



**CTU**

CZECH TECHNICAL  
UNIVERSITY  
IN PRAGUE

# Deep Learning Essentials

## 11. Implicit Layers

Root-finding layers, Differentiable rendering, ...

Lukáš Neumann

# Prerequisites

- **Explicit function**

- Explicit functions define the dependent variable directly in terms of the independent variable  $y = f(x)$
- E.g.  $y = 2x^2 + 3$

- **Implicit functions**

- Even though the direct dependence  $f(x)$  does not exist, you can still express the relationship between  $x$  and  $y$  as  $F(x, y) = 0$
- E.g.  $x^2 + y^2 = 4$

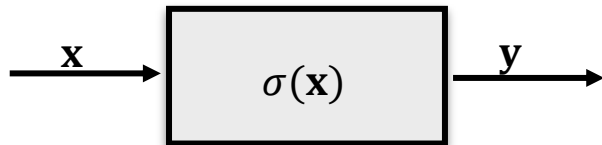
- **Implicit Differentiation**

$$\frac{d}{dx} F(x, y(x)) = \frac{\partial F}{\partial x} + \frac{\partial F}{\partial y} \frac{dy}{dx}$$

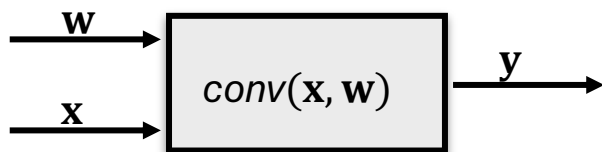
# Explicit vs Implicit layers

## Explicit layers

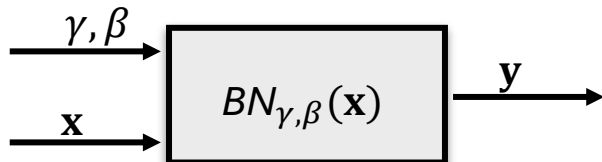
Sigmoid:



Convolution:

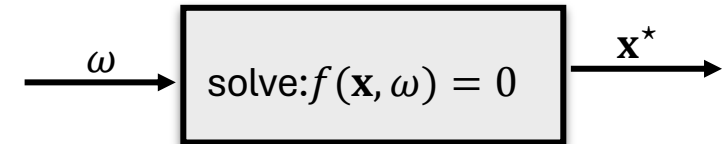


Batch-norm:

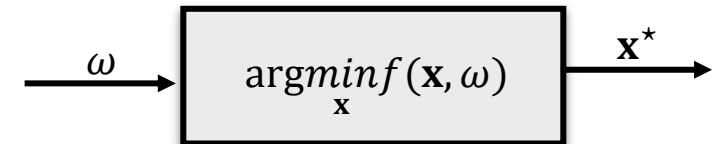


## Implicit layers

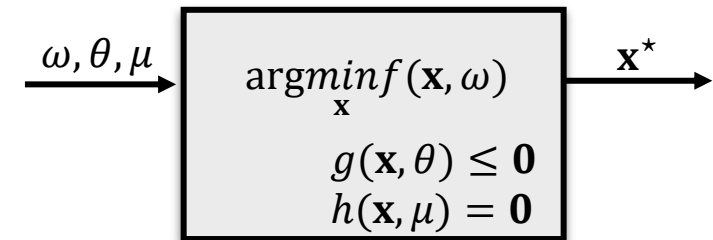
Root-finding layer:



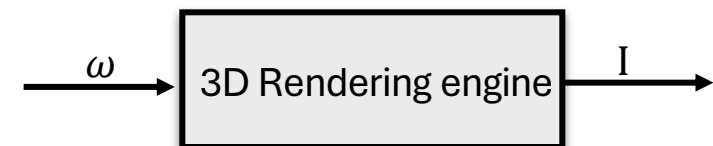
Unconstrained optimizer:



