Inverse Volume Rendering

Absorption + Emission, Automatic Differentiation

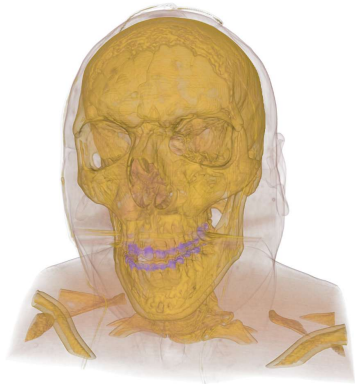
Sebastian Weiß

Lehrstuhl für Computergrafik
Technische Universität München

November 2020

Overview

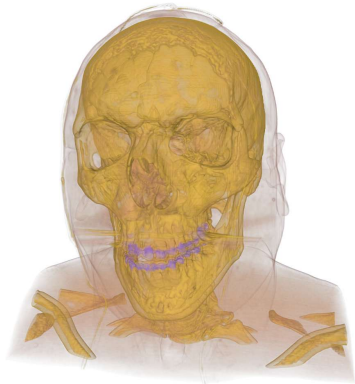
1. Models and Assumptions
   - 1.1. Absorption
   - 1.2. Emission
   - 1.3. Transfer Function



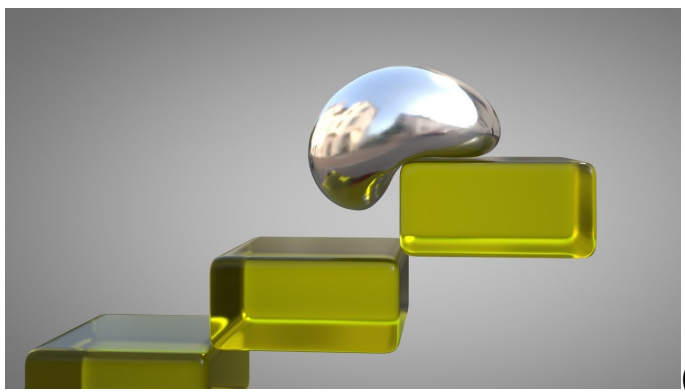
Overview

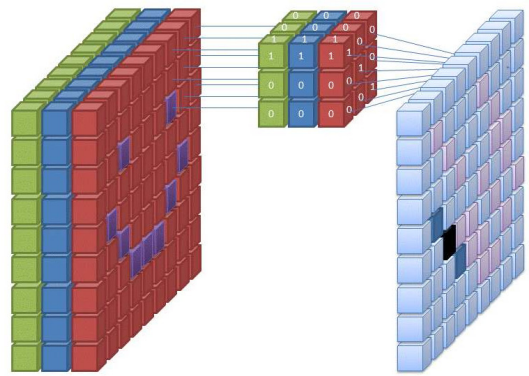
1. Models and Assumptions
   - 1.1. Absorption
   - 1.2. Emission
   - 1.3. Transfer Function





