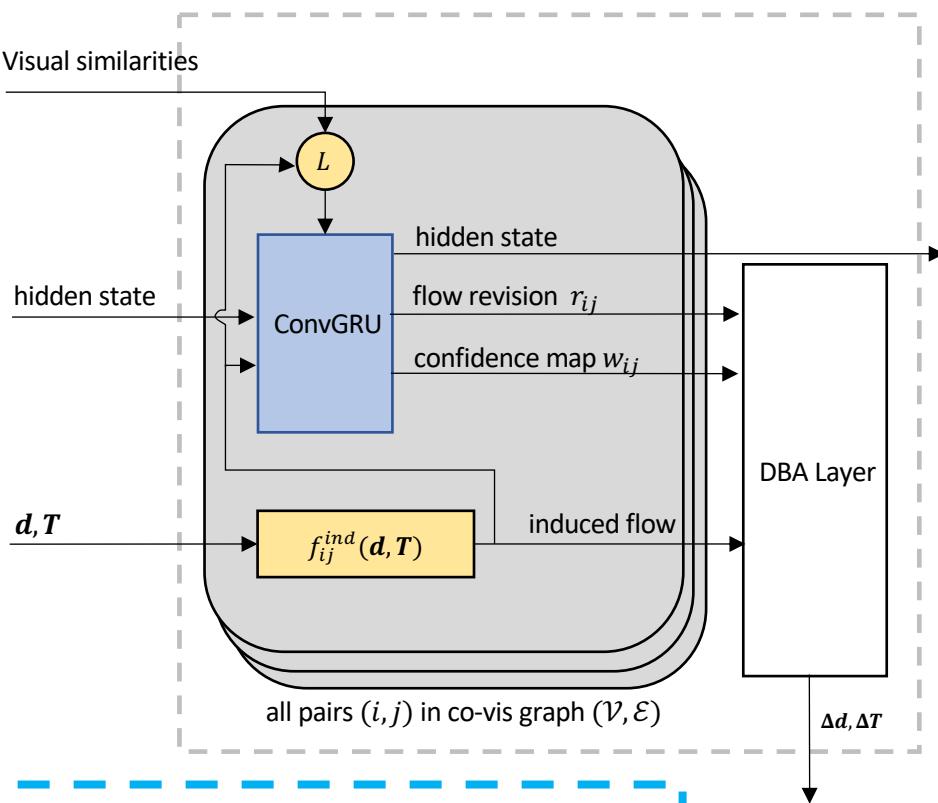
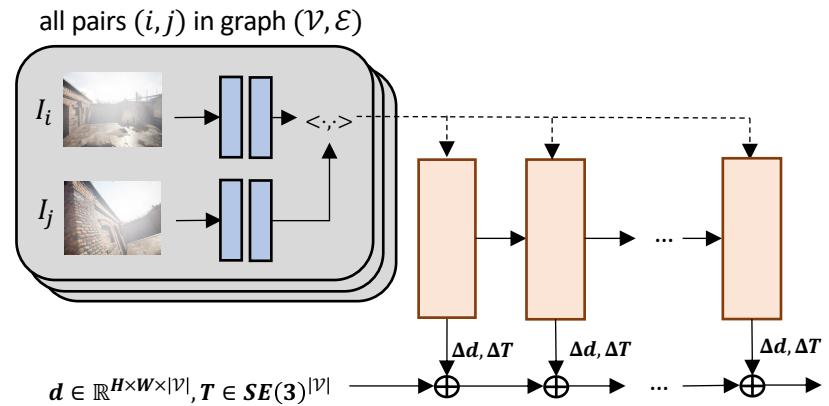
new induced flow between frame i, j

pixel confidence



DROID-SLAM: Architecture

- Recurrent Updates + Analytical Layer



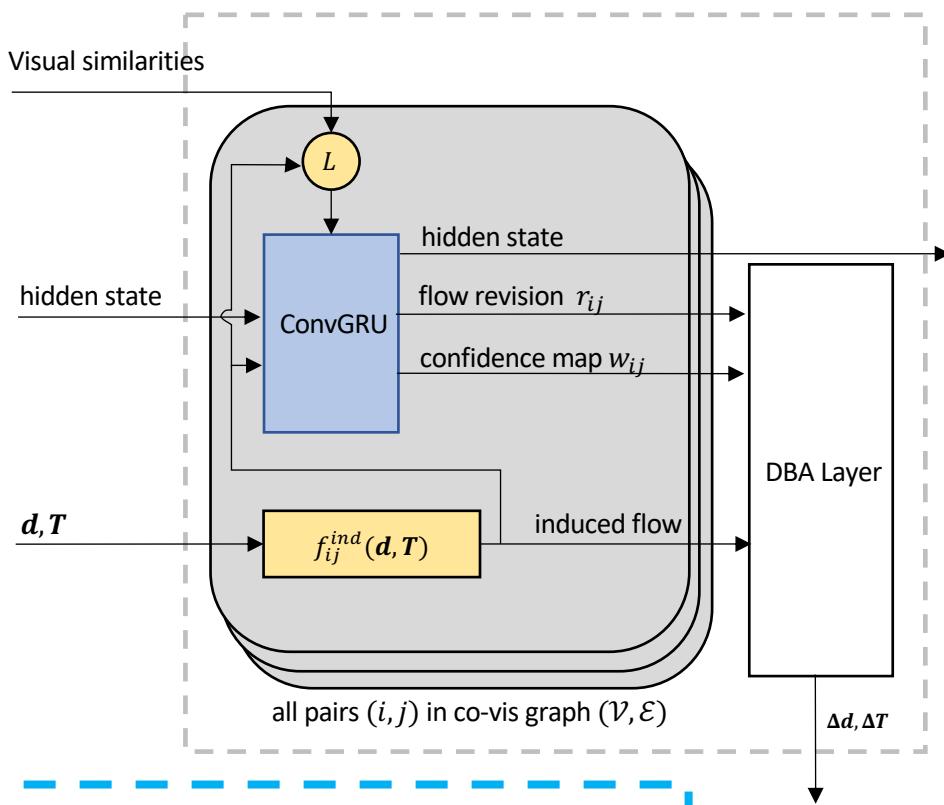
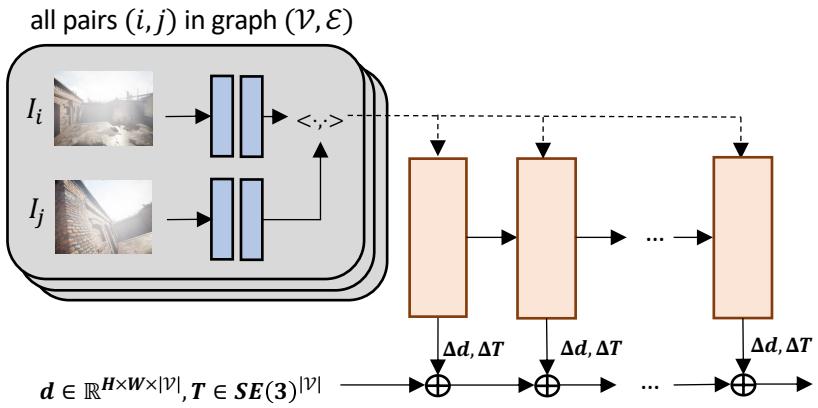
$$\min_{\Delta d, \Delta T} \sum_{(i,j) \in \mathcal{E}} \|f_{ij}^{ind}(d, T) + r_{ij} - f_{ij}^{ind}(d + \Delta d, T + \Delta T)\|_{diag(w_{ij})}^2$$

linearize →

$$\min_{\Delta d, \Delta T} \sum_{(i,j) \in \mathcal{E}} \left\| r_{ij} - \frac{\partial f_{ij}^{ind}(d, T)}{\partial d} \Delta d - \frac{\partial f_{ij}^{ind}(d, T)}{\partial T} \Delta T \right\|_{diag(w_{ij})}^2$$

DROID-SLAM: Architecture

- Recurrent Updates + Analytical Layer



$$\min_{\Delta d, \Delta T} \sum_{(i,j) \in \mathcal{E}} \|f_{ij}^{ind}(\mathbf{d}, \mathbf{T}) + r_{ij} - f_{ij}^{ind}(\mathbf{d} + \Delta \mathbf{d}, \mathbf{T} + \Delta \mathbf{T})\|_{diag(w_{ij})}^2$$

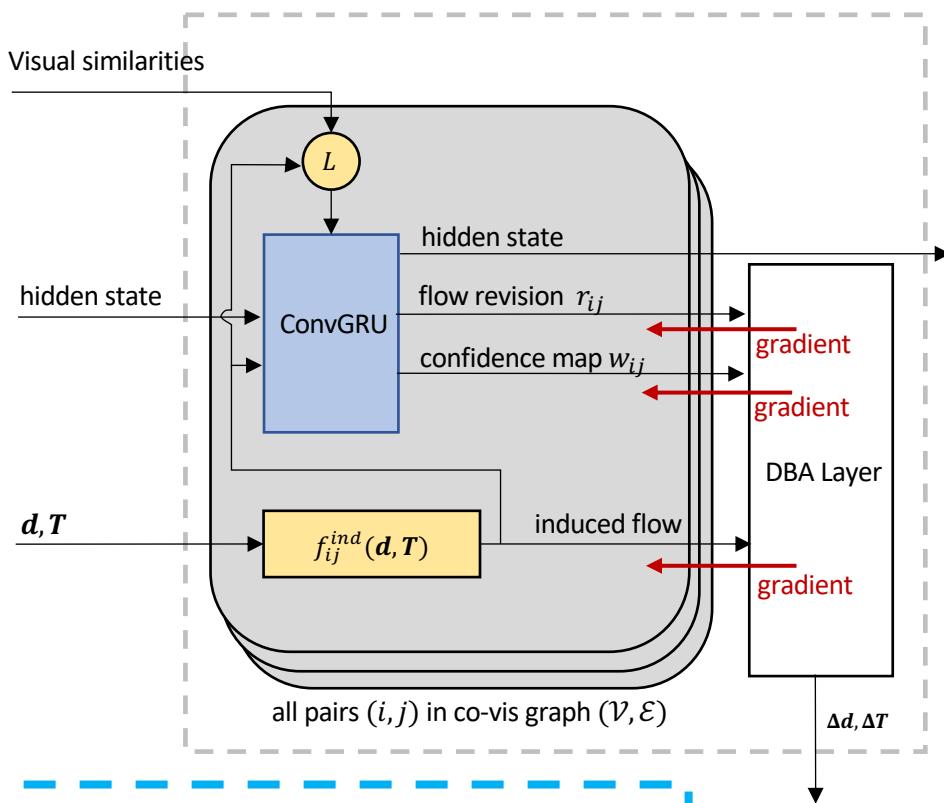
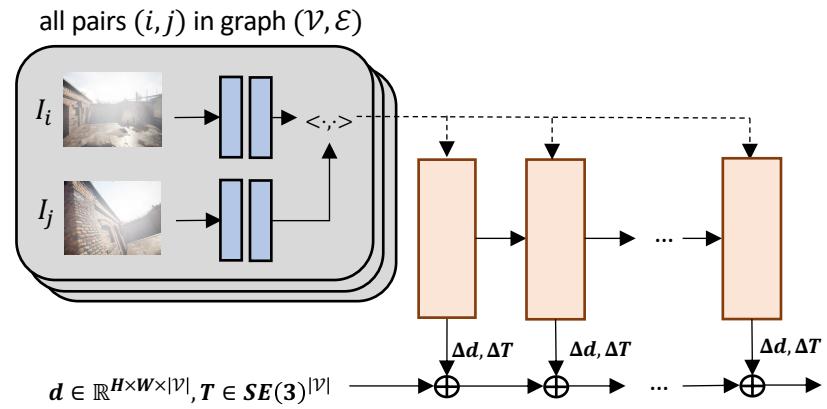
linearize

$$\min_{\Delta d, \Delta T} \sum_{(i,j) \in \mathcal{E}} \left\| r_{ij} - \frac{\partial f_{ij}^{ind}(\mathbf{d}, \mathbf{T})}{\partial \mathbf{d}} \Delta \mathbf{d} - \frac{\partial f_{ij}^{ind}(\mathbf{d}, \mathbf{T})}{\partial \mathbf{T}} \Delta \mathbf{T} \right\|_{diag(w_{ij})}^2$$

Linear least squares
Differentiable closed-form solution
i.e. Gauss-Newton step

DROID-SLAM: Architecture

- Recurrent Updates + Analytical Layer



$$\min_{\Delta d, \Delta T} \sum_{(i, j) \in \mathcal{E}} \|f_{ij}^{ind}(\mathbf{d}, \mathbf{T}) + r_{ij} - f_{ij}^{ind}(\mathbf{d} + \Delta \mathbf{d}, \mathbf{T} + \Delta \mathbf{T})\|_{diag(w_{ij})}^2$$

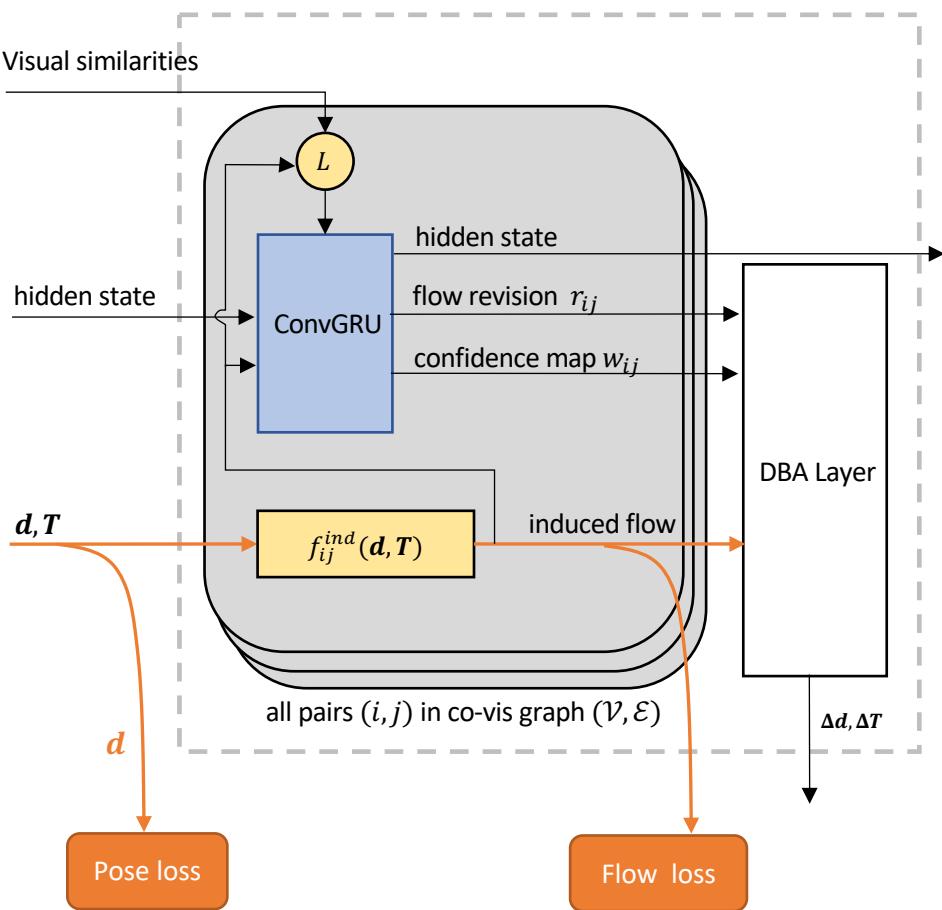
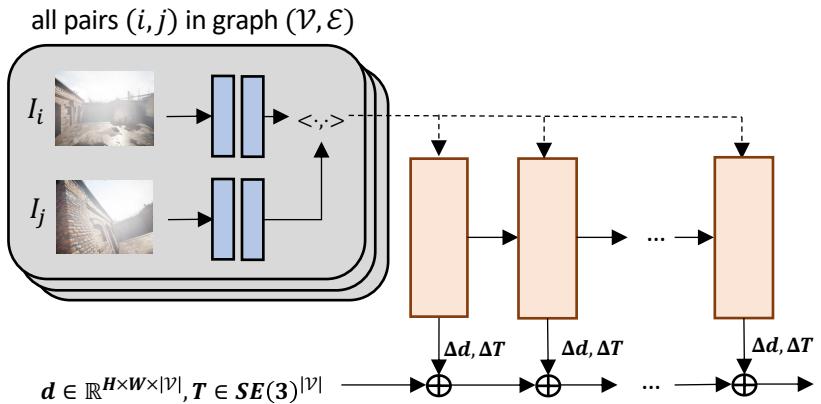
linearize

$$\min_{\Delta d, \Delta T} \sum_{(i, j) \in \mathcal{E}} \left\| r_{ij} - \frac{\partial f_{ij}^{ind}(\mathbf{d}, \mathbf{T})}{\partial \mathbf{d}} \Delta \mathbf{d} - \frac{\partial f_{ij}^{ind}(\mathbf{d}, \mathbf{T})}{\partial \mathbf{T}} \Delta \mathbf{T} \right\|_{diag(w_{ij})}^2$$

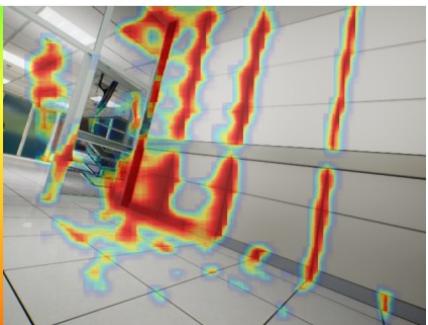
Linear least squares
Differentiable closed-form solution
i.e. Gauss-Newton step

DROID-SLAM: Architecture

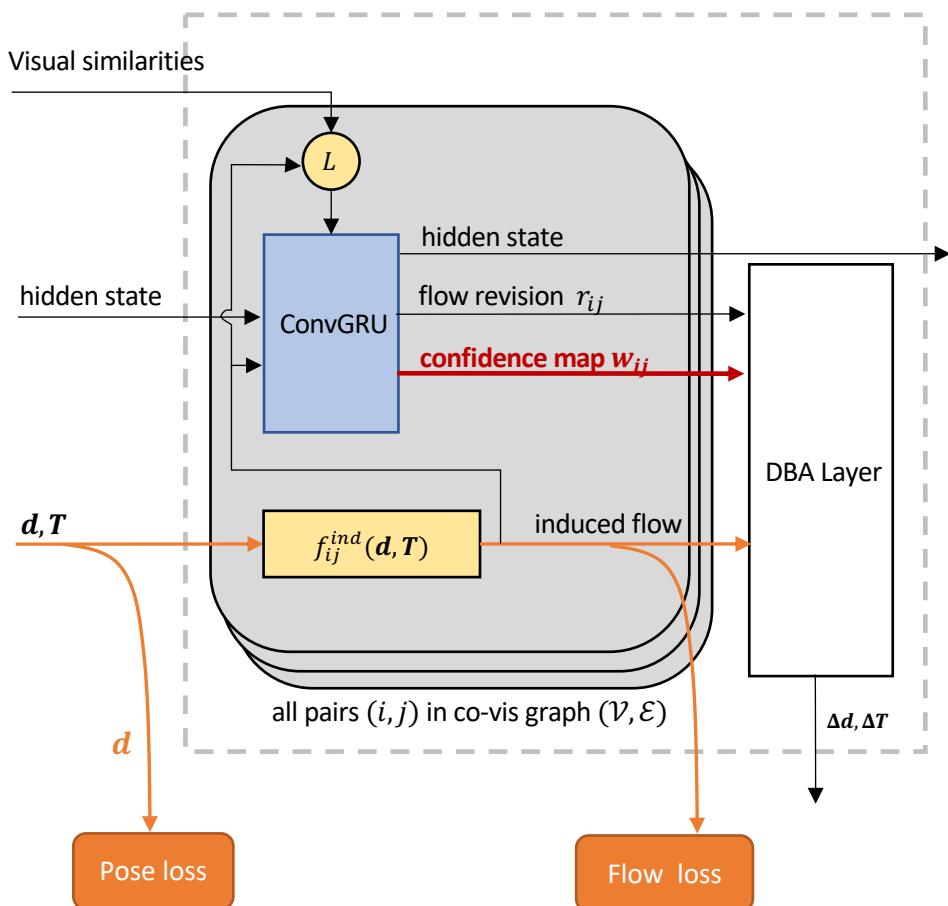
- Recurrent Updates + Analytical Layer



horizontal flow confidence

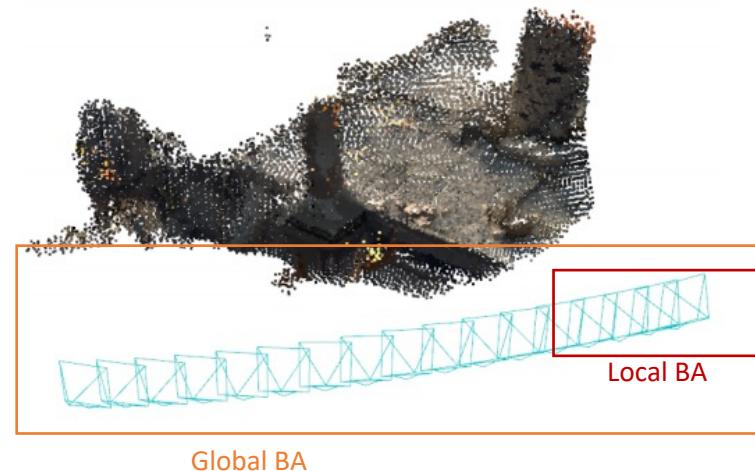


No direct supervision



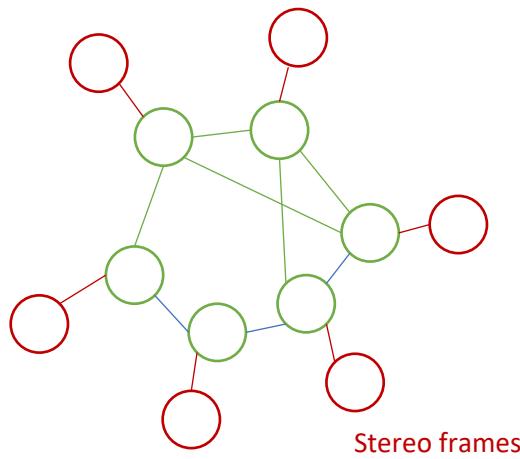
DROID-SLAM: Full System

- **Frontend:** feature extraction, local bundle adjustment
- **Backend:** global bundle adjustment
- **Building covisibility graph:** thresholding inter-frame flow magnitude
- Real time on 2 3090 GPUs (with custom GPU kernels)
- Trained only on monocular input



DROID-SLAM: extension to stereo and RGB-D

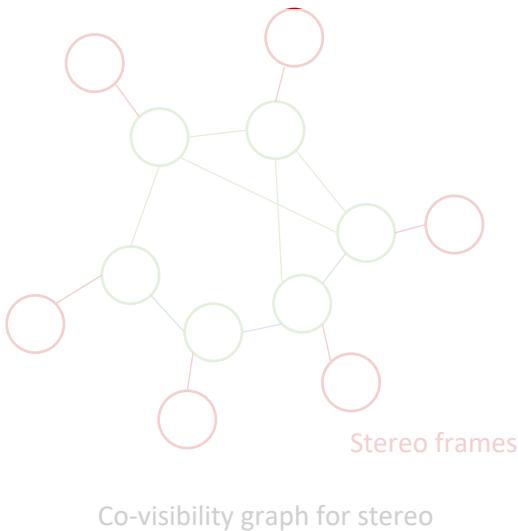
- **Stereo:** double the frames in graph, fixing relative poses between left & right frames



Co-visibility graph for stereo

DROID-SLAM: extension to stereo and RGB-D

- Stereo: double the frames in graph, fixing relative poses between left & right frames
- **RGB-D:** still estimate depth, but use sensor depth as a prior in DBA layer
 - Sensor depth can have noise and missing observations



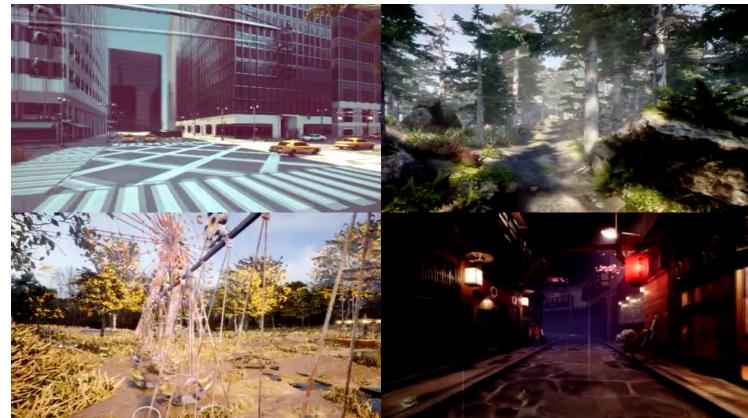
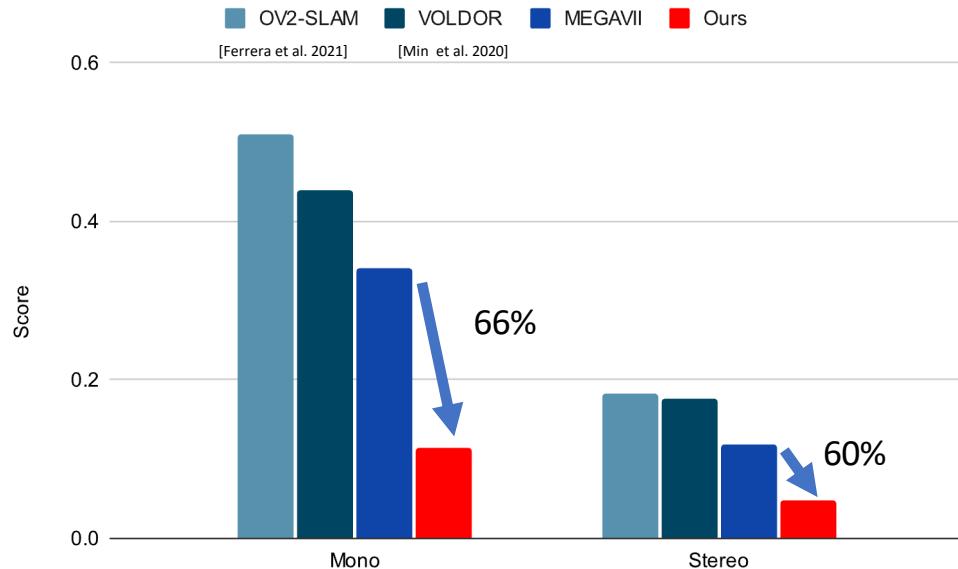
$$\min_{\Delta d, \Delta T} \sum_{(i,j) \in \mathcal{E}} \|f_{ij}^{ind}(\mathbf{d}, \mathbf{T}) + r_{ij} - f_{ij}^{ind}(\mathbf{d} + \Delta \mathbf{d}, \mathbf{T} + \Delta \mathbf{T})\|_{diag(w_{ij})}^2 + \|\mathbf{d} + \Delta \mathbf{d} - \hat{\mathbf{d}}\|^2$$

Sensor depth $\hat{\mathbf{d}}$ as a prior

DBA layer

No retraining needed for stereo or RGB-D

TartanAir – SLAM Challenge [Wang et al. 2020]



- Our system trained on TartanAir (training split) with monocular input
- **66%** lower error on monocular, **60%** lower error on stereo, **16x** faster

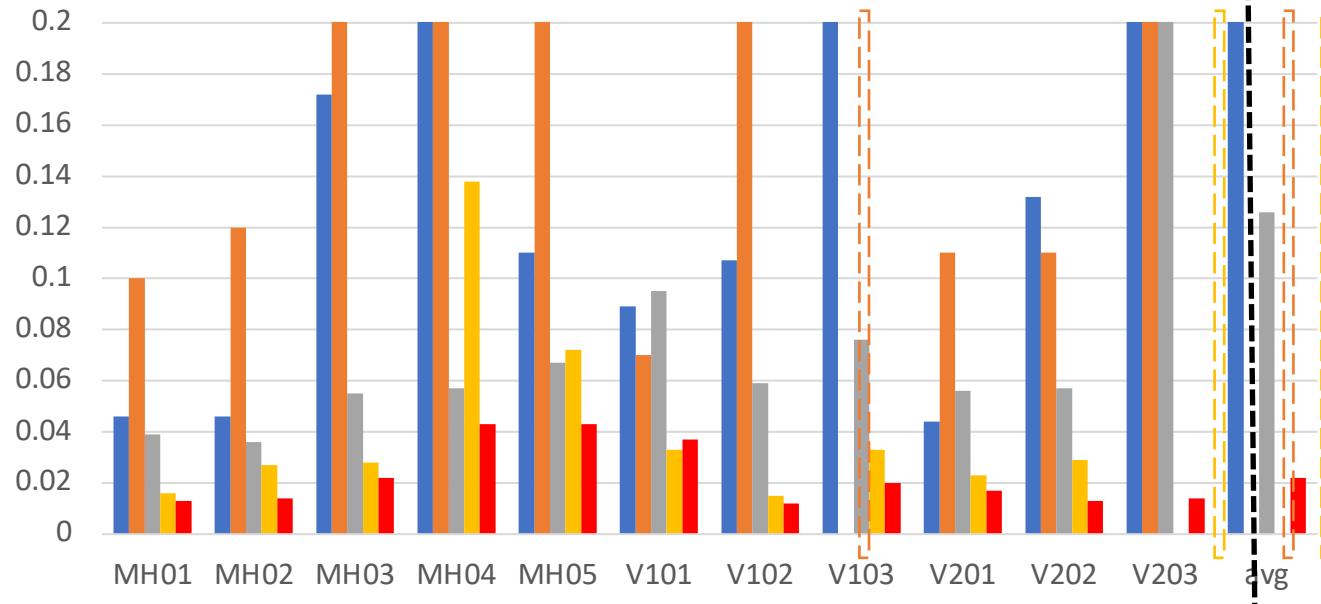
EuRoC MAV (Monocular)

[Burri et al. 2016]



V203

ATE



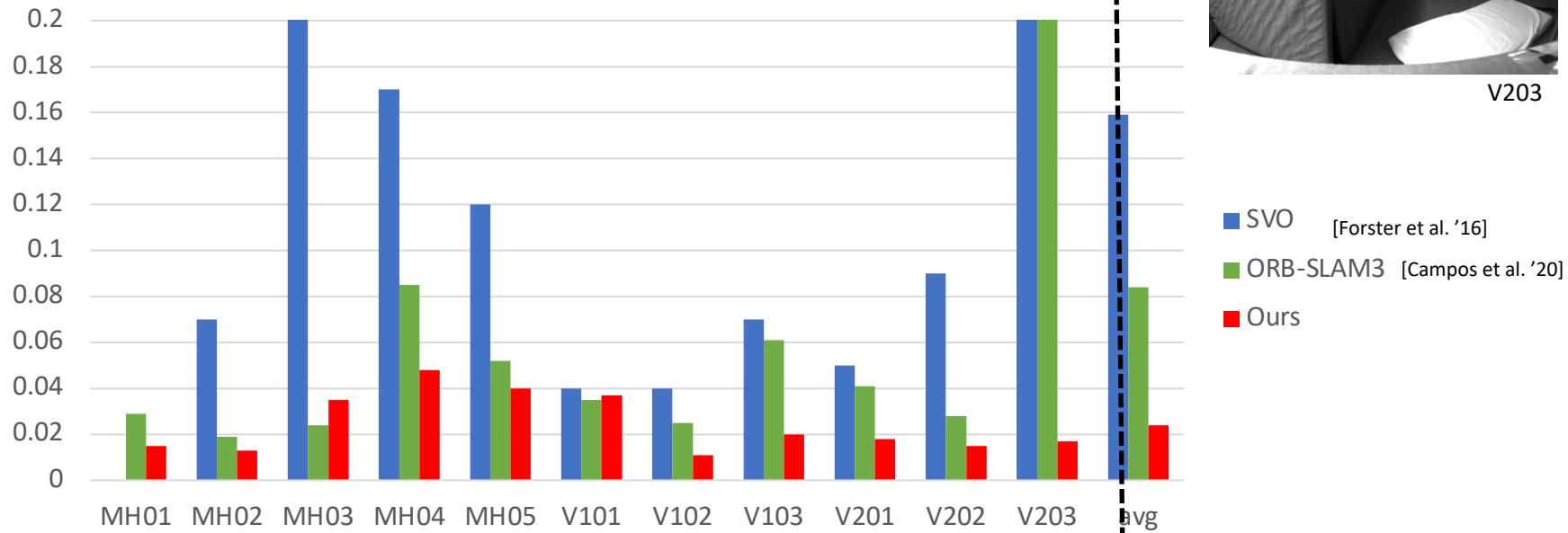
- Our system trained only on TartanAir
- **82% less error** among methods with zero failures
- **43% less error** than ORB-SLAM3 on its successful sequences

EuRoC MAV (Stereo)

[Burri et al. 2016]



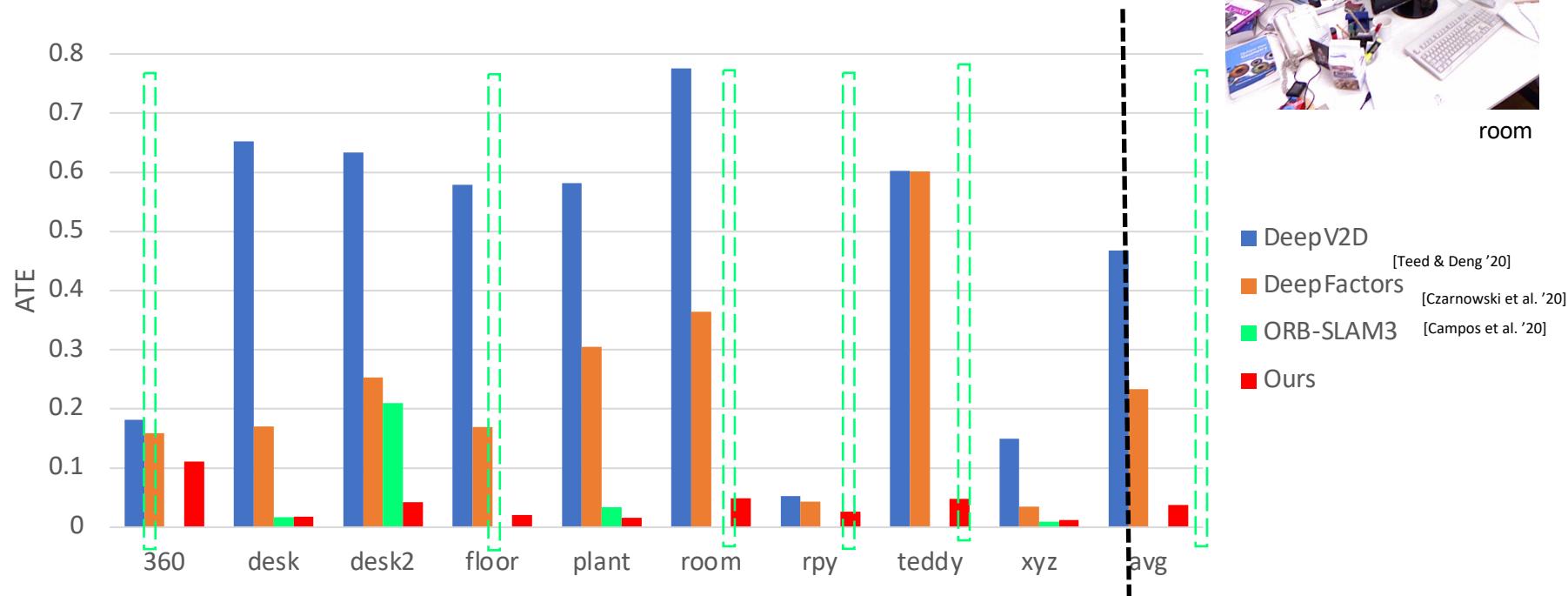
ATE



- Our system trained only on monocular TartanAir
- **71% less error** than ORB-SLAM3

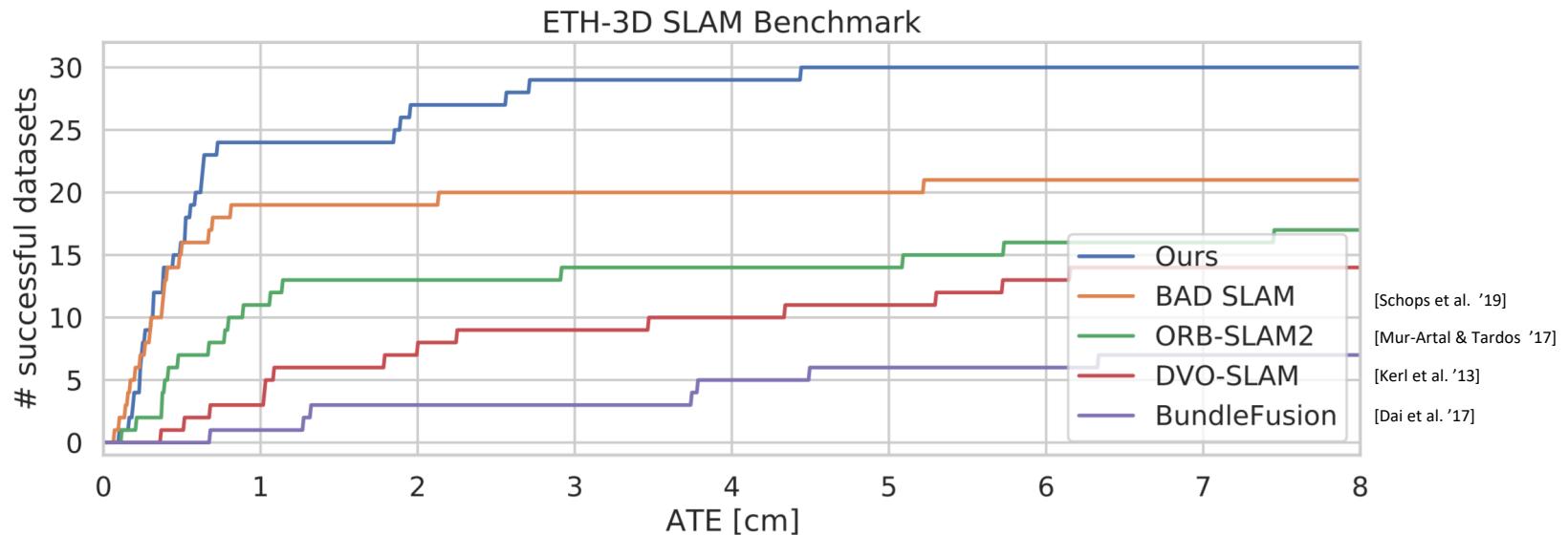
TUM-RGBD (Monocular)

[Sturm et al. 2012]



- Our system trained only on monocular TartanAir
- ***83% lower error*** than DeepFactors

ETH-3D SLAM (RGB-D)



- Our system trained only on monocular TartanAir
- Ranks 1st, 35% better AUC
- Successfully track 30/32 RGB-D datasets, next best method tracks 19/32

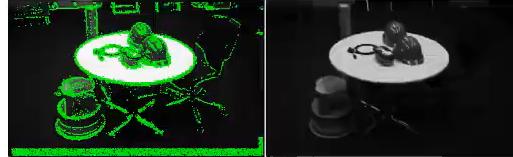
Strong Generalization

All results, across datasets and modalities (monocular, stereo, RGB-D),
are by *a single model*, trained only once, on synthetic data.

[ORB-SLAM3, Campos et al]



[DSO, Engel et al]



Tanks and Temples



iPhone 12



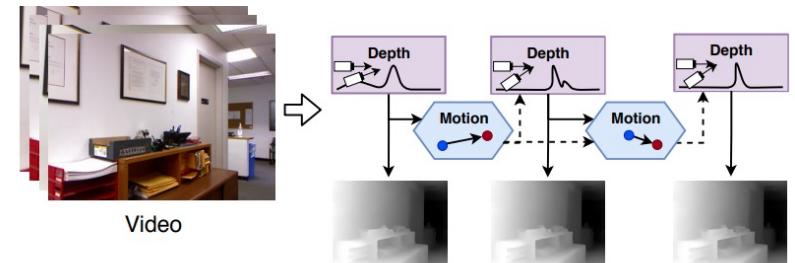
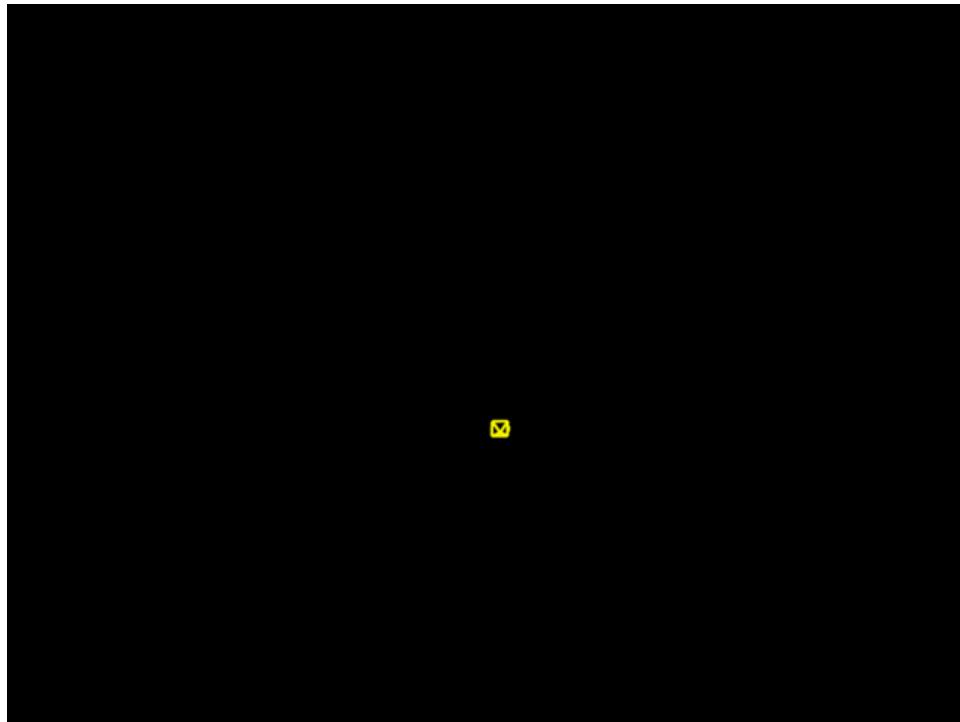
iPhone 12



iPhone 12



DeepV2D [ICLR 2020]: Video to Depth

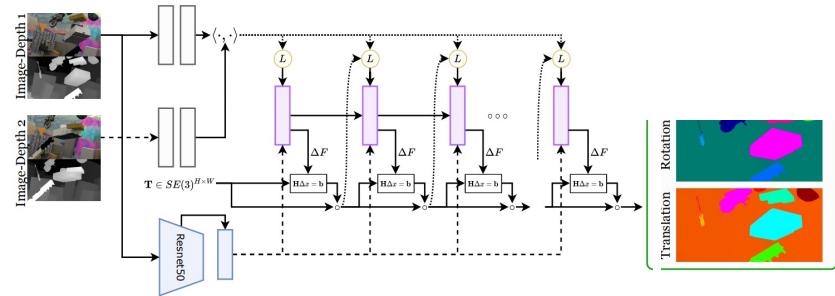


Recurrent unit + analytical layer (PnP)

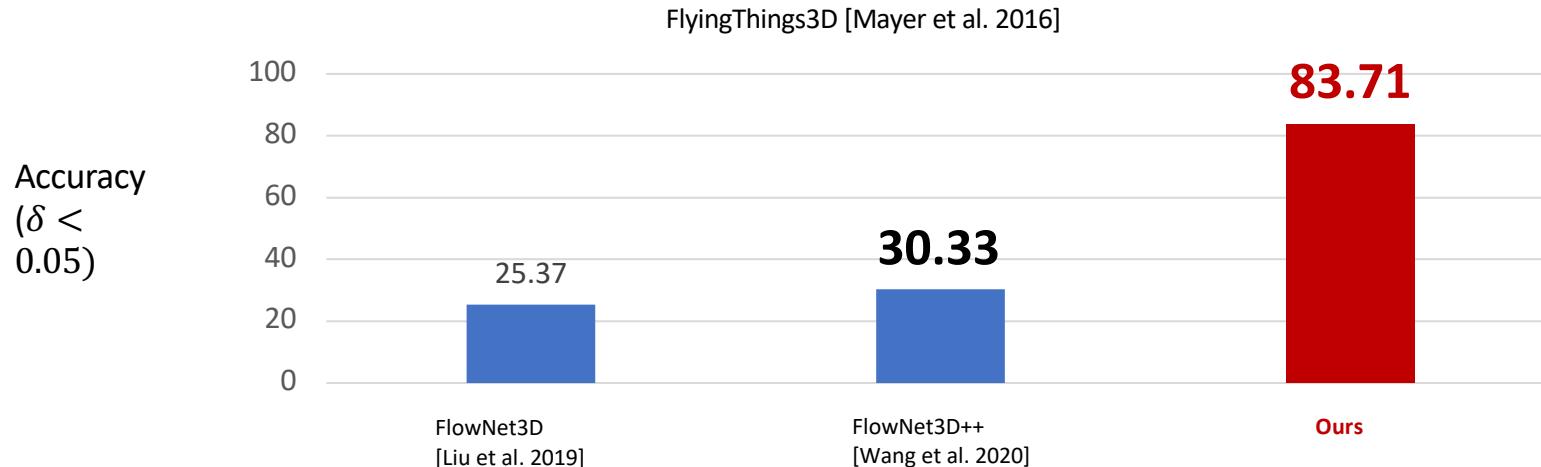
53% less error over prior SOTA on NYU Depth

RAFT-3D [CVPR 2021]: Scene Flow

Input: RGB-D video of dynamic scene
Output: per-pixel 3D motion



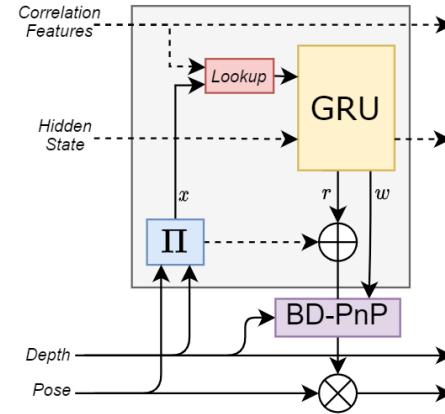
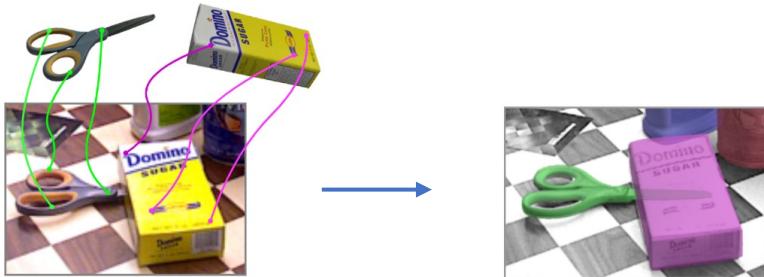
Recurrent unit + analytical layer (DBA w/ soft pixel grouping)



6D Multi-Object Pose [Lipson, Teed, Deng, CVPR 2022]

Input: RGB-D + known 3D models

Output: 6D object poses



Recurrent unit + analytical layer (Bidirectional PnP)

SOTA on the BOP benchmark (YCB-V, T-LESS, LINEMOD-Occluded)

CS 231N Lecture 16: 3D Vision





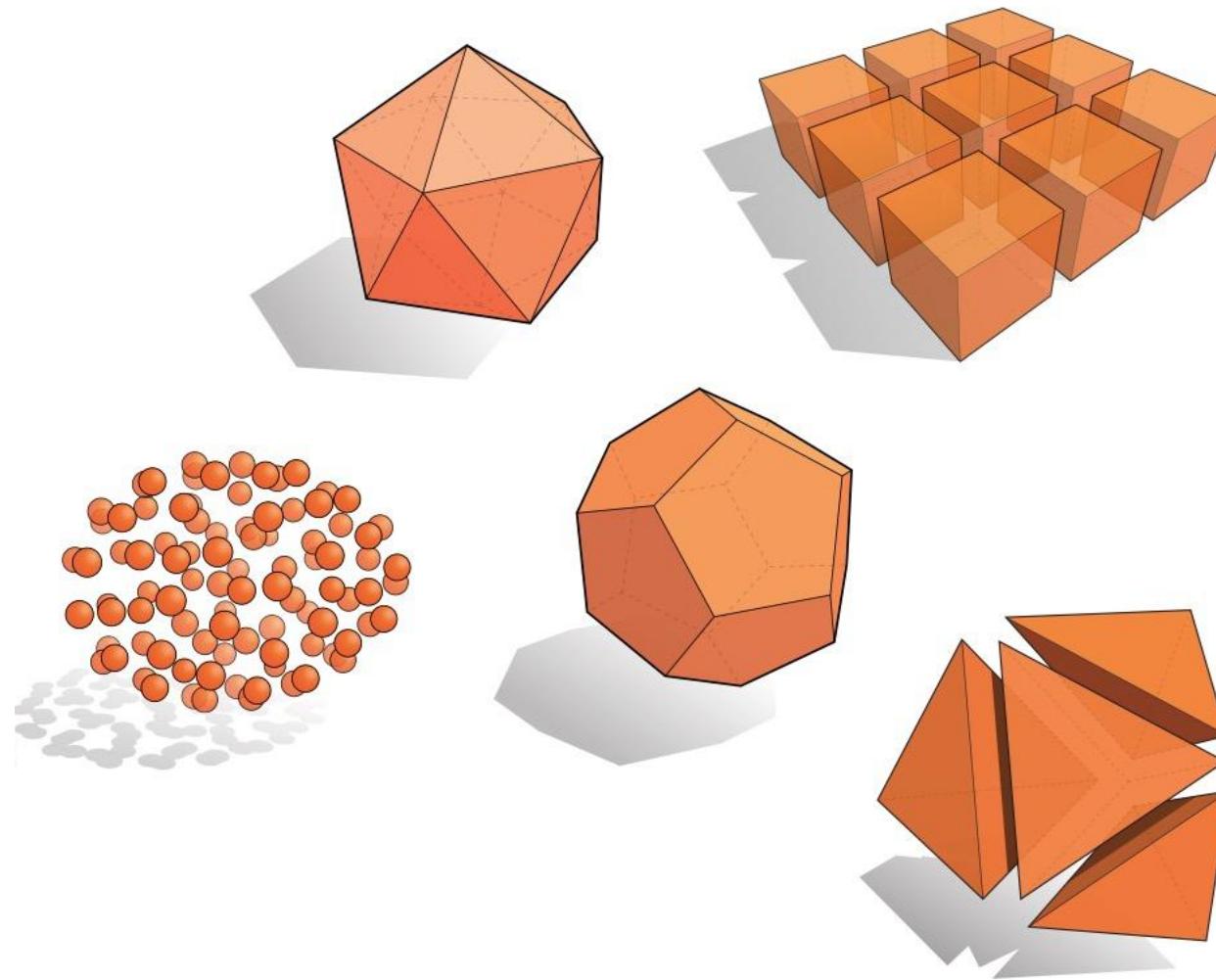


NATIONAL
GEOGRAPHIC

Photograph by Adriana Franco, Your Shot

© COPYRIGHT ADRIANA FRANCO. ALL RIGHTS RESERVED.