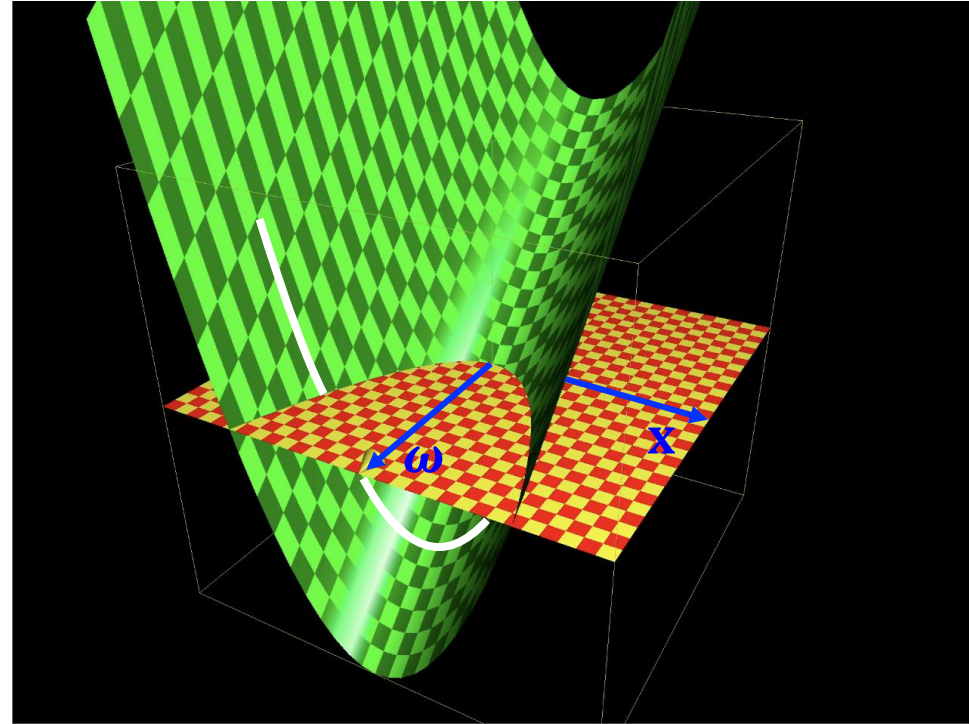
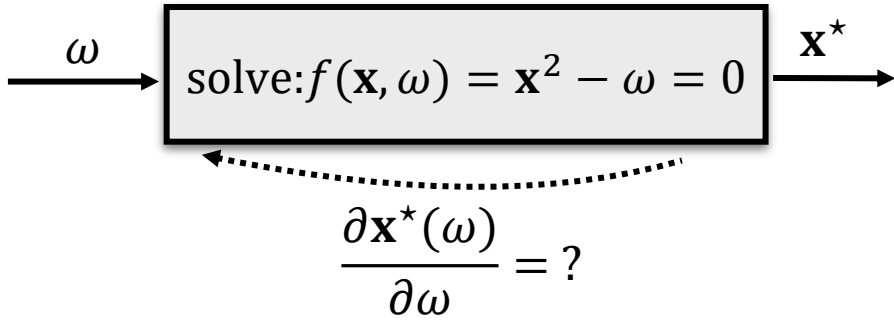
Constrained optimizer:



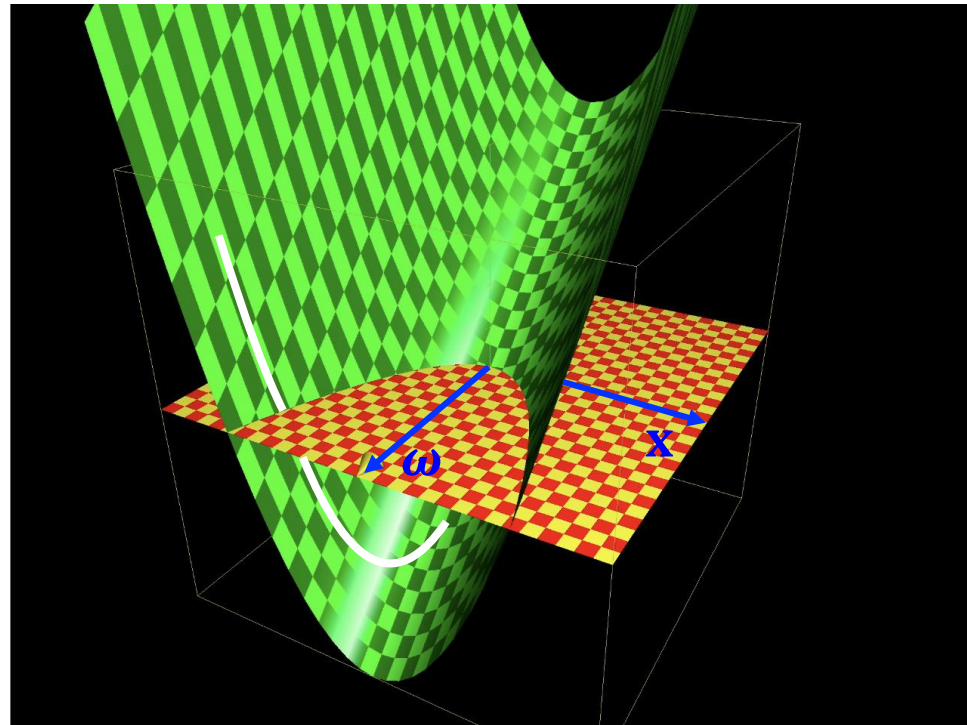
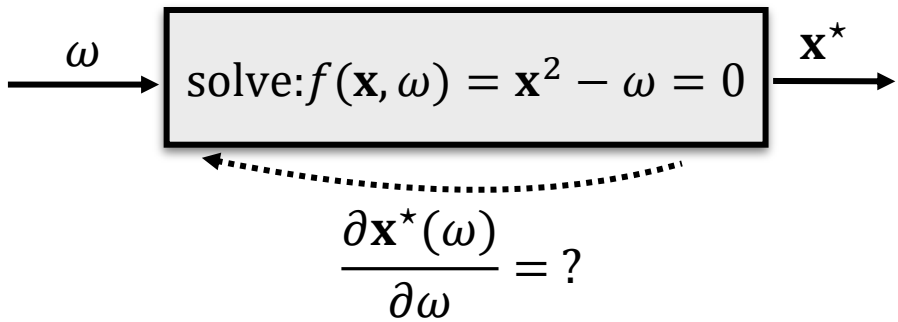
Differentiable Rendering:



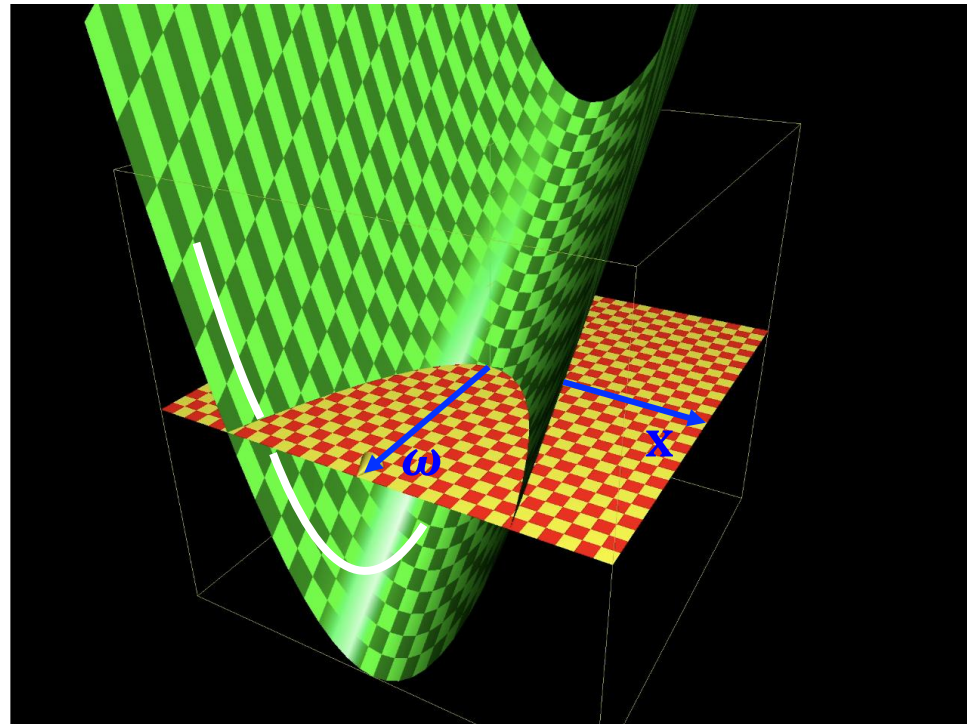
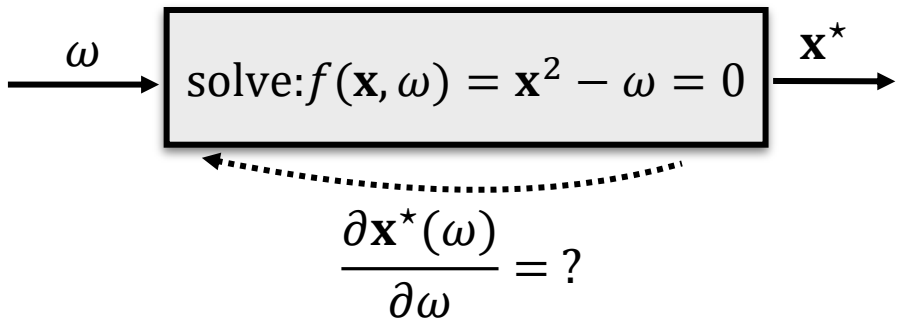
# Root-finding layer