2. Automatic Differentiation
   - 2.1. Forward Mode
   - 2.2. Backward Mode

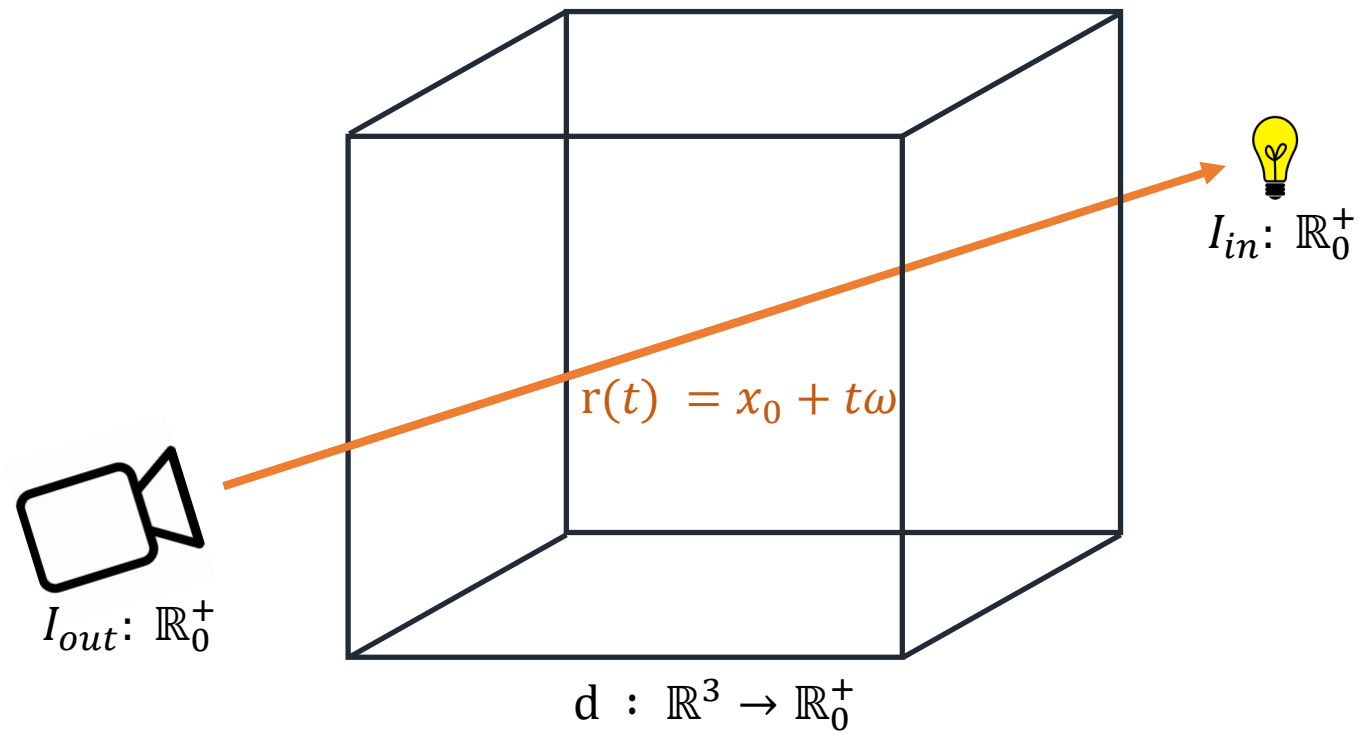

(1)


(2)

(1) https://www.youtube.com/watch?v=HwJyg5o3HGg
(2) https://commons.wikimedia.org/wiki/File:Convolutional_Neural_Network_with_Color_Image_Filter.gif

1. Models and Assumptions

1. Scalar Field $d : \mathrm{R}^3 \; ! \; \mathrm{R}_0^+$ of densities
2. Ray $r(t) = x_0 + t\,!\;;\;\; x_0\; 2\; \mathrm{R}^3;\; !\; 2\; S^2$
3. Received Illumination $I_{\mathrm{out}}(x_0;\, !\,)\; 2\; \mathrm{R}_0^+$



1.1 Only Absorption

Assume a medium that only absorbs incoming light, no emission or scattering.

$$I_{\text{out}}(x_0; \omega) = \int_0^1 g(t) \exp\left(-\int_0^t (v(r(t'))) dt'\right) dt$$

$$\text{with } g(s) = c(s)\,\tau(s) + \int_\Omega p(\omega; \omega') I_{\text{out}}(r(t); \omega') d\omega'$$

Nelson Max. Optical Models for Direct Volume Rendering. *TVCG* 1995
Kajiya, von Herzen. Ray Tracing Volume Densities. *Computer Graphics* 1984

1.1 Only Absorption

Assume a medium that only absorbs incoming light, no emission or scattering.

$$I_{\text{out}}(x_0; \omega) = \int_0^1 g(t) \exp\left(-\int_0^t \tau(v(r(t'))) dt'\right) dt$$

$$\text{with } g(s) = c(s)\,\sigma(s) + \int_\Omega p(\omega; \omega') I_{\text{out}}(r(t); \omega') d\omega'$$

Set $g(1) = I_{\text{in}}$ and zero everywhere else (only absorption).
Set $\tau(d) = d$

Nelson Max. Optical Models for Direct Volume Rendering. *TVCG* 1995
Kajiya, von Herzen. Ray Tracing Volume Densities. *Computer Graphics* 1984