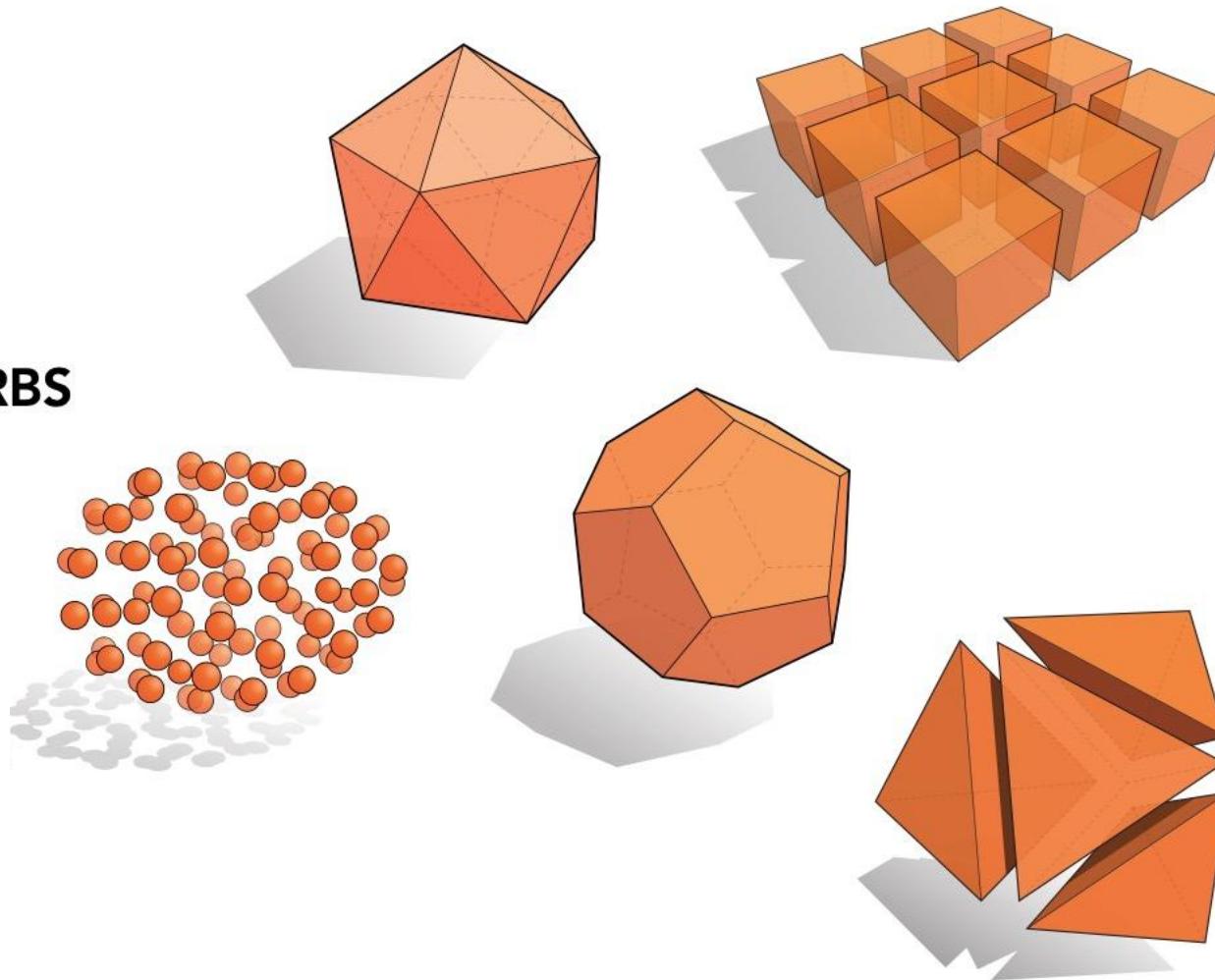
Many Ways to Represent Geometry



Many Ways to Represent Geometry

Explicit

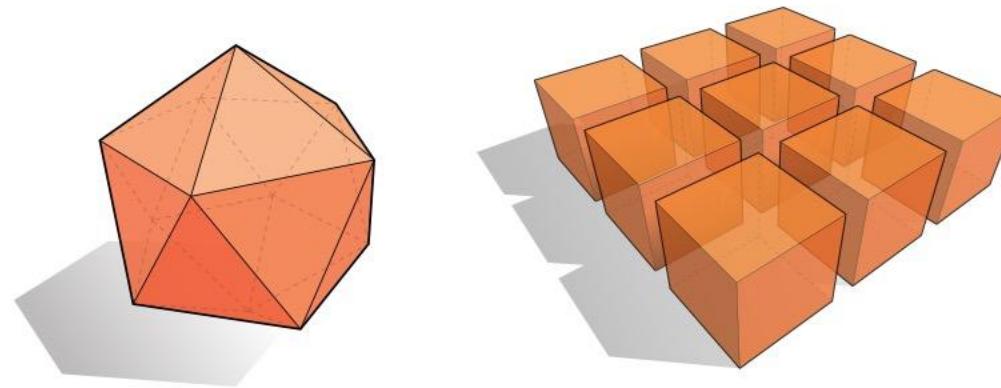
- point cloud
- polygon mesh
- subdivision, NURBS
- ...



Many Ways to Represent Geometry

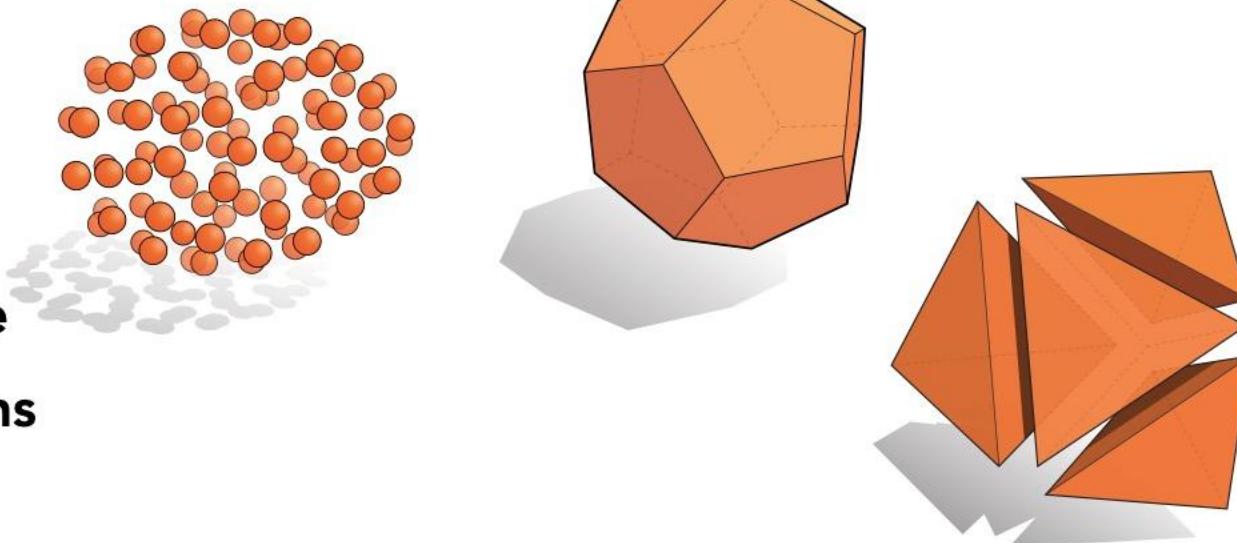
Explicit

- point cloud
- polygon mesh
- subdivision, NURBS
- ...



Implicit

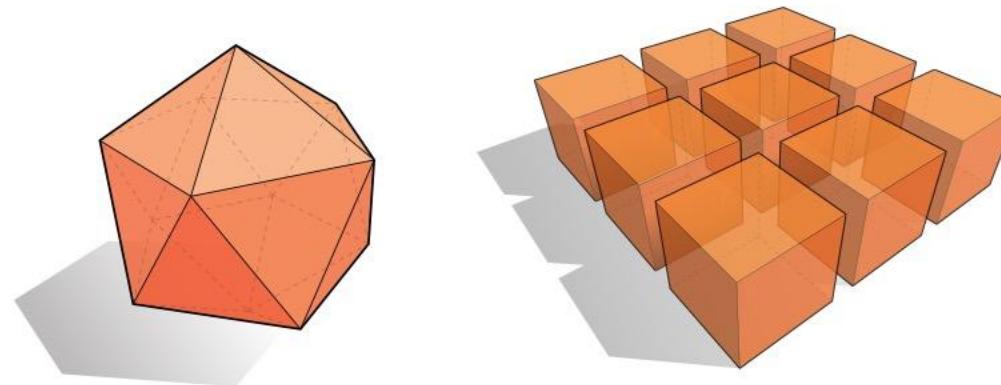
- level sets
- algebraic surface
- distance functions
- ...



Many Ways to Represent Geometry

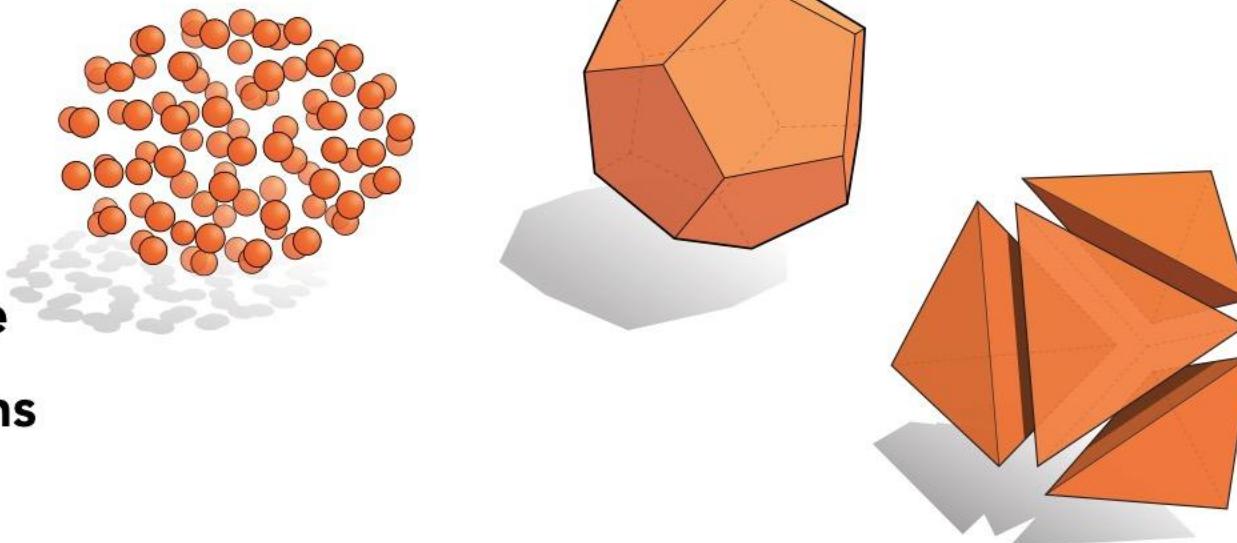
Explicit

- point cloud
- polygon mesh
- subdivision, NURBS
- ...



Implicit

- level sets
- algebraic surface
- distance functions
- ...

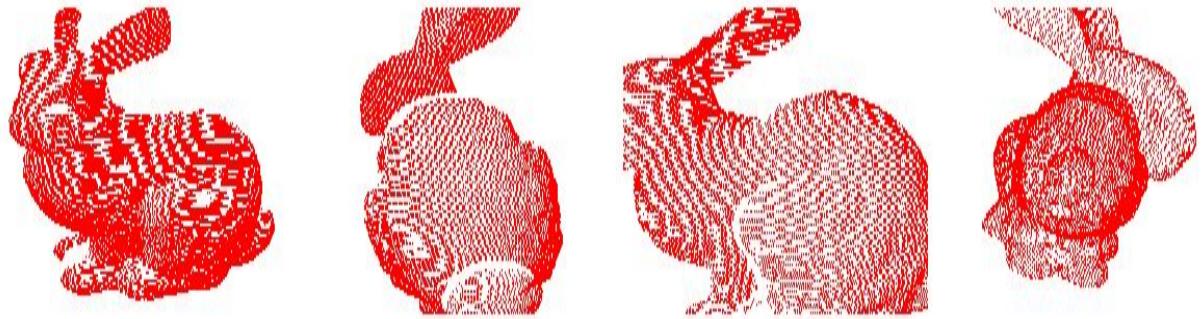


Each choice best suited to a different task/type of geometry

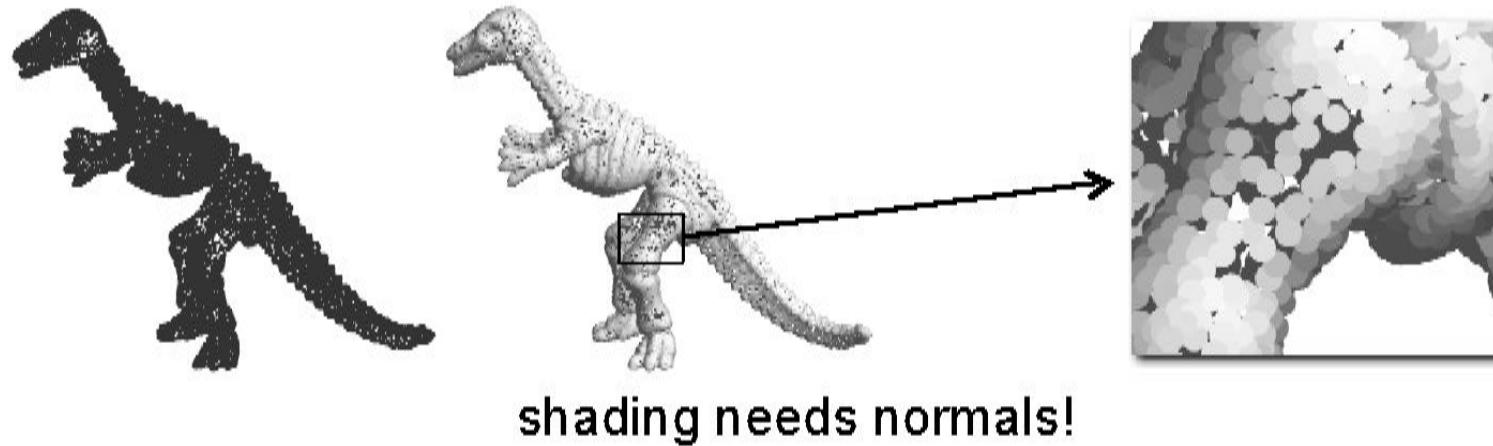
Representation Considerations

- Needs to be stored in the computer
- Creation of new shapes
 - Input metaphors, interfaces...
- Operations
 - Editing, simplification, smoothing, filtering, repairing...
- Rendering
 - Rasterization, ray tracing...
- Animation

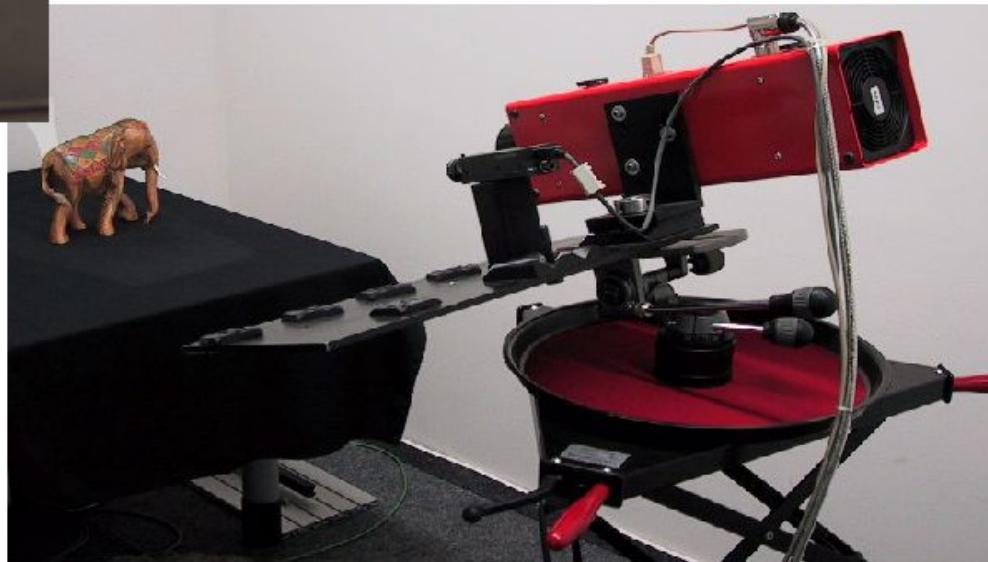
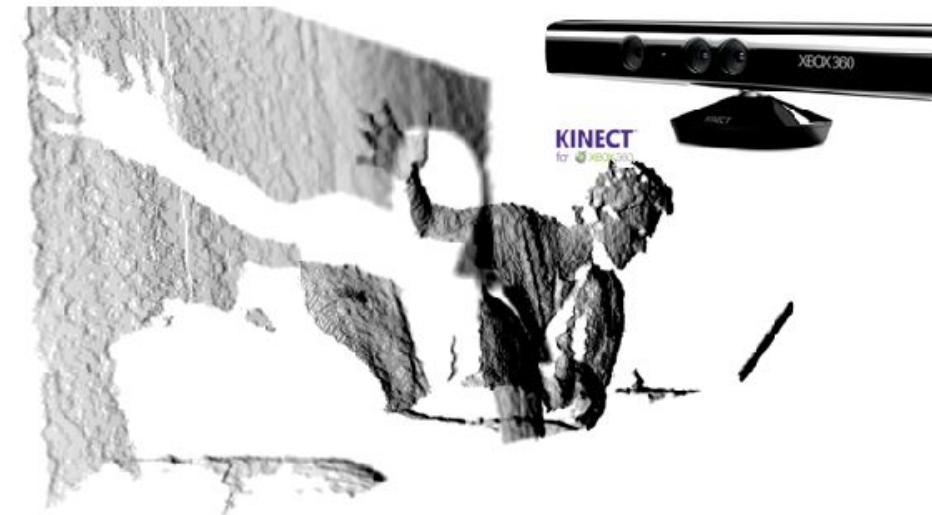
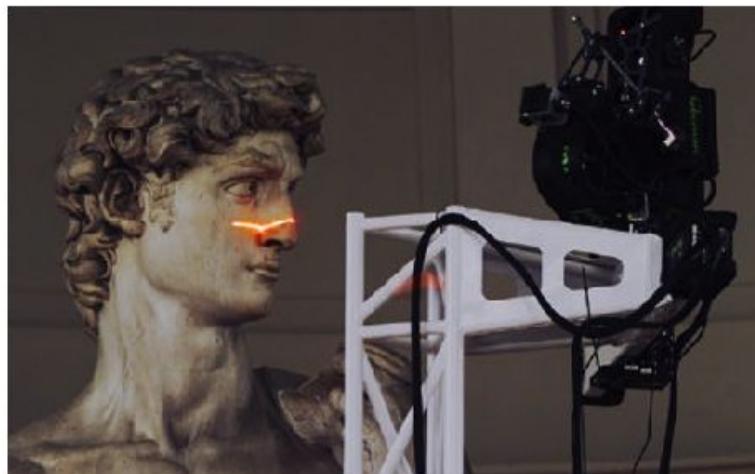
Point Clouds



- Simplest representation: **only points**, no connectivity
- Collection of (x,y,z) coordinates, possibly with normal
- Points with orientation are called **surfels**

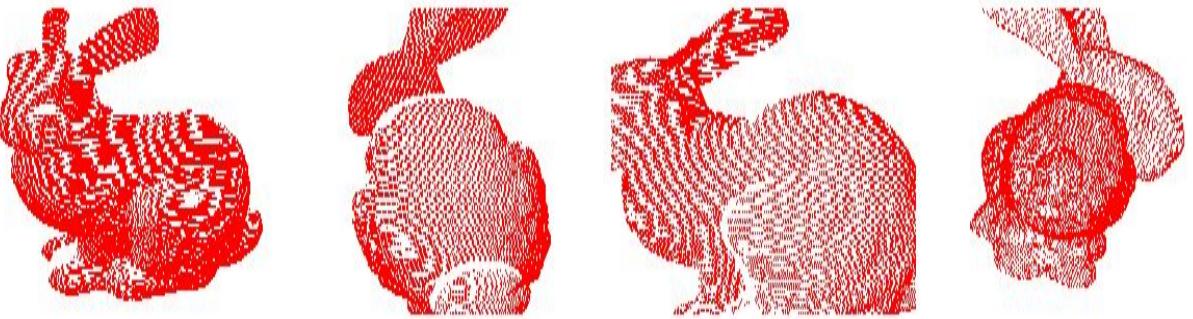


Output of Acquisition



Slide credit: Hao Su

Point Clouds

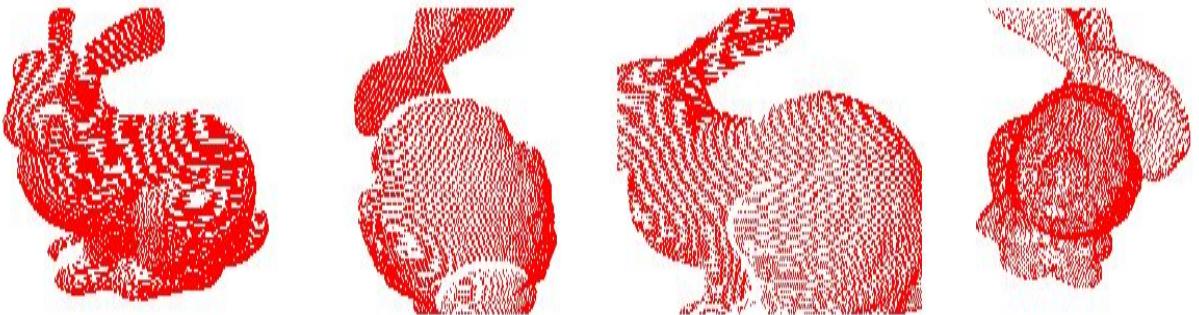


- Simplest representation: **only points**, no connectivity
- Collection of (x,y,z) coordinates, possibly with normal
- Points with orientation are called **surfels**
- Often results from scanners
- Potentially noisy
- Registration of multiple images

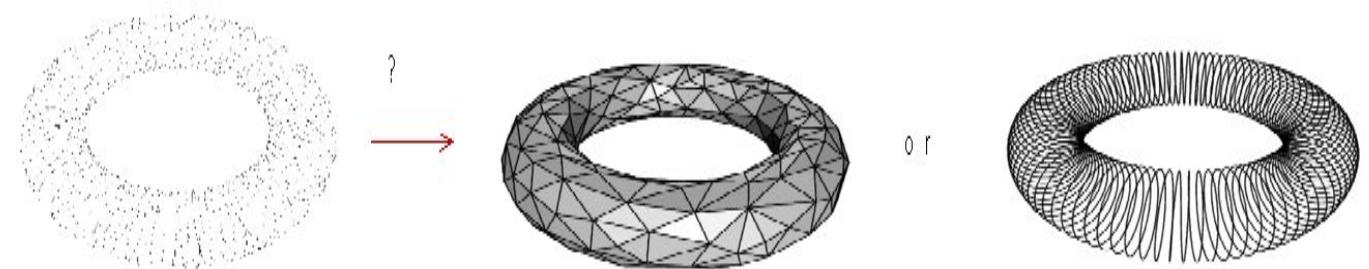


set of raw scans

Point Clouds

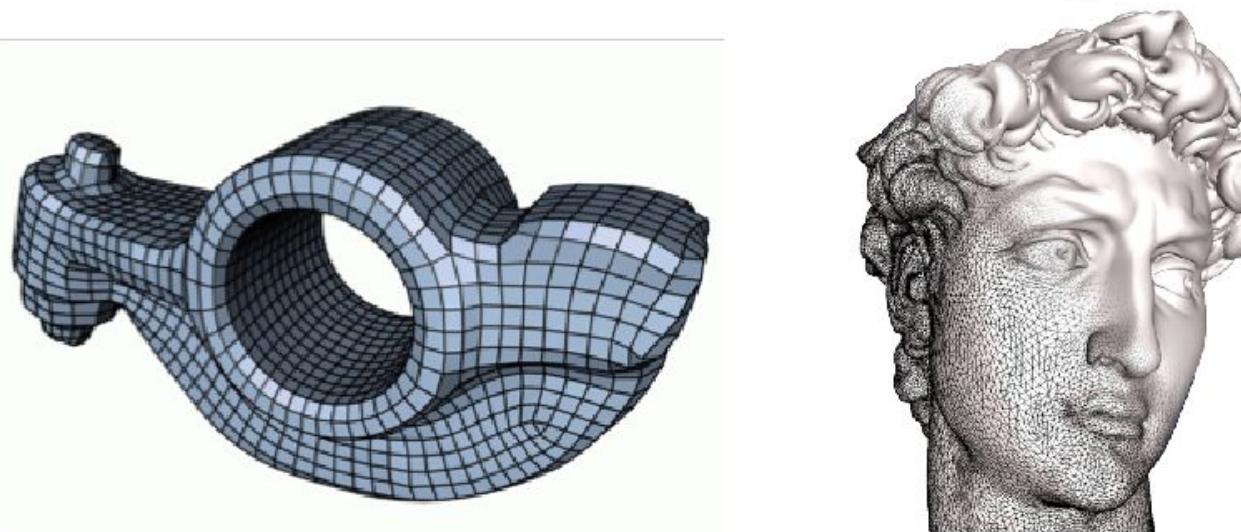
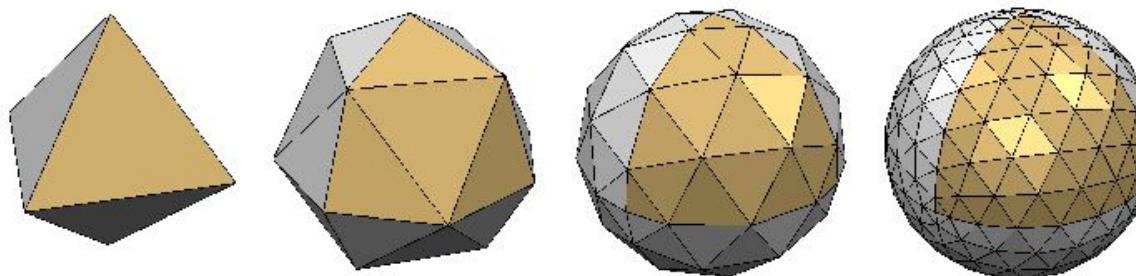


- Easily represent any kind of geometry
- Useful for large datasets
- Difficult to draw in undersampled regions
- Other limitations:
 - No simplification or subdivision
 - No direction smooth rendering
 - No topological information



Polygonal Meshes

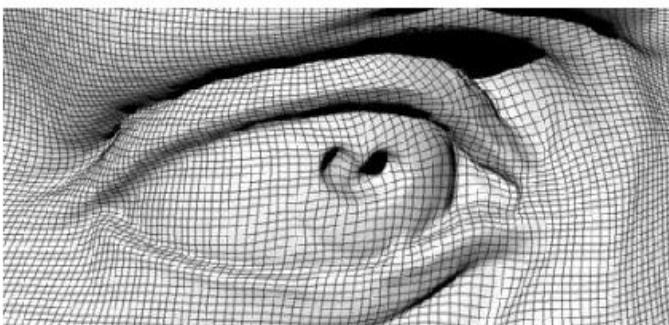
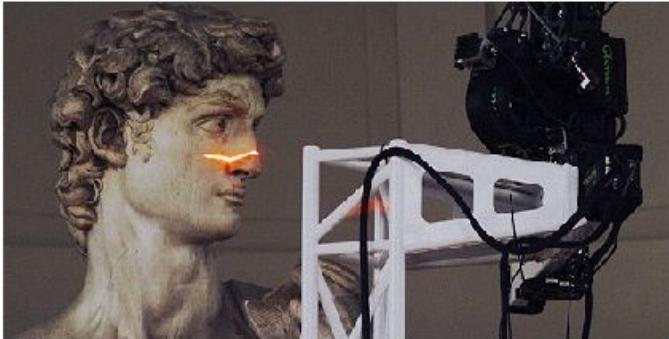
- Boundary representations of objects



A Large Triangle Mesh

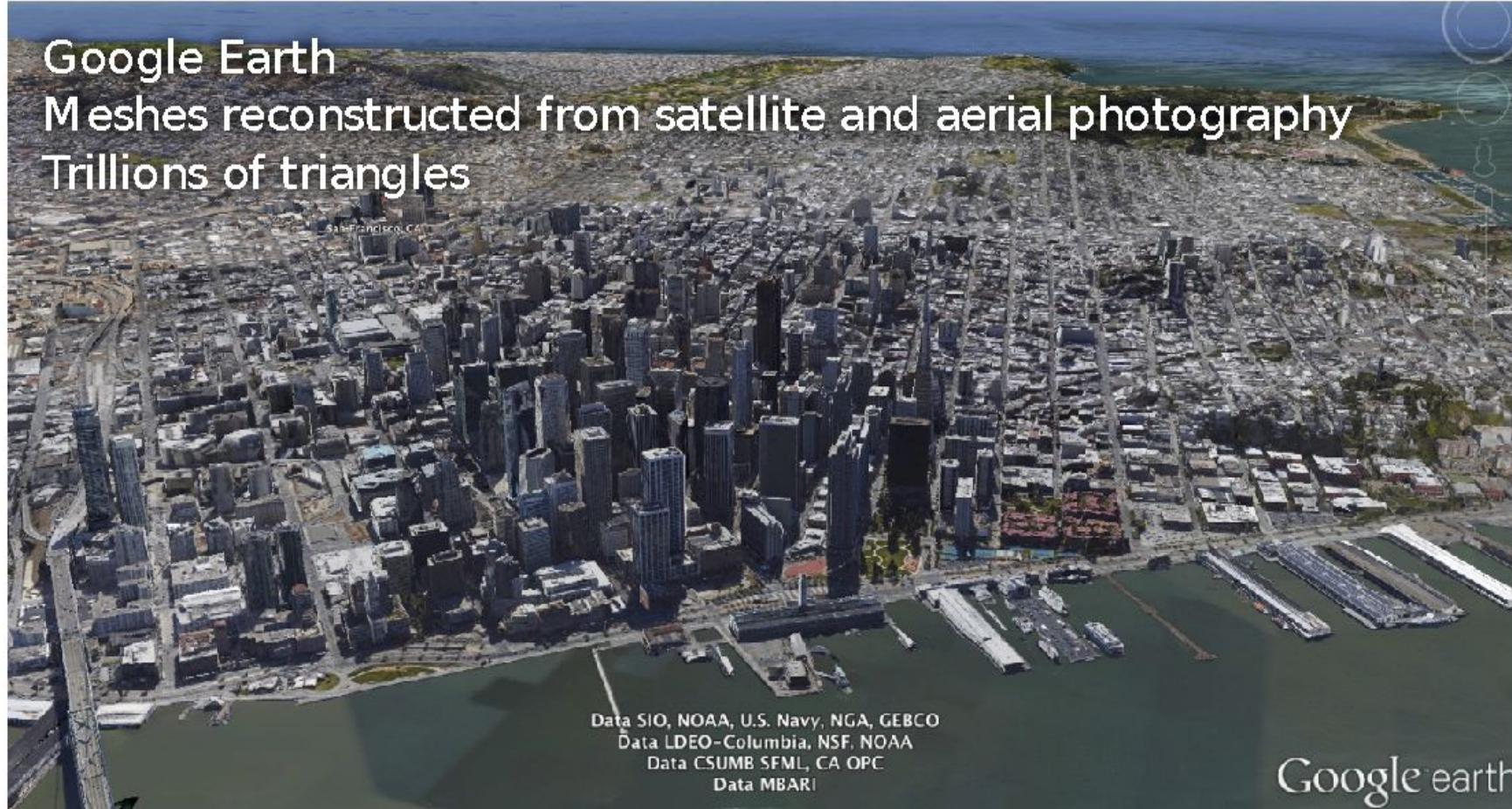
David

Digital Michelangelo Project
28,184,526 vertices
56,230,343 triangles



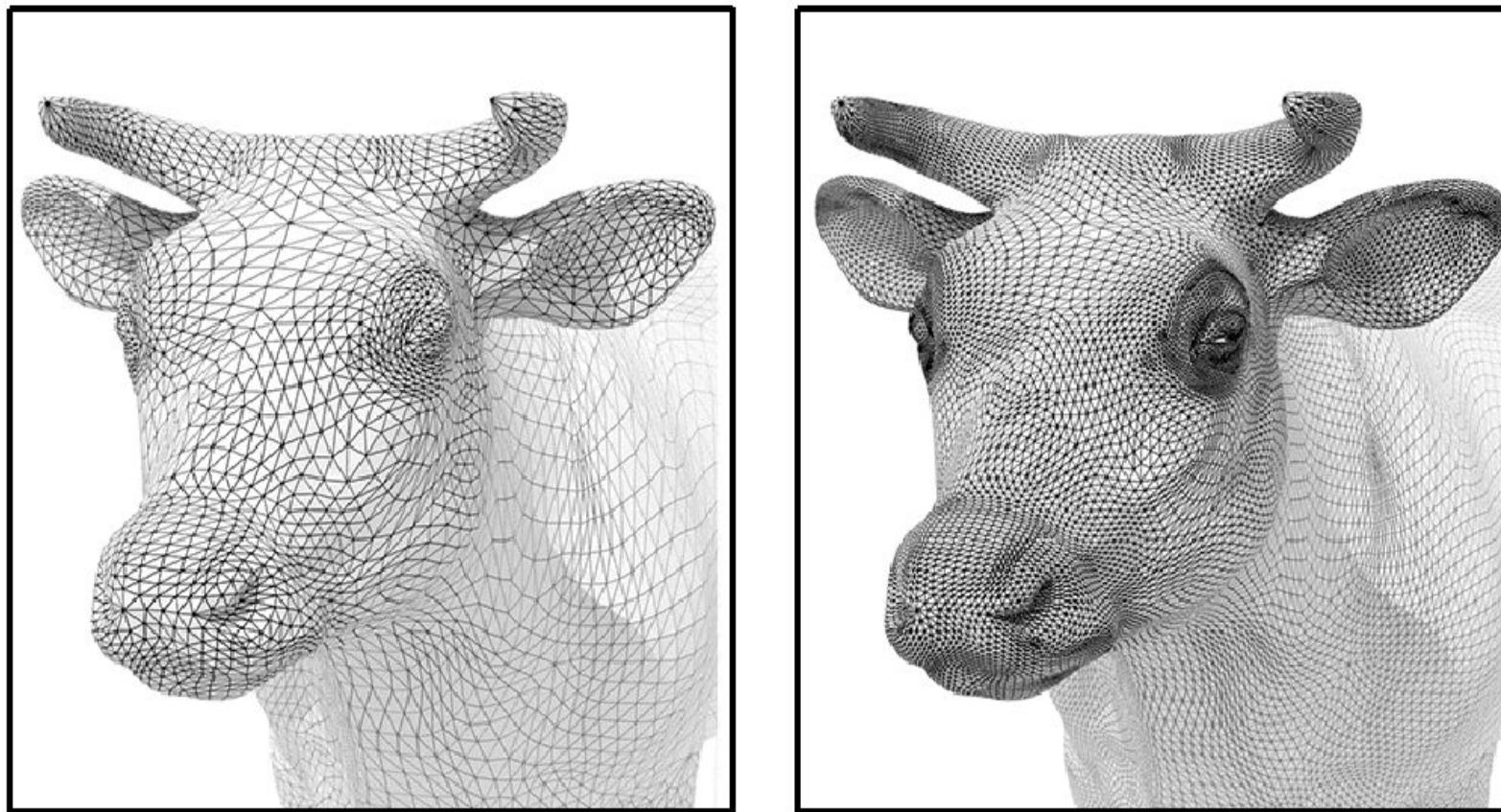
Slide credit: Ren Ng

A Very Large Triangle Mesh



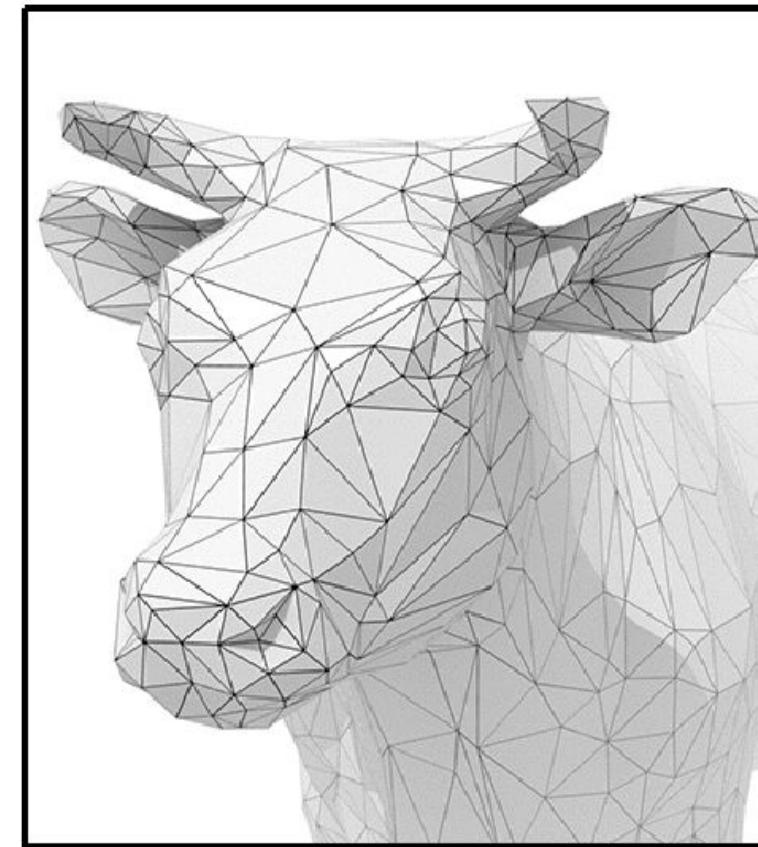
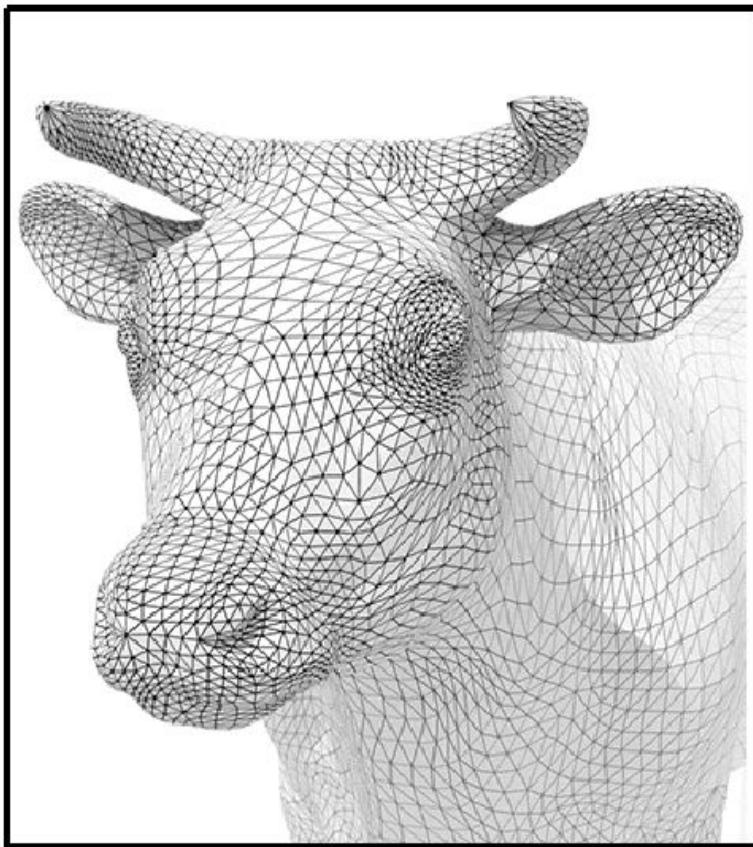
Slide credit: Ren Ng