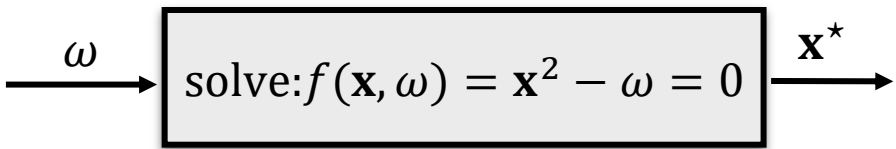
# Root-finding layer



# Root-finding layer



# Root-finding layer



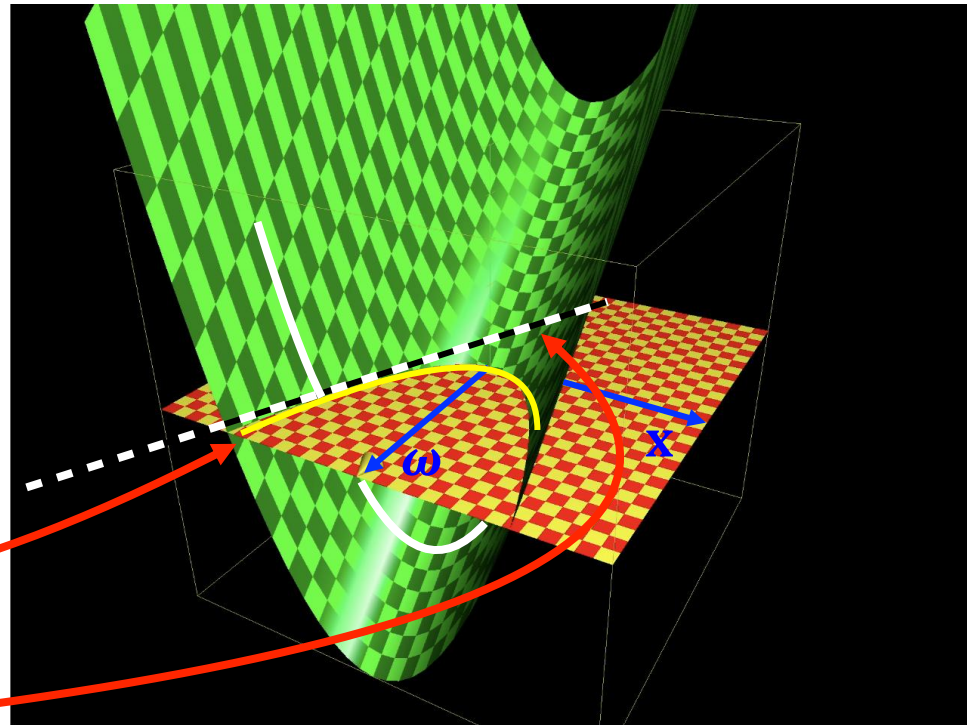
$$\frac{\partial \mathbf{x}^*(\omega)}{\partial \omega} = ?$$

While changing  $\omega$ , root function  $\mathbf{x}^*(\omega)$  follows the yellow curve

$$f(\mathbf{x}^*(\omega), \omega) = \mathbf{x}^*(\omega)^2 - \omega = 0$$

$$\frac{\partial f(\mathbf{x}^*(\omega), \omega)}{\partial \omega} = 0$$

Yellow space is locally defined as the direction in  $\omega$ -domain which locally preserves zero value of  $f(\mathbf{x}^*(\omega), \omega)$  function



# Root-finding layer

- Given parameters  $\omega$ , the code delivers solution  $\mathbf{x}^*$  such that  $f(\mathbf{x}^*, \omega) = 0$
- Often the function is a **compiled binary** without the access to its source
- We want to back-propagate through the layer  $\rightarrow$  we need to find  $\frac{\partial \mathbf{x}^*(\omega)}{\partial \omega}$

$$f(\mathbf{x}^*(\omega), \omega) = 0$$

Given  $\omega$ , the root function  $\mathbf{x}^*(\omega)$  satisfies this equation

$$\frac{\partial f(\mathbf{x}^*(\omega), \omega)}{\partial \omega} = 0$$

$\omega$  is allowed to **change only in directions** which **does not change** the value of  $f(\mathbf{x}^*(\omega), \omega)$  in order to stay within the solution manifold

$$f(\mathbf{x}^*(\omega), \omega) = \mathbf{x}^*(\omega)^2 - \omega = 0$$

$$\frac{\partial f(\mathbf{x}^*(\omega), \omega)}{\partial \omega} = 2\mathbf{x}^*(\omega) \cdot \frac{\partial \mathbf{x}^*(\omega)}{\partial \omega} - 1 = 0 \Rightarrow \boxed{\frac{\partial \mathbf{x}^*(\omega)}{\partial \omega} = \frac{1}{2\mathbf{x}^*(\omega)}}$$