1.1 Only Absorption

Assume a medium that only absorbs incoming light, no emission or scattering.

$$I_{\text{out}}(x_0; \omega) = \int_0^1 g(t) \exp\left(-\int_0^t \tau(v(r(t'))) dt'\right) dt$$

$$\text{with } g(s) = c(s)\,\tau(s) + \int_\Omega p(\omega; \omega') I_{\text{out}}(r(t); \omega') d\omega'$$

Set $g(1) = I_{\text{in}}$ and zero everywhere else (only absorption).
Set $\tau(d) = d$

) Transparency $T(s) = \exp\left(-\int_0^s d(r(t)) dt\right)$

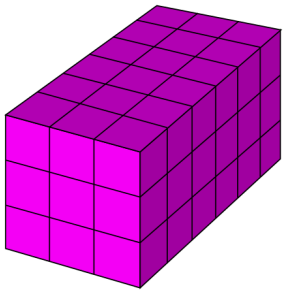
) Received Illumination $I_{\text{out}} = T(t_{\max}) I_{\text{in}}$

Nelson Max. Optical Models for Direct Volume Rendering. *TVCG* 1995
Kajiya, von Herzen. Ray Tracing Volume Densities. *Computer Graphics* 1984

1.1 Only Absorption

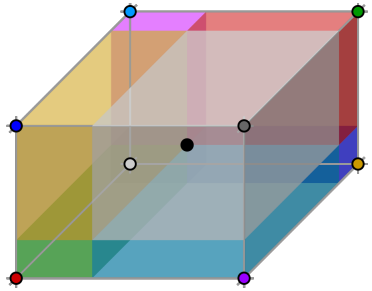
1.1 Only Absorption: Solutions

Densities on a grid v_i



(1)

Trilinear Interpolation $N_i(x)$



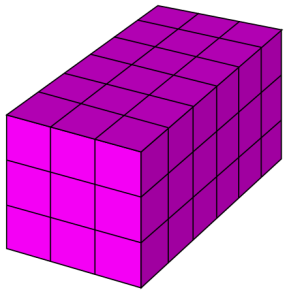
(2)

(1) https://en.wikipedia.org/wiki/Regular_grid

(2) https://en.wikipedia.org/wiki/Trilinear_interpolation

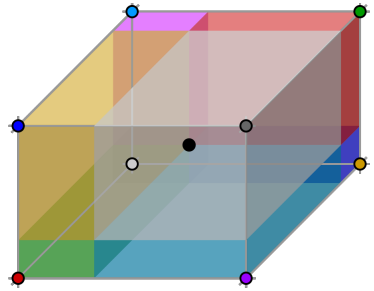
1.1 Only Absorption: Solutions

Densities on a grid v_i



(1)

Trilinear Interpolation $N_i(x)$



(2)

$$) \log I_{\text{out}} = \sum_{i \,2\, \text{voxels}} \sum_{j=1}^{8} v_{ij} \int_{t_0^{(i)}}^{t_1^{(i)}} N_j(r(t))dt$$

Assemble into matrices for all rays:

$$) \log I_{\text{out}} = Nv$$

(1) https://en.wikipedia.org/wiki/Regular_grid

(2) https://en.wikipedia.org/wiki/Trilinear_interpolation

1.1 Only Absorption: Differentiation

$$\log I_{\text{out}} = N \; v$$

output images — matrix of voxel-pixel contributions — voxel densities

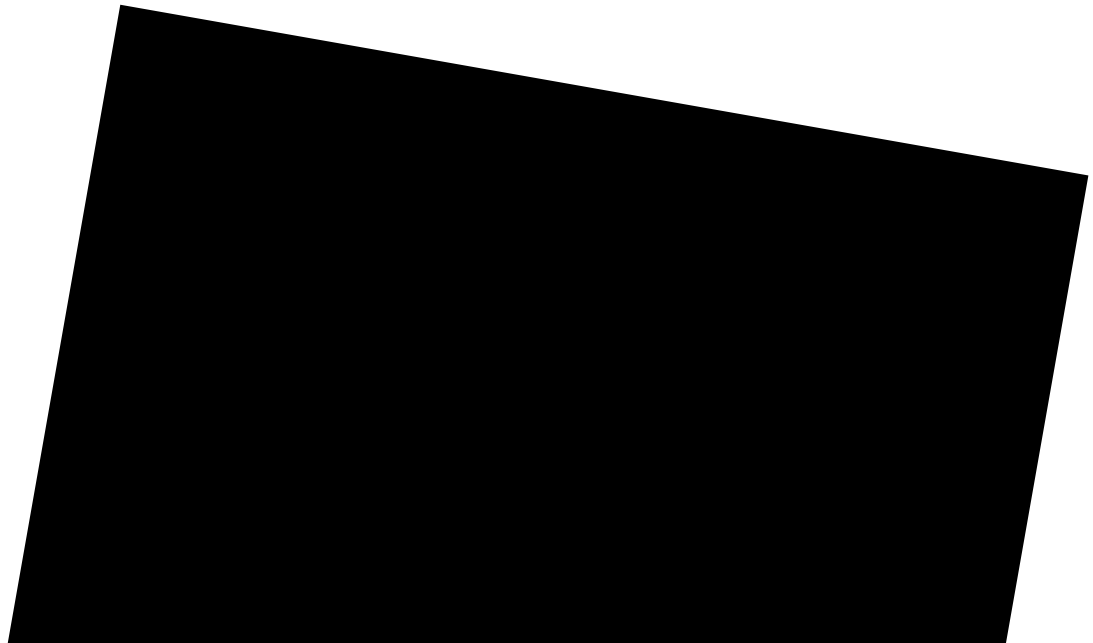
) Linear in v, simply solve it

1.1 Only Absorption: Differentiation

$$\log I_{\text{out}} = N \; v$$

output images — matrix of voxel-pixel contributions — voxel densities

) Linear in v, simply solve it



1.2 With Emission

Now every point in space also emits light $g(d)$
Usually dependent on (d): $g(d) := c(d)\ (d)$ for some color $c(d)$.

1.2 With Emission

Now every point in space also emits light $g(d)$
Usually dependent on (d): $g(d) := c(d)\ (d)$ for some color $c(d)$.

$$I_{\text{out}} = T(t_{\max})I_{\text{in}} + \int_0^{t_{\max}} g(t)T(t)dt$$

$$\text{with } T(s) = \exp\left(\int_0^{s} (v(r(t)))dt\right)$$

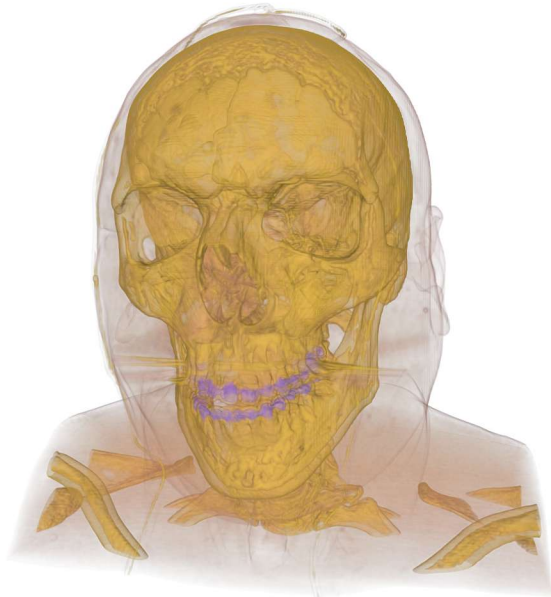
This is a nested integral and cannot be solved analytically.