Mesh Upsampling – Subdivision



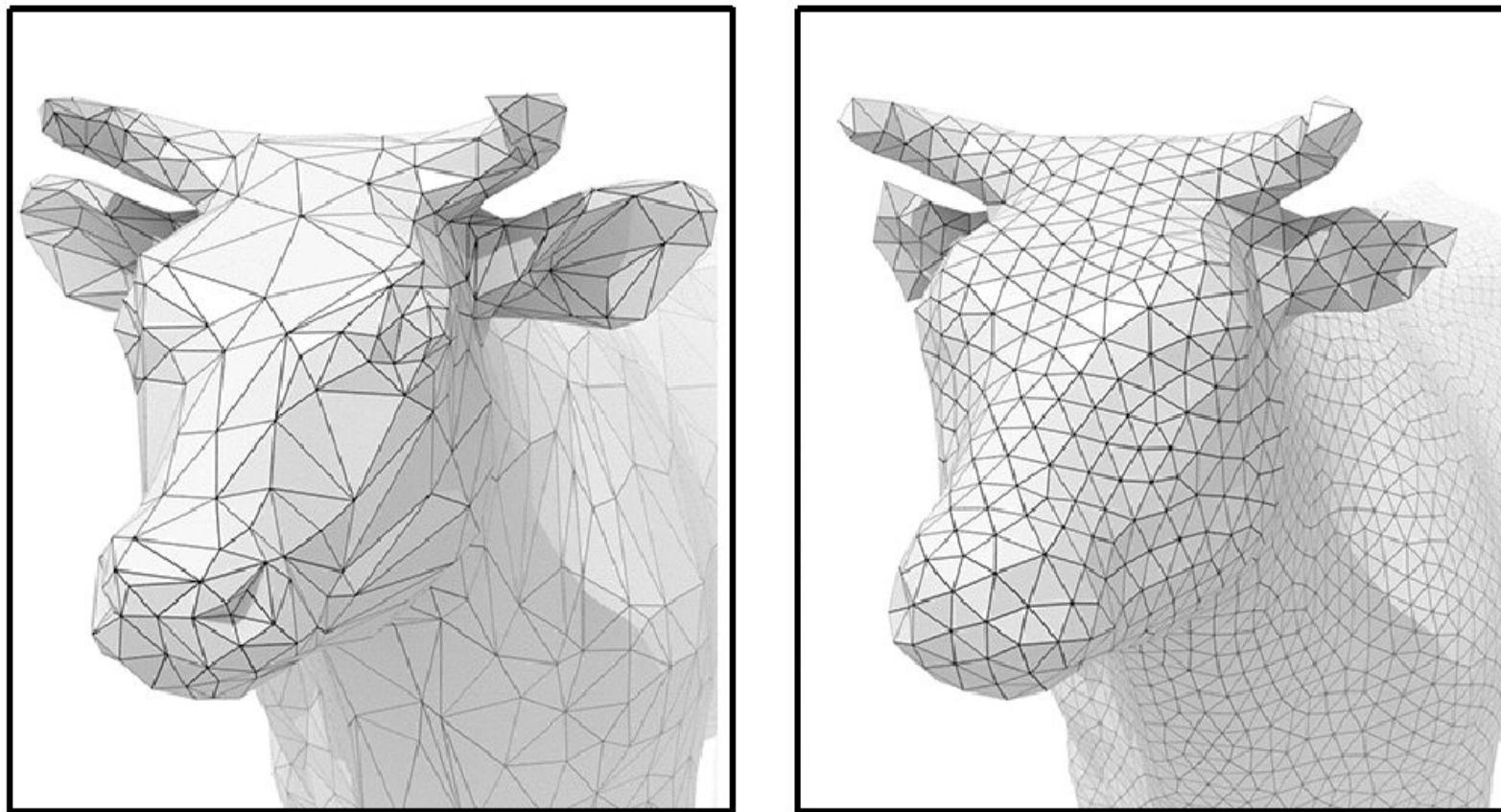
Increase resolution via interpolation

Mesh Downsampling – Simplification



Decrease resolution; try to preserve shape/appearance

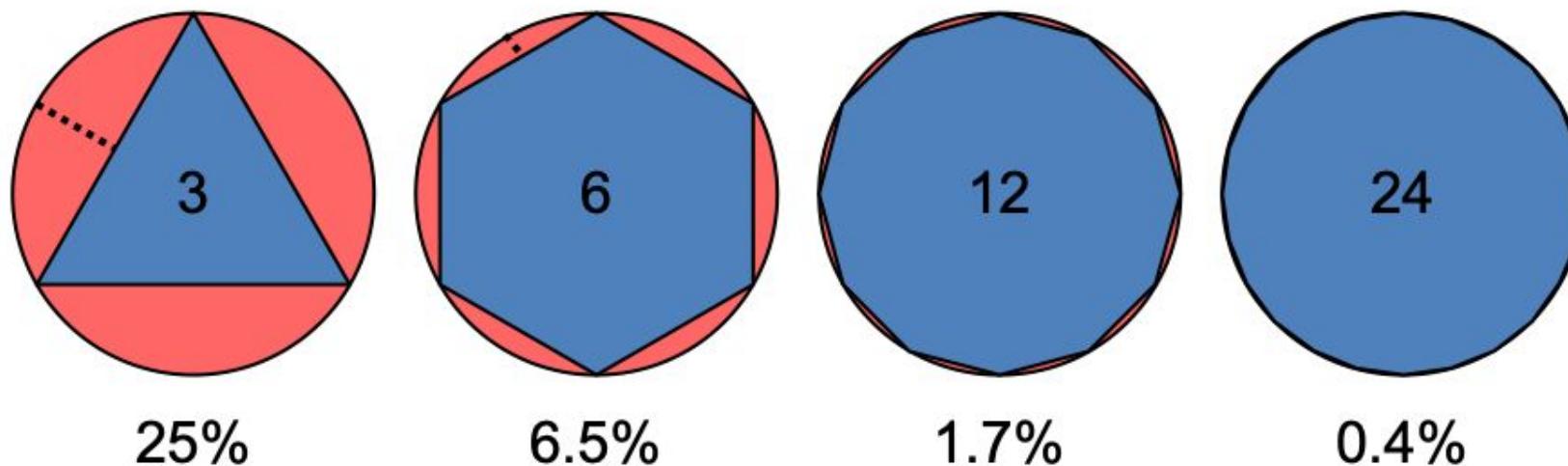
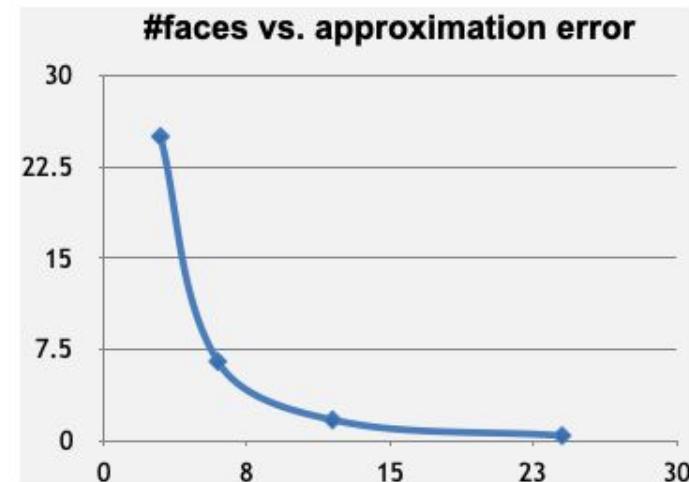
Mesh Regularization



Modify sample distribution to improve quality

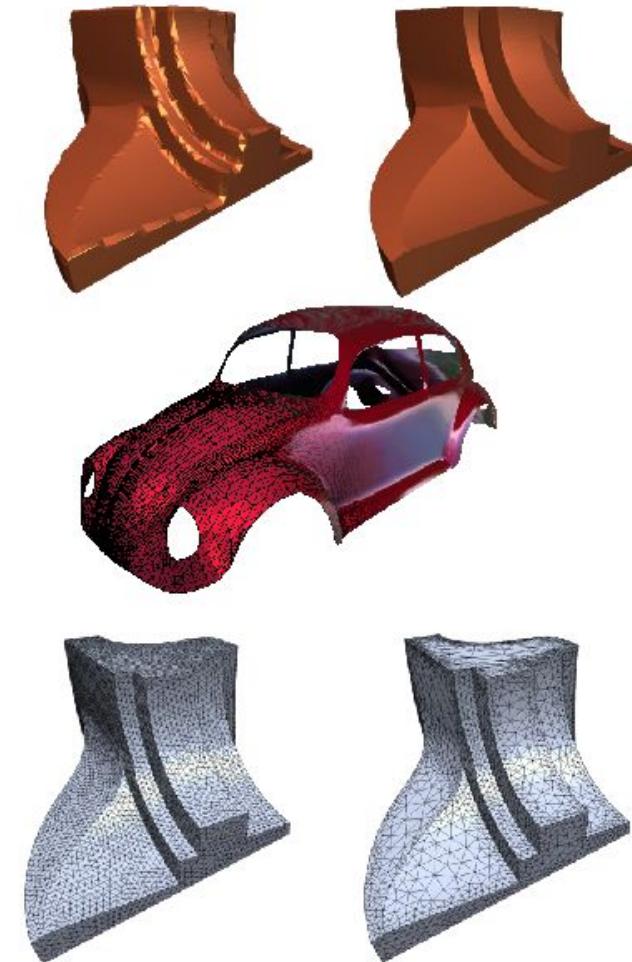
Meshes as Approximations of Smooth Surfaces

- Piecewise linear approximation
 - Error is $O(h^2)$

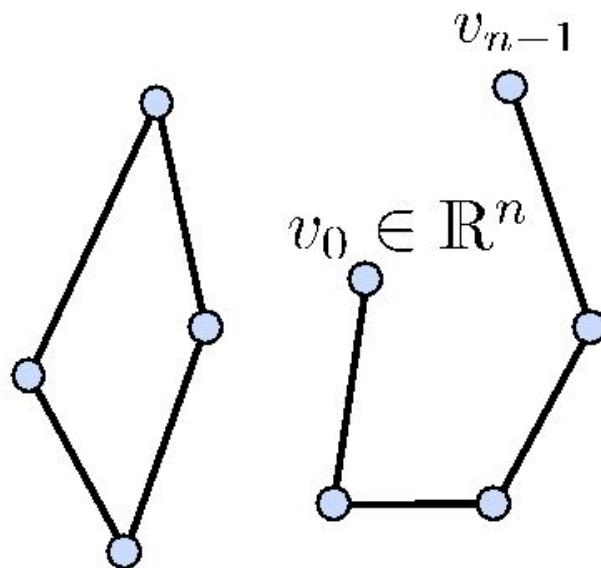


Polygonal Meshes

- Polygonal meshes are a good representation
 - approximation $O(h^2)$
 - arbitrary topology
 - adaptive refinement
 - efficient rendering

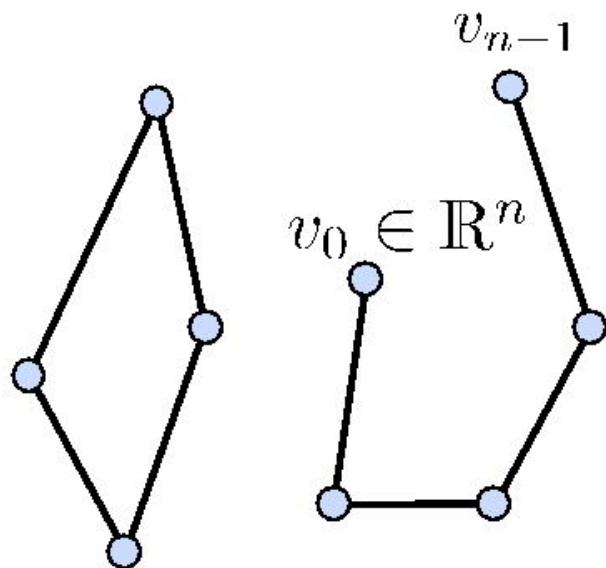


Polygon



- **Vertices:** v_0, v_1, \dots, v_{n-1}
- **Edges:** $\{(v_0, v_1), \dots, (v_{n-2}, v_{n-1})\}$

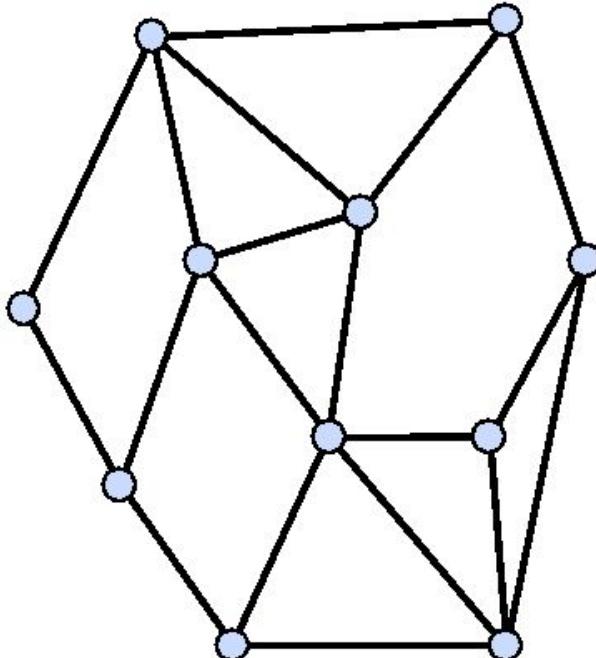
Polygon



- **Vertices:** v_0, v_1, \dots, v_{n-1}
- **Edges:** $\{(v_0, v_1), \dots, (v_{n-2}, v_{n-1})\}$
- **Closed:** $v_0 = v_{n-1}$
- **Planar:** all vertices on a plane
- **Simple:** not self-intersecting

Polygonal Mesh

- A finite set M of closed, simple polygons Q_i is a polygonal mesh
- The intersection of two polygons in M is either empty, a vertex, or an edge

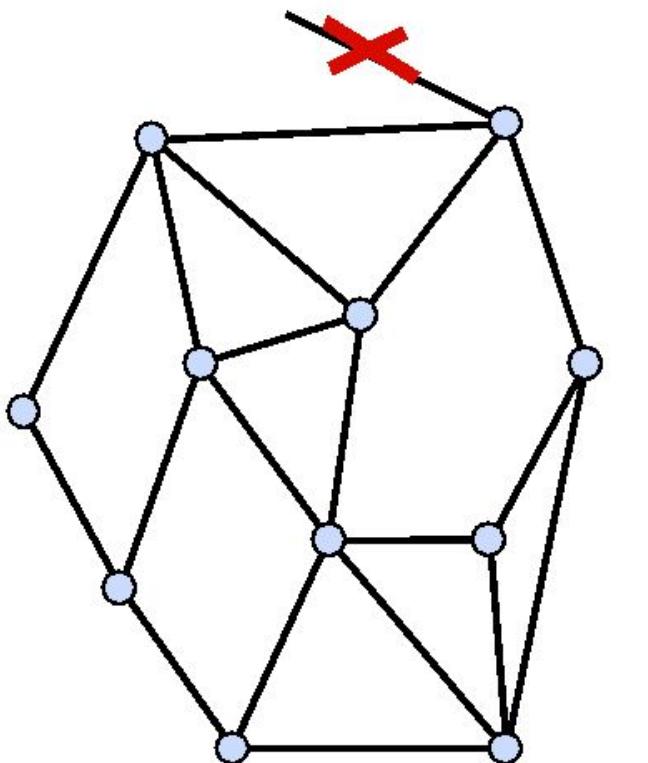


$$M = \langle V, E, F \rangle$$

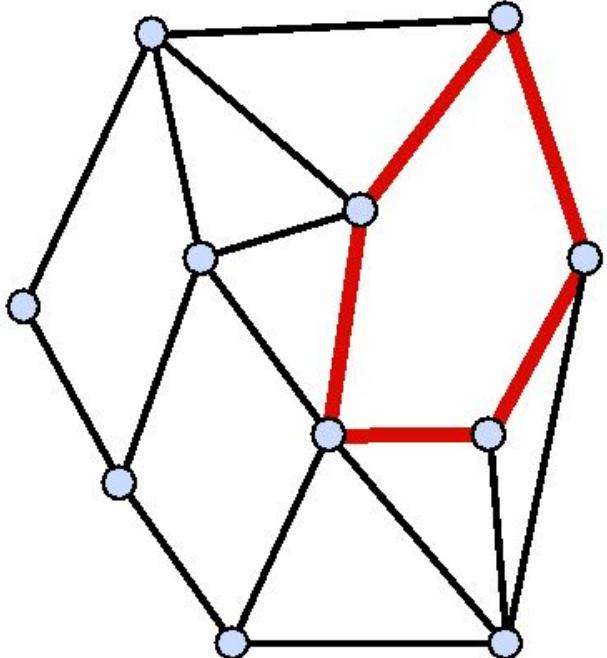
vertices edges faces

Polygonal Mesh

- A finite set M of closed, simple polygons Q_i is a polygonal mesh
- The intersection of two polygons in M is either empty, a vertex, or an edge
- Every edge belongs to at least one polygon

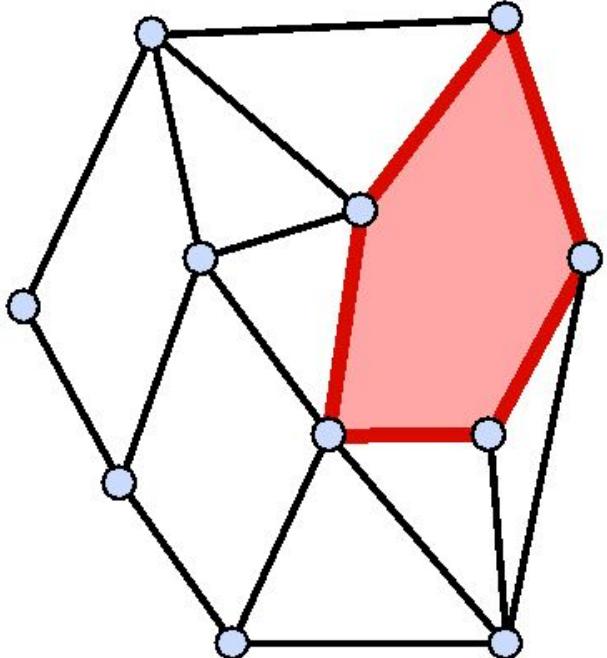


Polygonal Mesh



- A finite set M of closed, simple polygons Q_i is a polygonal mesh
- The intersection of two polygons in M is either empty, a vertex, or an edge
- Every edge belongs to at least one polygon
- Each Q_i defines a **face** of the polygonal mesh

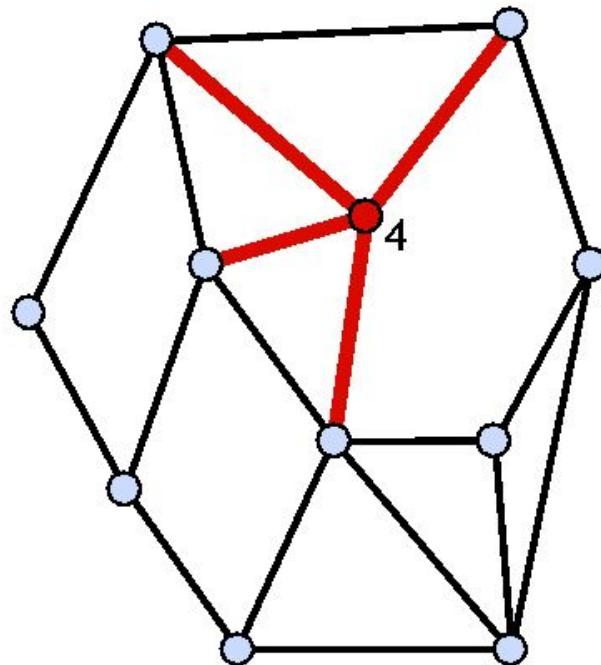
Polygonal Mesh



- A finite set M of closed, simple polygons Q_i is a polygonal mesh
- The intersection of two polygons in M is either empty, a vertex, or an edge
- Every edge belongs to at least one polygon
- Each Q_i defines a face of the polygonal mesh

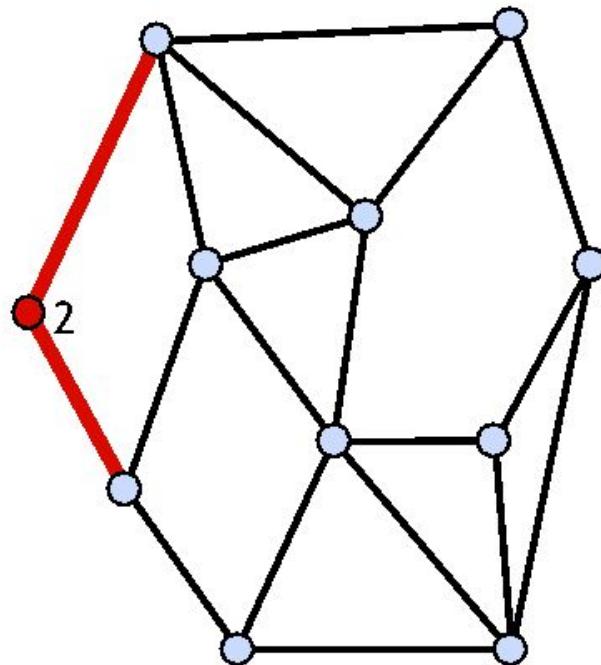
Polygonal Mesh

- Vertex **degree** or **valence** = number of incident edges



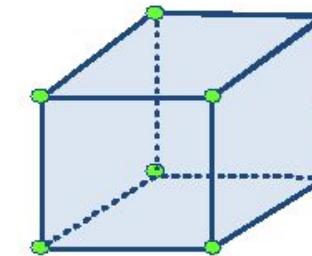
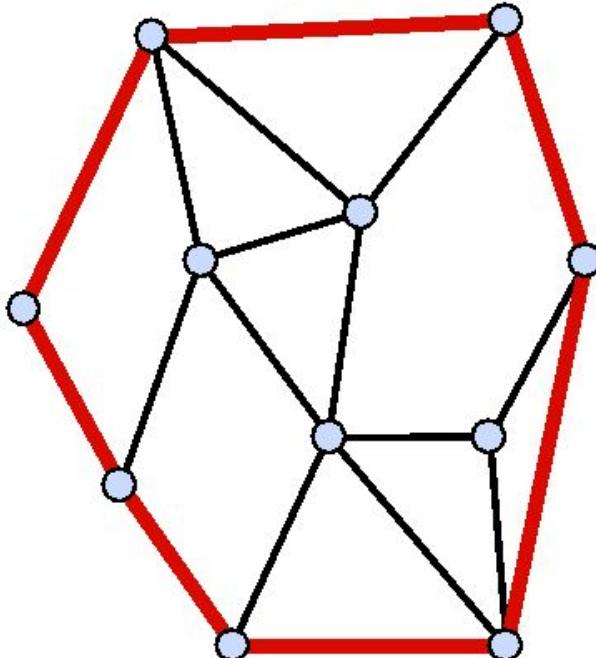
Polygonal Mesh

- Vertex degree or valence = number of incident edges



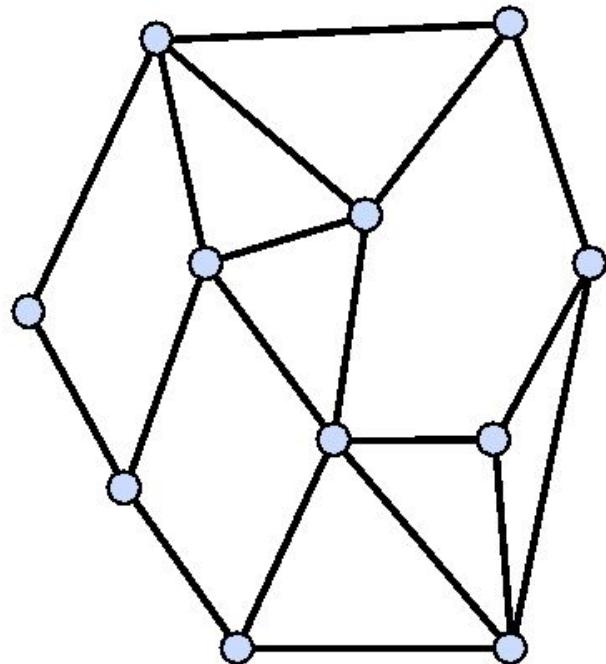
Polygonal Mesh

- Boundary: the set of all edges that belong to only one polygon
 - Either empty or forms closed loops
 - If empty, then the polygonal mesh is closed



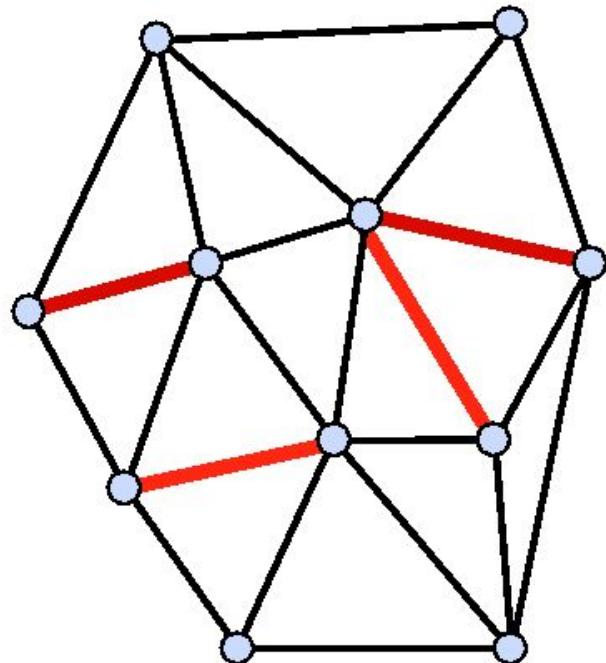
Triangulation

- Polygonal mesh where every face is a triangle



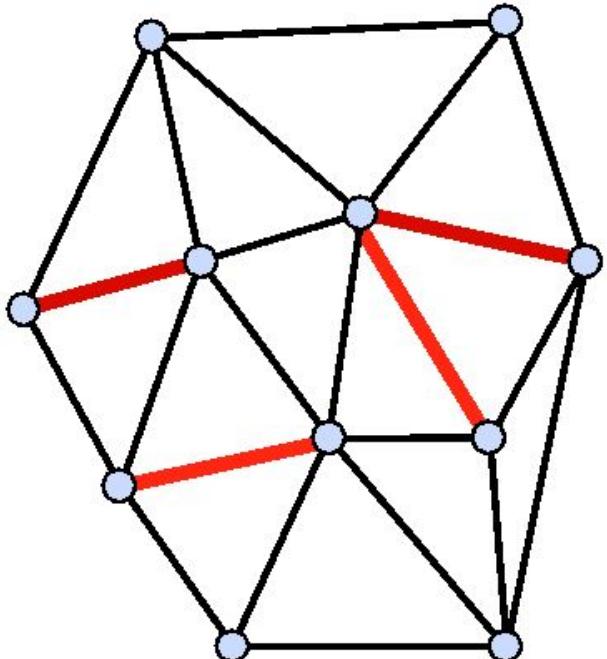
Triangulation

- Polygonal mesh where every face is a triangle



Triangulation

- Polygonal mesh where every face is a triangle
- Simplifies data structures
- Simplifies rendering
- Simplifies algorithms
- Each face planar and convex
- Any polygon can be triangulated



Data Structures

- What should be stored?
- Geometry: 3D coordinates



Data Structures



- What should be stored?
 - Geometry: 3D coordinates
 - Connectivity
 - Adjacency relationships

Data Structures



- What should be stored?
 - Geometry: 3D coordinates
 - Connectivity
 - Adjacency relationships
 - Attributes
 - Normal, color, texture coordinates
 - Per vertex, face, edge

Simple Data Structures: Indexed Face Set

- Used in formats
 - OBJ, OFF, WRL
- Storage
 - Vertex: position
 - Face: vertex indices
 - 12 bytes per vertex
 - 12 bytes per face
 - $36*v$ bytes for the mesh
- No explicit neighborhood info

Vertices			
v0	x0	y0	z0
v1	x1	x1	z1
v2	x2	y2	z2
v3	x3	y3	z3
v4	x4	y4	z4
v5	x5	y5	z5
v6	x6	y6	z6
...

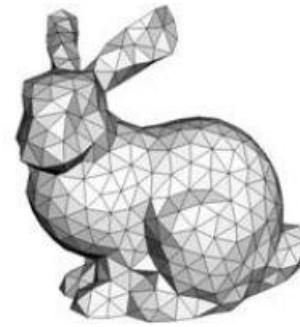
Triangles			
t0	v0	v1	v2
t1	v0	v1	v3
t2	v2	v4	v3
t3	v5	v2	v6
...

Shape Representations

Non-parametric



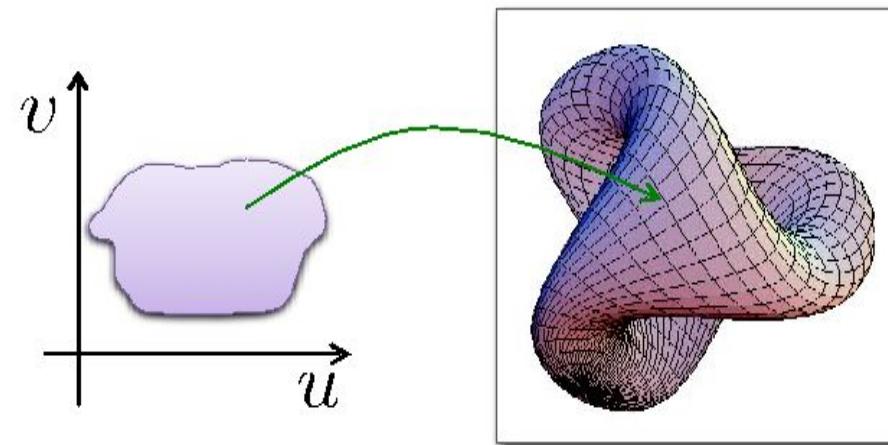
Points



Meshes

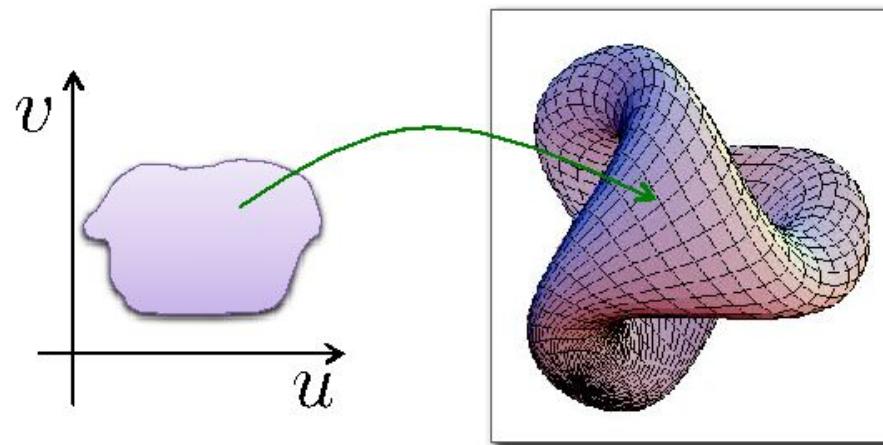
Parametric Representation

- Range of a function $f : X \rightarrow Y, X \subseteq \mathbb{R}^m, Y \subseteq \mathbb{R}^n$
- Surface in 3D: $m = 2, n = 3$



Parametric Representation

- Range of a function $f : X \rightarrow Y, X \subseteq \mathbb{R}^m, Y \subseteq \mathbb{R}^n$
- Surface in 3D: $m = 2, n = 3$



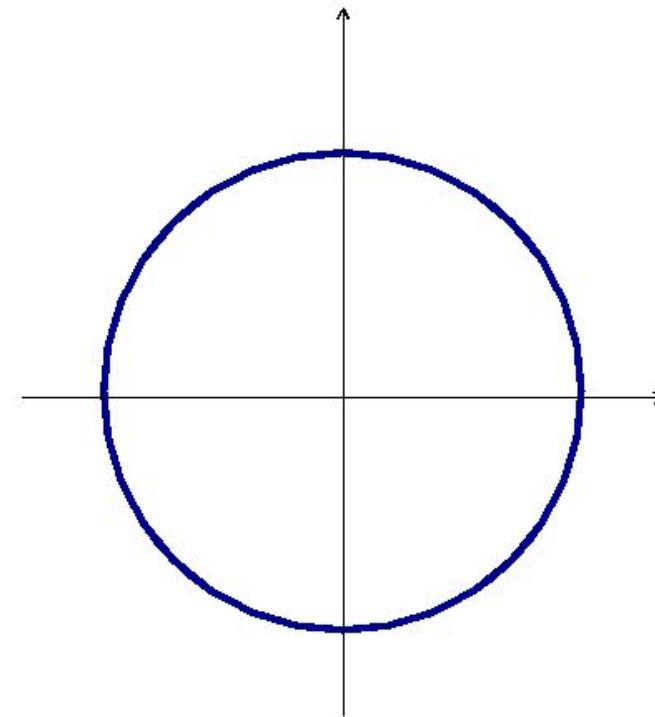
$$s(u, v) = (x(u, v), y(u, v), z(u, v))$$

Parametric Curves

- Example: Explicit curve/circle in 2D

$$\mathbf{p} : \mathbb{R} \rightarrow \mathbb{R}^2$$

$$t \mapsto \mathbf{p}(t) = (x(t), y(t))$$



Parametric Curves

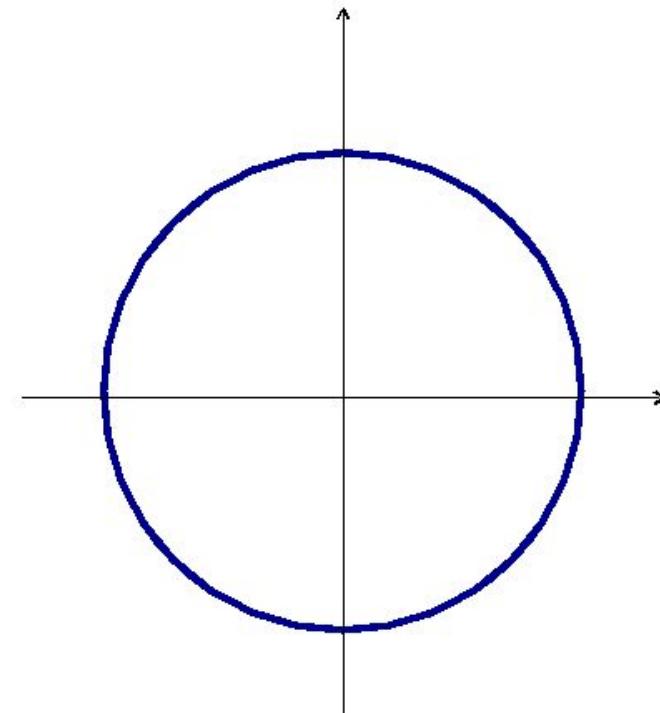
- Example: Explicit curve/circle in 2D

$$\mathbf{p} : \mathbb{R} \rightarrow \mathbb{R}^2$$

$$t \mapsto \mathbf{p}(t) = (x(t), y(t))$$

$$\mathbf{p}(t) = r (\cos(t), \sin(t))$$

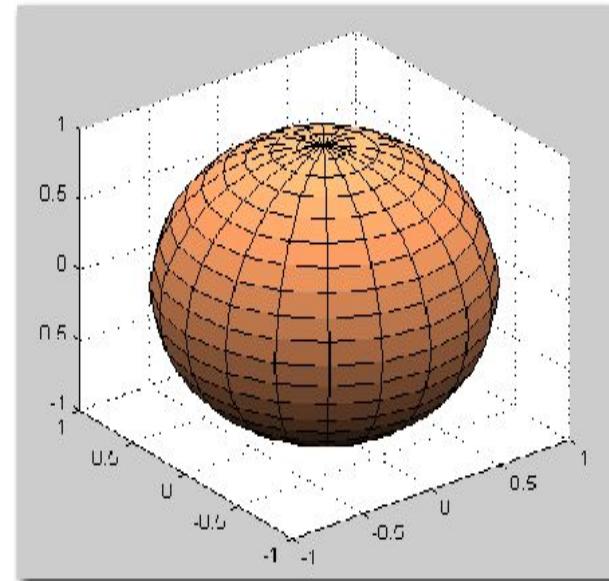
$$t \in [0, 2\pi)$$



Parametric Surfaces

- Sphere in 3D

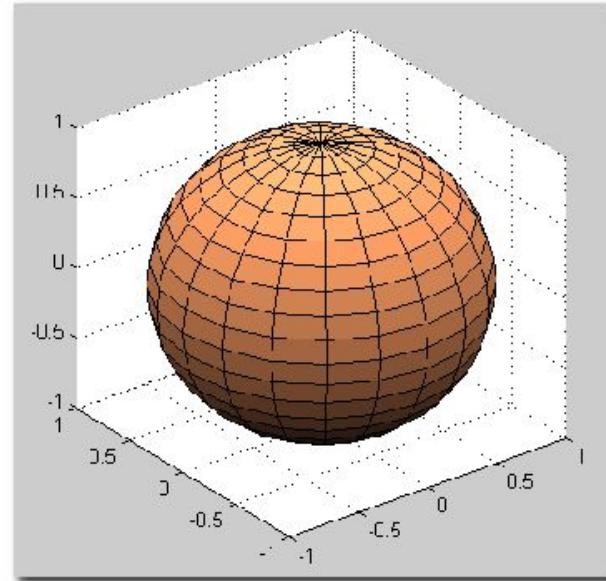
$$s : \mathbb{R}^2 \rightarrow \mathbb{R}^3$$



Parametric Surfaces

- Sphere in 3D

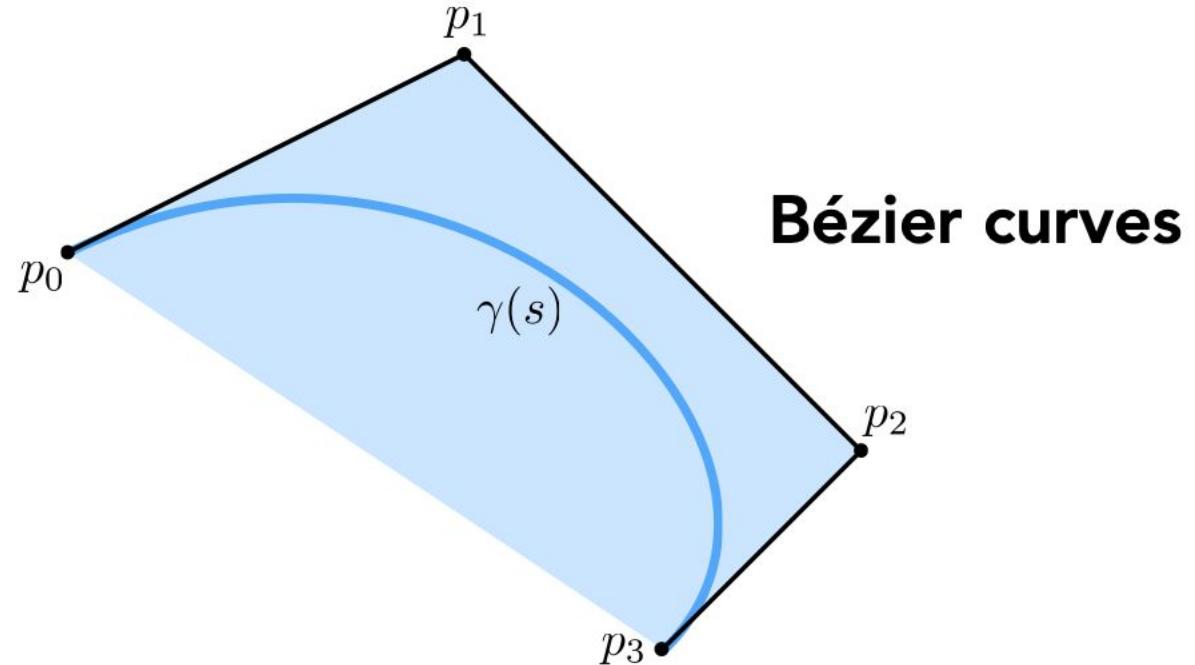
$$s : \mathbb{R}^2 \rightarrow \mathbb{R}^3$$



$$s(u, v) = r (\cos(u) \cos(v), \sin(u) \cos(v), \sin(v))$$

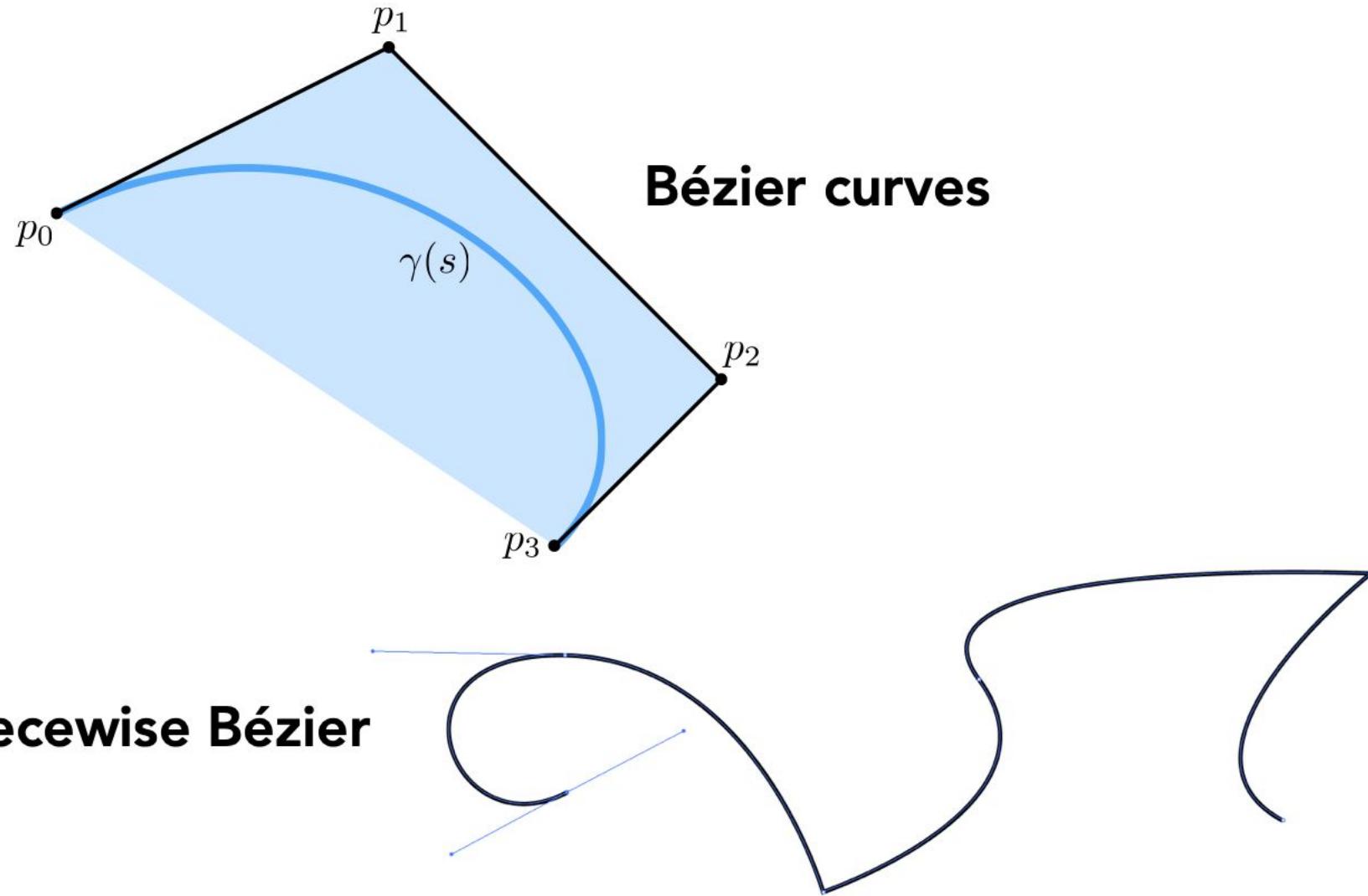
$$(u, v) \in [0, 2\pi) \times [-\pi/2, \pi/2]$$

Bézier Curves



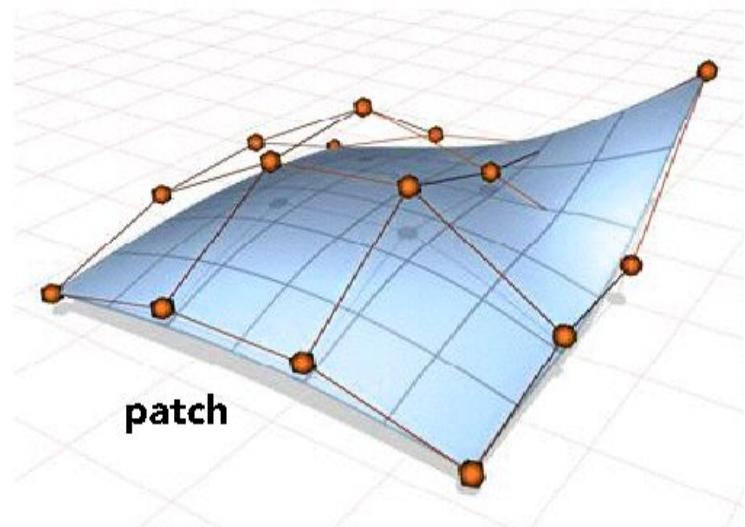
Bézier curves

Bézier Curves



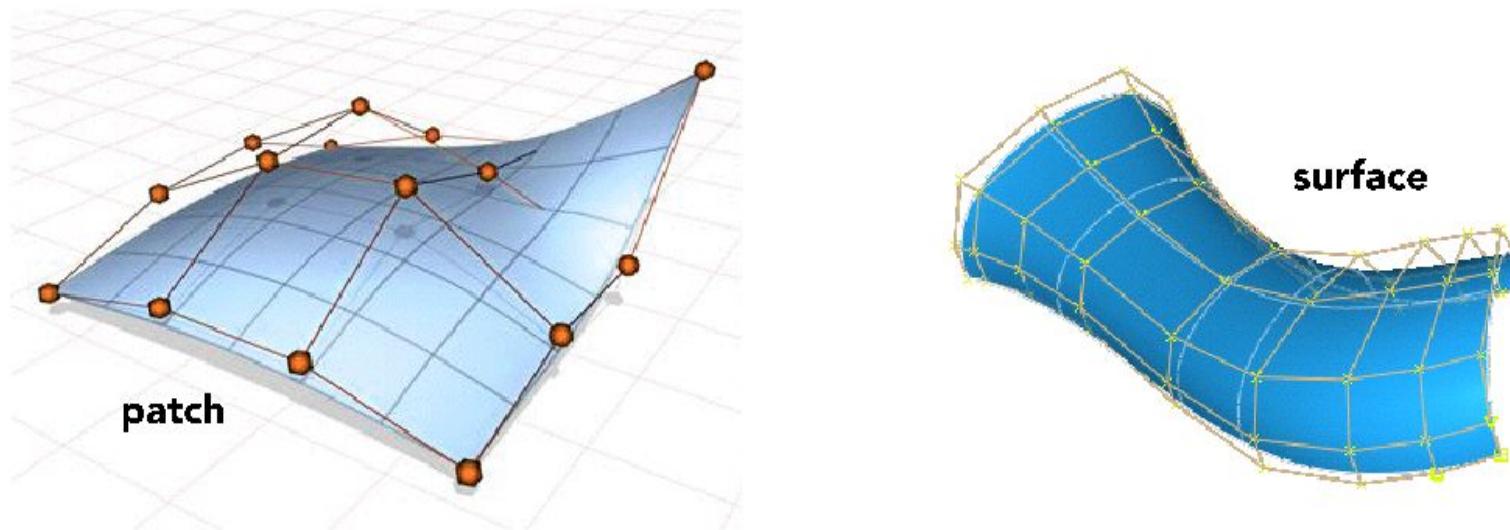
Bézier Surfaces

Use tensor product of Bézier curves to get a patch:



Bézier Surfaces

Use tensor product of Bézier curves to get a patch:

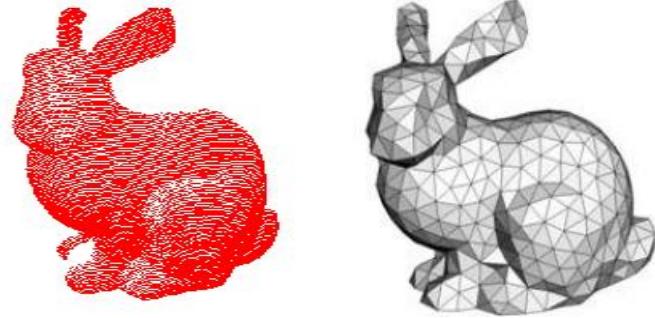


Multiple Bézier patches form a surface

Shape Representations

Non-parametric

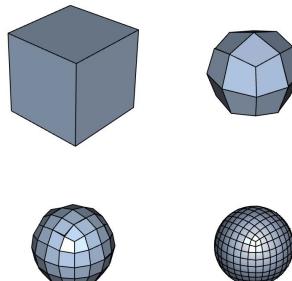
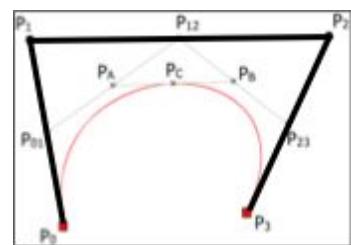
Explicit



Points

Meshes

Parametric



Splines

Subdivision
Surfaces

“Explicit” Representations of Geometry

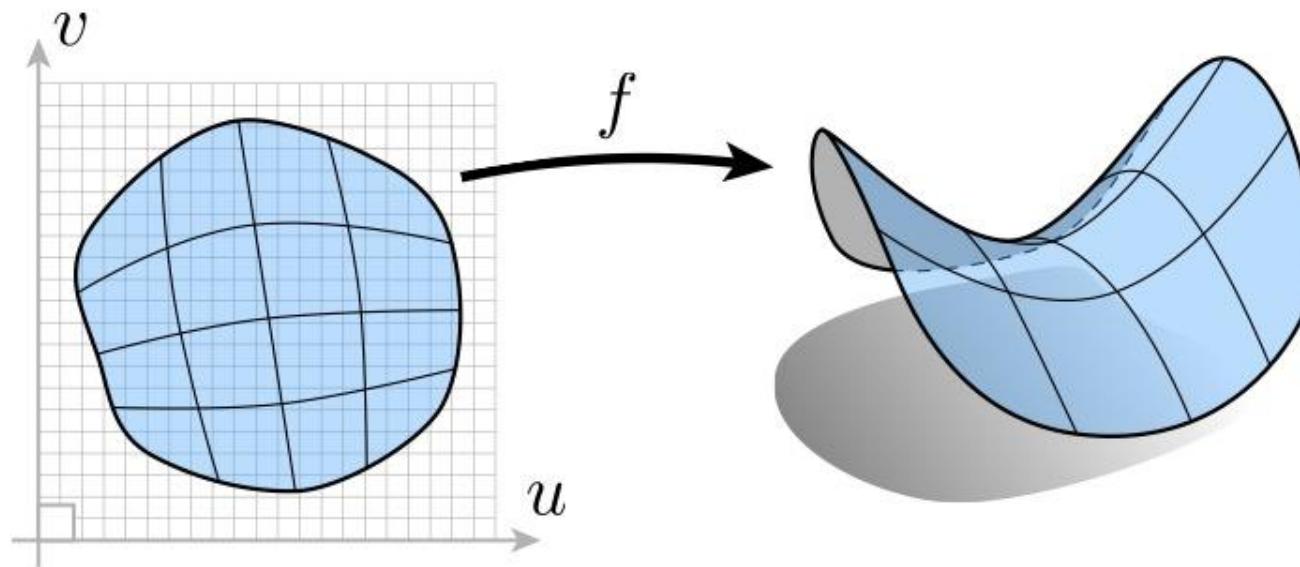
All points are given directly

“Explicit” Representations of Geometry

All points are given directly

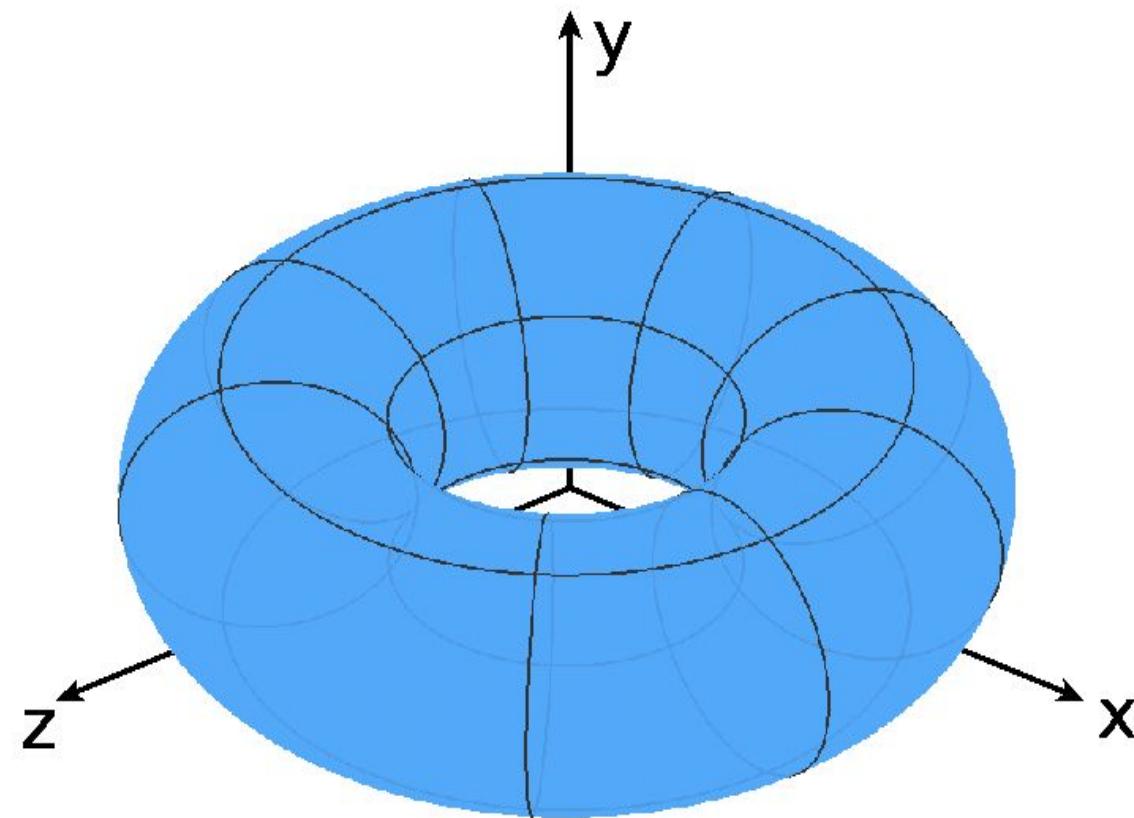
Generally:

$$f : \mathbb{R}^2 \rightarrow \mathbb{R}^3; (u, v) \mapsto (x, y, z)$$



Explicit Surface – Sampling Is Easy

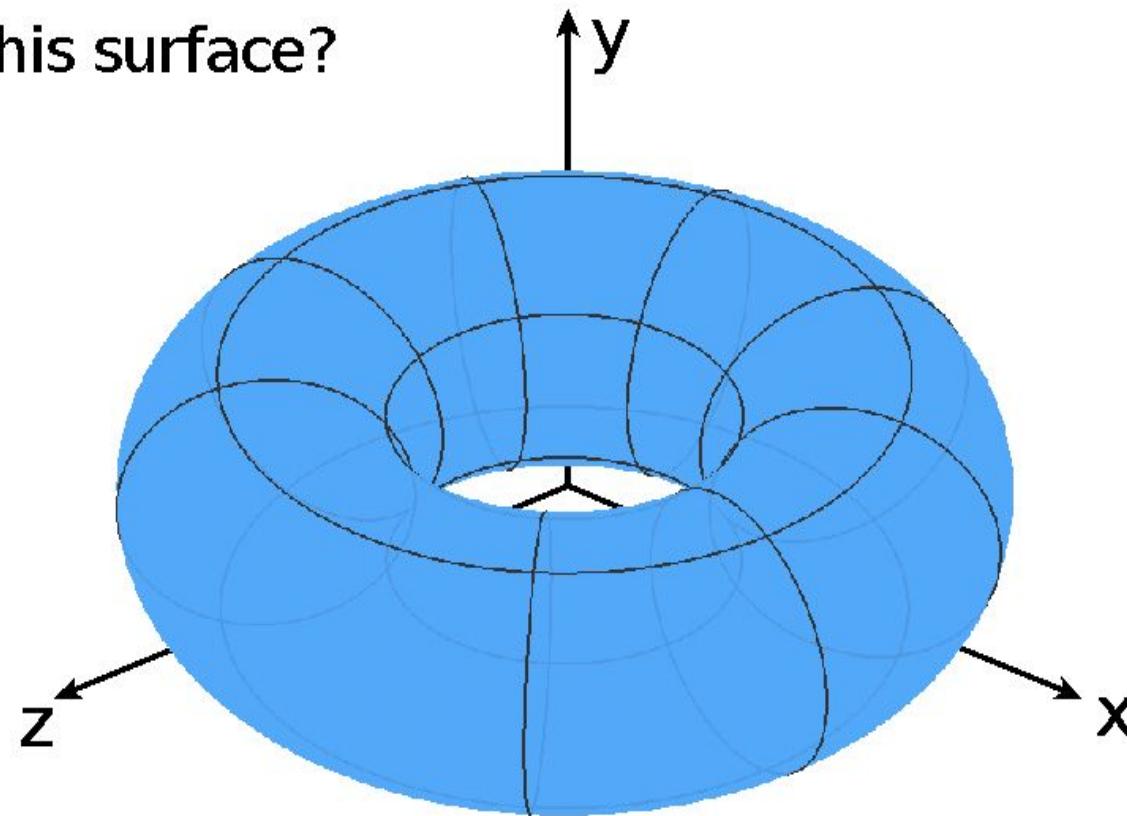
$$f(u, v) = ((2 + \cos u) \cos v, (2 + \cos u) \sin v, \sin u)$$



Explicit Surface – Sampling Is Easy

$$f(u, v) = ((2 + \cos u) \cos v, (2 + \cos u) \sin v, \sin u)$$

What points lie on this surface?

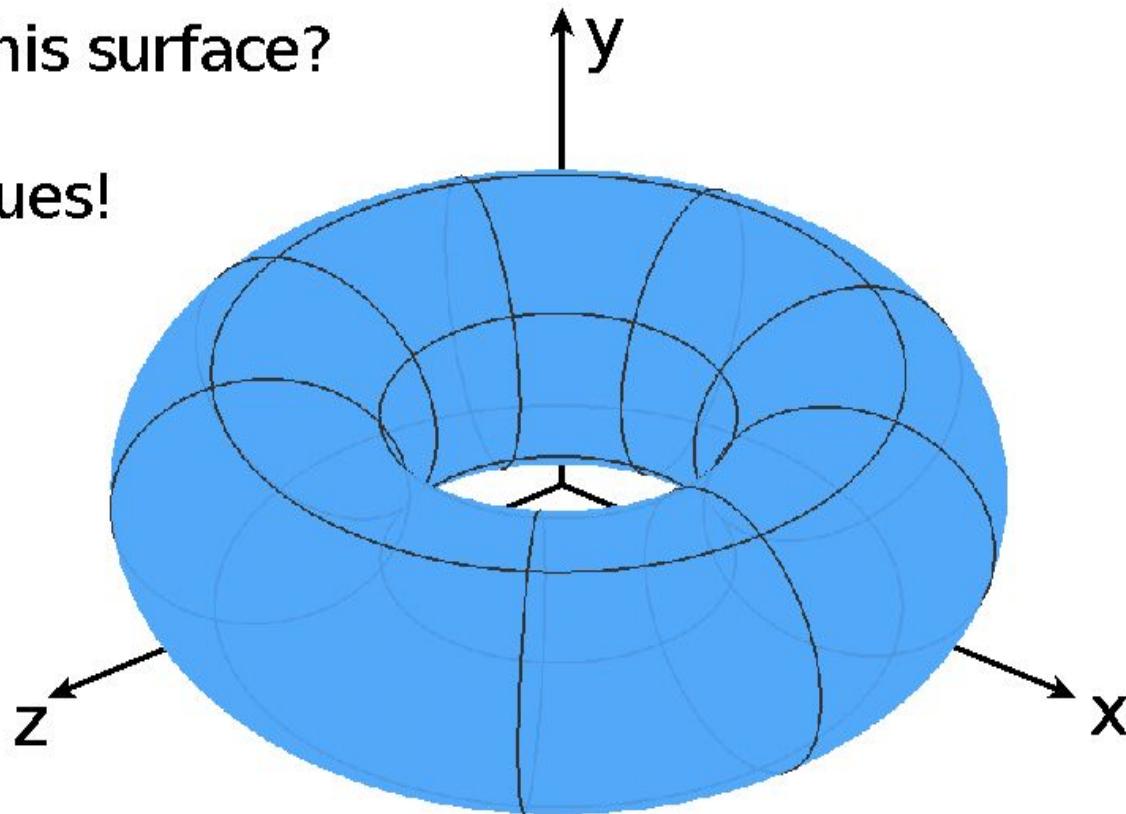


Explicit Surface – Sampling Is Easy

$$f(u, v) = ((2 + \cos u) \cos v, (2 + \cos u) \sin v, \sin u)$$

What points lie on this surface?

Just plug in (u, v) values!

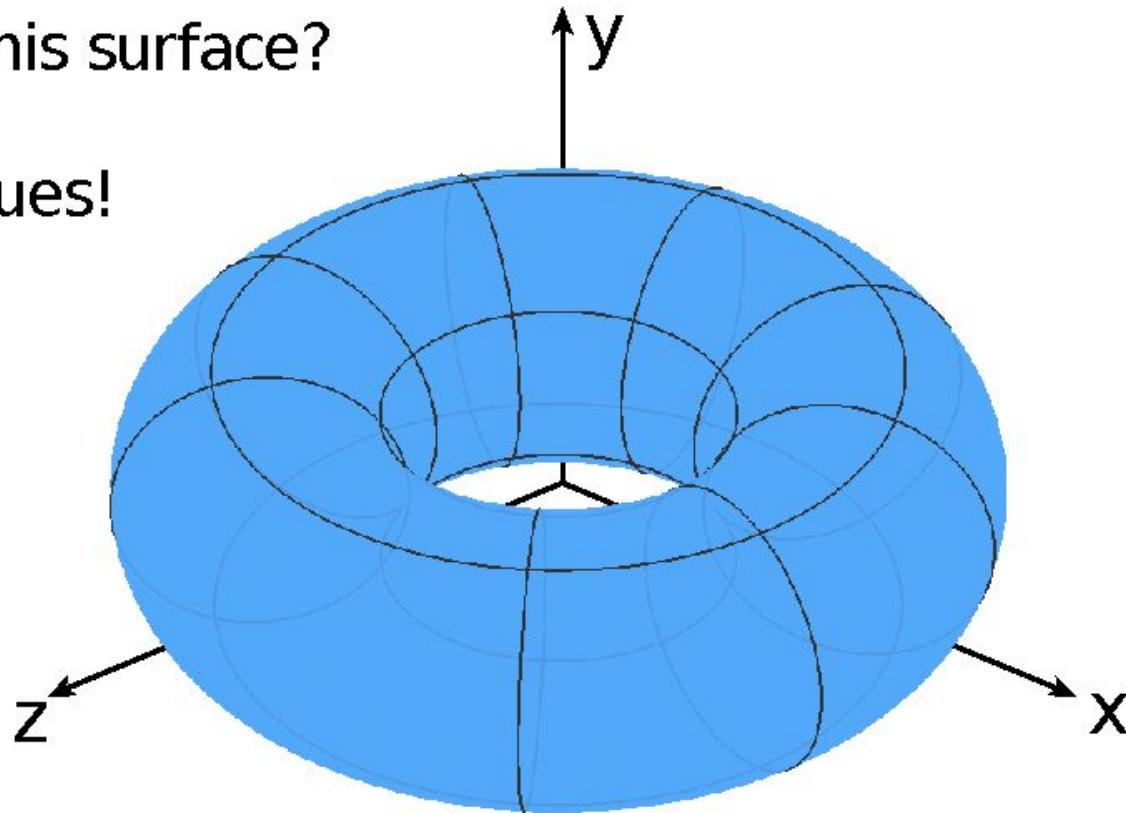


Explicit Surface – Sampling Is Easy

$$f(u, v) = ((2 + \cos u) \cos v, (2 + \cos u) \sin v, \sin u)$$

What points lie on this surface?

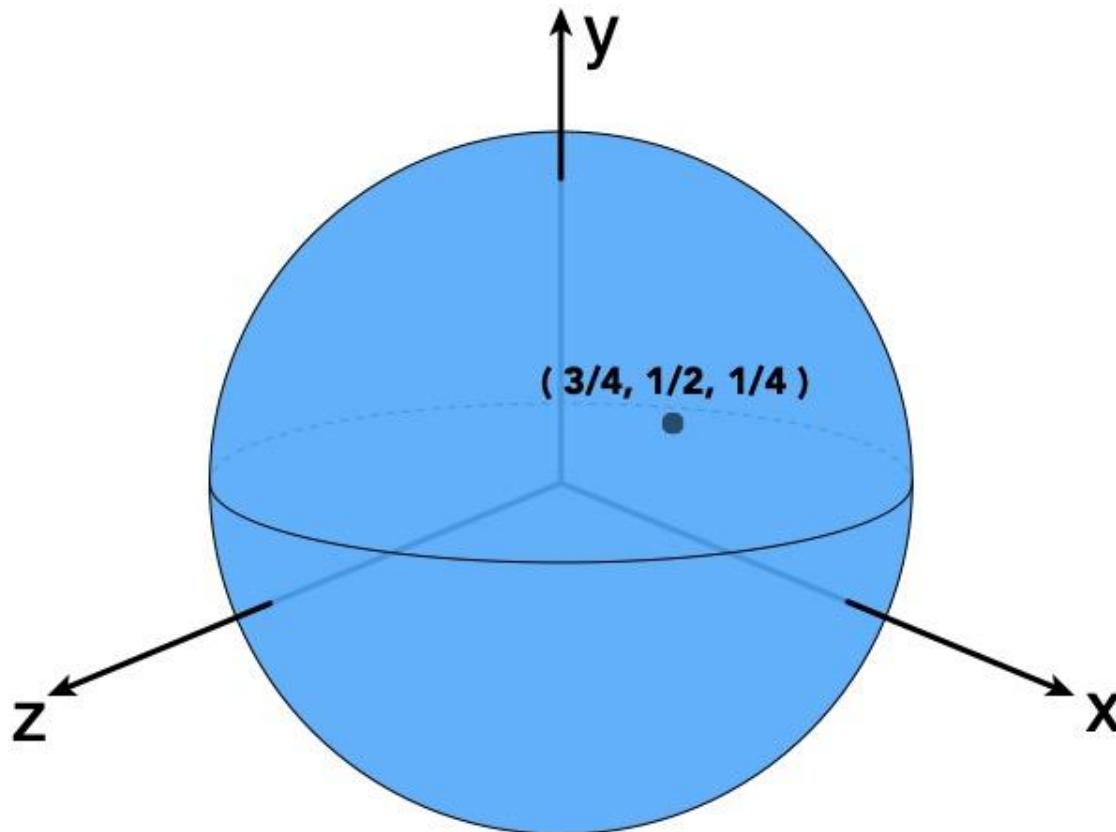
Just plug in (u, v) values!



Explicit representations make some tasks easy

Explicit Surface – Inside/Outside Test Hard

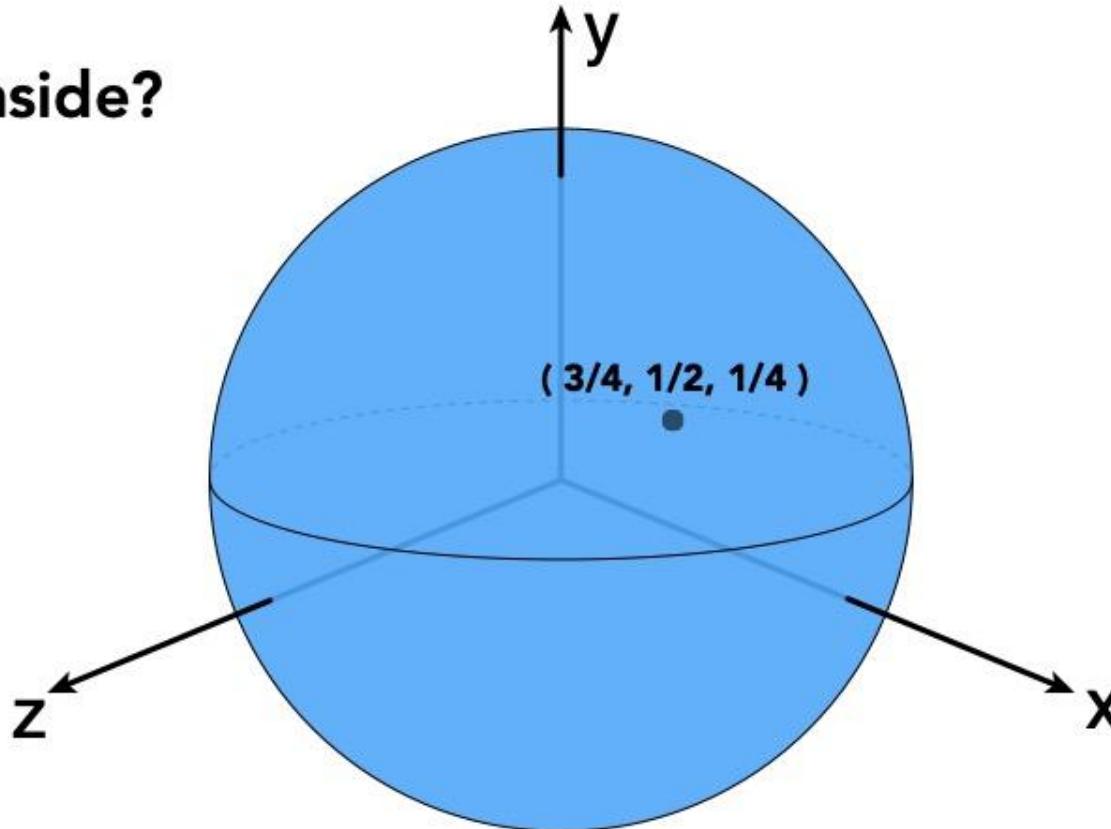
$$f(u, v) = (\cos u \sin v, \sin u \sin v, \cos v)$$



Explicit Surface – Inside/Outside Test Hard

$$f(u, v) = (\cos u \sin v, \sin u \sin v, \cos v)$$

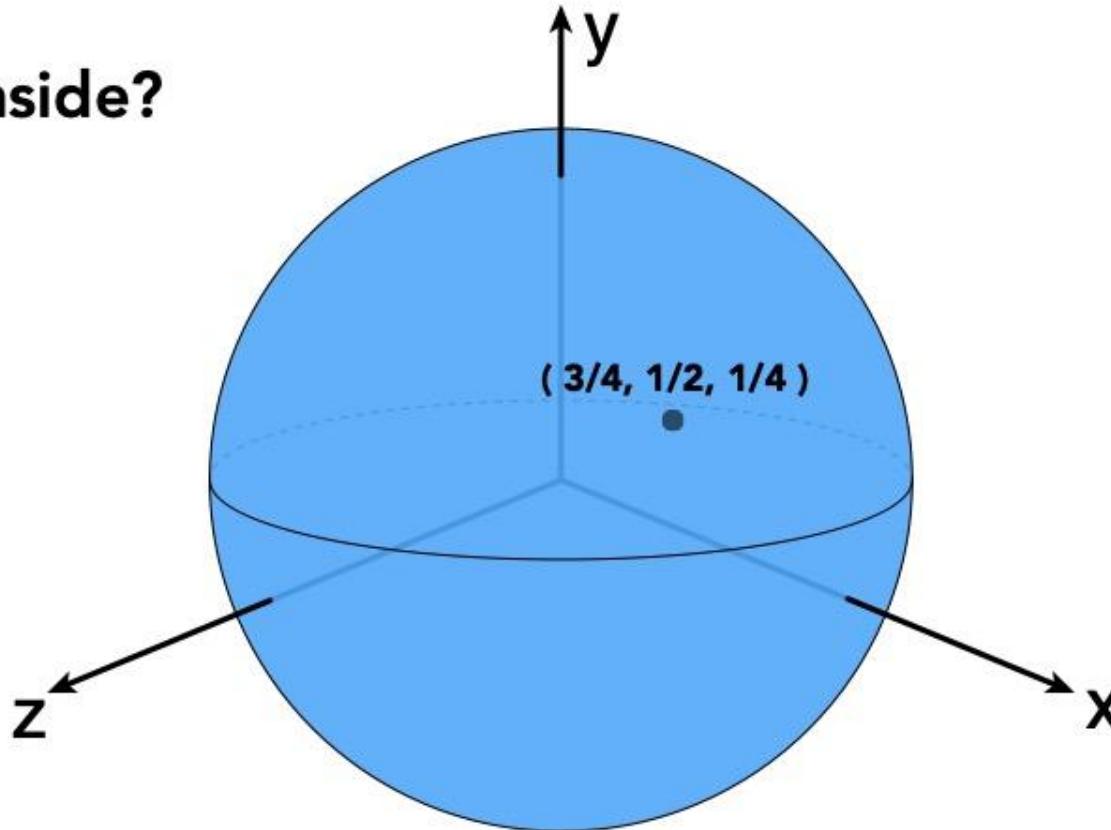
Is $(3/4, 1/2, 1/4)$ inside?



Explicit Surface – Inside/Outside Test Hard

$$f(u, v) = (\cos u \sin v, \sin u \sin v, \cos v)$$

Is $(3/4, 1/2, 1/4)$ inside?



Some tasks are hard with explicit representations

“Implicit” Representations of Geometry

Based on classifying points

- Points satisfy some specified relationship

“Implicit” Representations of Geometry

Based on classifying points

- Points satisfy some specified relationship

E.g. sphere: all points in 3D, where $x^2+y^2+z^2 = 1$

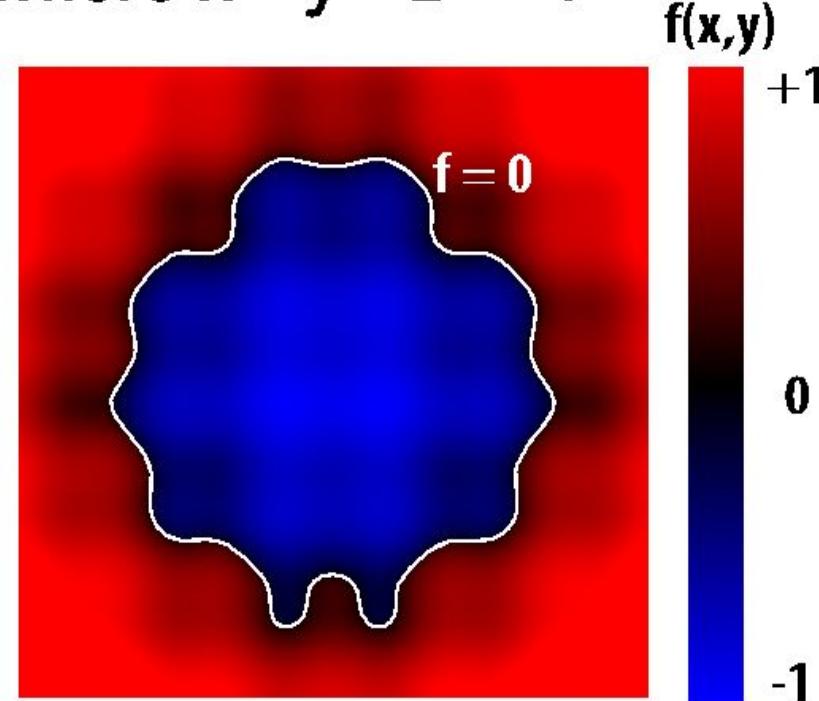
“Implicit” Representations of Geometry

Based on classifying points

- Points satisfy some specified relationship

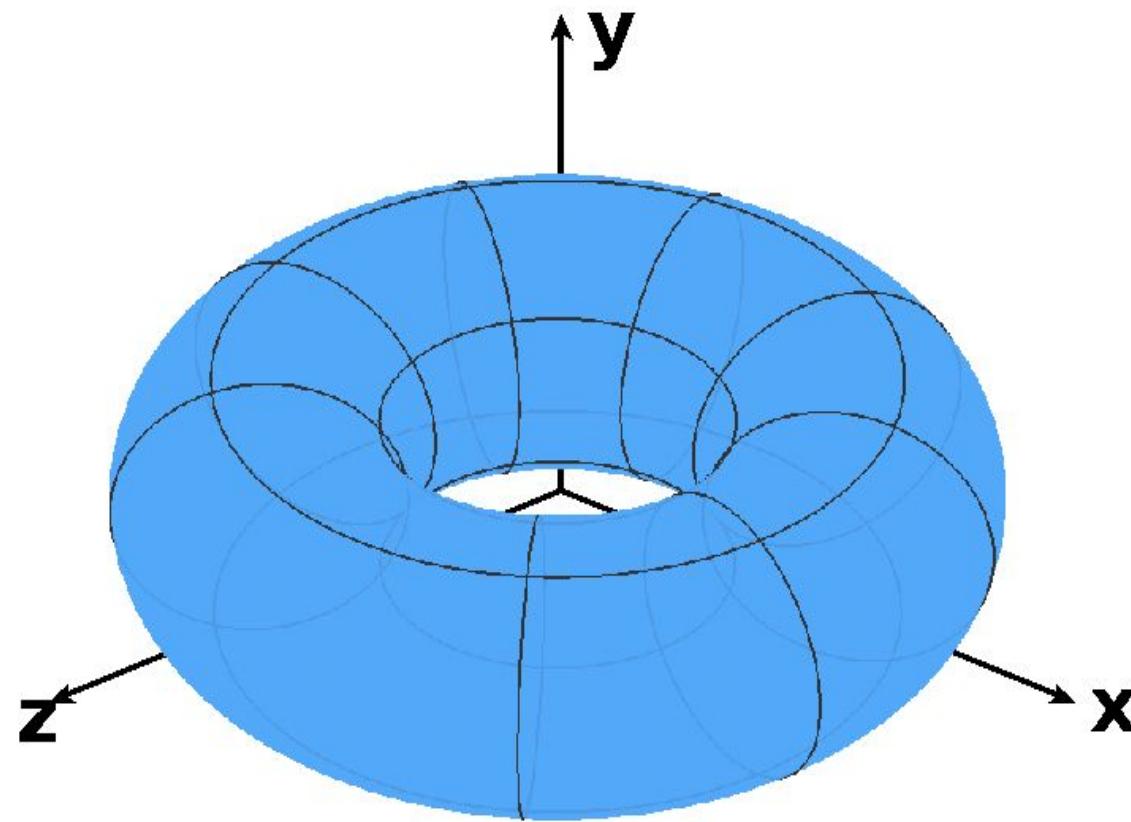
E.g. sphere: all points in 3D, where $x^2+y^2+z^2 = 1$

More generally, $f(x,y,z) = 0$



Implicit Surface – Sampling Can Be Hard

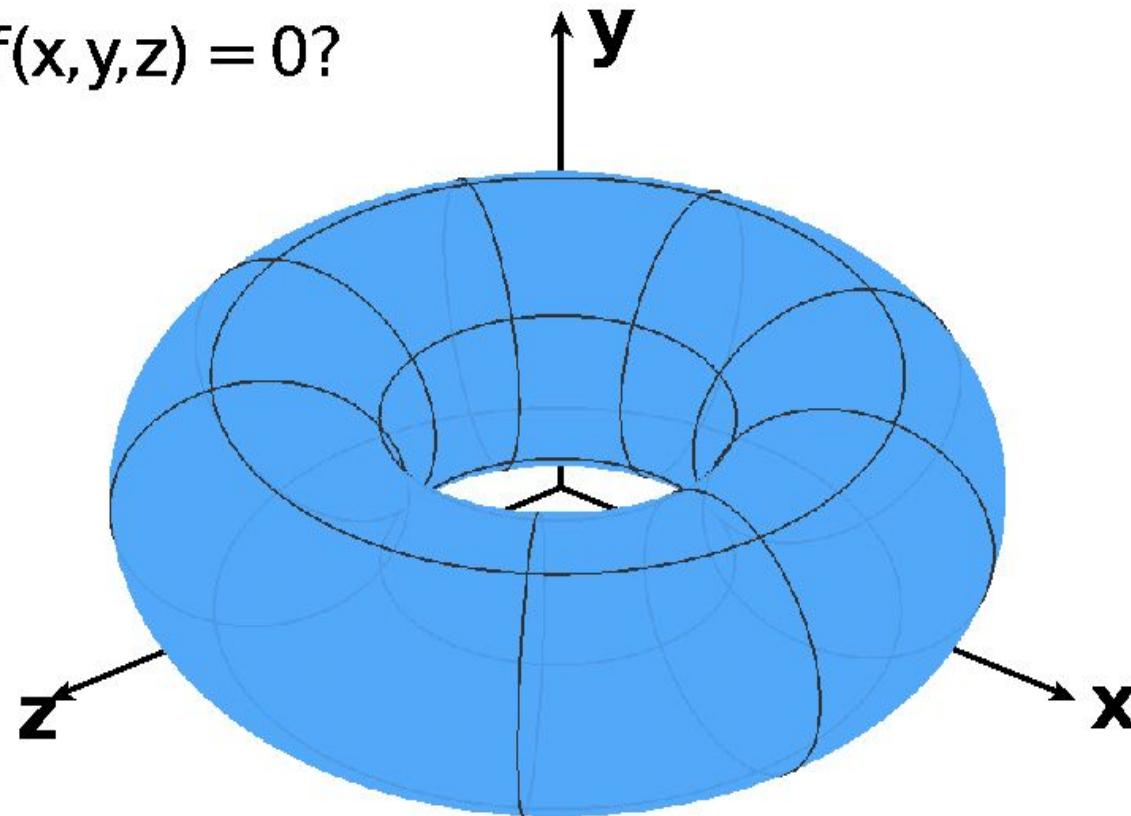
$$f(x, y, z) = (2 - \sqrt{x^2 + y^2})^2 + z^2 - 1$$



Implicit Surface – Sampling Can Be Hard

$$f(x, y, z) = (2 - \sqrt{x^2 + y^2})^2 + z^2 - 1$$

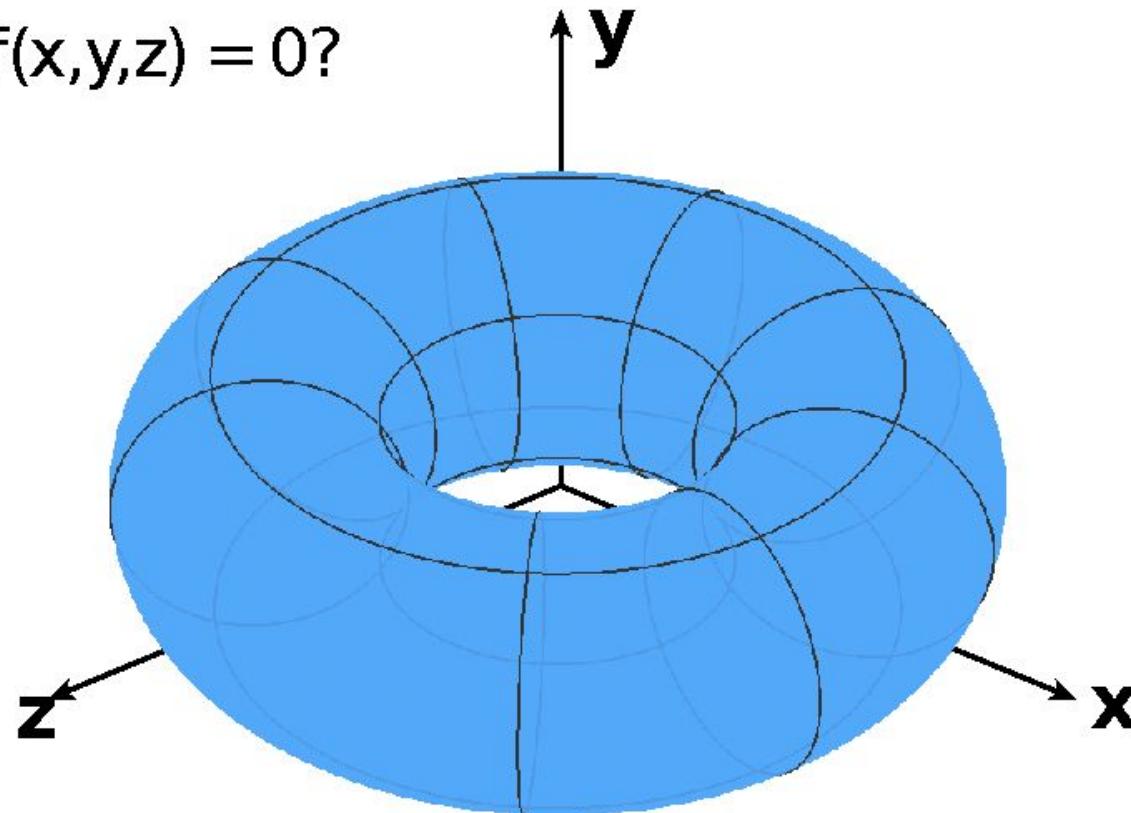
What points lie on $f(x, y, z) = 0$?



Implicit Surface – Sampling Can Be Hard

$$f(x, y, z) = (2 - \sqrt{x^2 + y^2})^2 + z^2 - 1$$

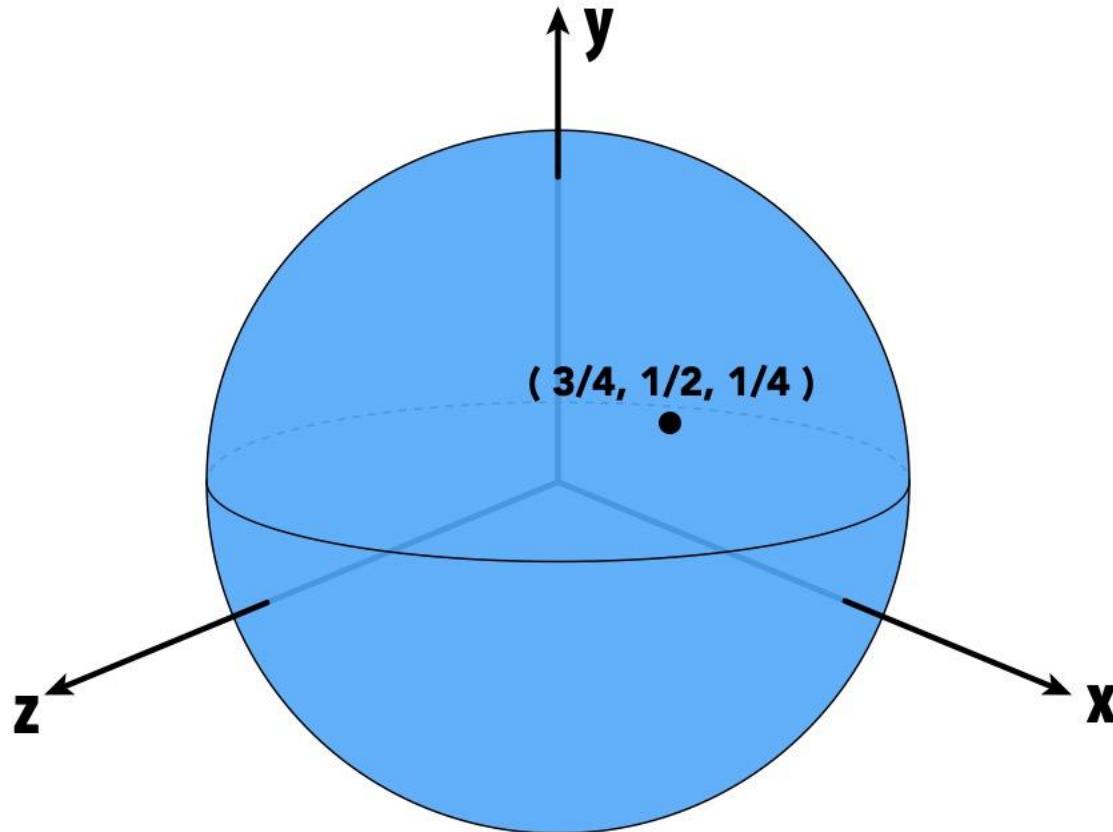
What points lie on $f(x, y, z) = 0$?



Some tasks are hard with implicit representations

Implicit Surface – Inside/Outside Tests Easy

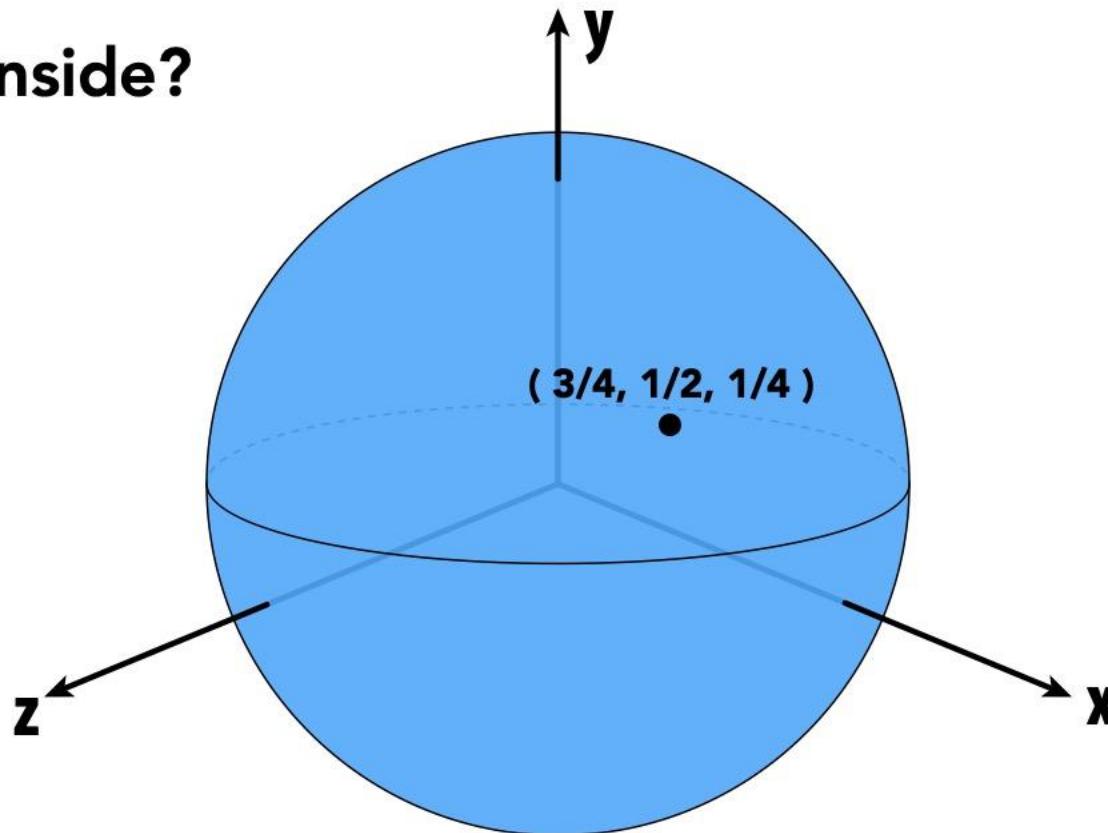
$$f(x, y, z) = x^2 + y^2 + z^2 - 1$$



Implicit Surface – Inside/Outside Tests Easy

$$f(x, y, z) = x^2 + y^2 + z^2 - 1$$

Is $(3/4, 1/2, 1/4)$ inside?