# Implicit Differentiation

- Differentiation of compound function yields manifold that we search for

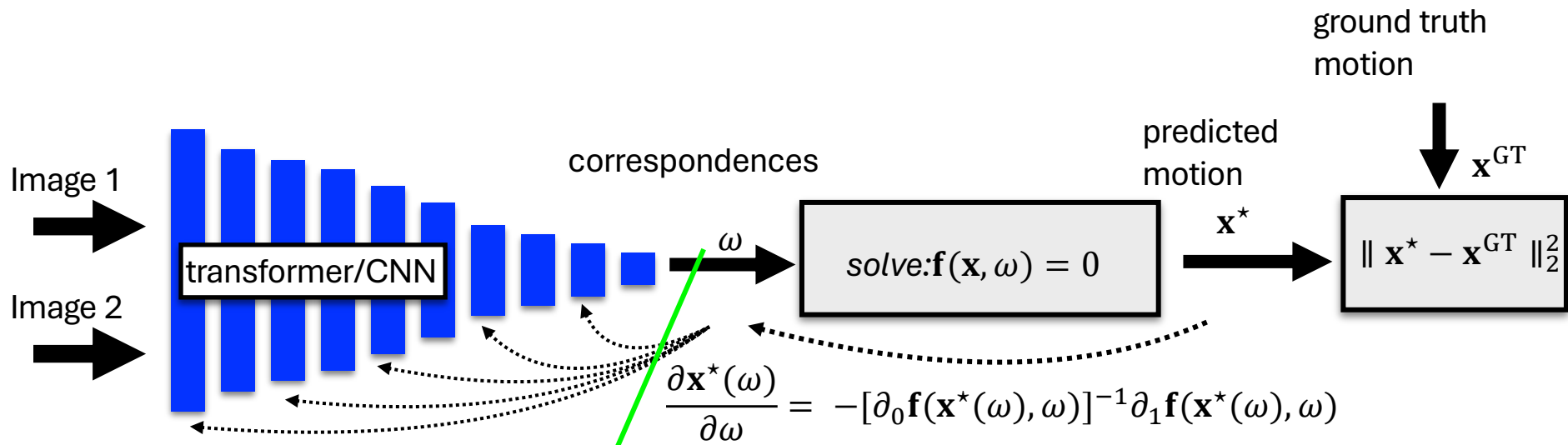
$$\frac{\partial f(\mathbf{x}^*(\omega), \omega)}{\partial \omega} = \partial_1 f(\mathbf{x}^*(\omega), \omega) \frac{\partial \mathbf{x}^*(\omega)}{\partial \omega} + \partial_2 f(\mathbf{x}^*(\omega), \omega) = 0$$

$$\frac{\partial \mathbf{x}^*(\omega)}{\partial \omega} = -[\partial_1 f(\mathbf{x}^*(\omega), \omega)]^{-1} \partial_2 f(\mathbf{x}^*(\omega), \omega)$$

using notation  $\frac{\partial f(a,b)}{\partial a} = \partial_1 f(a,b)$ ,  $\frac{\partial f(a,b)}{\partial b} = \partial_2 f(a,b)$

- Both terms  $\partial_1 f(\mathbf{x}^*(\omega), \omega)$  and  $\partial_2 f(\mathbf{x}^*(\omega), \omega)$  can be calculated using Autograd

# Image Correspondences



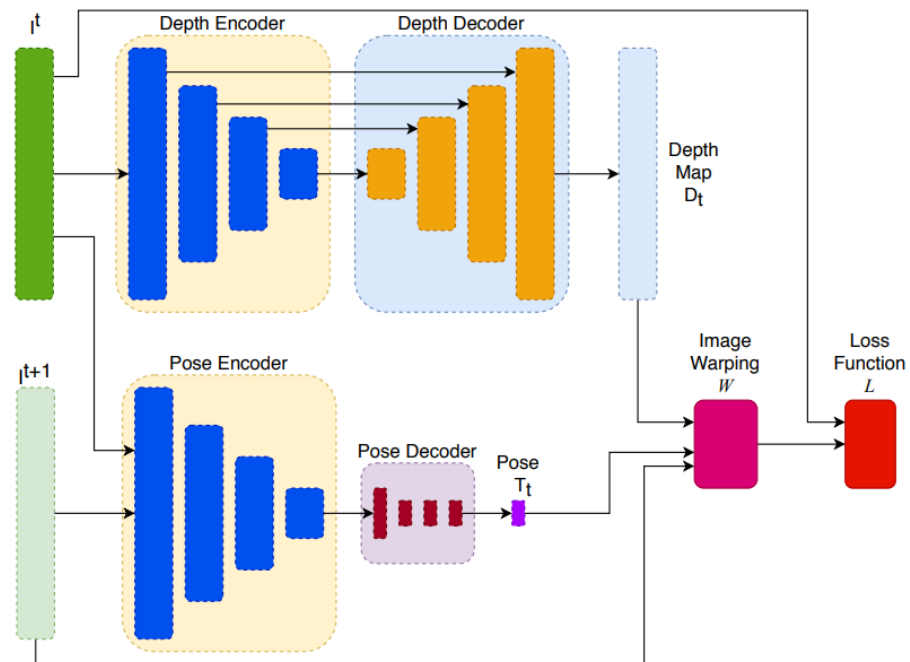
$f(\mathbf{x}, \omega) = 0$  is set of non-linear equations