1.2 With Emission

$$(d) = d; c(d) = d$$

1.2 With Emission

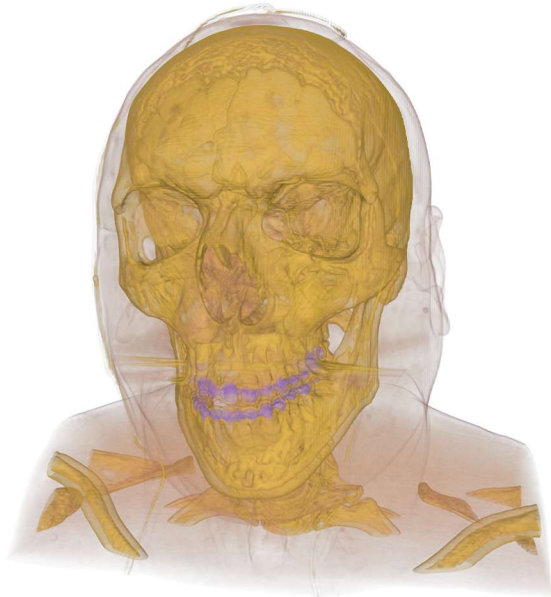
$(d) = d; c(d) = d$



$(d); c(d)$ arbitrary

1.2 With Emission

$(d) = d; c(d) = d$



$(d); c(d)$ arbitrary

2. Automatic Differentiation

$$x_0 = \text{const}$$
$$x_1 = f_1(x_0; w_1)$$
$$x_2 = f_2(x_1; w_2)$$
$$\vdots$$
$$x_k = f_k(x_{k-1}; w_k)$$

with

$k \in \mathbb{N}$	number of operations
$x_i \in \mathbb{R}^n$	state variables
$x_k \in \mathbb{R}^m$	size of the output / last state
$w_i \in \mathbb{R}^p$	parameters

2. Automatic Differentiation

$$
\begin{aligned}
x_0 &= \text{const} \\
x_1 &= f_1(x_0; w_1) \\
x_2 &= f_2(x_1; w_2) \\
&\vdots \\
x_k &= f_k(x_{k-1}; w_k)
\end{aligned}
\qquad \Big) \quad \frac{\mathrm{d}x_k}{\mathrm{d}w_1}
$$

with

$k \in \mathbb{N}$	number of operations
$x_i \in \mathbb{R}^n$	state variables
$x_k \in \mathbb{R}^m$	size of the output / last state
$w_i \in \mathbb{R}^p$	parameters

2.1 AutoDiff: Forward Mode

$$
\begin{aligned}
x_0 &= \text{const} \\
x_1 &= f_1(x_0; w_1) \\
x_2 &= f_2(x_1; w_2) \\
&\vdots \\
x_k &= f_n(x_{k\ \ 1}; w_k)
\end{aligned}
$$

2.1 AutoDiff: Forward Mode

$$x_0 = \text{const}$$
$$x_1 = f_1(x_0; w_1)$$
$$x_2 = f_2(x_1; w_2)$$
$$\vdots$$
$$x_k = f_n(x_{k-1}; w_k)$$

Replace each $x_i; w_i$ by

$$\tilde{x}_i = \langle x_i; \frac{dx_i}{dw_1} \rangle \; ; \; \tilde{w}_i = \langle w_i; \frac{dw_i}{dw_1} \rangle$$

Note that $\tilde{x}_0 = \langle x_0; 0 \rangle; \tilde{w}_1 = \langle w_1; 1 \rangle$

2.1 AutoDiff: Forward Mode

$$x_0 = \text{const}$$
$$x_1 = f_1(x_0; w_1)$$
$$x_2 = f_2(x_1; w_2)$$
$$\vdots$$
$$x_k = f_n(x_{k-1}; w_k)$$

Replace each $x_i; w_i$ by

$$\tilde{x}_i = x_i; \frac{\mathrm{d}x_i}{\mathrm{d}w_1} \quad ; \quad \tilde{w}_i = w_i; \frac{\mathrm{d}w_i}{\mathrm{d}w_1}$$

Note that $\tilde{x}_0 = hx_0; 0i; \tilde{w}_1 = hw_1; 1i$

Replace a function $c = f(a; b)$ by $\tilde{c} = \tilde{f}(\tilde{a}; \tilde{b})$ with

$$c; \frac{\mathrm{d}c}{\mathrm{d}c_1} = f(a; b);$$
$$J_{fja}(a; b)\frac{\mathrm{d}a_i}{\mathrm{d}w_1} + J_{fjb}(a; b)\frac{\mathrm{d}b_i}{\mathrm{d}w_1}$$