Implicit Surface – Inside/Outside Tests Easy

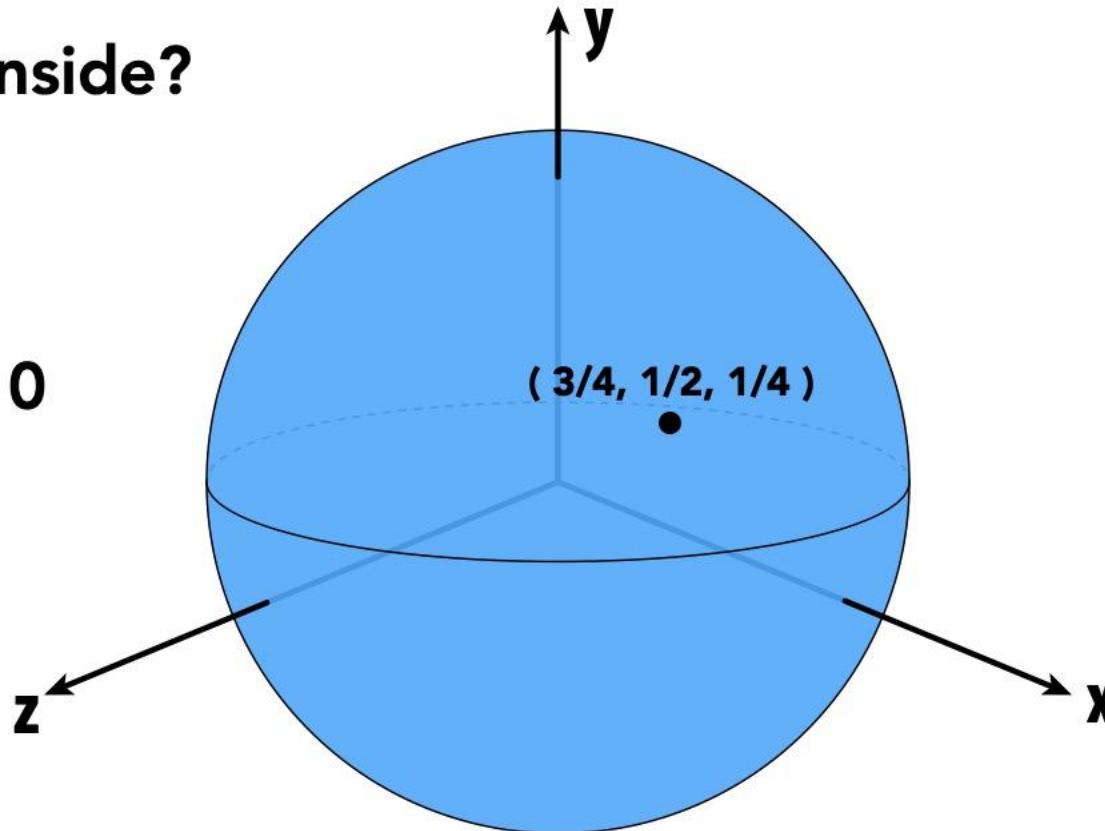
$$f(x, y, z) = x^2 + y^2 + z^2 - 1$$

Is $(3/4, 1/2, 1/4)$ inside?

Just plug it in:

$$f(x, y, z) = -1/8 < 0$$

Yes, inside.



Implicit Surface – Inside/Outside Tests Easy

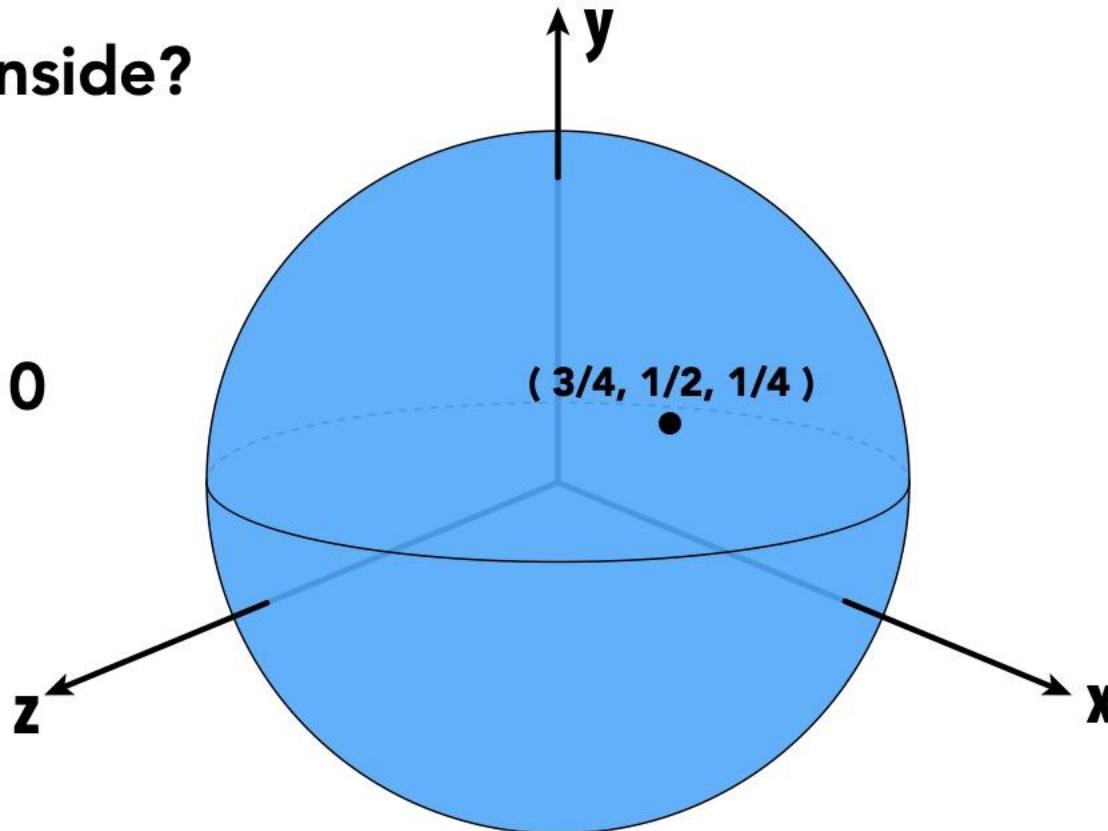
$$f(x, y, z) = x^2 + y^2 + z^2 - 1$$

Is $(3/4, 1/2, 1/4)$ inside?

Just plug it in:

$$f(x, y, z) = -1/8 < 0$$

Yes, inside.



Implicit representations make some tasks easy

Algebraic Surfaces (Implicit)

Surface is zero set of a polynomial in x, y, z



$$x^2 + y^2 + z^2 = 1$$

Algebraic Surfaces (Implicit)

Surface is zero set of a polynomial in x, y, z



$$x^2 + y^2 + z^2 = 1$$



$$(R - \sqrt{x^2 + y^2})^2 + z^2 = r^2$$



$$(x^2 + \frac{9y^2}{4} + z^2 - 1)^3 =$$

$$x^2 z^3 + \frac{9y^2 z^3}{80}$$

Algebraic Surfaces (Implicit)

Surface is zero set of a polynomial in x, y, z



$$x^2 + y^2 + z^2 = 1$$



$$(R - \sqrt{x^2 + y^2})^2 + z^2 = r^2$$



$$(x^2 + \frac{9y^2}{4} + z^2 - 1)^3 =$$

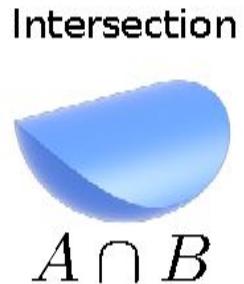
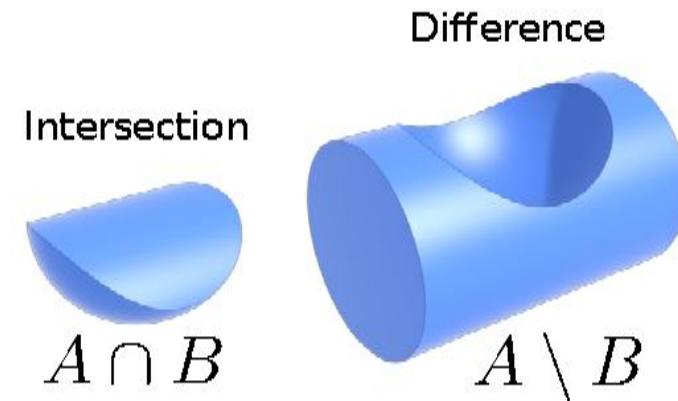
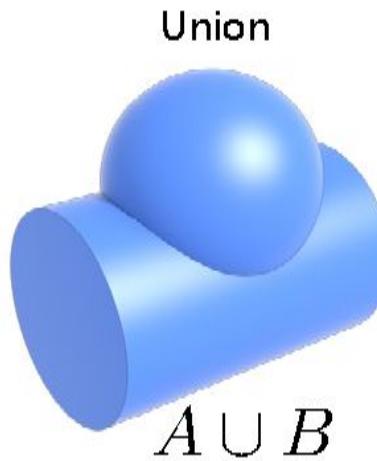
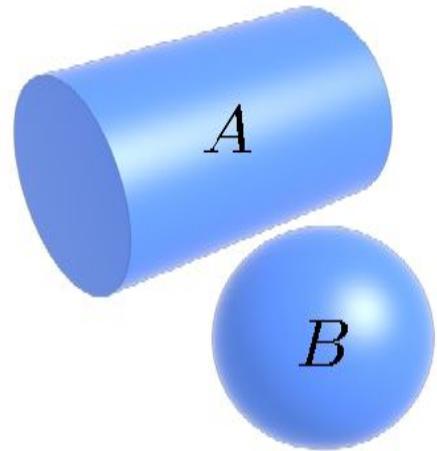
$$x^2 z^3 + \frac{9y^2 z^3}{80}$$



More complex shapes?

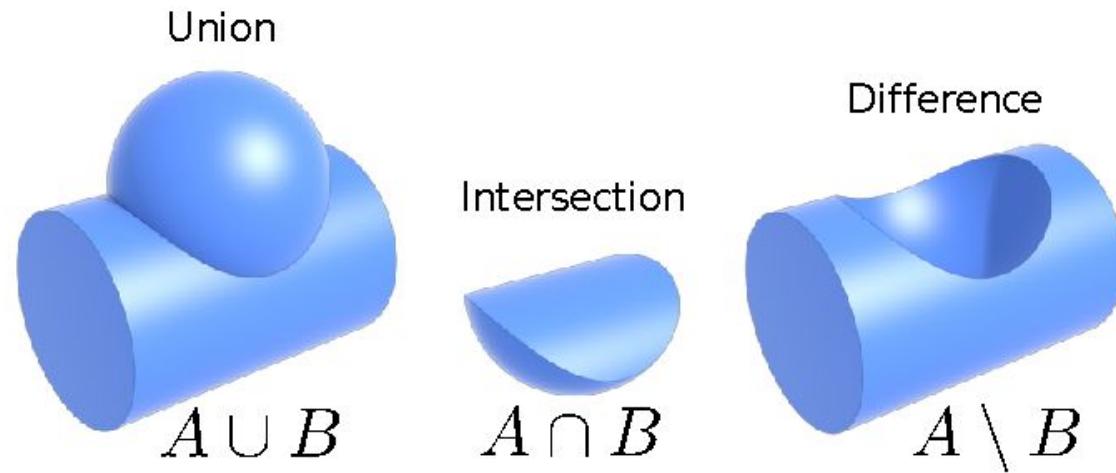
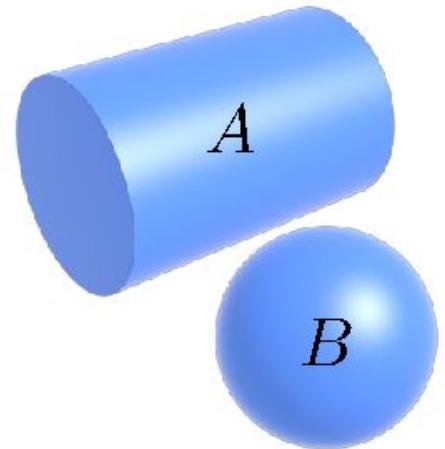
Constructive Solid Geometry (Implicit)

Combine implicit geometry via Boolean operations

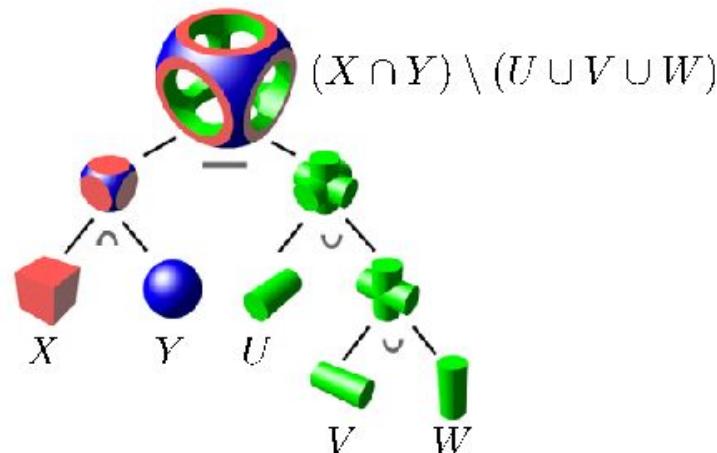


Constructive Solid Geometry (Implicit)

Combine implicit geometry via Boolean operations



Boolean expressions:

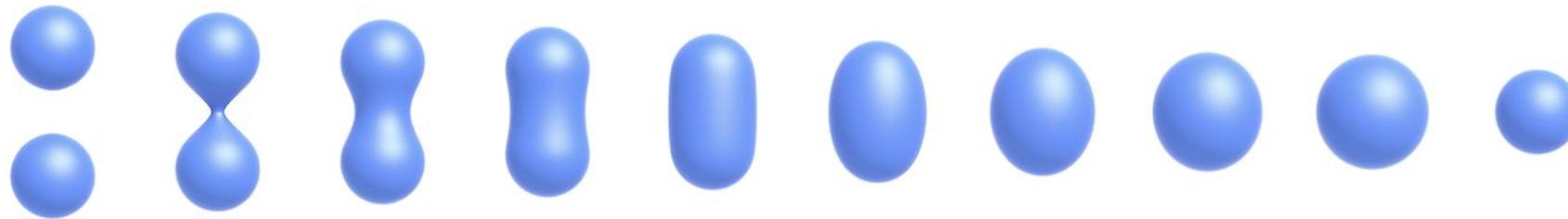


Distance Functions (Implicit)

Instead of Booleans, gradually blend surfaces together using

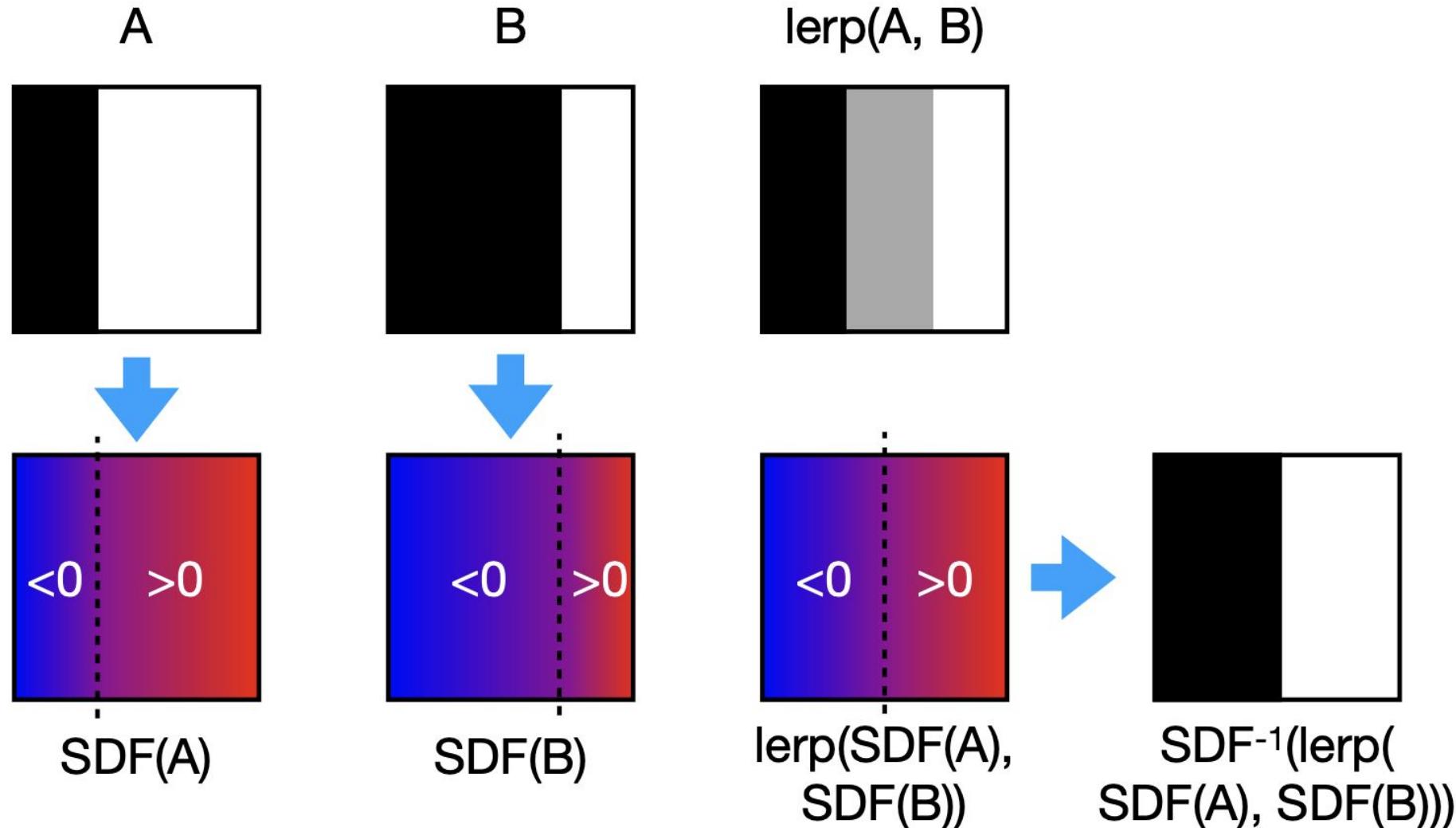
Distance functions:

giving minimum distance (could be **signed** distance)
from anywhere to object



Distance Functions (Implicit)

An Example: Blending (linear interp.) a moving boundary

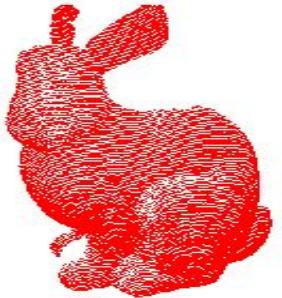
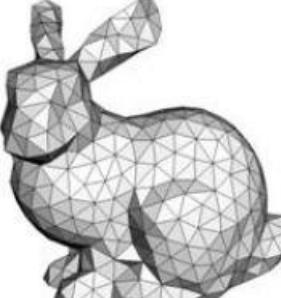
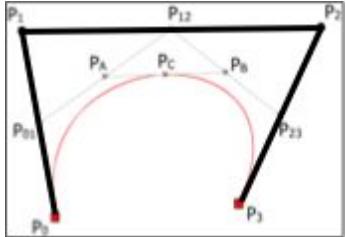
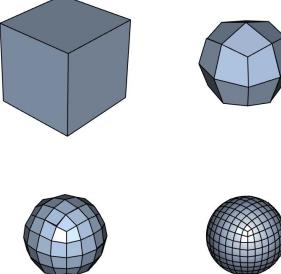
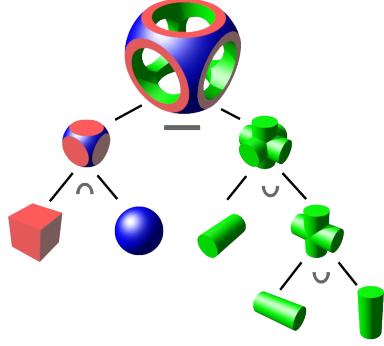


Scene of Pure Distance Functions (Not Easy!)



See <http://iquilezles.org/www/material/nvscene2008/nvscene2008.htm>

Shape Representations

	Explicit	Implicit
Non-parametric	 Points	 Meshes
Parametric	 Splines	 $x^2 + y^2 + z^2 = 1$ Algebraic Surfaces
	 Subdivision Surfaces	 Constructive Solid Geometry

Level Set Methods (Implicit)

Implicit surfaces have some nice features (e.g., merging/splitting)

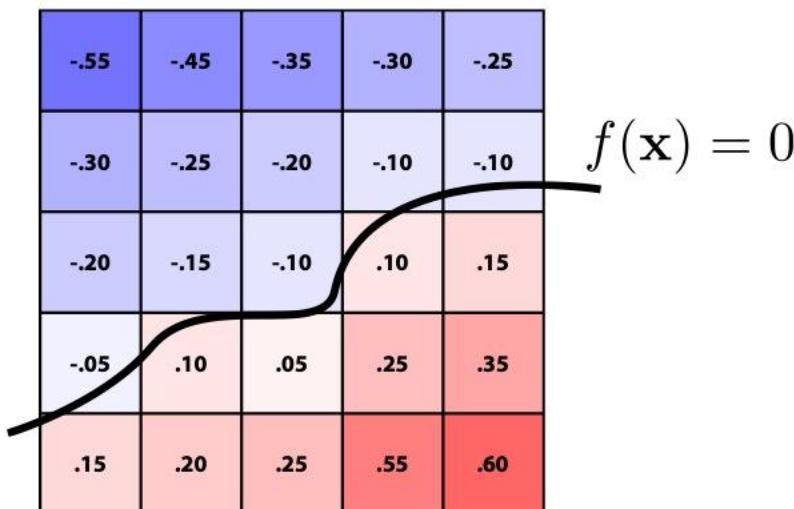
But, hard to describe complex shapes in closed form

Level Set Methods (Implicit)

Implicit surfaces have some nice features (e.g., merging/splitting)

But, hard to describe complex shapes in closed form

Alternative: store a grid of values approximating function

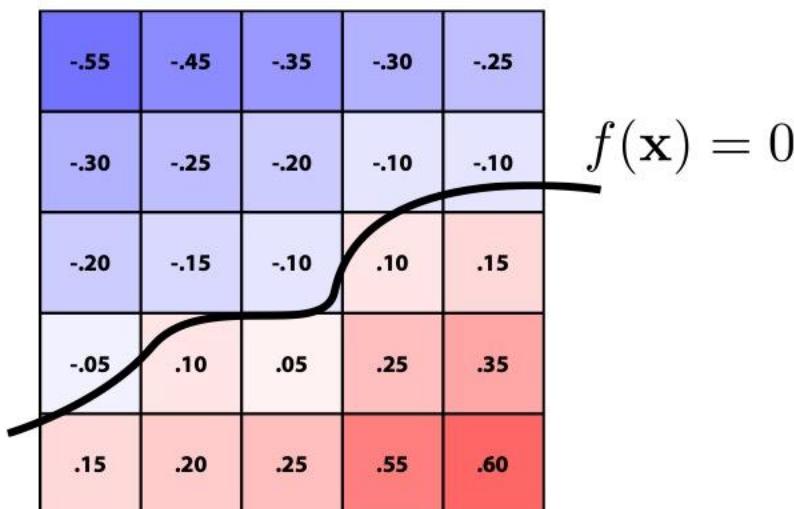


Level Set Methods (Implicit)

Implicit surfaces have some nice features (e.g., merging/splitting)

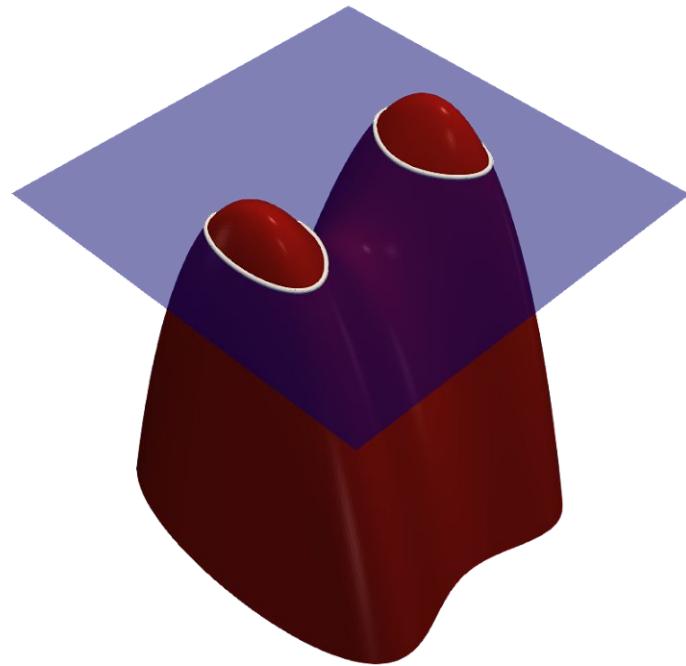
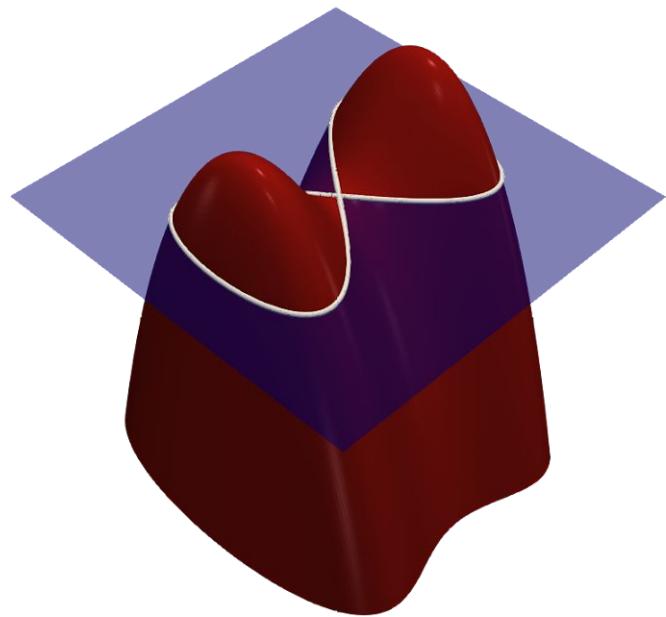
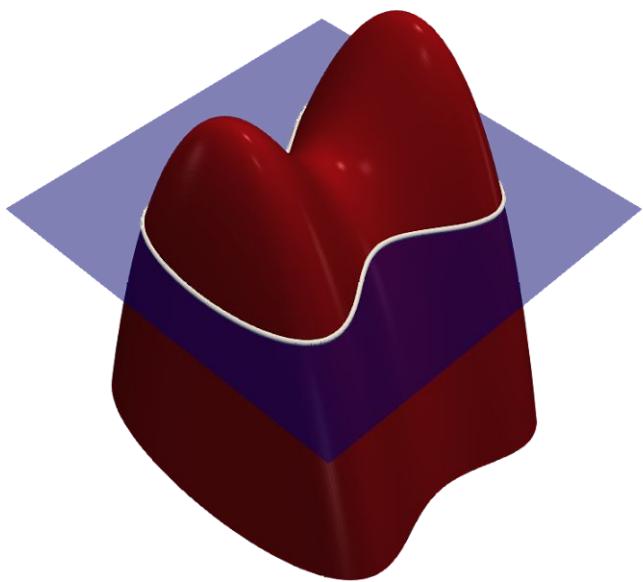
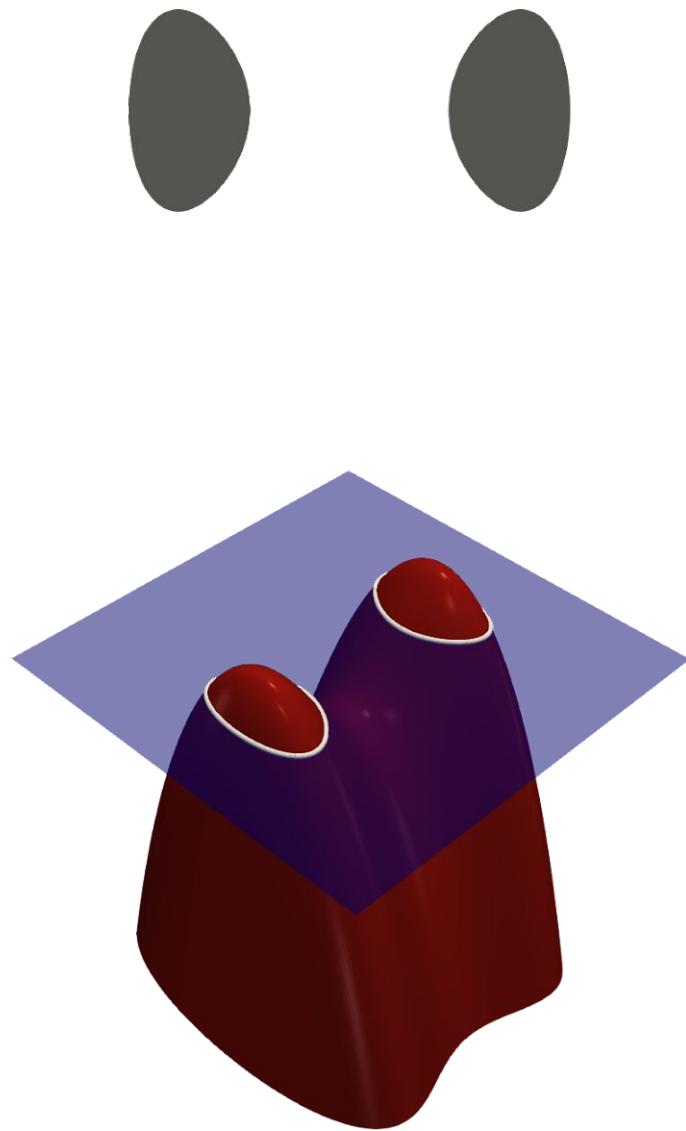
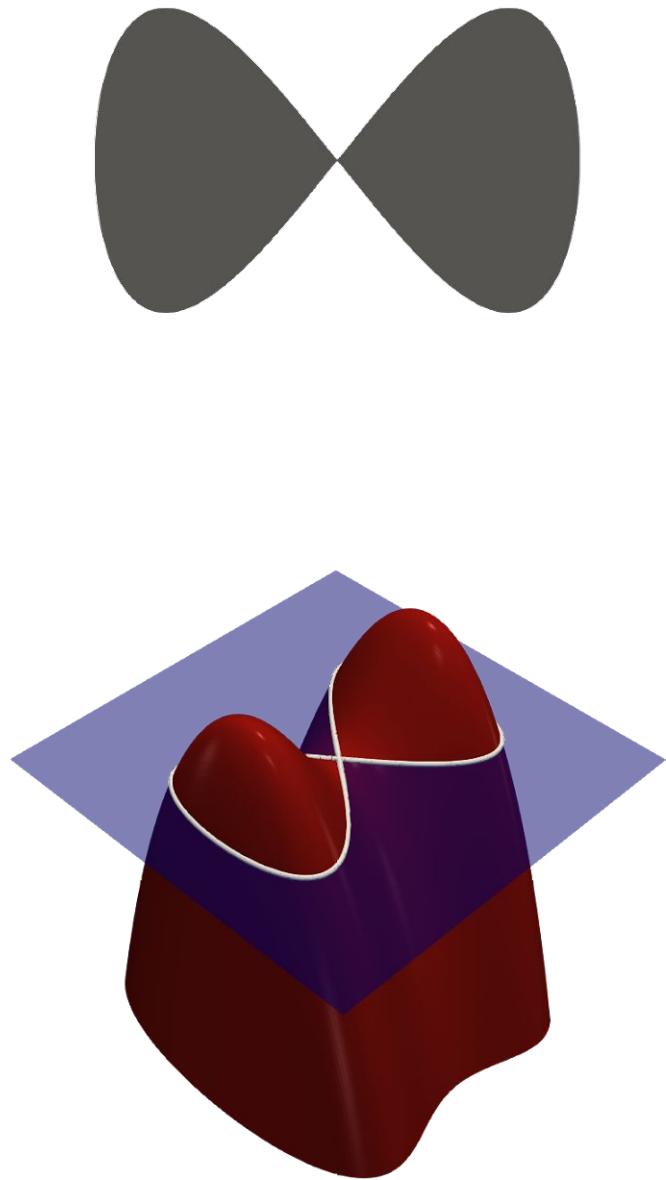
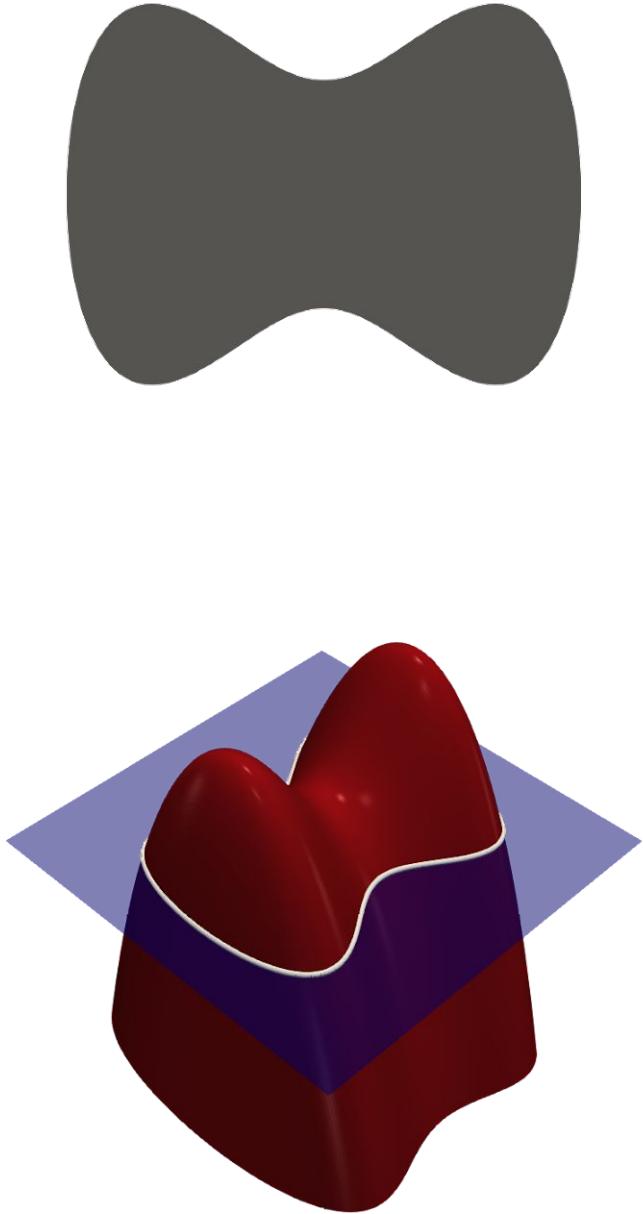
But, hard to describe complex shapes in closed form

Alternative: store a grid of values approximating function



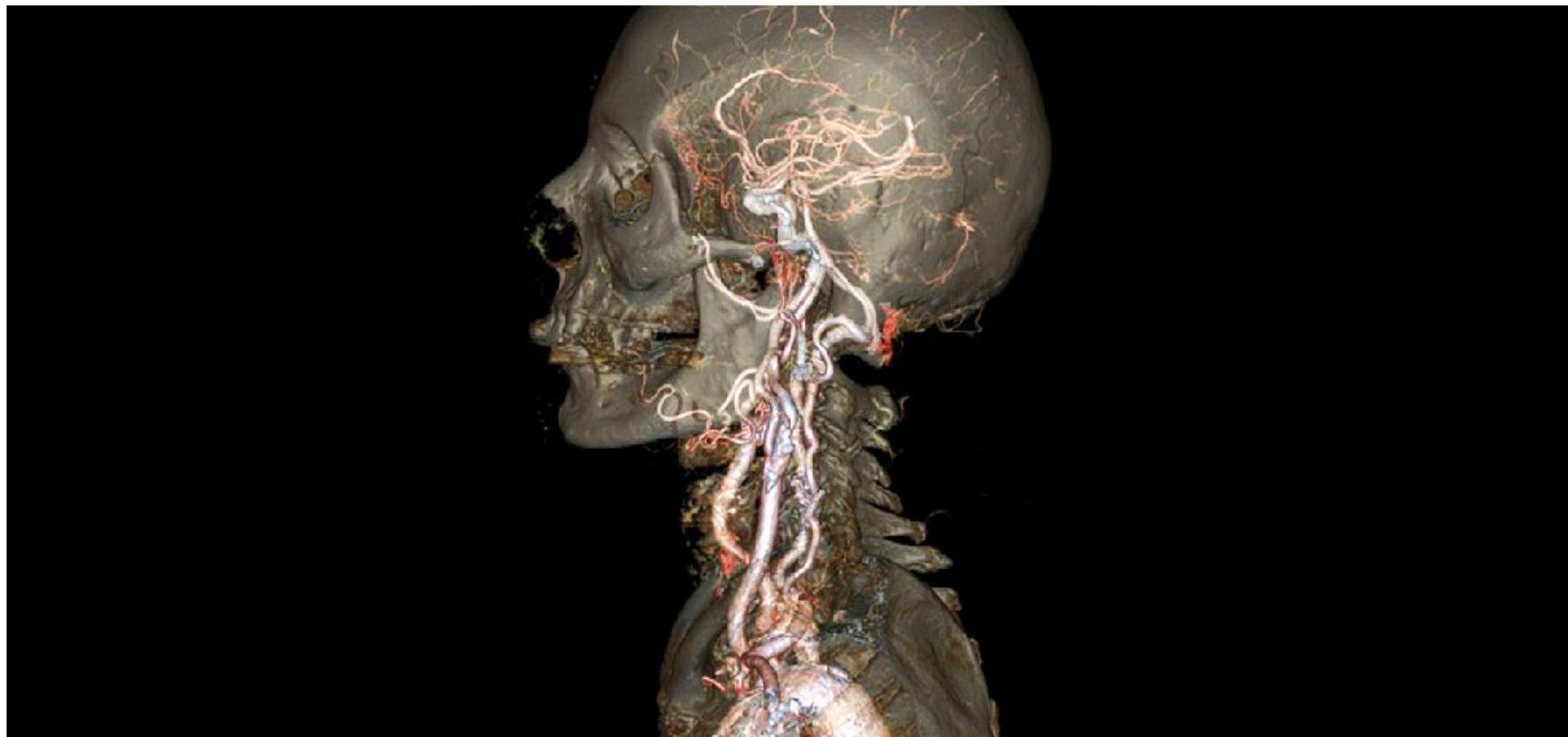
Surface is found where interpolated values equal zero

Provides much more explicit control over shape (like a texture)



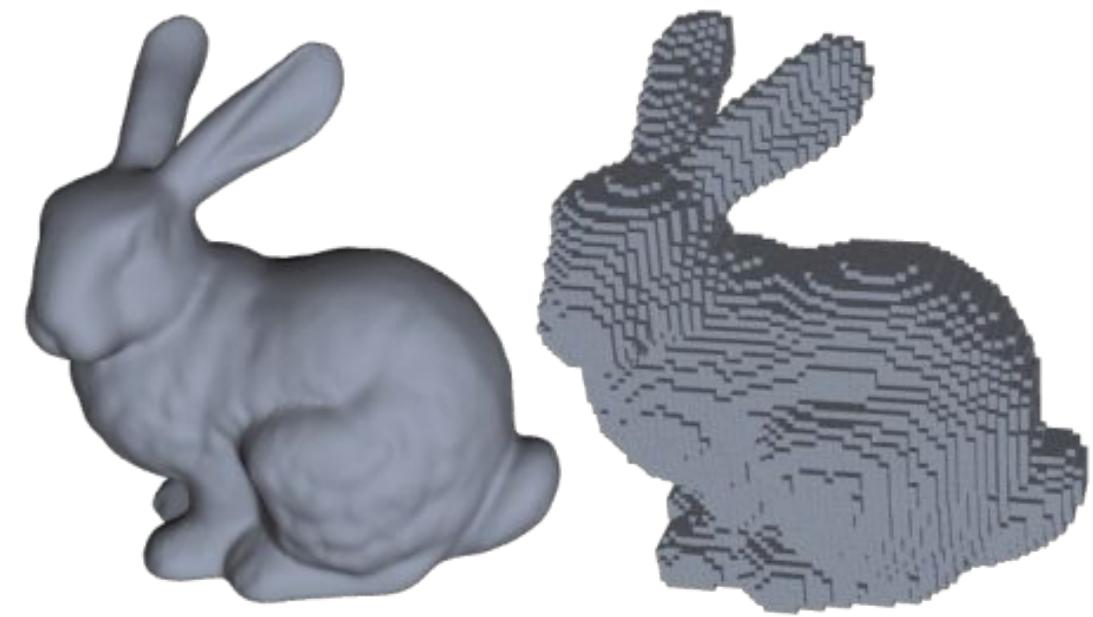
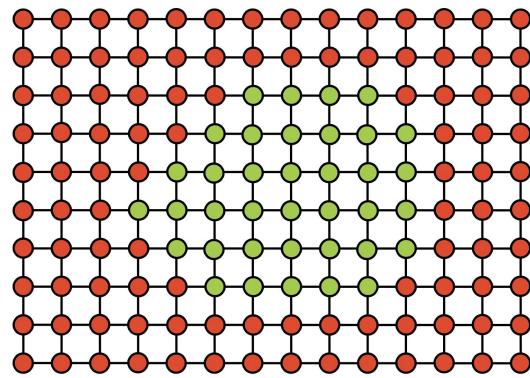
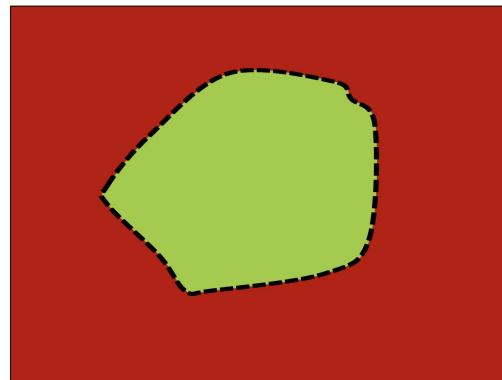
Level Sets from Medical Data (CT, MRI, etc.)

Level sets encode, e.g., constant tissue density



Related Representation: Voxels

- Binary thresholding the volumetric grid



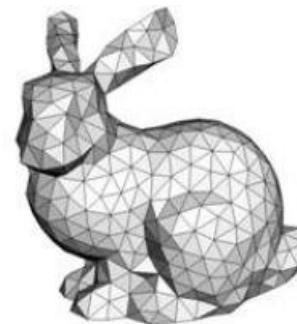
Shape Representations

Non-parametric

Explicit



Points

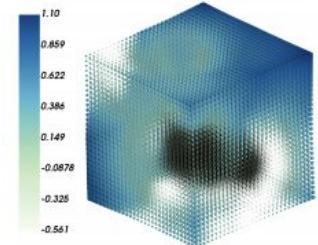


Meshes

Implicit (Eulerian)

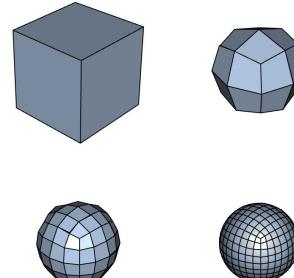
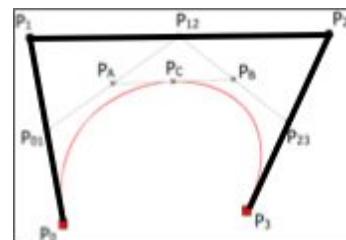


Voxels



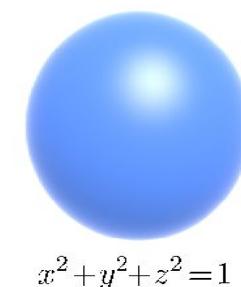
Level Sets

Parametric

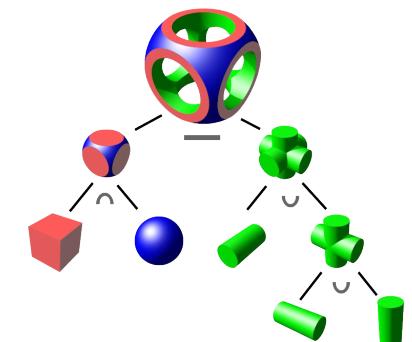


Splines

Subdivision
Surfaces



Algebraic
Surfaces



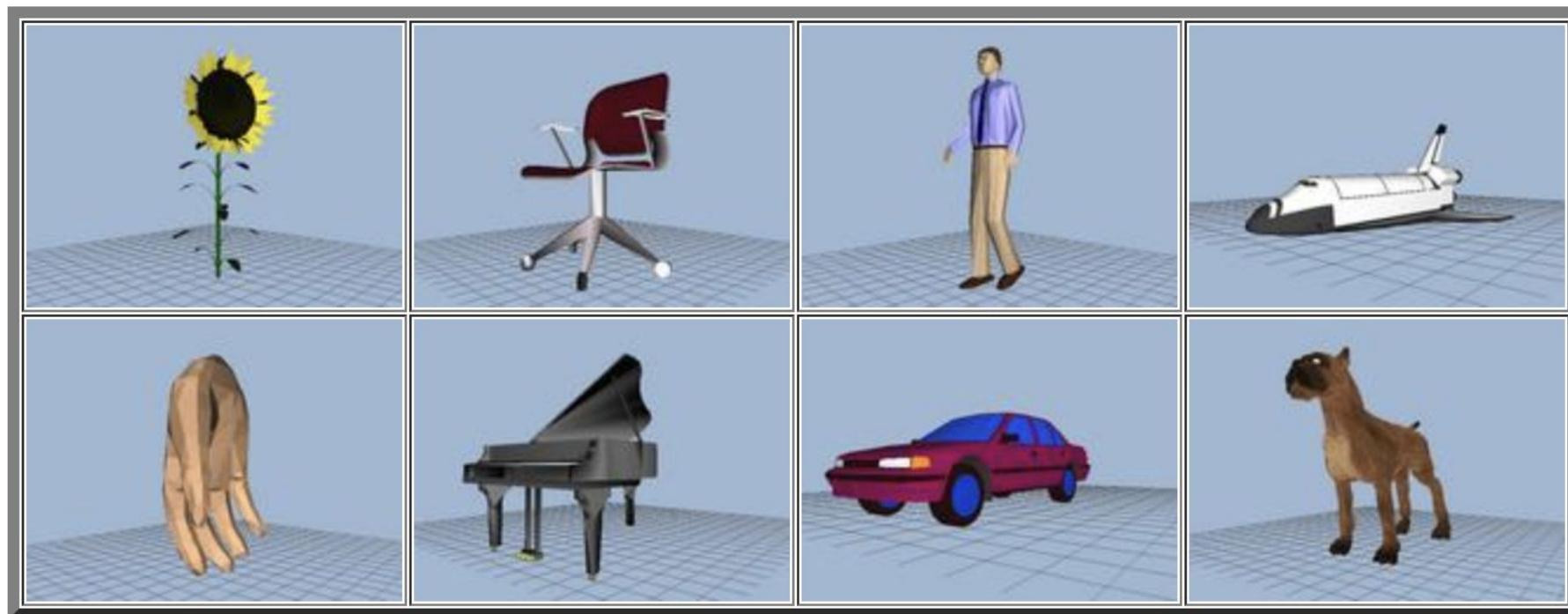
Constructive
Solid
Geometry

AI + Shapes

- In 3D ...