# Monocular Depth Estimation

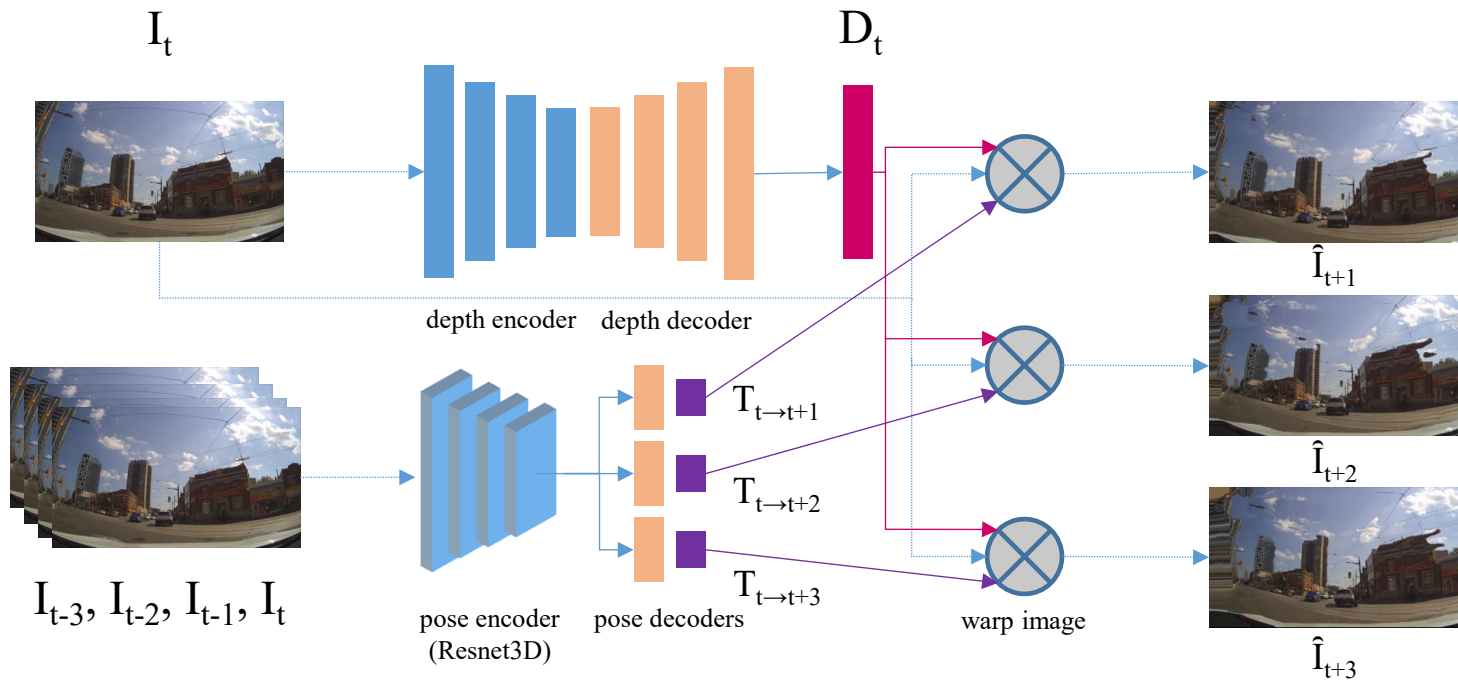


Godard et al., "Digging into self-supervised monocular depth estimation", ICCV 2019