2.1 AutoDiff: Forward Mode

$$x_0 = \text{const}$$
$$x_1 = f_1(x_0; w_1)$$
$$x_2 = f_2(x_1; w_2)$$
$$\vdots$$
$$x_k = f_n(x_{k-1}; w_k)$$

Replace each $x_i; w_i$ by

$$\tilde{x}_i = \langle x_i; \frac{\mathrm{d}x_i}{\mathrm{d}w_1} \rangle \quad ; \quad \tilde{w}_i = \langle w_i; \frac{\mathrm{d}w_i}{\mathrm{d}w_1} \rangle$$

Note that $\tilde{x}_0 = \langle x_0; 0 \rangle; \tilde{w}_1 = \langle w_1; 1 \rangle$

Replace a function $c = f(a; b)$ by $\tilde{c} = \tilde{f}(\tilde{a}; \tilde{b})$ with

$$\langle c; \frac{\mathrm{d}c}{\mathrm{d}c_1} \rangle = \langle f(a; b); J_{fja}(a; b)\frac{\mathrm{d}a_i}{\mathrm{d}w_1} + J_{fjb}(a; b)\frac{\mathrm{d}b_i}{\mathrm{d}w_1} \rangle$$

- $O(np)$ memory
- $O(kn^2 + knp)$ computations

2.2 AutoDiff: Backward Mode / Adjoint Method

$$
\begin{aligned}
x_0 &= \text{const} \\
x_1 &= f_1(x_0; w_1) \\
x_2 &= f_2(x_1; w_2) \\
&\vdots \\
x_k &= f_n(x_{k\ \ 1}; w_k)
\end{aligned}
$$

2.2 AutoDiff: Backward Mode / Adjoint Method

$$
\begin{aligned}
x_0 &= \text{const} \\
x_1 &= f_1(x_0; w_1) \\
x_2 &= f_2(x_1; w_2) \\
&\vdots \\
x_k &= f_n(x_{k-1}; w_k)
\end{aligned}
$$

Assume $x_k \in \mathbb{R}$ is a scalar ($m = 1$), define

$$\hat{x}_i = \frac{\mathrm{d}x_k}{\mathrm{d}x_i} \in \mathbb{R}^n; \quad \hat{w}_i = \frac{\mathrm{d}x_k}{\mathrm{d}w_i} \in \mathbb{R}^p;$$

Note: $\hat{x}_k = 1$.

2.2 AutoDiff: Backward Mode / Adjoint Method

$$x_0 = \text{const}$$
$$x_1 = f_1(x_0; w_1)$$
$$x_2 = f_2(x_1; w_2)$$
$$\vdots$$
$$x_k = f_n(x_{k-1}; w_k)$$

Assume $x_k \in \mathbb{R}$ is a scalar ($m = 1$), define

$$\hat{x}_i = \frac{\mathrm{d}x_k}{\mathrm{d}x_i} \in \mathbb{R}^n; \quad \hat{w}_i = \frac{\mathrm{d}x_k}{\mathrm{d}w_i} \in \mathbb{R}^p;$$

Note: $\hat{x}_k = 1$.

1. Forward Pass, compute x_k normally, but save intermediate results

2.2 AutoDiff: Backward Mode / Adjoint Method

$$
\begin{aligned}
x_0 &= \text{const} \\
x_1 &= f_1(x_0; w_1) \\
x_2 &= f_2(x_1; w_2) \\
&\vdots \\
x_k &= f_n(x_{k-1}; w_k)
\end{aligned}
$$

Assume $x_k \in \mathbb{R}$ is a scalar ($m = 1$), define

$$\hat{x}_i = \frac{\mathrm{d}x_k}{\mathrm{d}x_i} \in \mathbb{R}^n; \quad \hat{w}_i = \frac{\mathrm{d}x_k}{\mathrm{d}w_i} \in \mathbb{R}^p;$$

Note: $\hat{x}_k = 1$.