Princeton Shape Benchmark

- 1814 Models
- 182 Categories



Datasets Prior to 2014

Benchmarks	Types	# models	# classes	Avg # models per class
SHREC14LSGTB	Generic	8,987	171	53
PSB	Generic	907+907 (train+test)	90+92 (train+test)	10+10 (train+test)
SHREC12GTB	Generic	1200	60	20
TSB	Generic	10,000	352	28
CCCC	Generic	473	55	9
WMB	Watertight (articulated)	400	20	20
MSB	Articulated	457	19	24
BAB	Architecture	2257	183+180 (function+form)	12+13 (function+form)
ESB	CAD	867	45	19

Table 1. Source datasets from SHREC 2014: *Princeton Shape Benchmark (PSB)* [27], *SHREC 2012 generic Shape Benchmark (SHREC12GTB)* [16], *Toyohashi Shape Benchmark (TSB)* [29], *Konstanz 3D Model Benchmark (CCCC)* [32], *Watertight Model Benchmark (WMB)* [31], *McGill 3D Shape Benchmark (MSB)* [37], *Bonn Architecture Benchmark (BAB)* [33], *Purdue Engineering Shape Benchmark (ESB)* [9].

Datasets for 3D Objects

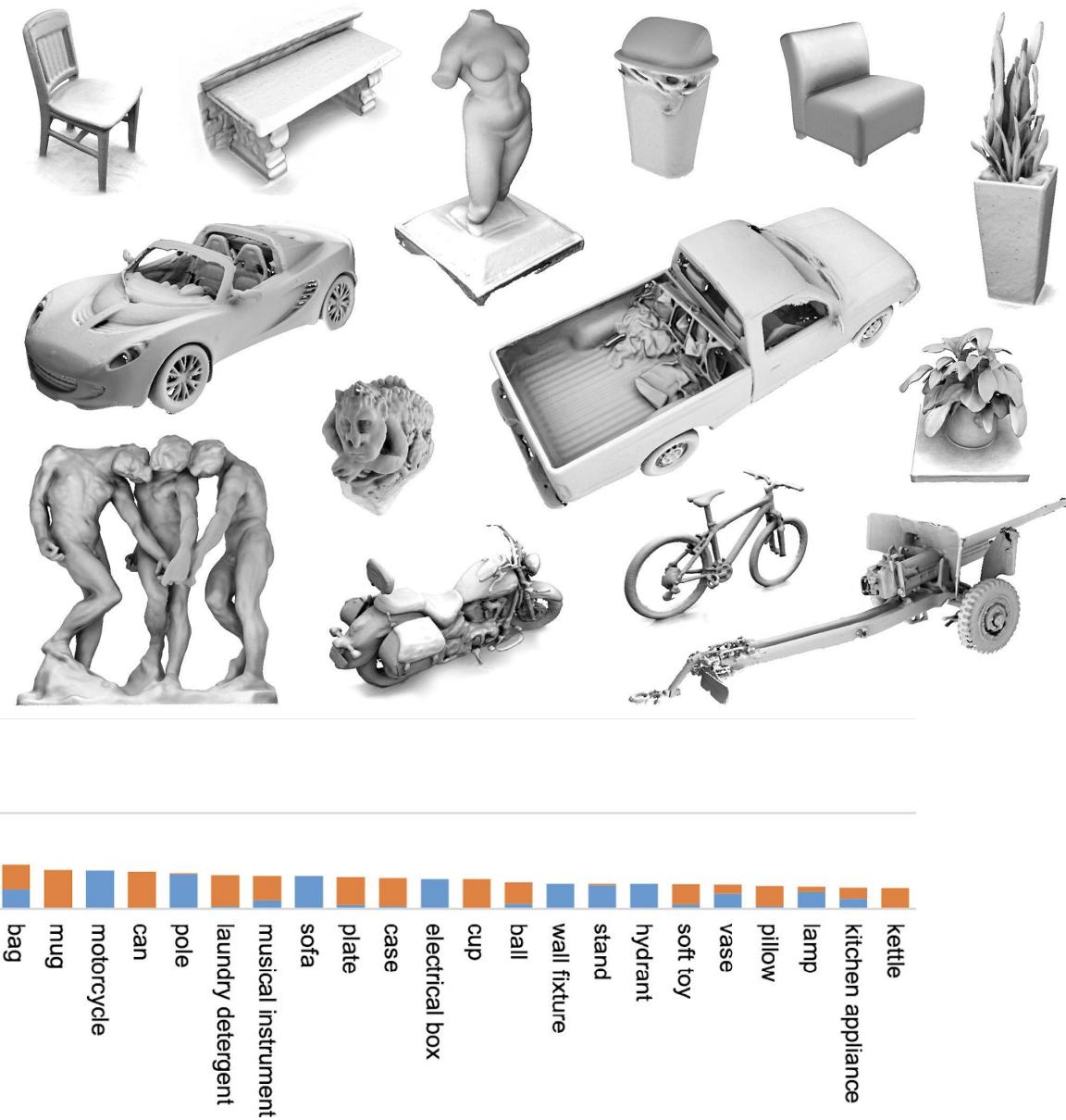
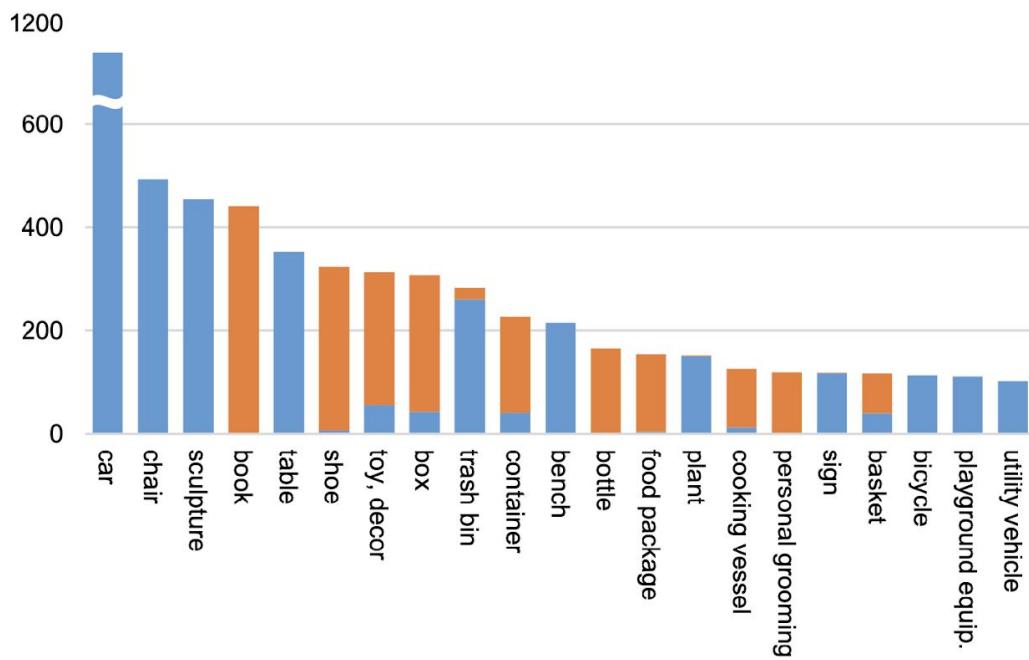
- Large-scale Synthetic Objects: ShapeNet, 3M models
- ModelNet: absorbed by ShapeNet
- ShapeNetCore: 51.3K models in 55 categories



Chang et al. ShapeNet. arXiv 2015
Wu et al. 3D ShapeNets. CVPR 2015

Object Scan

- 10,933 RGBD scans
- 441 models



Pascal 3D+

- Retrieve a nearest-neighbor 3D model for objects in real images
- 8,505 PASCAL images (13,898 instances) + 22,394 ImageNet images
- 12 rigid categories, 3,000+ instances per category on average

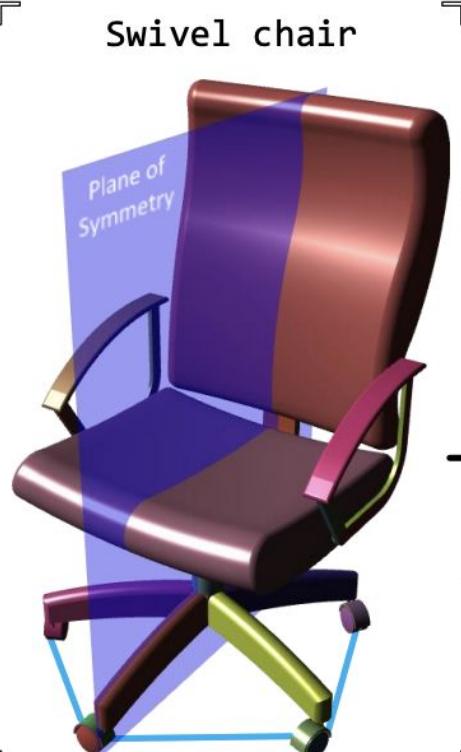


Pix3D

- 10,069 images
- 395 shapes (IKEA furniture + 3D scan)



Link to WordNet Taxonomy Alignment+Symmetry



Part Hierarchy Part Correspondences

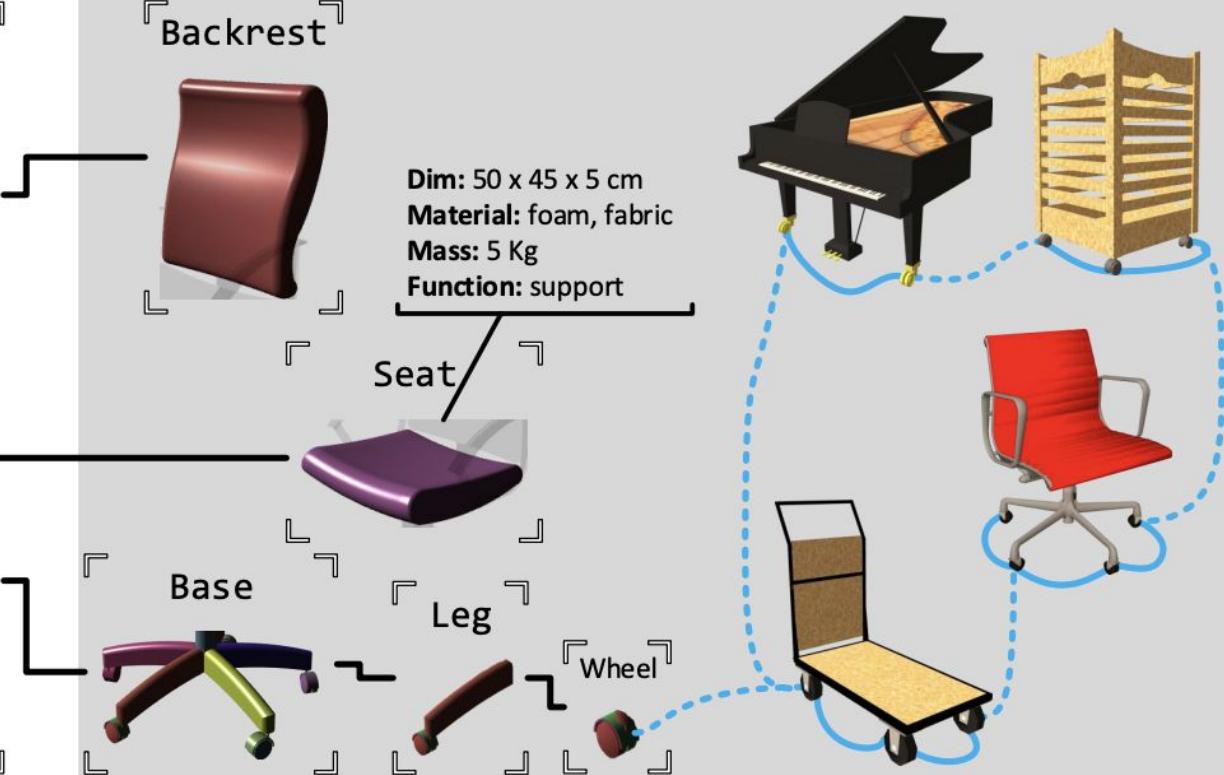
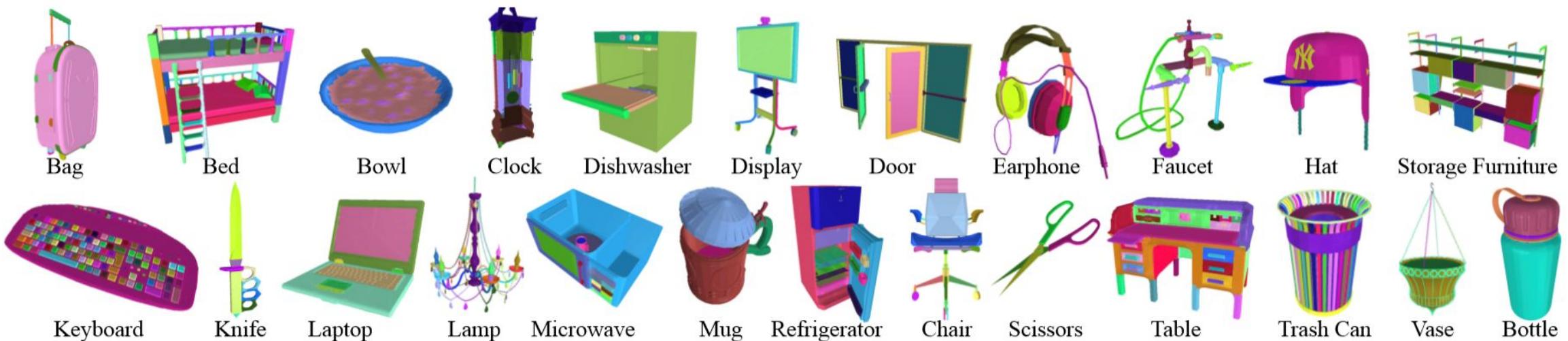


Figure from the ShapeNet paper, Chang et al. arXiv 2015

Datasets for 3D Object Parts

Fine-grained Parts: PartNet

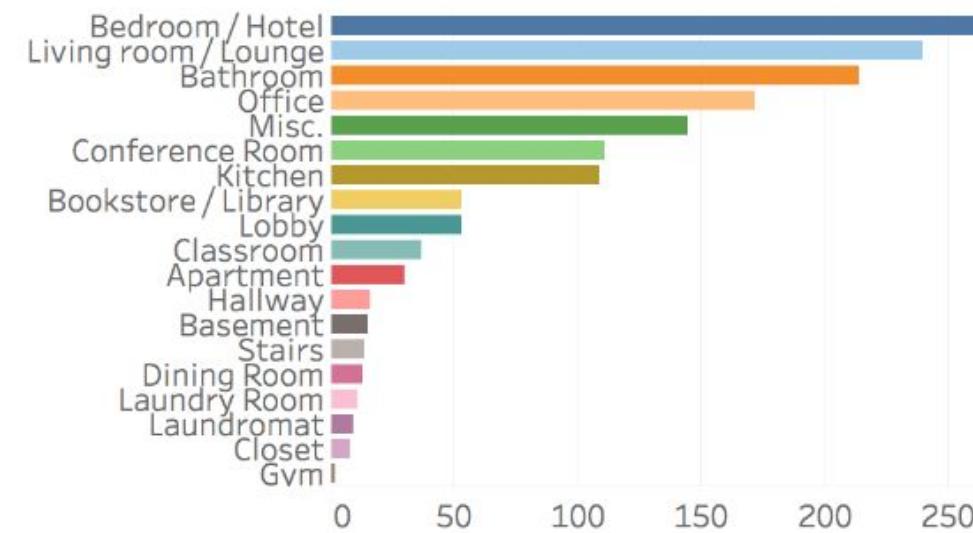
- Fine-grained (+mobility)
- Instance-level
- Hierarchical



Datasets for Indoor 3D Scenes

Large-scale Scanned Real Scenes: ScanNet

- 2.5M Views in 1,500 RGBD scans
- 3D camera poses
- Surface reconstructions
- Instance-level semantic segmentations



Physical Interaction with Articulated Objects

**300+ door
annotations**

**support
articulated
objects**

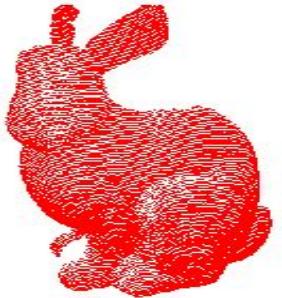
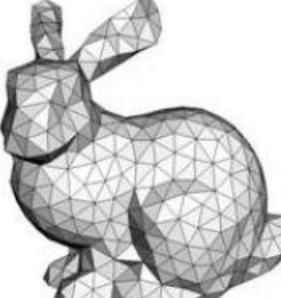
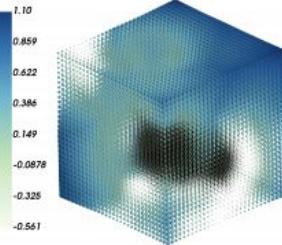
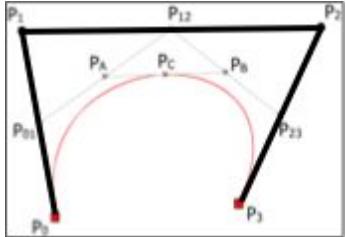
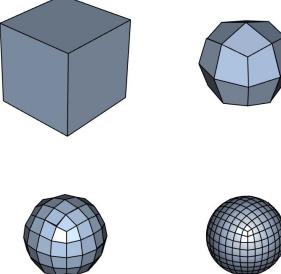
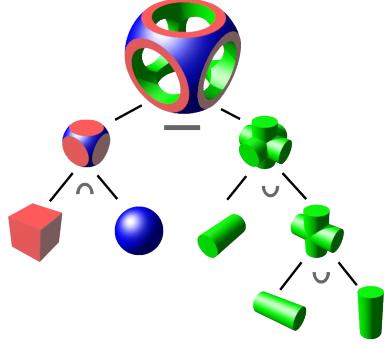
(cabinets, doors, fridge,
oven, window etc.)



AI + Geometry

- $P(S)$ or $P(S|c)$ --- Generative models
 - Learning (conditional) shape priors
 - Shape generation, completion, & geometry data processing
- $P(c|S)$ --- Discriminative models
 - Learning shape descriptors
 - Shape classification, segmentation, view estimation, etc.
- Joint modeling of 3D and 2D data
 - Large-scale 2D datasets & very good pretrained models
 - Differentiable projection/back-projection & differentiable/neural rendering

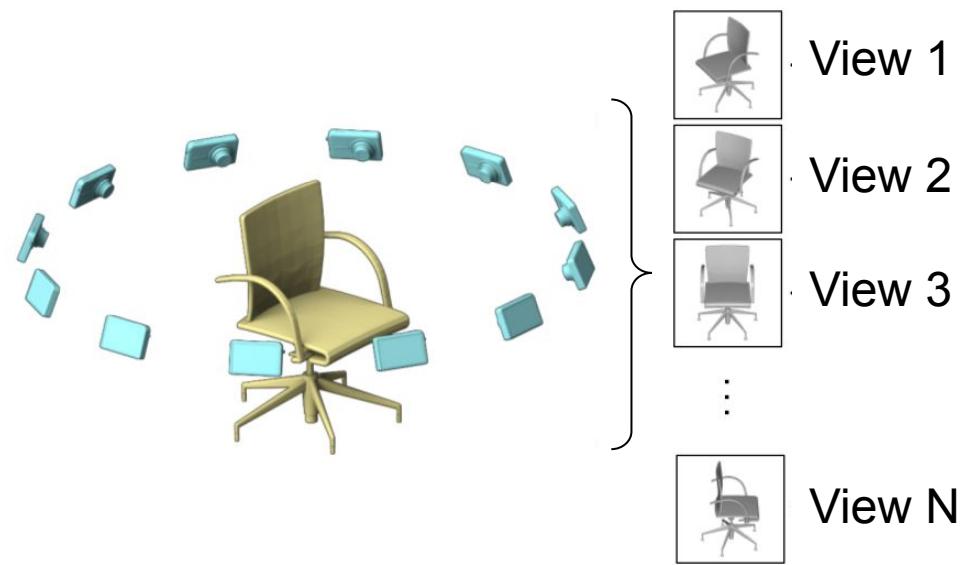
Which Shape Representation?

		Explicit	Implicit (Eulerian)
Non-parametric	Points		
	Meshes		
Parametric	Splines		 $x^2 + y^2 + z^2 = 1$
	Subdivision Surfaces		

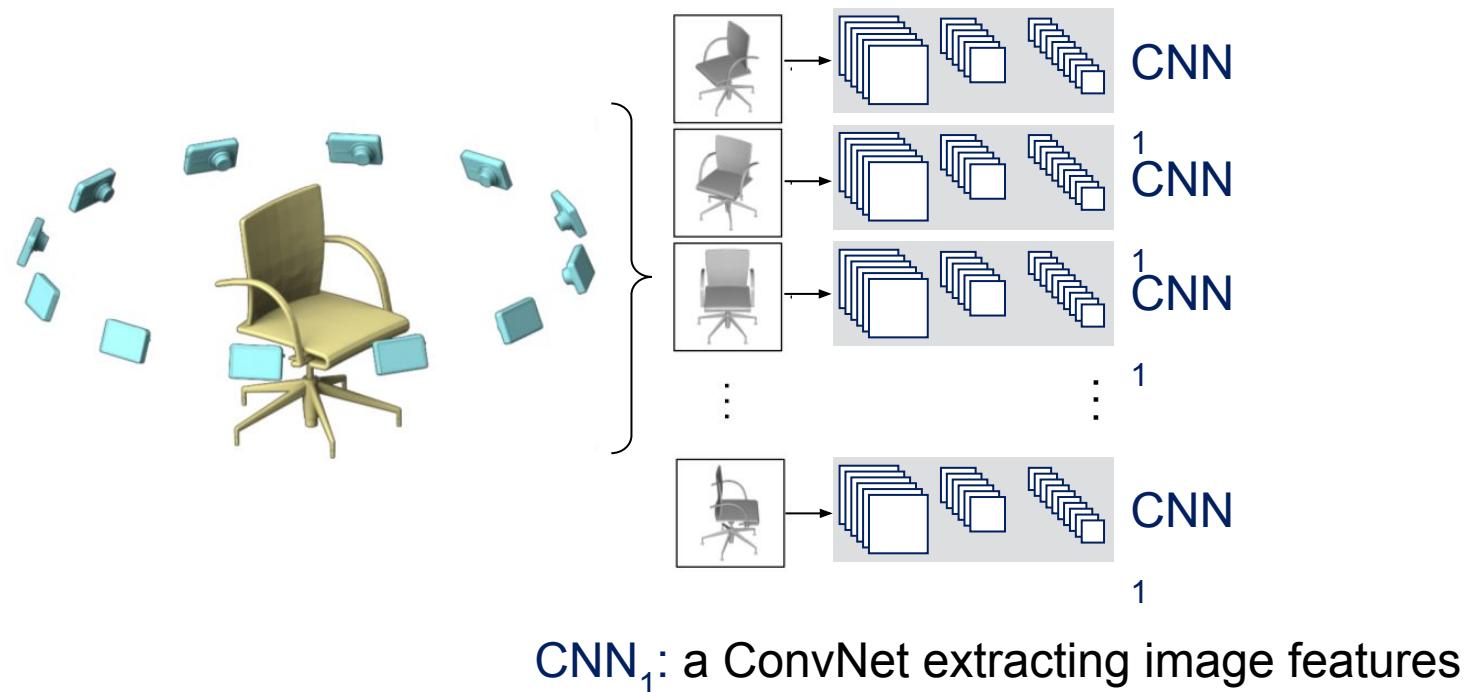
Multi-View CNN



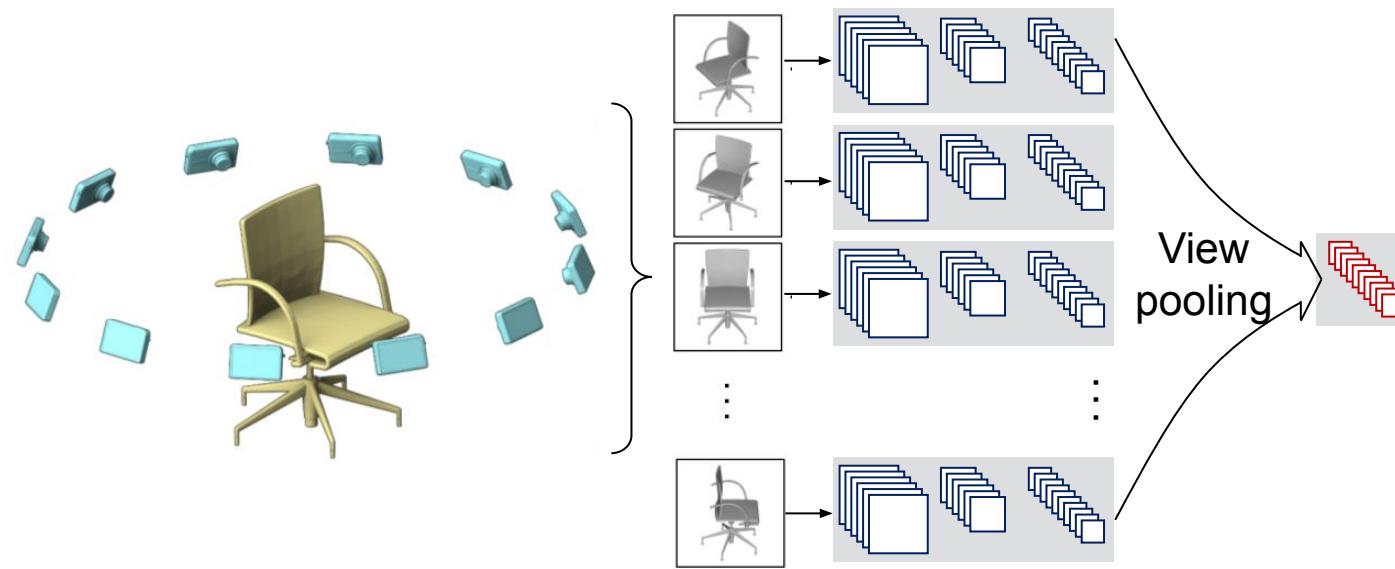
Multi-View CNN



Multi-View CNN

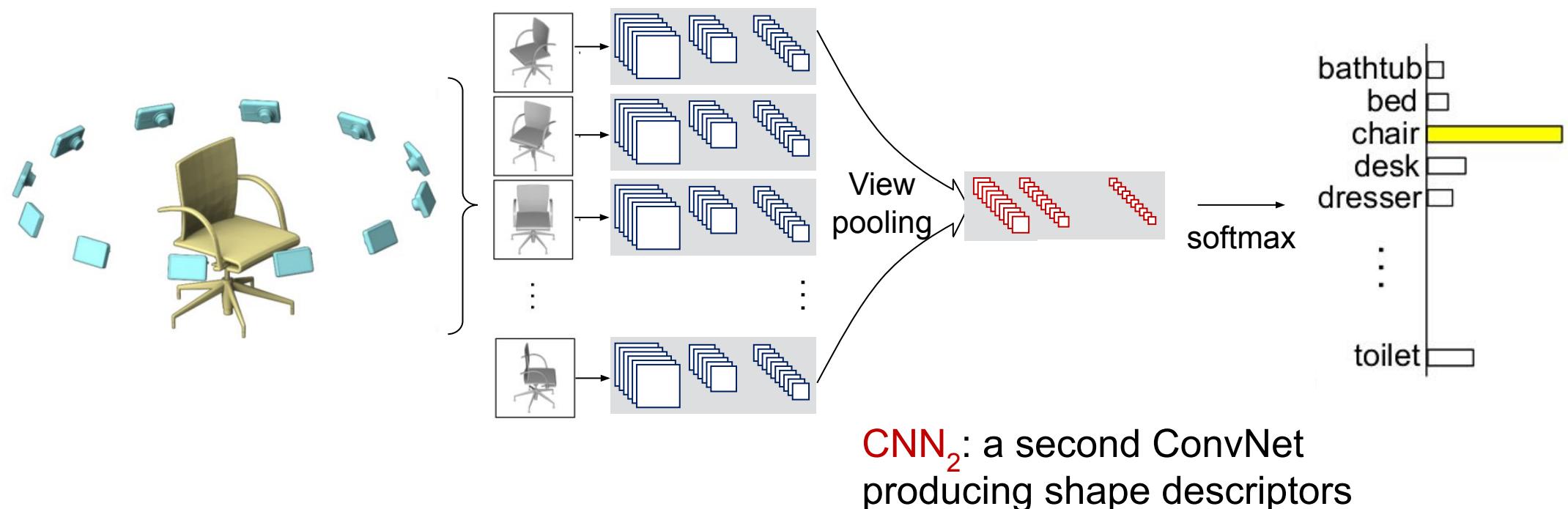


Multi-View CNN



View pooling: element-wise
max-pooling across all views

Multi-View CNN



Experiments – Classification & Retrieval

Non-dee
p {

Method	Classification	Retrieval
	(Accuracy)	(mAP)
SPH	68.2%	33.3%
LFD	75.5%	40.9%
3D ShapeNets	77.3%	49.2%
FV, 12 views	84.8%	43.9%
CNN, 12 views	88.6%	62.8%
MVCNN, 12 views	89.9%	70.1%
MVCNN+metric, 12 views	89.5%	80.2%
MVCNN, 80 views	90.1%	70.4%
MVCNN+metric, 80 views	90.1%	79.5%

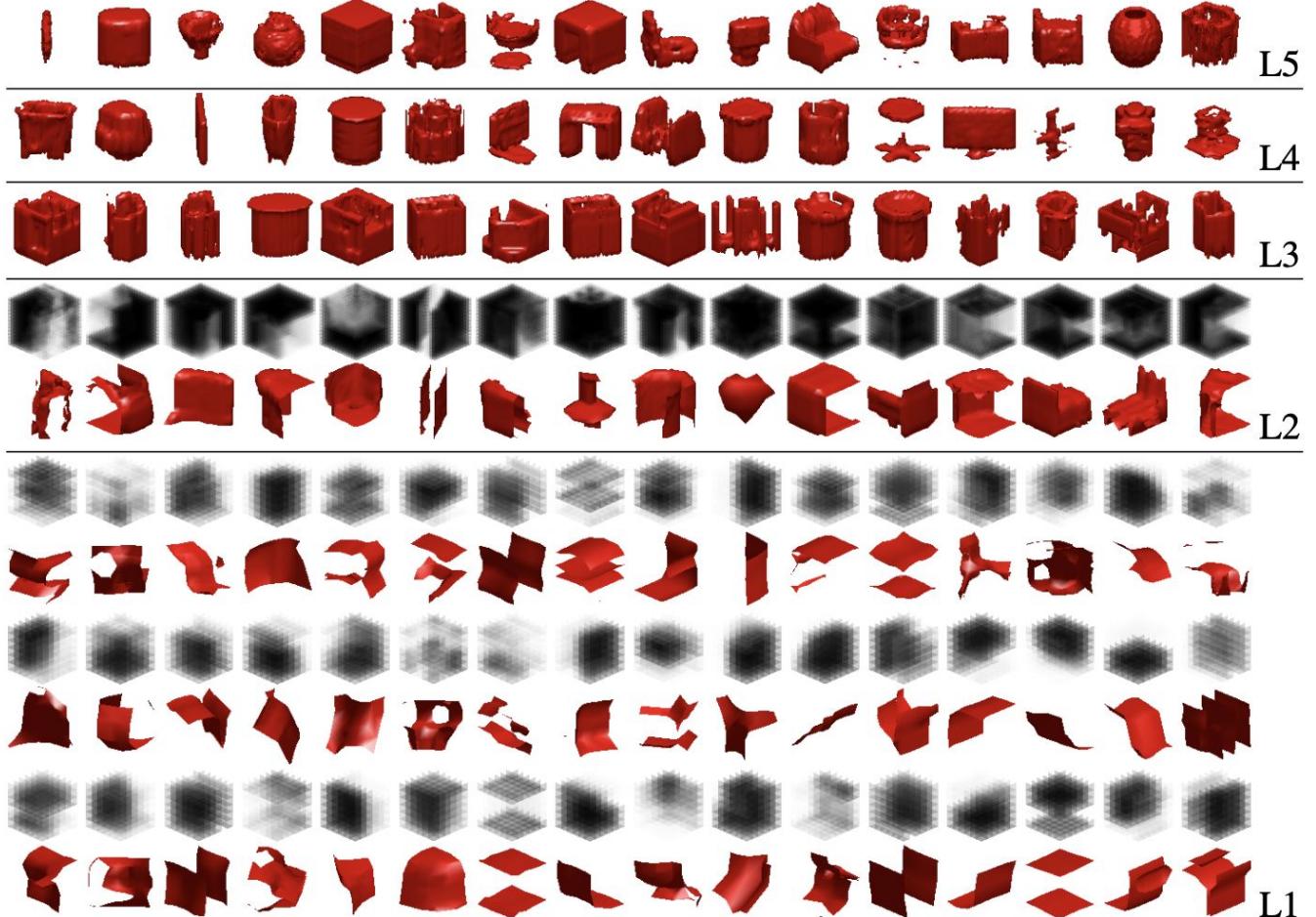
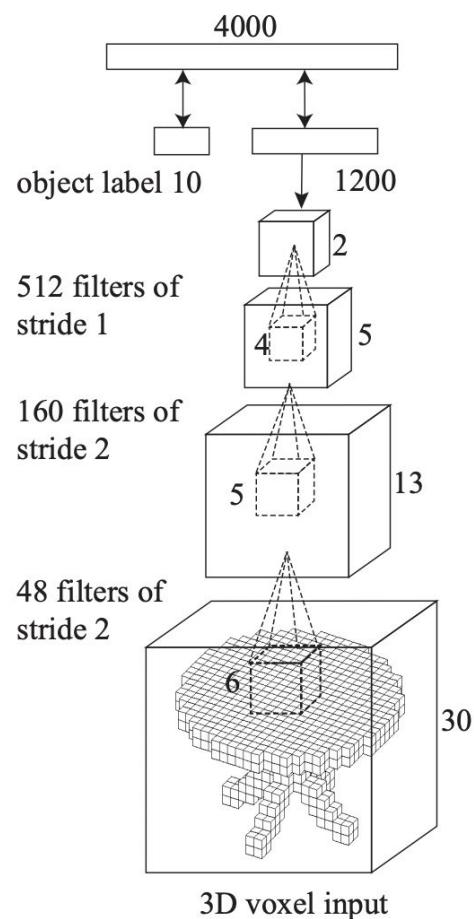
On ModelNet

Multi-View Representations

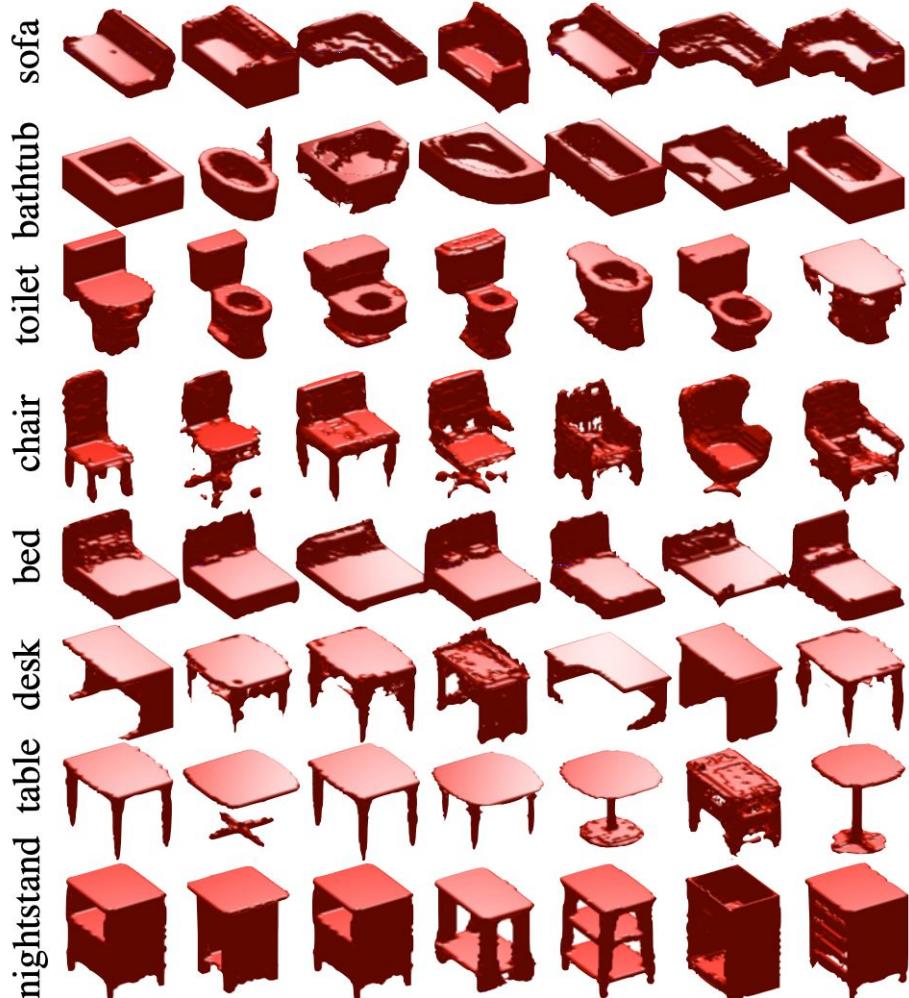
- Indeed gives good performance
- Can leverage vast literature of image classification
- Can use pertained features
- Need projection
- What if the input is noisy and/or incomplete? e.g., point cloud

Pixels -> Voxels

- 3D Conv Deep Belief Networks (CDBN)



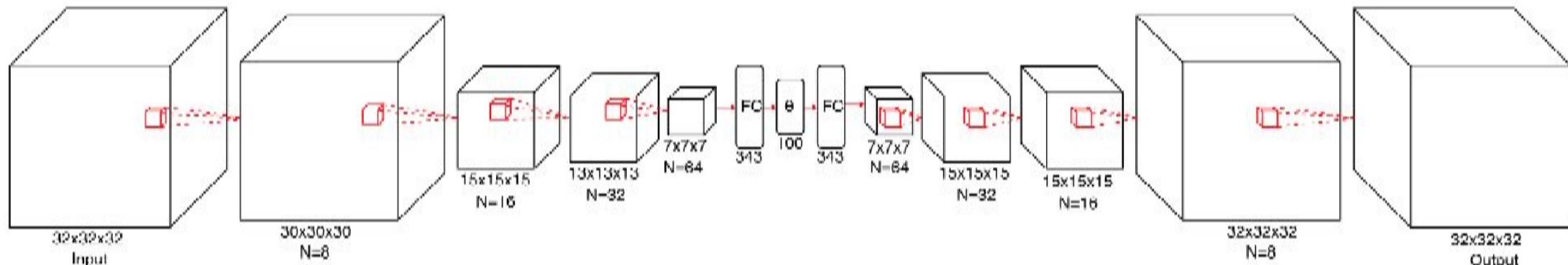
Generative Modeling



	10 classes	SPH [18]	LFD [8]	Ours
classification	79.79 %	79.87 %	83.54%	
retrieval AUC	45.97%	51.70%	69.28%	
retrieval MAP	44.05%	49.82%	68.26%	
	40 classes	SPH [18]	LFD [8]	Ours
classification	68.23%	75.47%	77.32%	
retrieval AUC	34.47%	42.04%	49.94%	
retrieval MAP	33.26%	40.91%	49.23%	

Table 1: Shape Classification and Retrieval Results.