# Monocular Depth Estimation

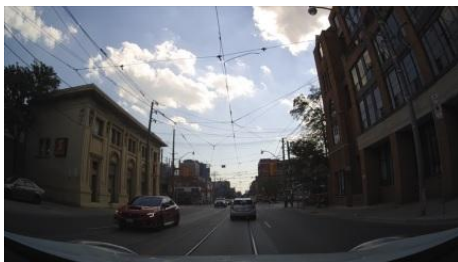


# Ego-Vehicle Motion Prediction

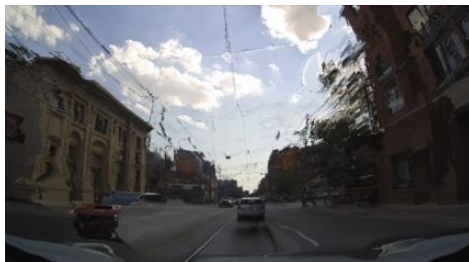
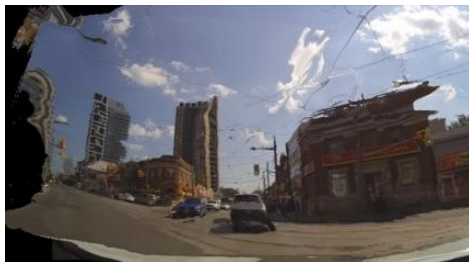