1. Forward Pass, compute x_k normally, but save intermediate results
2. Backward Pass: For $i = k; \ldots; 1$

$$
\begin{aligned}
\hat{x}_{i-1} &= J_{f_i | x_{i-1}}(x_{i-1}; w_i)^T \hat{x}_i \\
\hat{w}_i &= J_{f_i | w_i}(x_{i-1}; w_i)^T \hat{x}_i
\end{aligned}
$$

2.2 AutoDiff: Backward Mode / Adjoint Method

$$x_0 = \text{const}$$
$$x_1 = f_1(x_0; w_1)$$
$$x_2 = f_2(x_1; w_2)$$
$$\vdots$$
$$x_k = f_n(x_{k-1}; w_k)$$

Assume $x_k \in \mathbb{R}$ is a scalar ($m = 1$), define

$$\hat{x}_i = \frac{dx_k}{dx_i} \in \mathbb{R}^n; \quad \hat{w}_i = \frac{dx_k}{dw_i} \in \mathbb{R}^p;$$

Note: $\hat{x}_k = 1$.

1. Forward Pass, compute x_k normally, but save intermediate results
2. Backward Pass: For $i = k; \ldots; 1$

$$\hat{x}_{i-1} = J_{f_i j x_{i-1}}(x_{i-1}; w_i)^T \hat{x}_i$$
$$\hat{w}_i = J_{f_i j w_i}(x_{i-1}; w_i)^T \hat{x}_i$$

$\Rightarrow$ $O(kn)$ memory
$\Rightarrow$ $O(kn^2 + np)$ computations

2.3 AutoDiff: Comparison

Forward Mode

- I
  -) $O(np)$ memory
  -) $O(kn^2 + knp)$ ops

Backward Mode

- I
  -) $O(kn)$ memory
  -) $O(kn^2 + np)$ ops

2.3 AutoDiff: Comparison

Forward Mode

-) $O(np)$ memory
) $O(kn^2 + knp)$ ops
- Arbitrary number of outputs

Backward Mode

-) $O(kn)$ memory
) $O(kn^2 + np)$ ops
- Only one output! Or repeat backward pass

2.3 AutoDiff: Comparison

Forward Mode

-) $O(np)$ memory
) $O(kn^2 + knp)$ ops
- Arbitrary number of outputs
- Compile-time: implement with operator overloading

Backward Mode

-) $O(kn)$ memory
) $O(kn^2 + np)$ ops
- Only one output! Or repeat backward pass
- Dynamic computation graph Dynamic memory

2.3 AutoDiff: Comparison

Forward Mode

-) $O(np)$ memory
) $O(kn^2 + knp)$ ops
- Arbitrary number of outputs
- Compile-time: implement with operator overloading
- Expensive for large p

Backward Mode

-) $O(kn)$ memory
) $O(kn^2 + np)$ ops
- Only one output! Or repeat backward pass
- Dynamic computation graph Dynamic memory
- Lots of memory

2.3 AutoDiff: Comparison

Forward Mode

-) $O(np)$ memory
) $O(kn^2 + knp)$ ops
- Arbitrary number of outputs
- Compile-time: implement with operator overloading
- Expensive for large p

) use if k large, p small

Backward Mode

-) $O(kn)$ memory
) $O(kn^2 + np)$ ops
- Only one output! Or repeat backward pass
- Dynamic computation graph Dynamic memory
- Lots of memory

) use if k small, p large

2.3 AutoDiff: Comparison

Forward Mode

-) $O(np)$ memory
) $O(kn^2 + knp)$ ops
- Arbitrary number of outputs
- Compile-time: implement with operator overloading
- Expensive for large p

) use if k large, p small

) Sensitivity Analysis in simulations

Backward Mode

-) $O(kn)$ memory
) $O(kn^2 + np)$ ops
- Only one output! Or repeat backward pass
- Dynamic computation graph Dynamic memory
- Lots of memory

) use if k small, p large

) Neural networks

2.3 AutoDiff: Comparison

Forward Mode

-) $O(np)$ memory
) $O(kn^2 + knp)$ ops
- Arbitrary number of outputs
- Compile-time: implement with operator overloading
- Expensive for large p

) use if k large, p small

) Sensitivity Analysis in simulations

Backward Mode

-) $O(kn)$ memory
) $O(kn^2 + np)$ ops
- Only one output! Or repeat backward pass
- Dynamic computation graph Dynamic memory
- Lots of memory

) use if k small, p large

) Neural networks

For DVR we need some hybrid method...