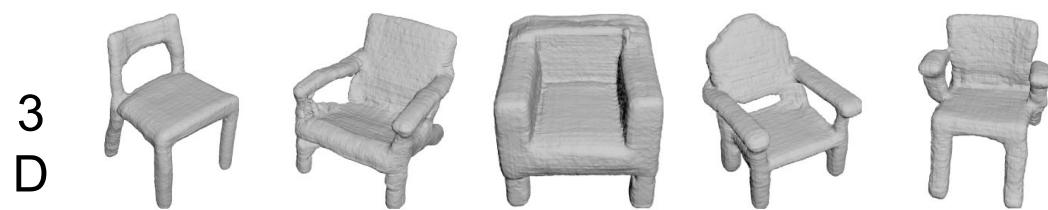
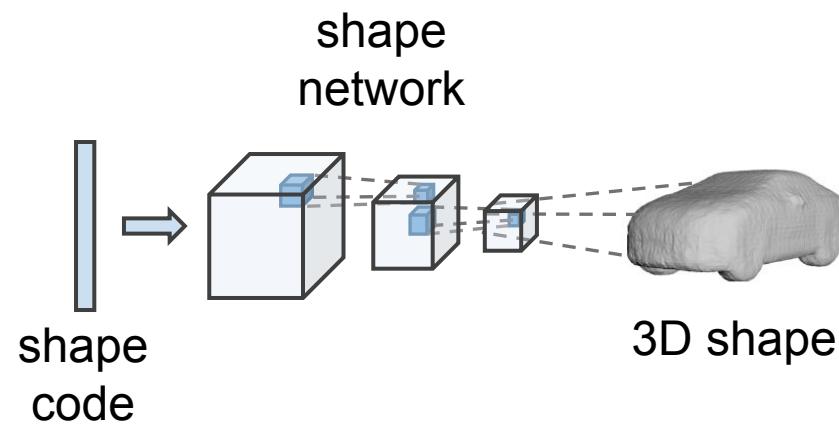
Volumetric Autoencoders



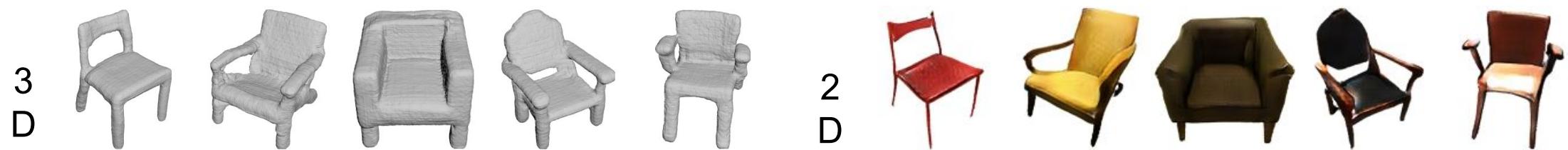
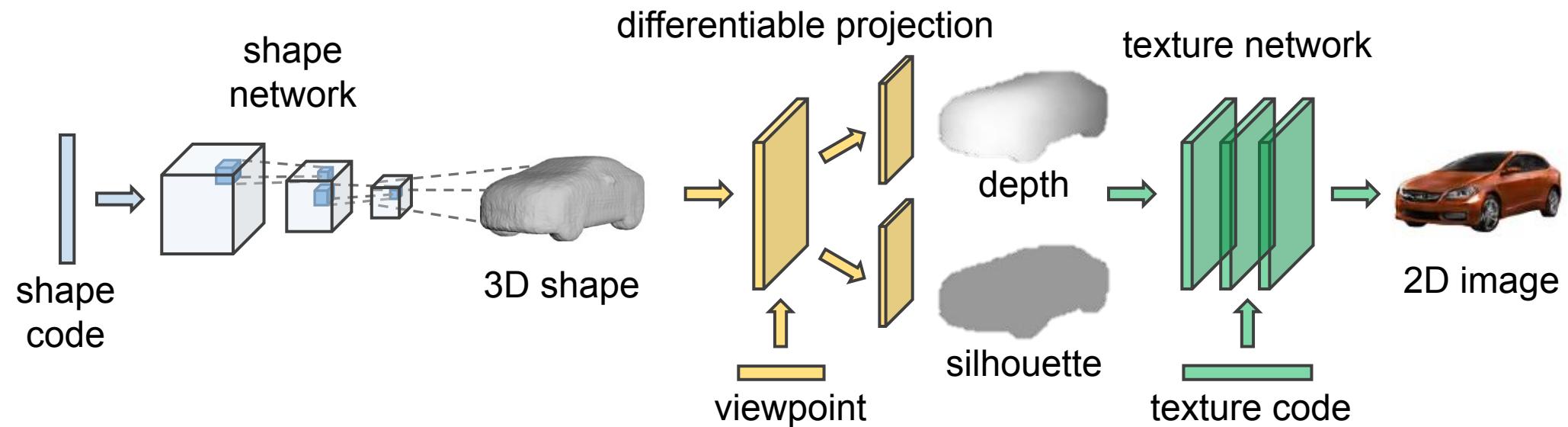
$$\text{Binary Cross-Entropy Loss: } \mathcal{L} = -l \log(o) - (1-l) \log(1-o)$$



3D-GANs



Visual Object Networks



Editing viewpoint, shape, and



Interpolation in the latent



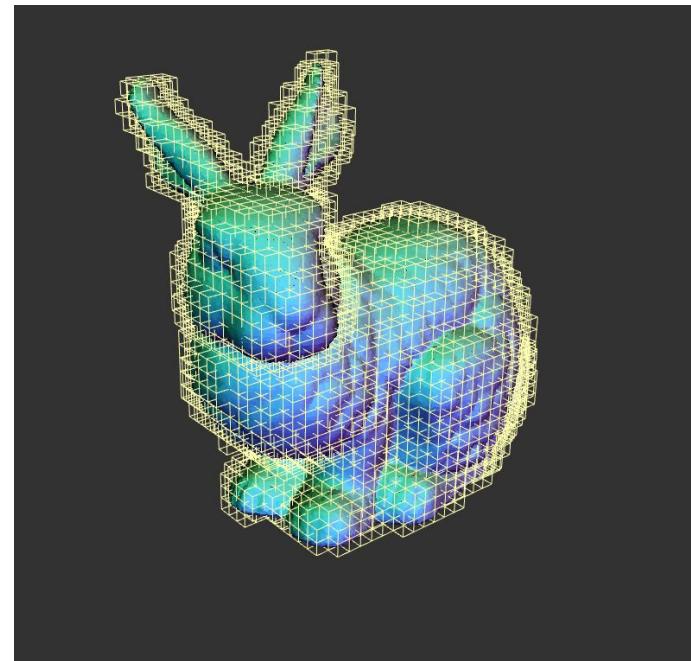
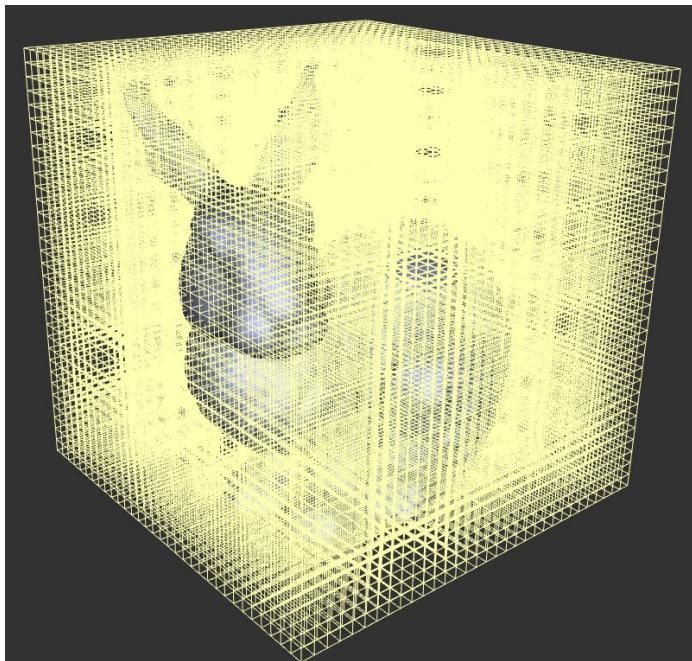
Transferring shape and
texture

shape
image



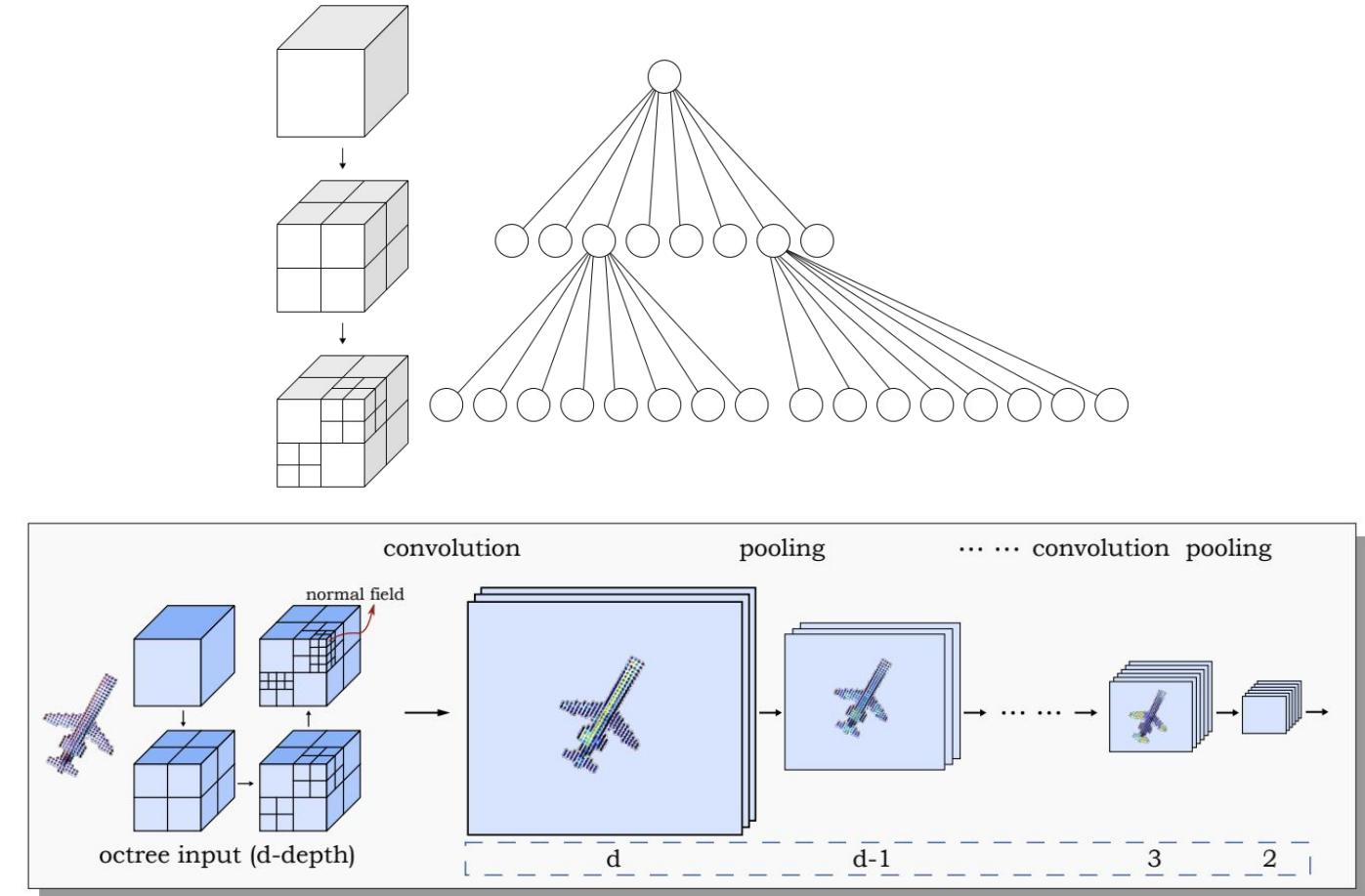
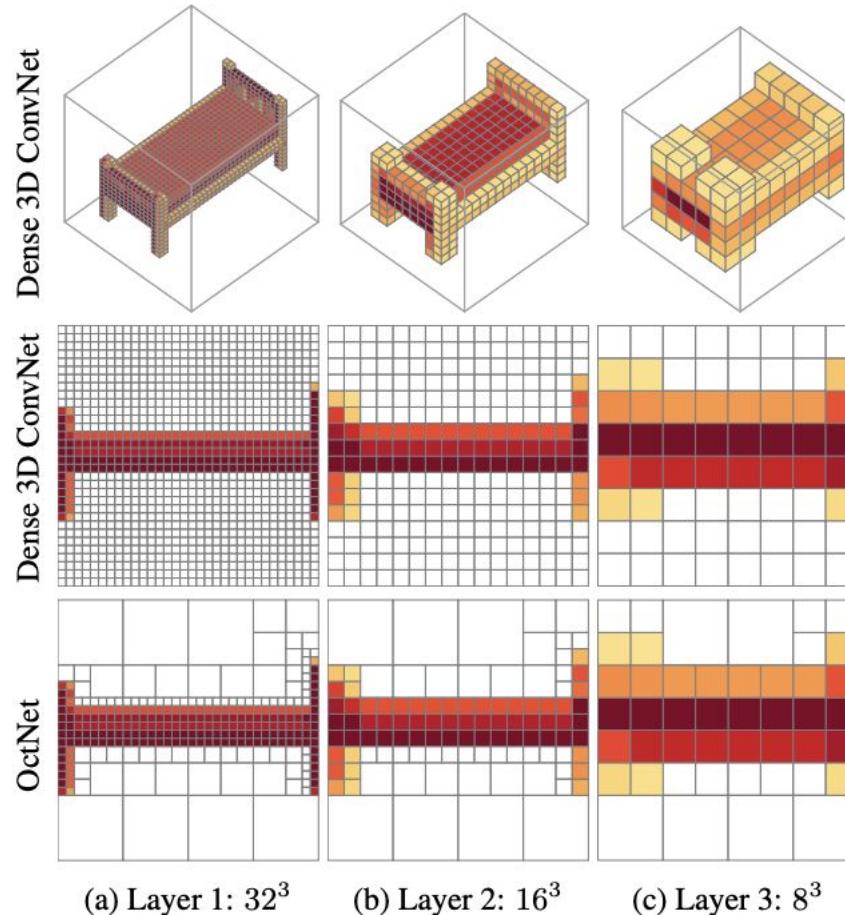
Octave Tree Representations

- Store the sparse surface signals
- Constrain the computation near the surface

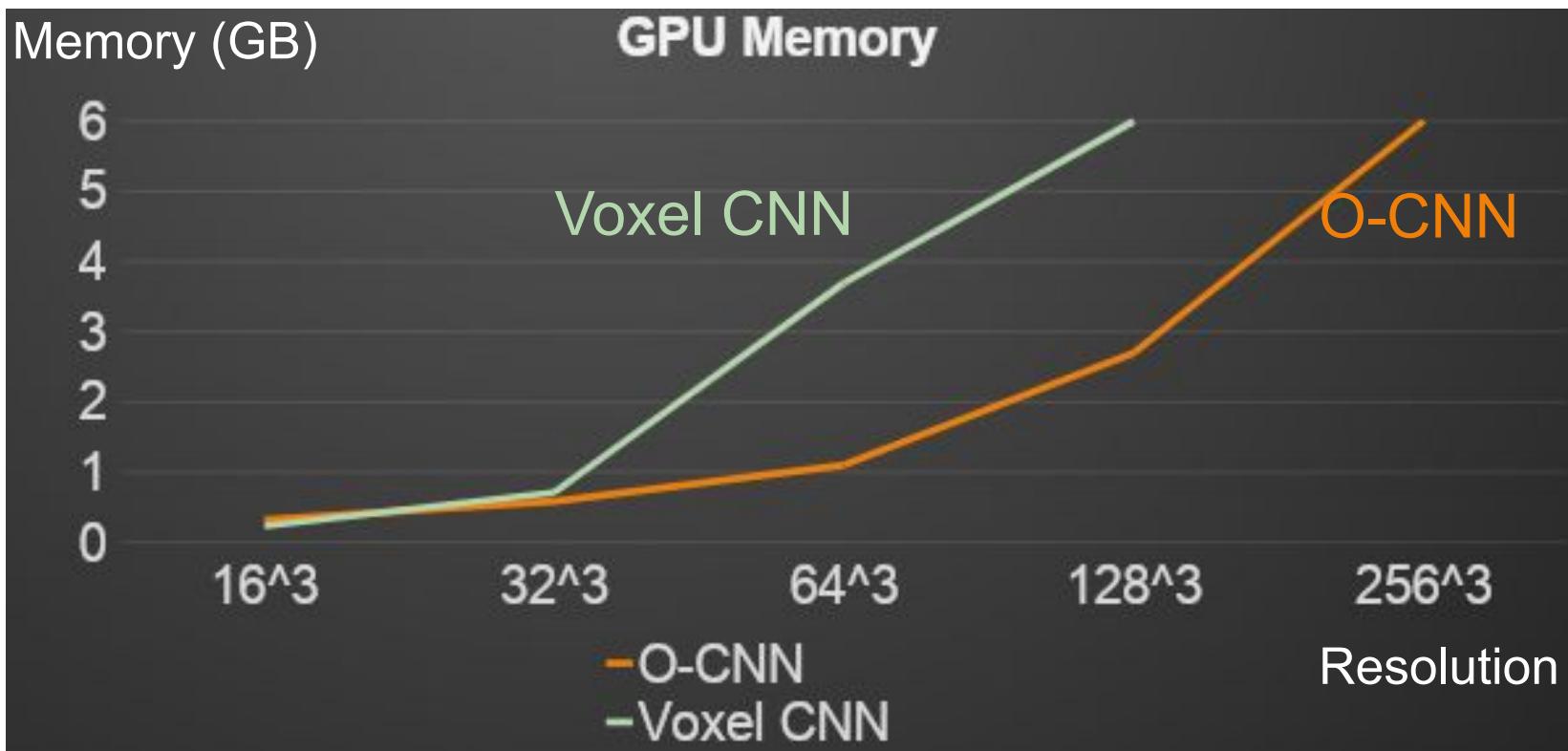


Slide Credit: Hao Su

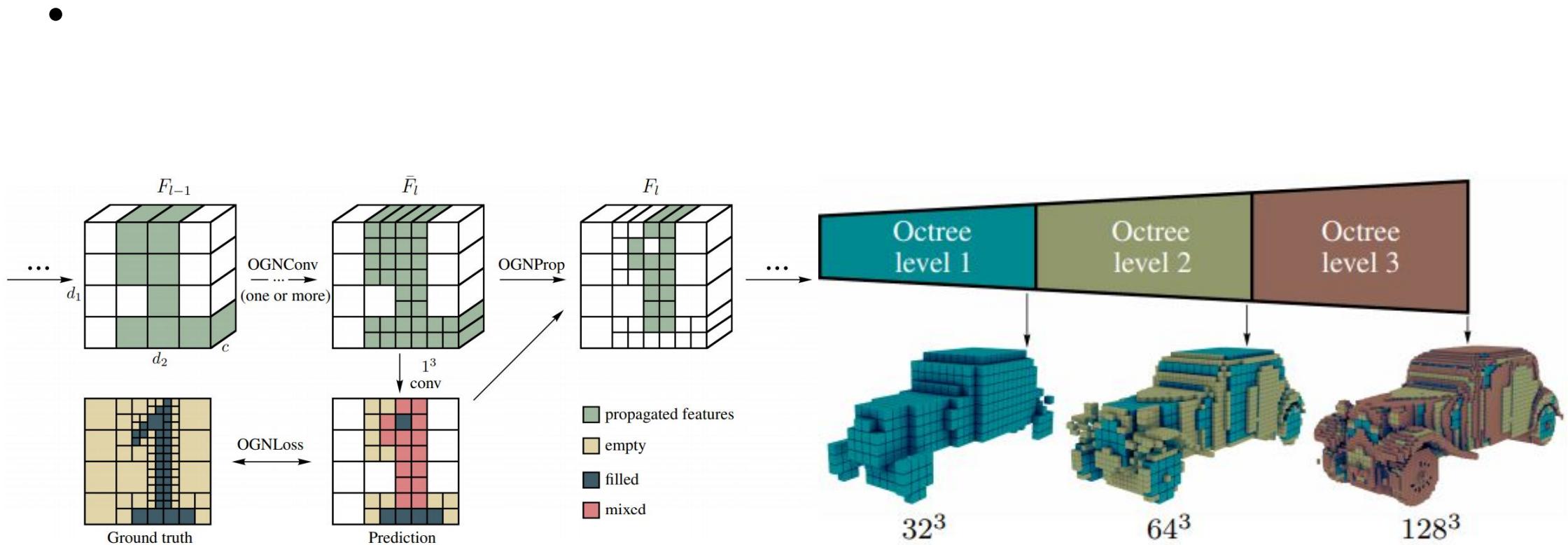
Octree: Recursively Partition the Space



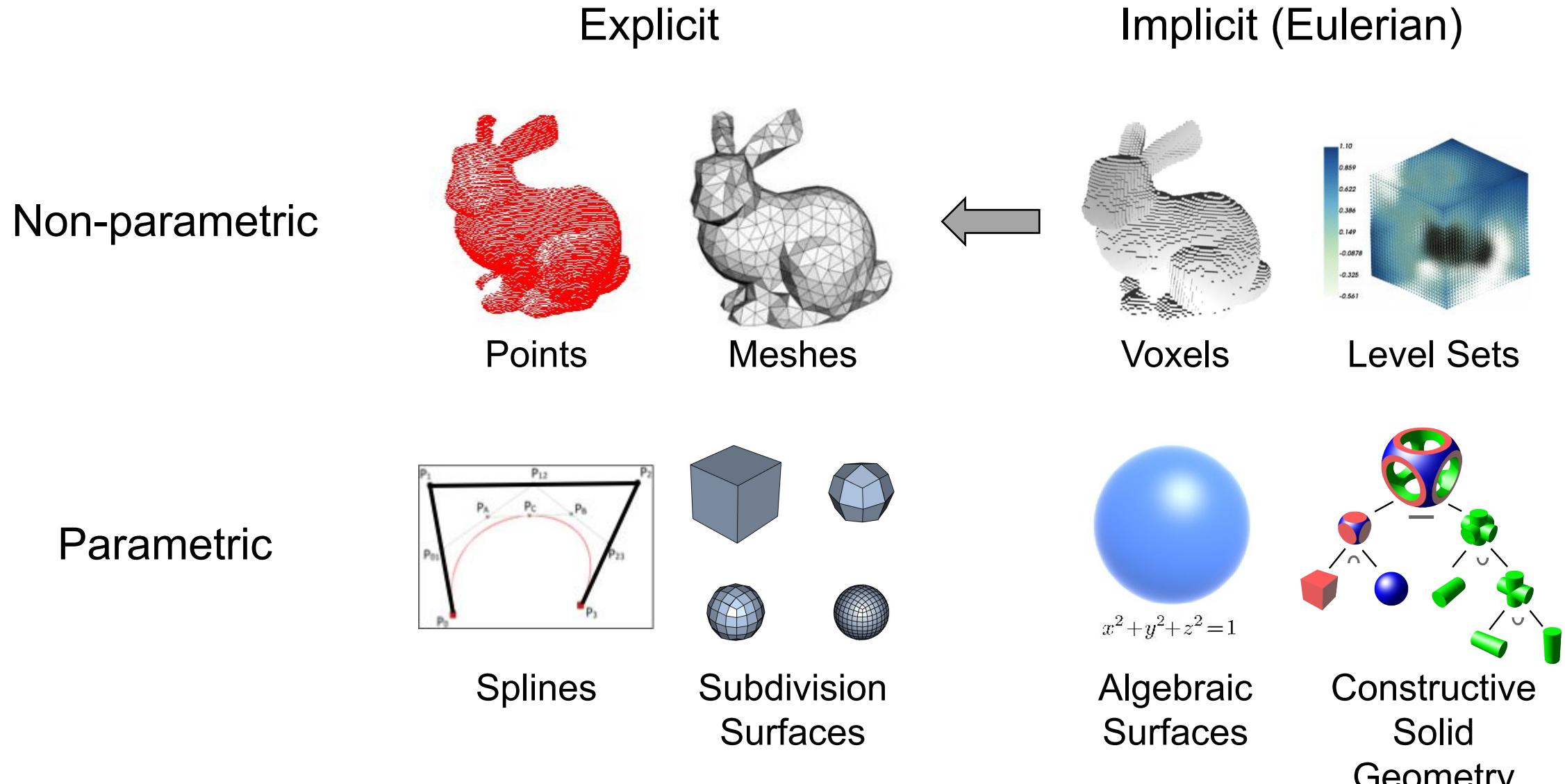
Memory Efficiency



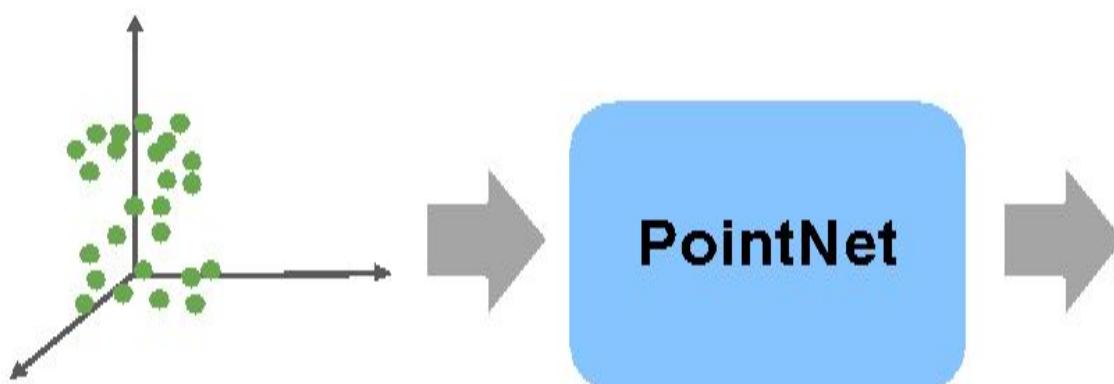
Octree Generating Networks



Eulerian \rightarrow Lagrangian

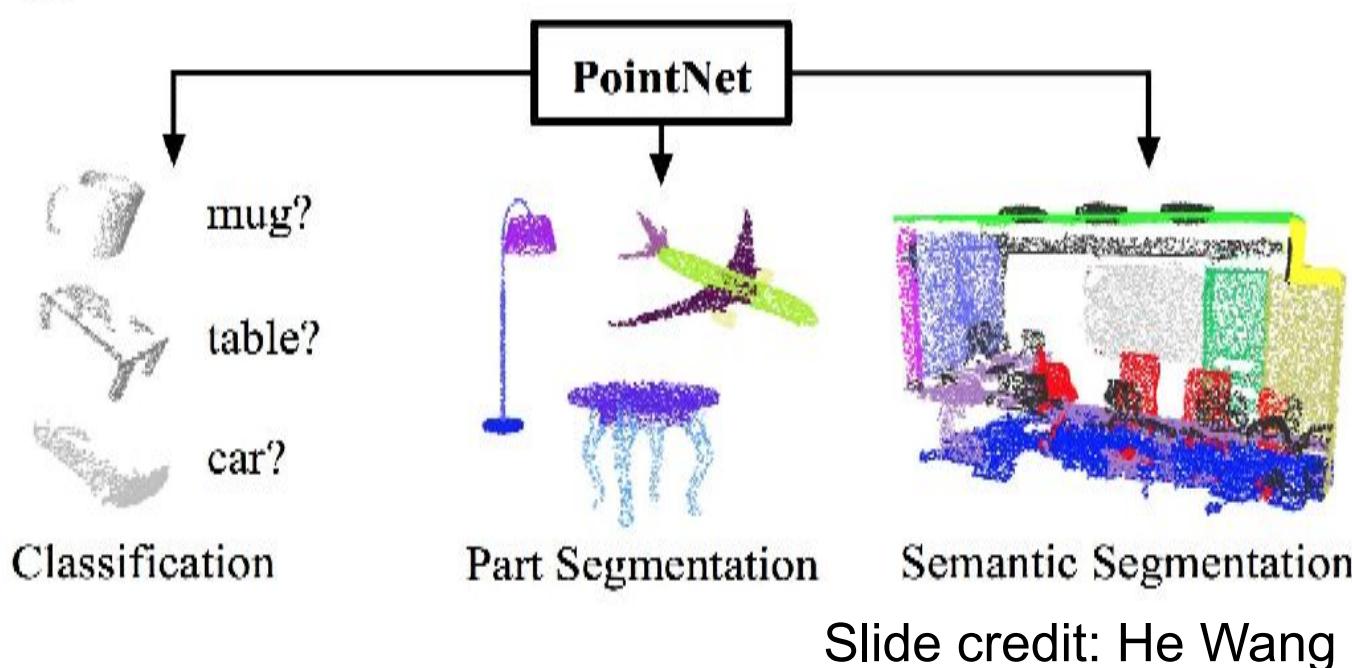


PointNet: First Learning Tool for Point Clouds



End-to-end learning for irregular point data

Unified framework for various tasks



Charles R. Qi, Hao Su, Kaichun Mo, Leonidas J. Guibas.
PointNet: Deep Learning on Point Sets for 3D
Classification and Segmentation. (CVPR '17)

Slide credit: He Wang

Invariances

The model has to respect key desiderata for point clouds:

Point Permutation Invariance

Point cloud is a set of **unordered** points

Sampling Invariance

Output a function of the underlying geometry and **not the sampling**

Permutation Invariance: Symmetric Functions

$$f(x_1, x_2, \dots, x_n) \equiv f(x_{\pi_1}, x_{\pi_2}, \dots, x_{\pi_n}), \quad x_i \in \mathbb{R}^D$$

Examples:

$$f(x_1, x_2, \dots, x_n) = \max\{x_1, x_2, \dots, x_n\}$$

$$f(x_1, x_2, \dots, x_n) = x_1 + x_2 + \dots + x_n$$

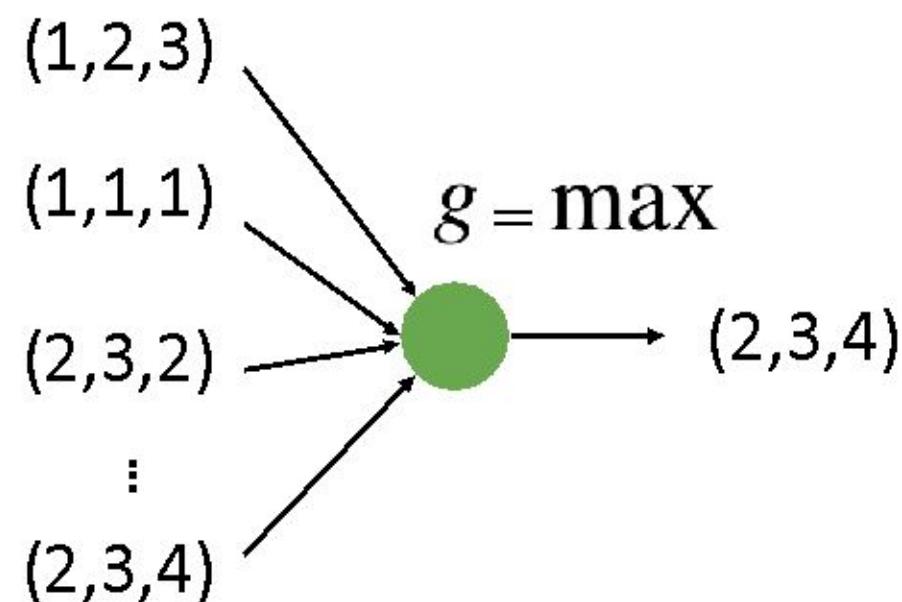
...

How can we construct a universal family of symmetric functions by neural networks?

Construct Symmetric Functions by Neural Networks

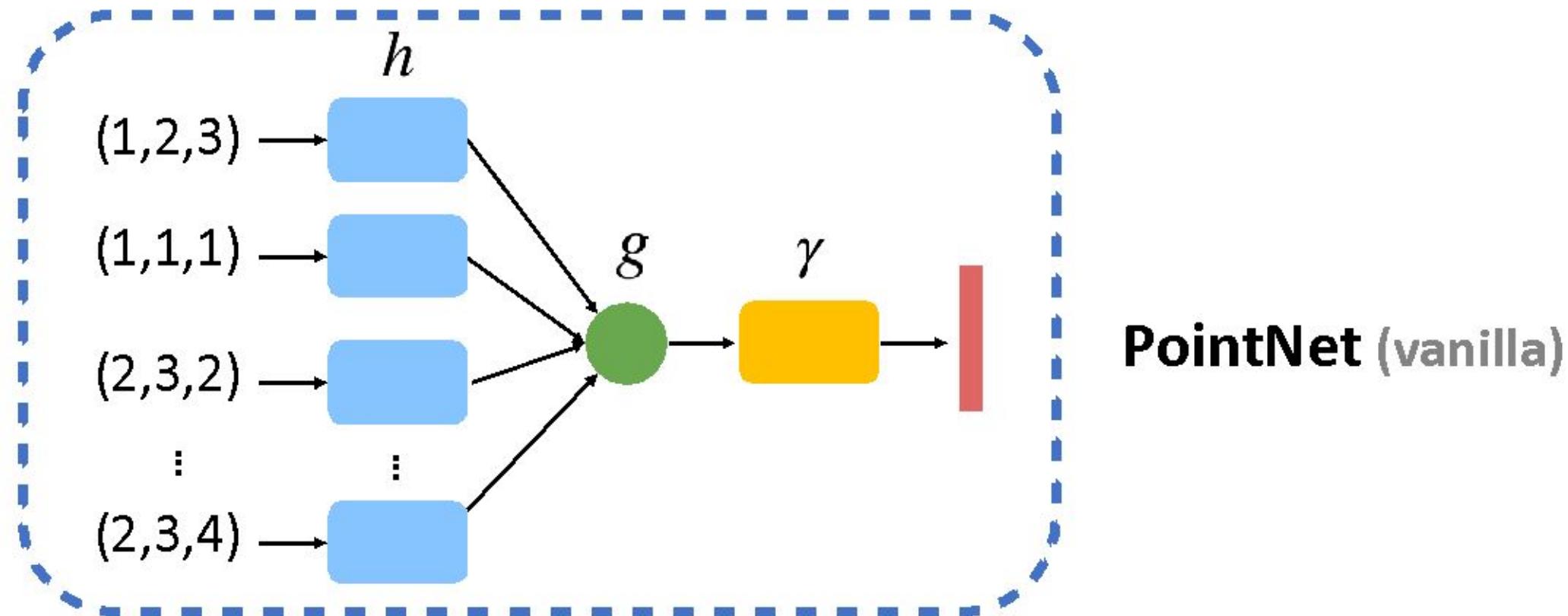
Simplest form: directly aggregate all points with a symmetric operator g

Just discovers simple extreme/aggregate properties of the geometry.



Construct Symmetric Functions by Neural Networks

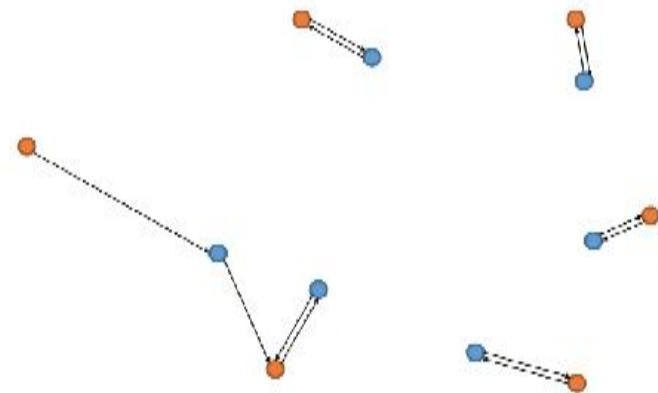
$f(x_1, x_2, \dots, x_n) = \gamma \circ g(h(x_1), \dots, h(x_n))$ is symmetric if g is symmetric



Distance Metrics for Point Cloud

Chamfer distance We define the Chamfer distance between $S_1, S_2 \subseteq \mathbb{R}^3$ as:

$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2$$



Distance Metrics for Point Cloud

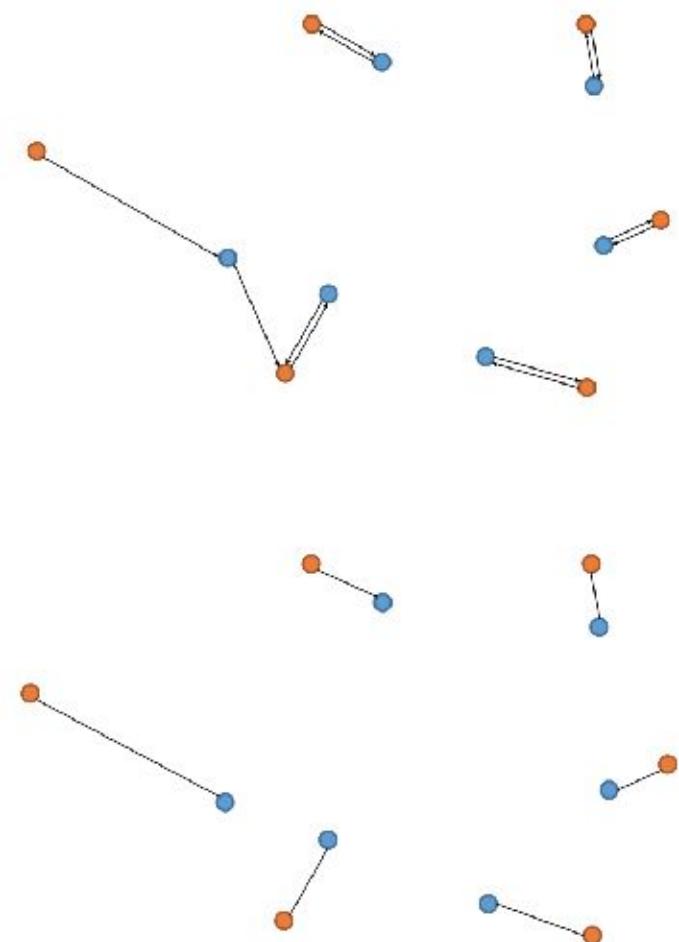
Chamfer distance We define the Chamfer distance between $S_1, S_2 \subseteq \mathbb{R}^3$ as:

$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2$$

Earth Mover's distance Consider $S_1, S_2 \subseteq \mathbb{R}^3$ of equal size $s = |S_1| = |S_2|$. The EMD between A and B is defined as:

$$d_{EMD}(S_1, S_2) = \min_{\phi: S_1 \rightarrow S_2} \sum_{x \in S_1} \|x - \phi(x)\|_2$$

where $\phi : S_1 \rightarrow S_2$ is a bijection.



Distance Metrics for Point Cloud

Chamfer distance We define the Chamfer distance between $S_1, S_2 \subseteq \mathbb{R}^3$ as:

$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2$$

Earth Mover's distance Consider $S_1, S_2 \subseteq \mathbb{R}^3$ of equal size $s = |S_1| = |S_2|$. The EMD between A and B is defined as:

$$d_{EMD}(S_1, S_2) = \min_{\phi: S_1 \rightarrow S_2} \sum_{x \in S_1} \|x - \phi(x)\|_2$$

where $\phi : S_1 \rightarrow S_2$ is a bijection.

Sum of closest distances

Inensitive to sampling
(only relatively)

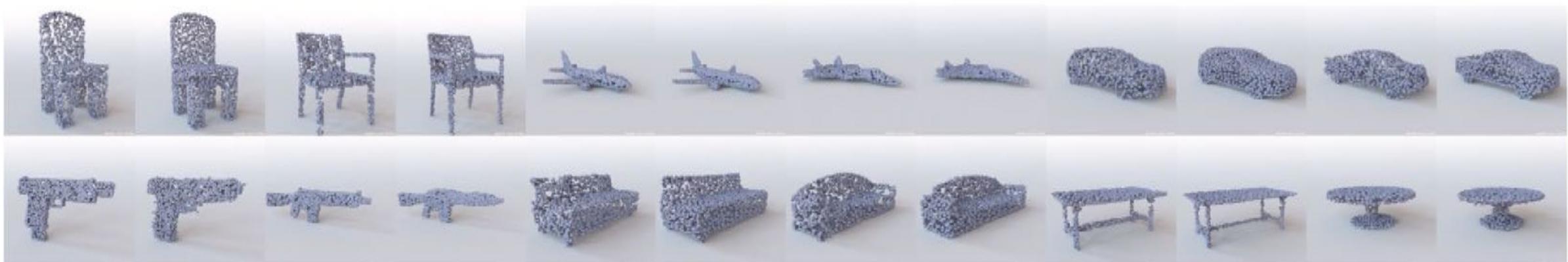
Sum of matched closest distances

Sensitive to sampling

Point Cloud AE

Encoder: PointNet

Decoder: MLP

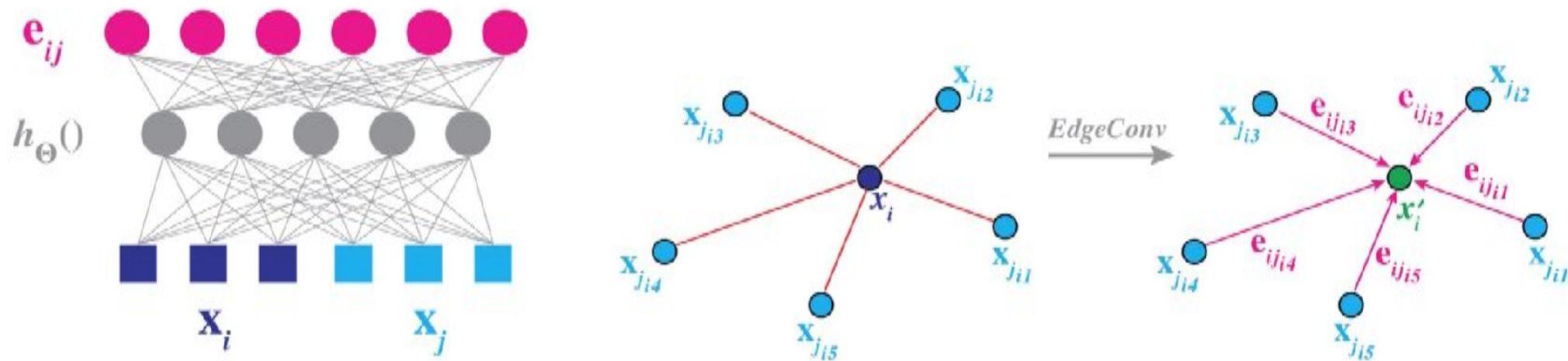


ICML 2018, Learning Representations and Generative Models for
3D Point Clouds, Panos Achlioptas, et. al.

Slide credit: He Wang

Graph NNs on Point Clouds

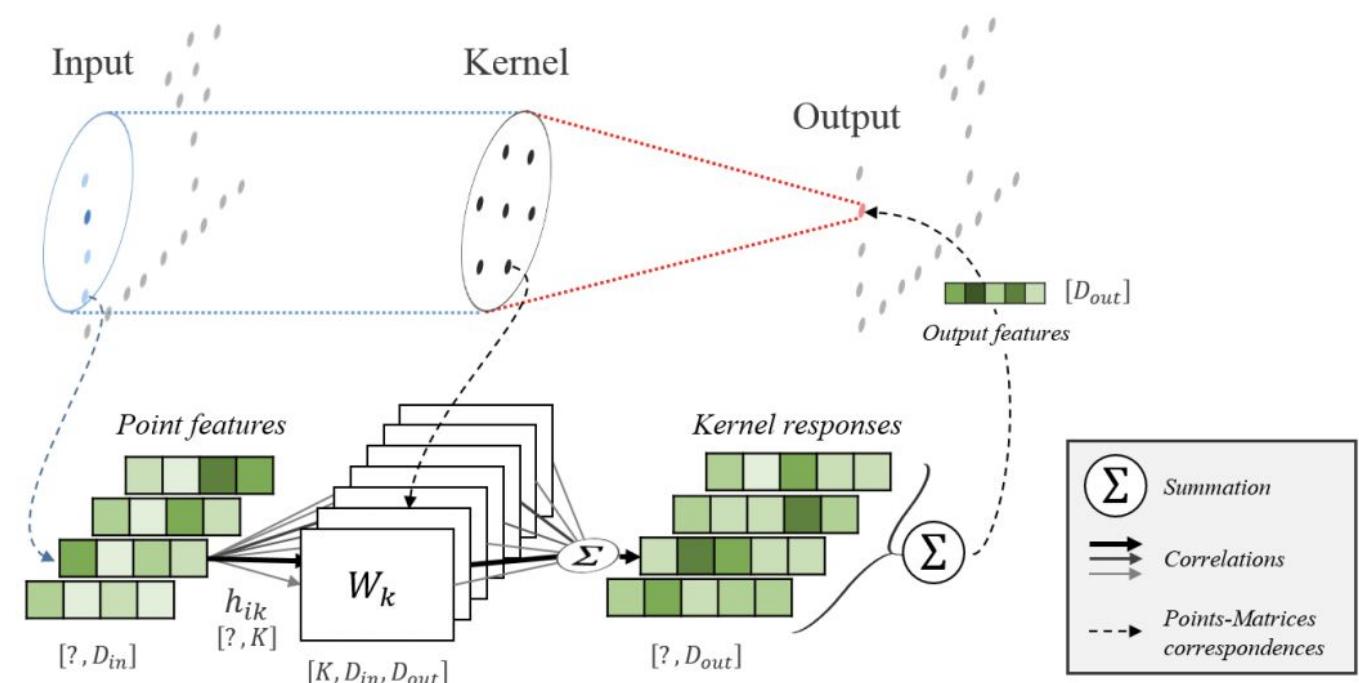
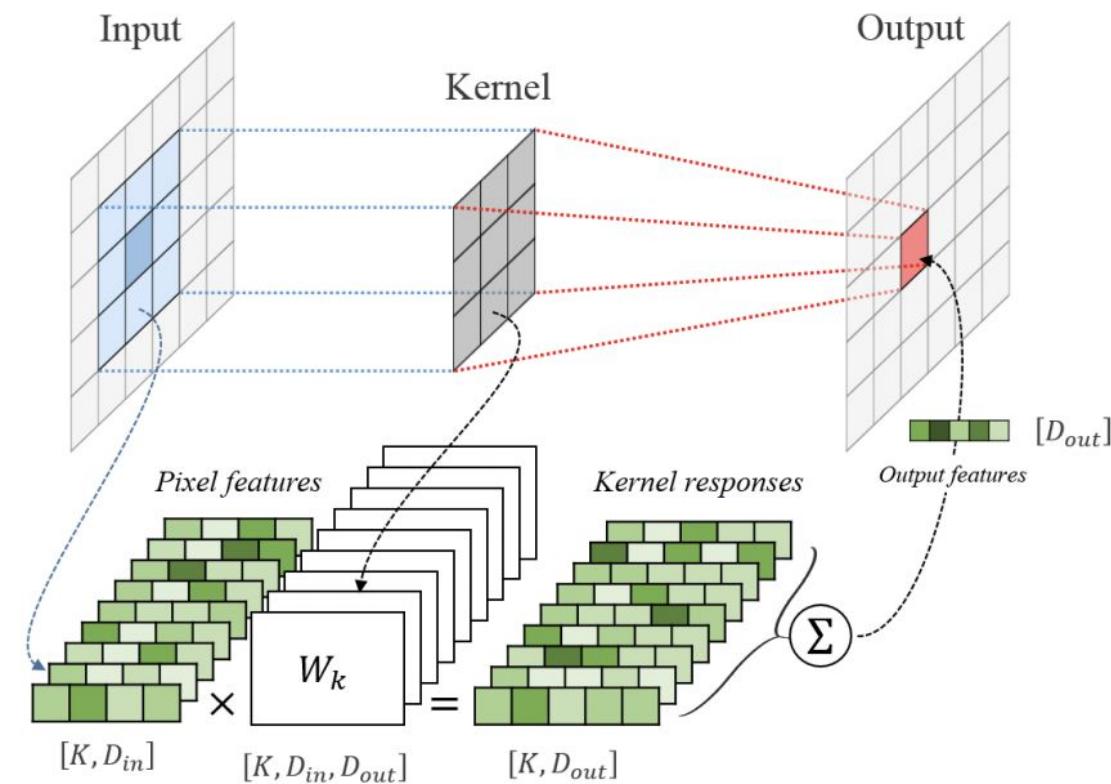
- Points -> Nodes
- Neighborhood -> Edges
- Graph NNs for point cloud processing



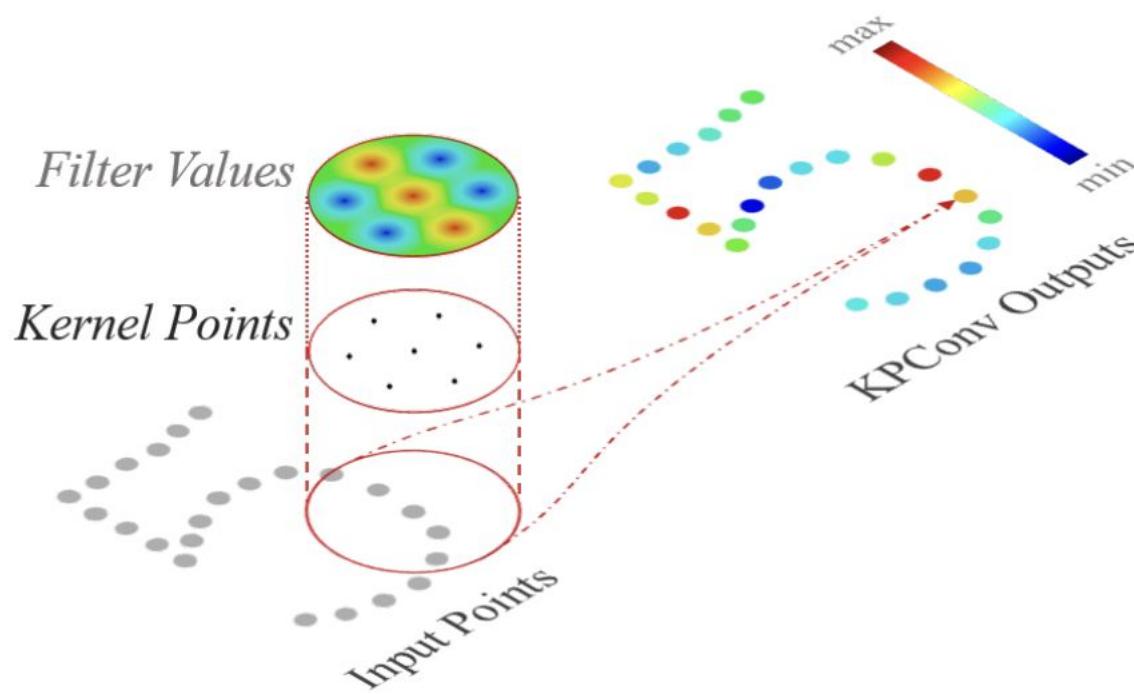
Message-Passing GNNs are not Geometry-Aware

- Points are **sampled** from surfaces.
- Ideally, features describe the geometry of underlying surface. They should be sample invariant.
- Message-passing GNNs do not address sample invariance.
- **Solution:** Estimate the continuous kernel and point density for continuous convolution (KPConv)

Kernel Point Convolution (KPConv)