# Ego-Vehicle Motion Prediction

Observed



Predicted @ +1.5s

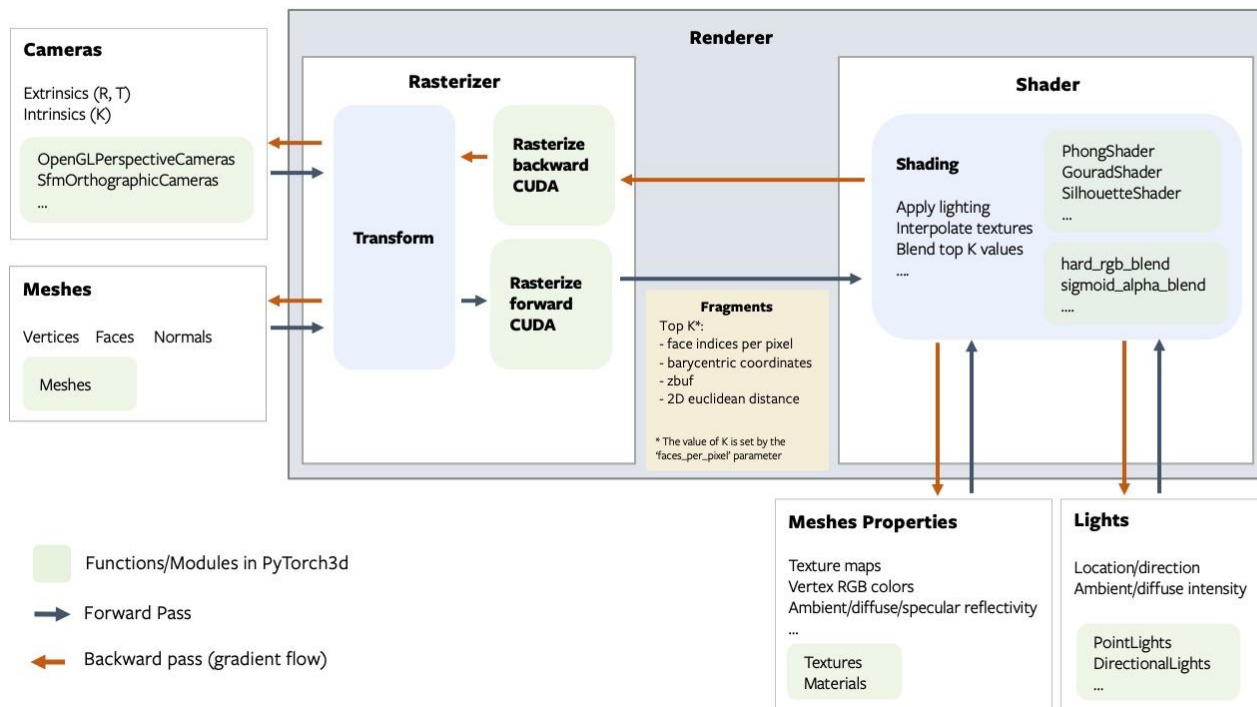


Actual @ +1.5s



# Differentiable Rendering

- Pytorch3D



# Mesh R-CNN

Input Image



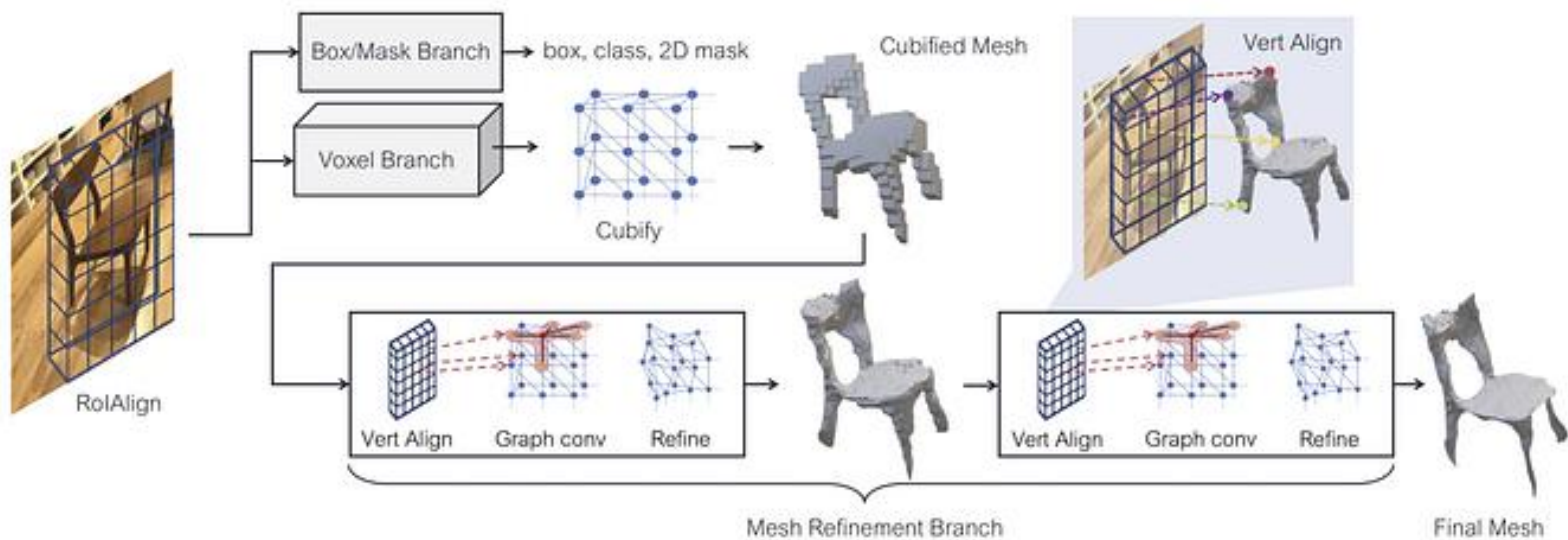
2D Recognition



3D Meshes

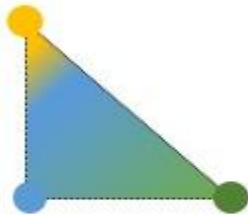
3D Voxels

# Mesh R-CNN



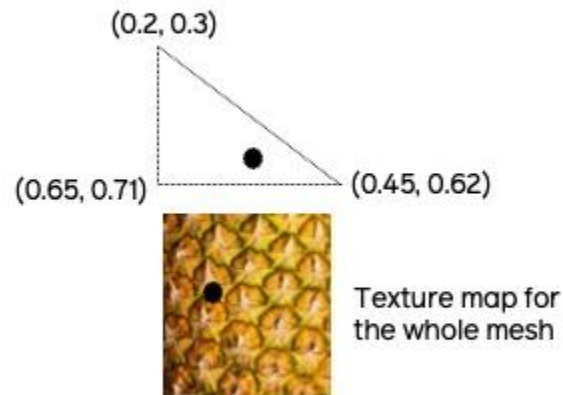
# Differentiable Rendering

## Vertex textures



Textures: (N, V, D)

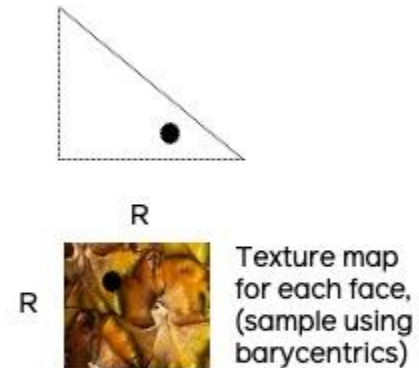
## Texture map + Vertex UV coordinates



UV coords: (N, V, 2)

Texture map: (N, H, W, 3)

## Texture Atlas [1]



Atlas: (N, F, R, R, 3)

# Summary

- Implicit layers allow incorporating prior knowledge  
→ less overfitting
- Many different implicit layers, choose the one which fits your problem the best  
→ “Always use the right tool”



# Competencies gained for the test

- Explicit vs Implicit Layers
- Useful implicit layers (root-finding, optimization, differentiable rendering)