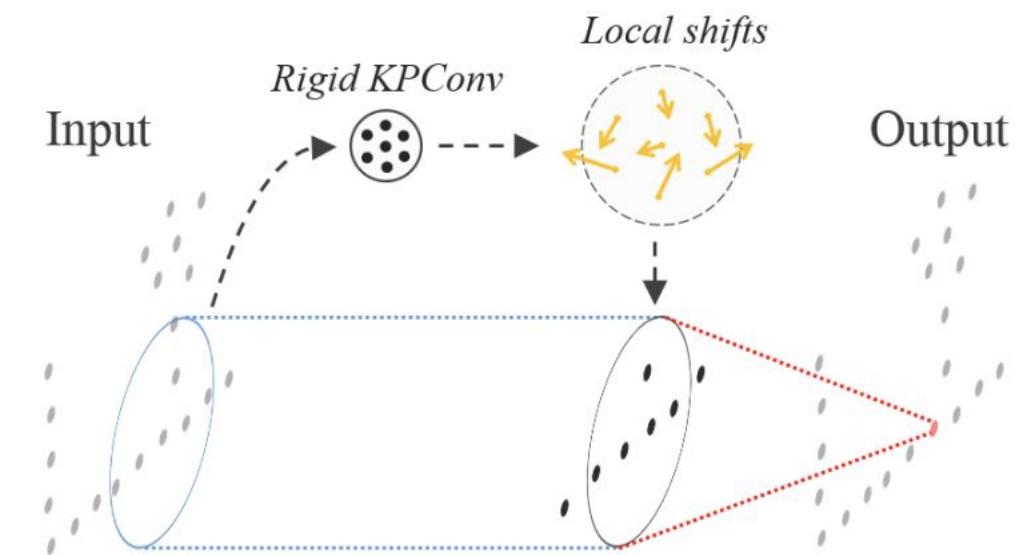


Deformable Kernel for Deformable Objects



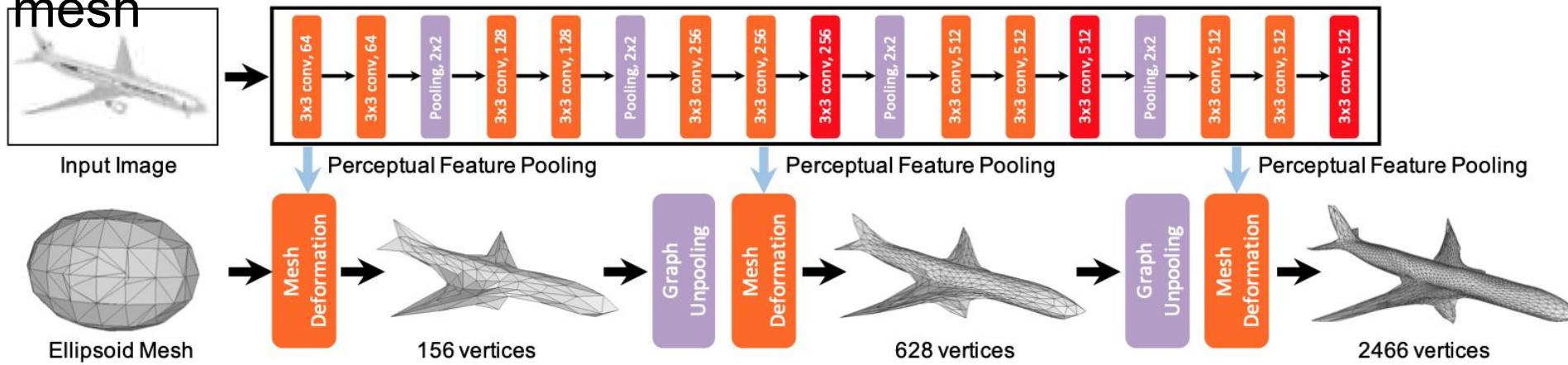
Deformable point-based kernel

- 3D version of 2D deformable convolution

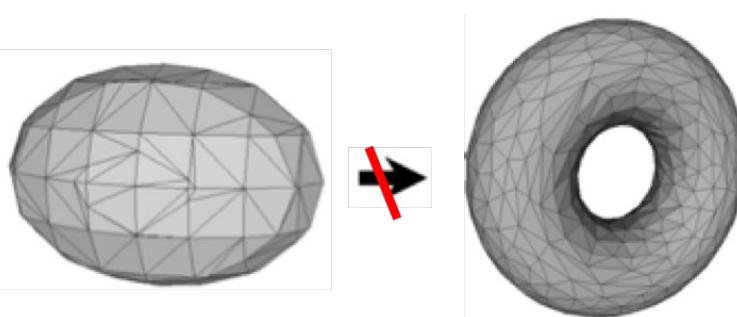


Pixel2Mesh

Learn to deform a template mesh

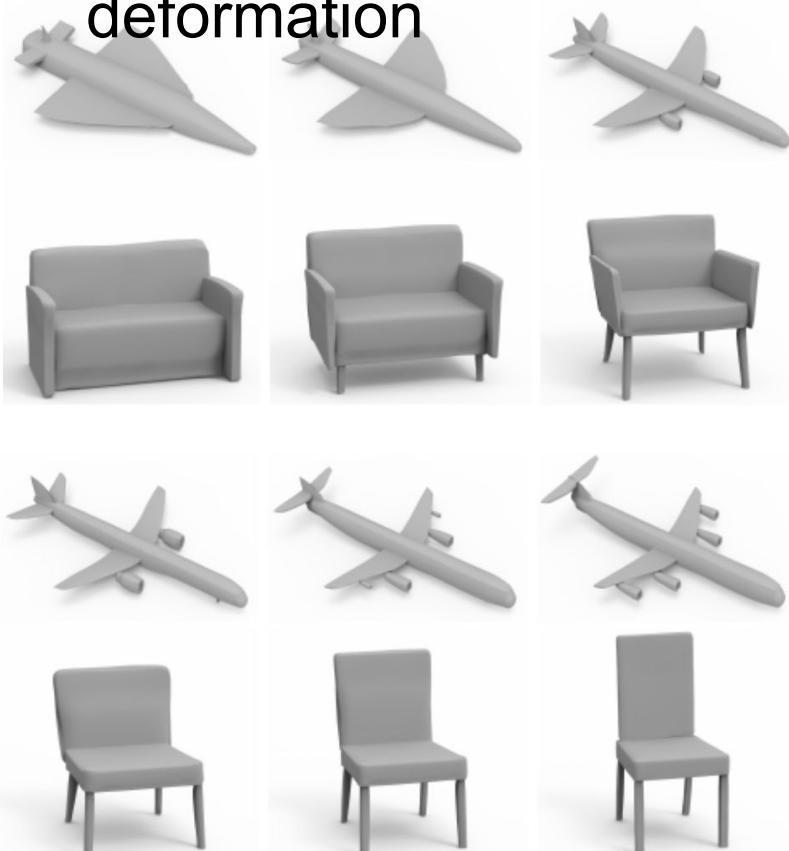


Cannot change the topology of the template mesh

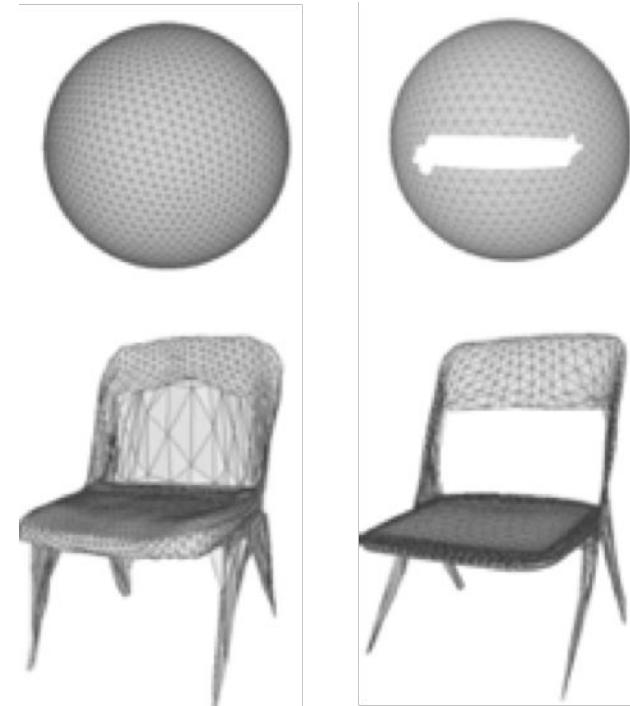


More on Mesh Deformation

Part-level
deformation



Modify the topology of
the template mesh

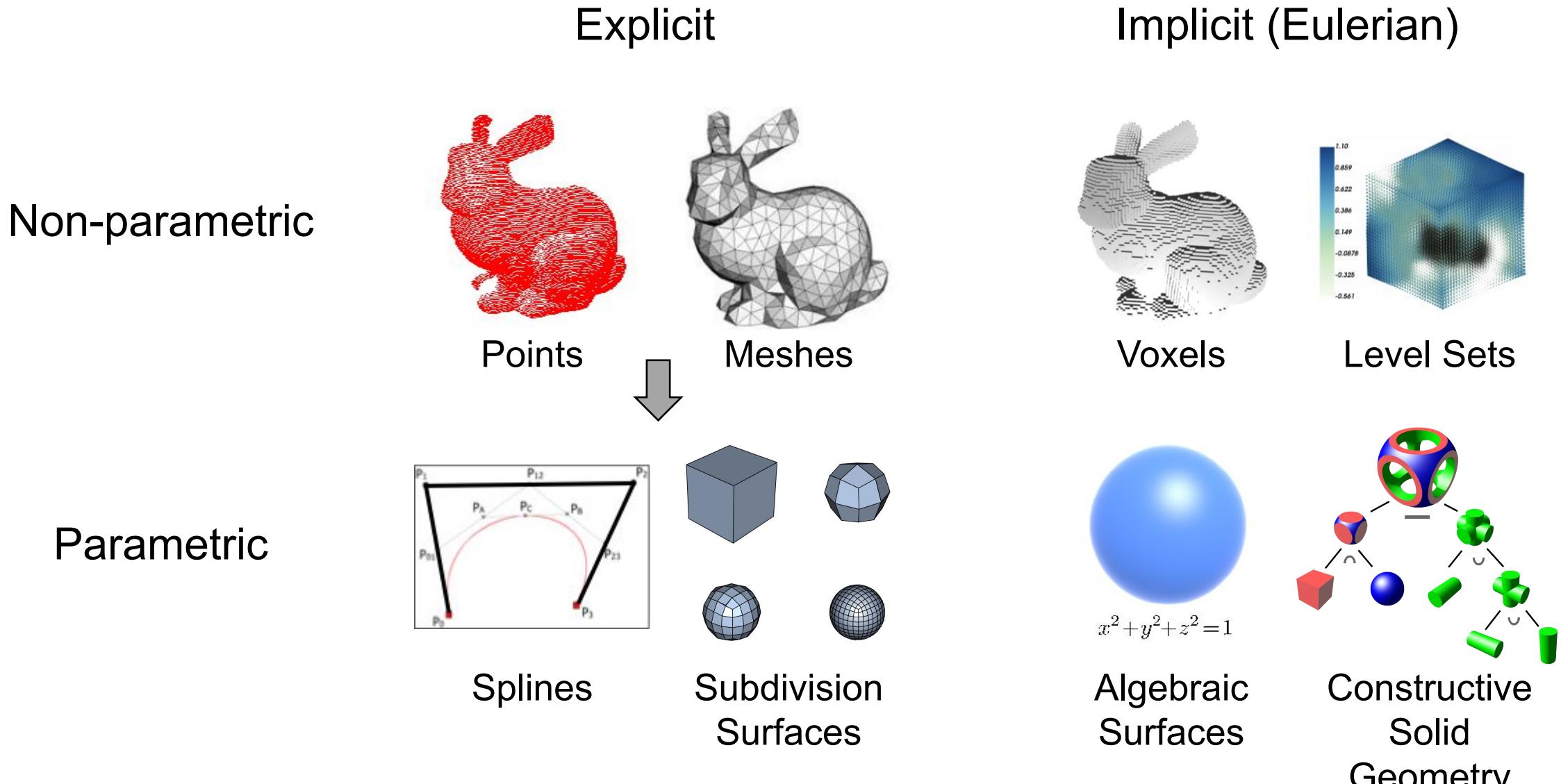


Gao et al. SDM-NET: Deep generative network for structured deformable mesh. SIGGRAPH Asia 2019

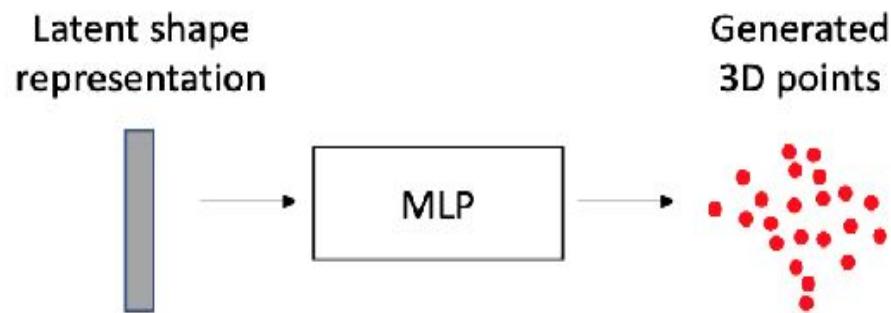
Pan et al., Deep Mesh Reconstruction from Single RGB Images via Topology Modification Networks. ICCV 2019

Slide: Hao Su

Non-Parametric → Parametric

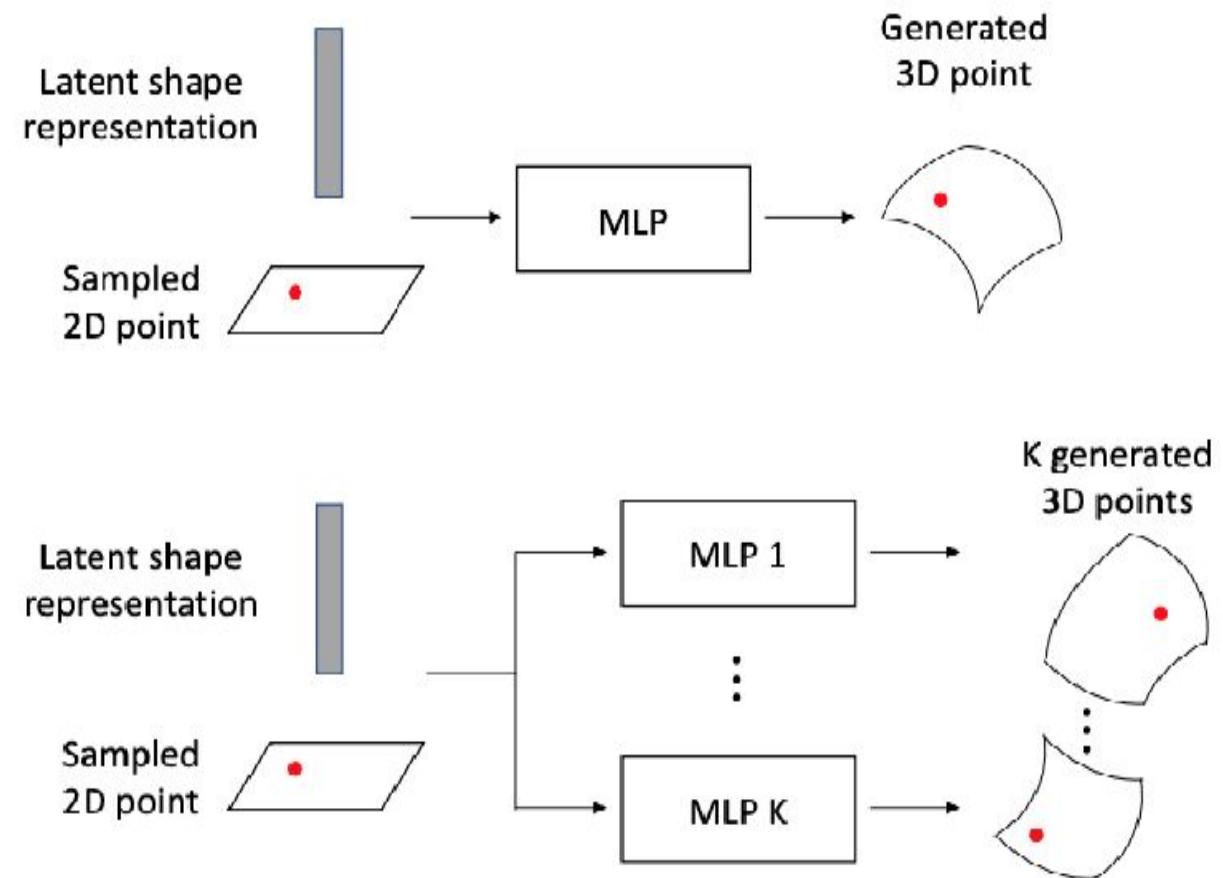


Parametric Decoder: AtlasNet



Given the output points form a smooth surface,
enforce such a parametrization as input.

$$\text{MLP}(z, u, v) \rightarrow \text{point}$$

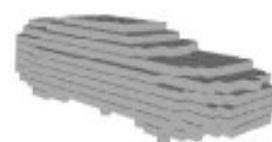
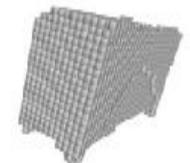
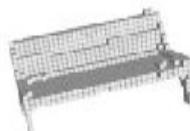


Results

Input image



Voxel



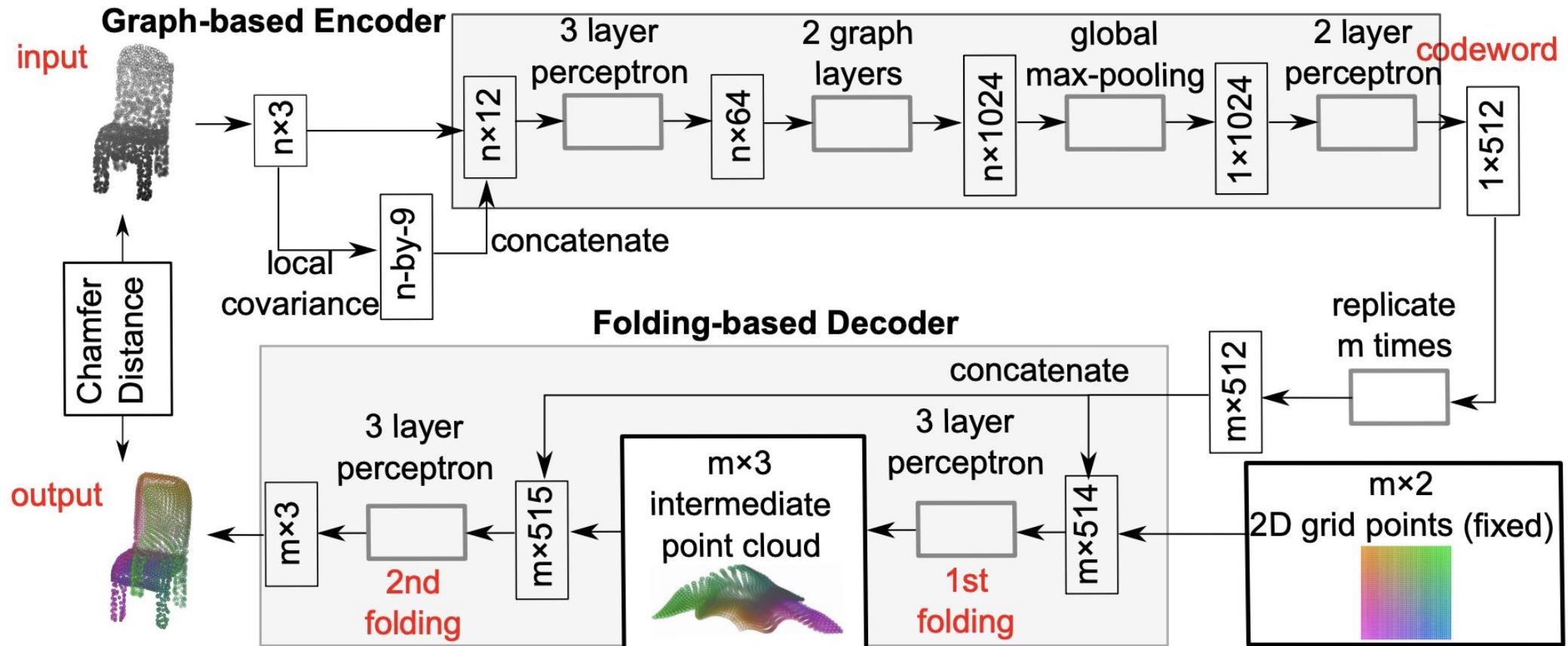
Point cloud



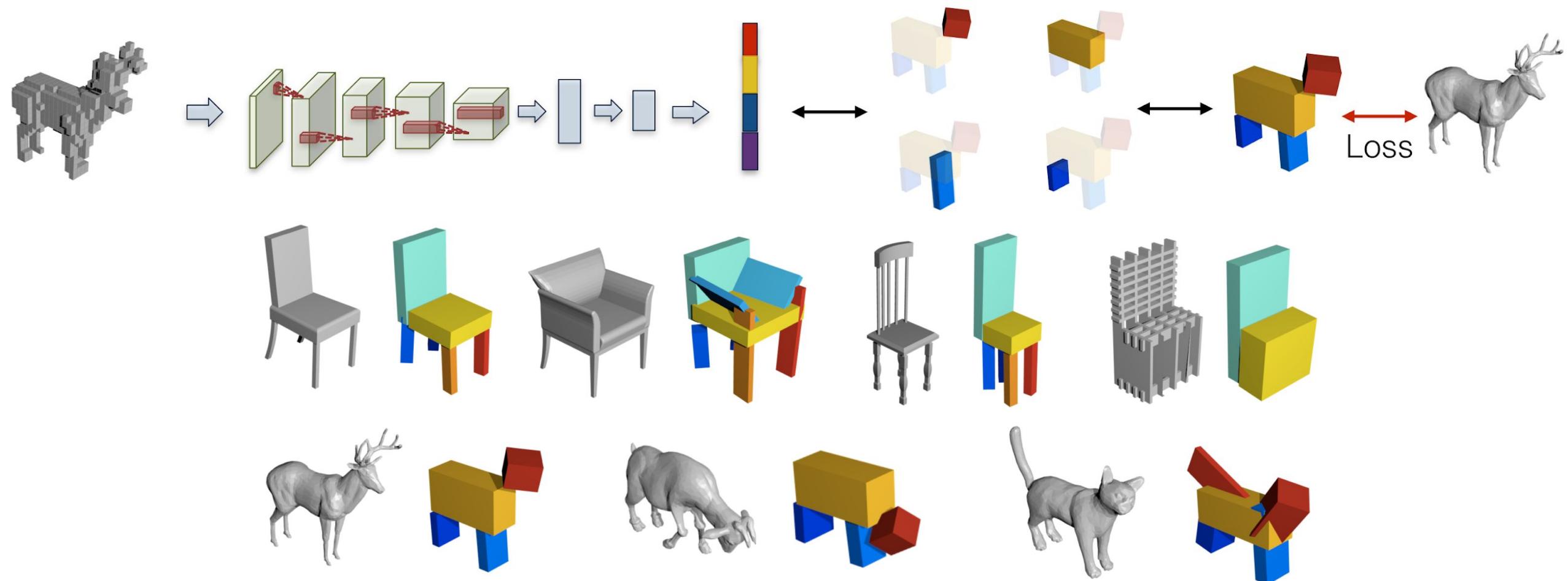
AtlasNet



FoldingNet

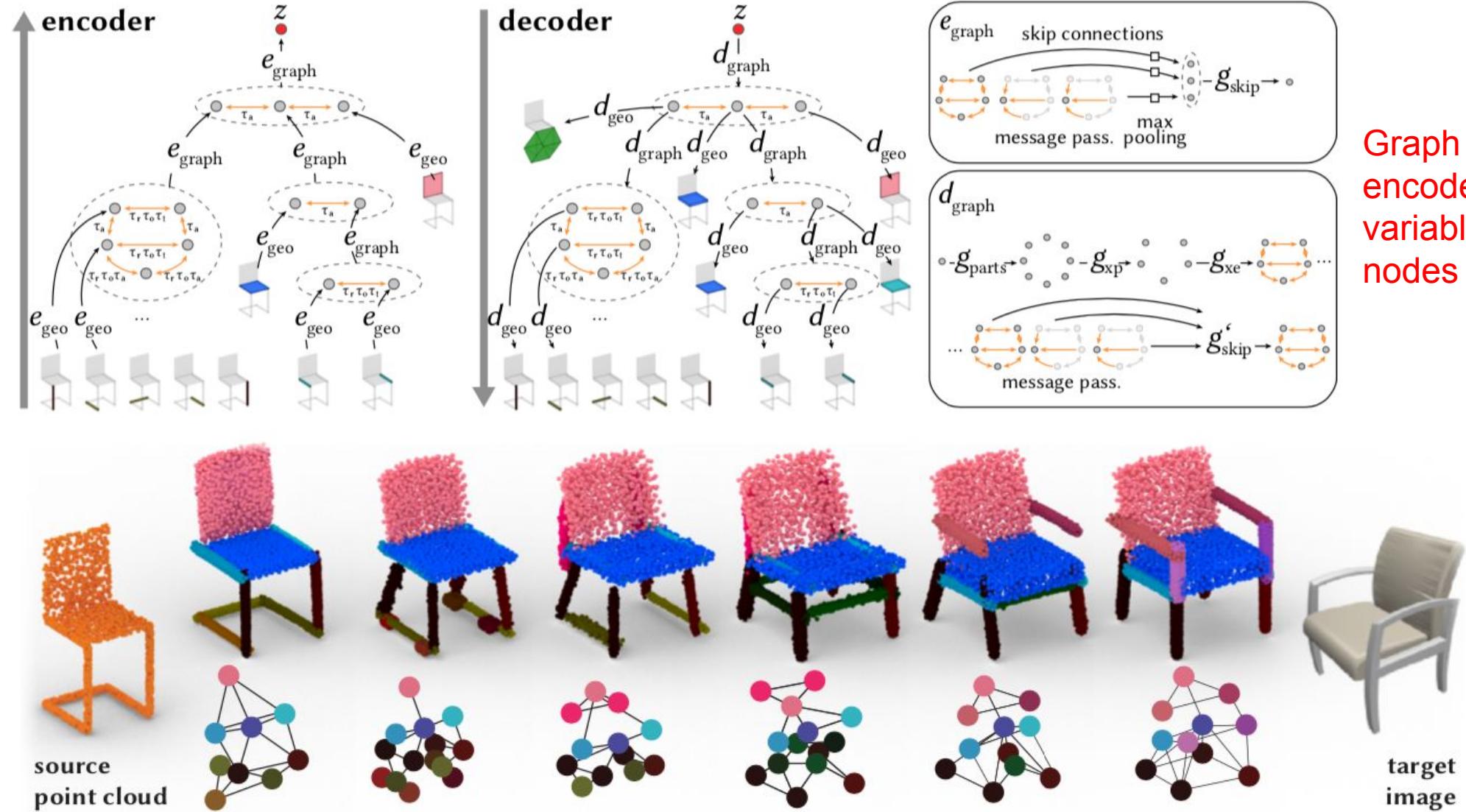


Assembling Volumetric Primitives

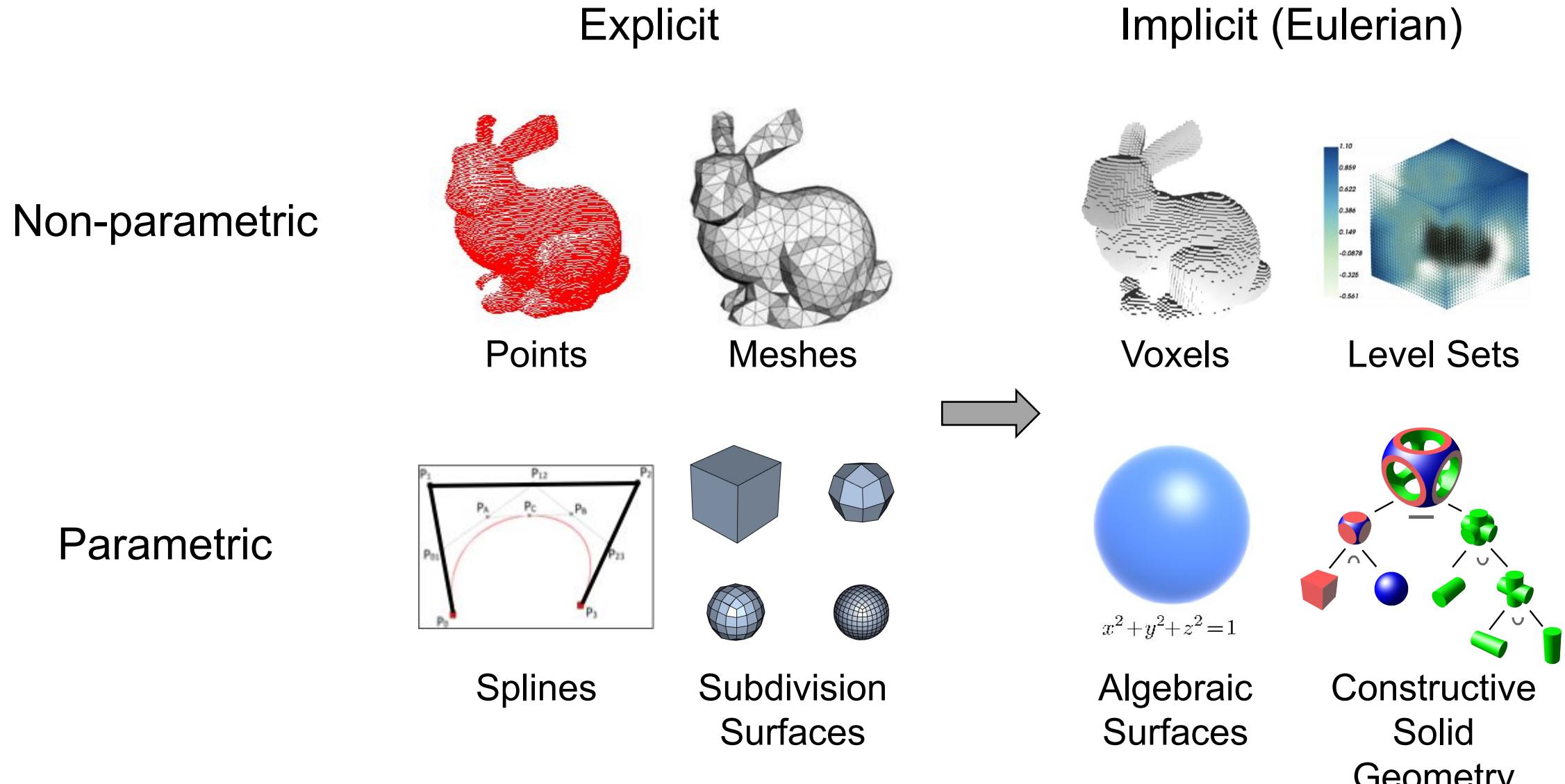


Incorporating Shape Structure

StructureNet



Explicit \rightarrow Implicit

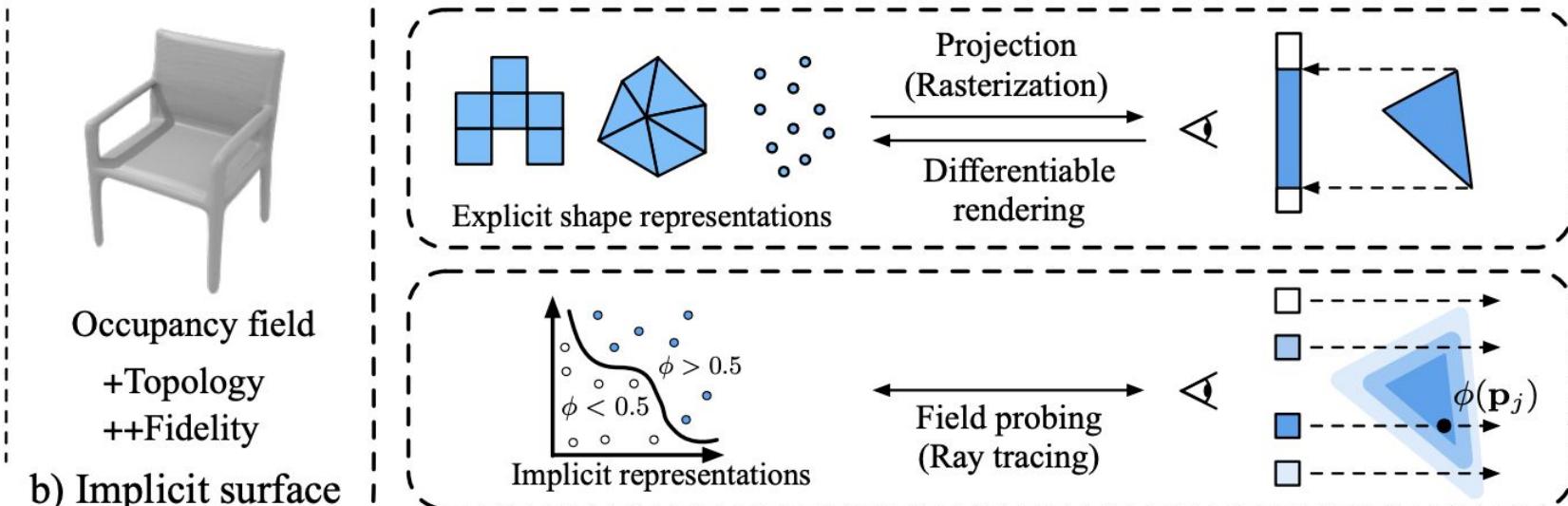


Deep Implicit Functions



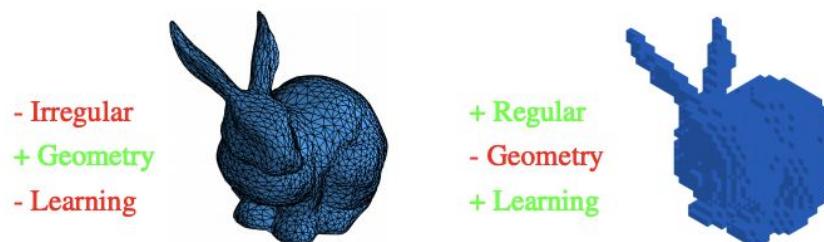
Voxel
+Topology
-Fidelity
Point cloud
+Topology
-Fidelity
Mesh
-Topology
+Fidelity

a) Explicit representation



b) Implicit surface

Liu et al. Learning to Infer Implicit Surfaces without 3D Supervision. NeurIPS



(a) Explicit representations

(b) Voxels

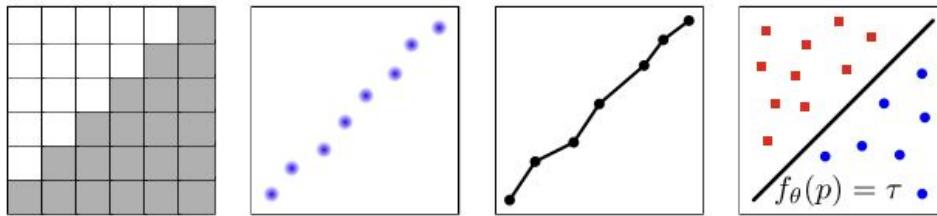
(c) Point cloud

(d) Level set

Figure 2. Four common representations of 3D shape along with their advantages and disadvantages.

Deep Level Sets: Implicit Surface Representations for 3D Shape Inference.
2019

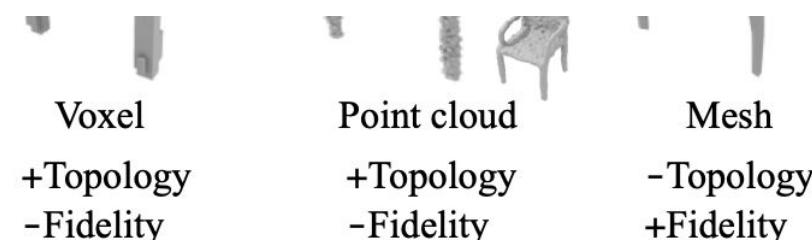
DeepSDF. CVPR
2019



Occupancy Networks
CVPR 2019



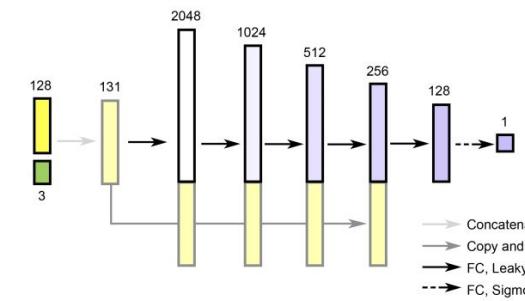
(a) Voxel (b) Point (c) Mesh (d) Ours



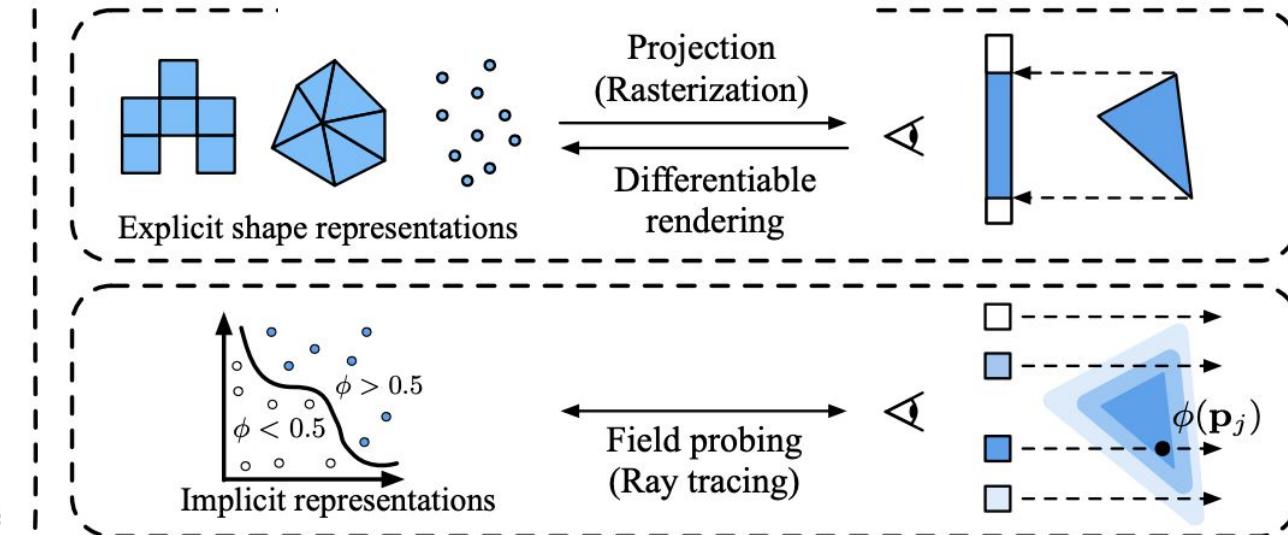
a) Explicit representation

Occupancy field
+Topology
++Fidelity

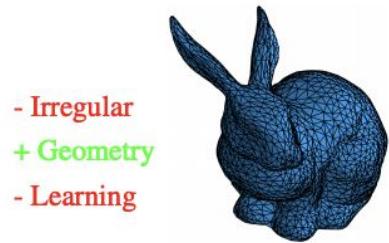
b) Implicit surface



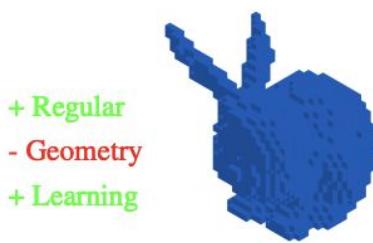
Chen and Zhang.
Learning Implicit
Fields
CVPR 2019



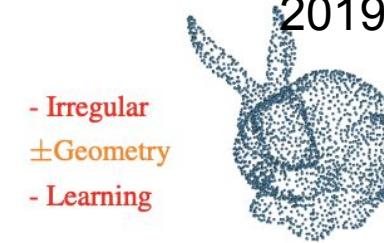
Liu et al. Learning to Infer Implicit Surfaces without 3D Supervision. NeurIPS



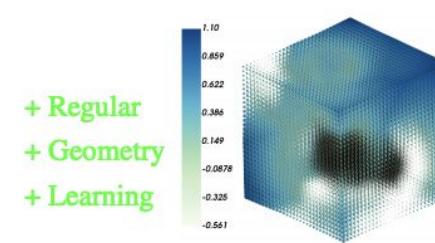
(a) Explicit representations



(b) Voxels



(c) Point cloud



(d) Level set

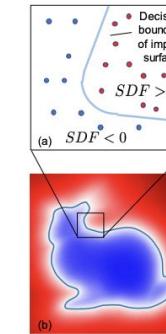


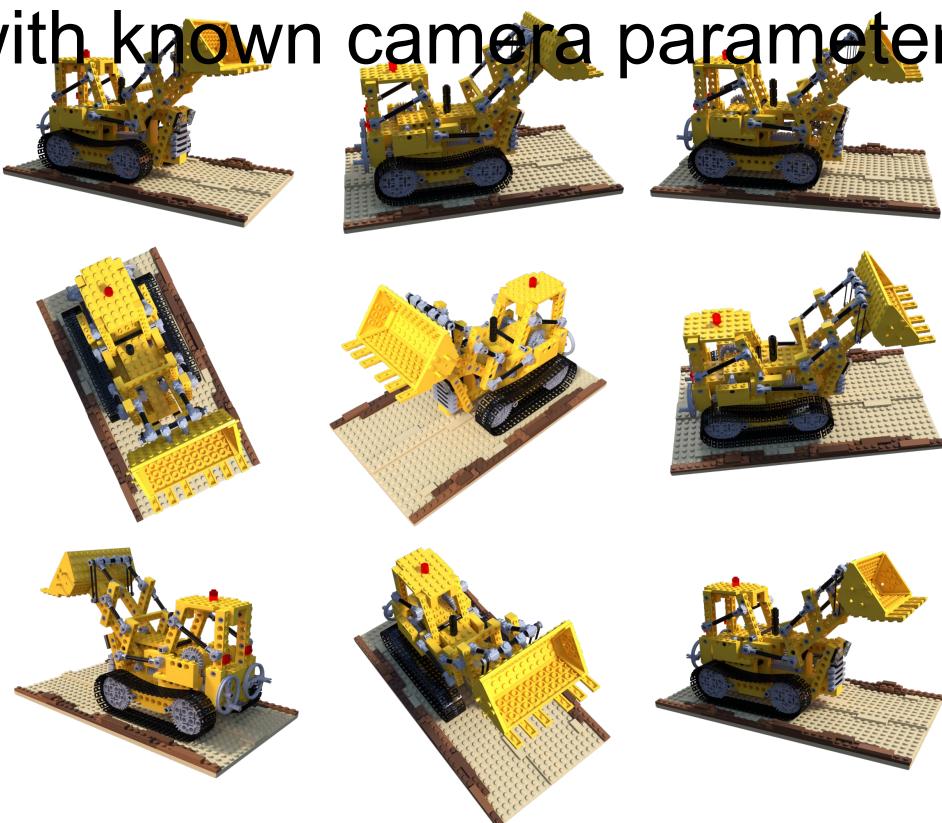
Figure 2. Four common representations of 3D shape along with their advantages and disadvantages.

Deep Level Sets: Implicit Surface Representations for 3D Shape Inference.
2019

DeepSDF. CVPR
2019

Neural Radiance Fields (NeRF) for View Synthesis

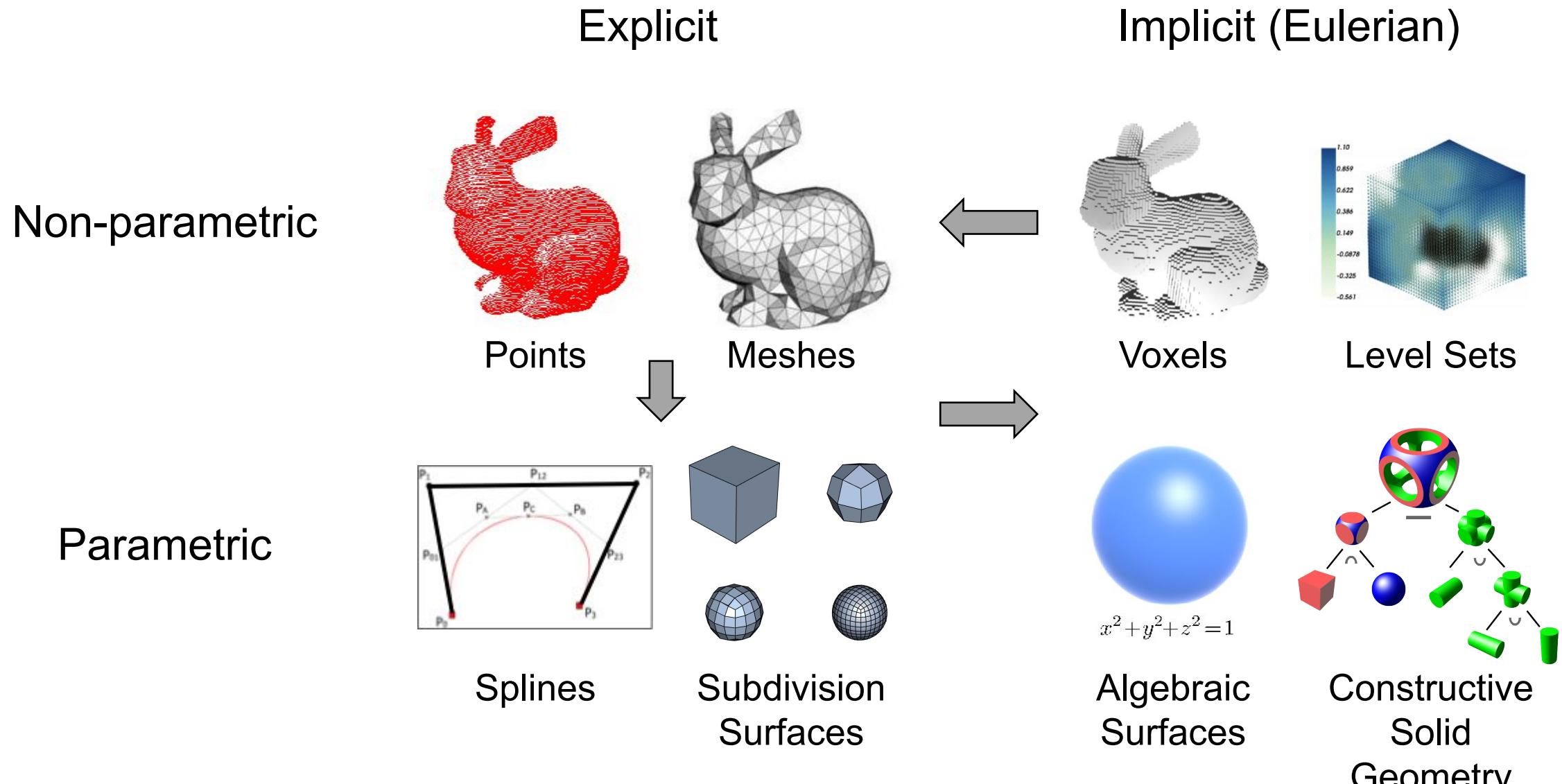
Input: Many images of the same scene
(with known camera parameters)



Output: Images showing the scene from novel viewpoints



Shape Representations



Next:
Human-Centered Artificial Intelligence