Real-time Rendering

Mechanical engineering, Ungsig Nam

CS580 Student Presentation

10th May, 2016

Brief review of previous talk

- Noise Filtering in Monte Calro Rendering
  - Random Parameter Filtering(RPF)
  - Non-local means Filtering(NLM)
- Compare the characteristics of RPF and NLM

Presentation papers

- Peiran Ren et al. Global Illumination with Radiance Regression Functions, SIGGRAPH 2013
- Lei Yang et al. Image-Based Bidirectional Scene Reprojection, SIGGRAPH Asia 2011

1. Global Illumination with Radiance Regression Functions

Motivation

- Precomputed radiance transfer (PRT) is successful approach for indirect illumination, BUT in real-time rendering

- cannot deal with dynamic local light sources
- cannot deal with high frequency glossy interreflections.

→ Solve these problem with radiance regression functions(RRF)

Key idea

- Regression Function with augmented attributes
- Neural network structure & Training
- Input space partitioning & RRF combination

Radiance Regression Functions(RRF)

$$s(\mathbf{x}_p, \mathbf{v}, \mathbf{l}) = \int_{\Omega^+} \rho(\mathbf{x}_p, \mathbf{v}, \mathbf{v}_i)(\mathbf{n} \cdot \mathbf{v}_i) s_i(\mathbf{x}_p, \mathbf{v}_i) d\mathbf{v}_i$$

$$= \underbrace{s^0(\mathbf{x}_p, \mathbf{v}, \mathbf{l})}_{\text{direct}} + \underbrace{s^+(\mathbf{x}_p, \mathbf{v}, \mathbf{l})}_{\text{indirect}},$$

$\mathbf{x}_p$: surface point
v: viewing direction
l: position of the point light
ρ: BRDF
n: surface normal

Radiance Regression Functions(RRF)

$$\rho(\mathbf{x}_p, \mathbf{v}, \mathbf{v}_i) = \rho_c(\mathbf{v}, \mathbf{v}_i, \mathbf{a}(\mathbf{x}_p))$$

ρ_c: closed-form of **ρ**
$\mathbf{a}(\mathbf{x}_p)$: a set of reflectance parameters

$\mathbf{x}_p$: surface point
v: viewing direction
l: position of the point light
ρ: BRDF
n: surface normal

$$s^+(\mathbf{x}_p, \mathbf{v}, \mathbf{l}) = \int_{\Omega^+} \rho_c(\mathbf{v}, \mathbf{v}_i, \mathbf{a}(\mathbf{x}_p))(\mathbf{n}(\mathbf{x}_p) \cdot \mathbf{v}_i) s_i^+(\mathbf{x}_p, \mathbf{v}_i) \mathrm{d}\mathbf{v}_i$$

$$\rightarrow \; s^+(\mathbf{x}_p, \mathbf{v}, \mathbf{l}) = s_a^+(\mathbf{x}_p, \mathbf{v}, \mathbf{l}, \mathbf{n}(\mathbf{x}_p), \mathbf{a}(\mathbf{x}_p))$$

$$s_a^+(\mathbf{x}_p, \mathbf{v}, \mathbf{l}, \mathbf{n}(\mathbf{x}_p), \mathbf{a}(\mathbf{x}_p)) \approx \Phi(\mathbf{x}_p^i, \mathbf{v}^i, \mathbf{l}^i, \mathbf{n}^i, \mathbf{a}^i)$$

New function s_a^+ with an expanded set of attributes
→ n and a do not need to be inferred from training data in regression function Φ

Radiance Regression Functions(RRF)

$$\mathbf{x}^i = [\mathbf{x}_p^i, \mathbf{v}^i, \mathbf{l}^i, \mathbf{n}^i, \mathbf{a}^i]^T, \ \mathbf{y}^i = s^+(\mathbf{x}_p^i, \mathbf{v}^i, \mathbf{l}^i)$$

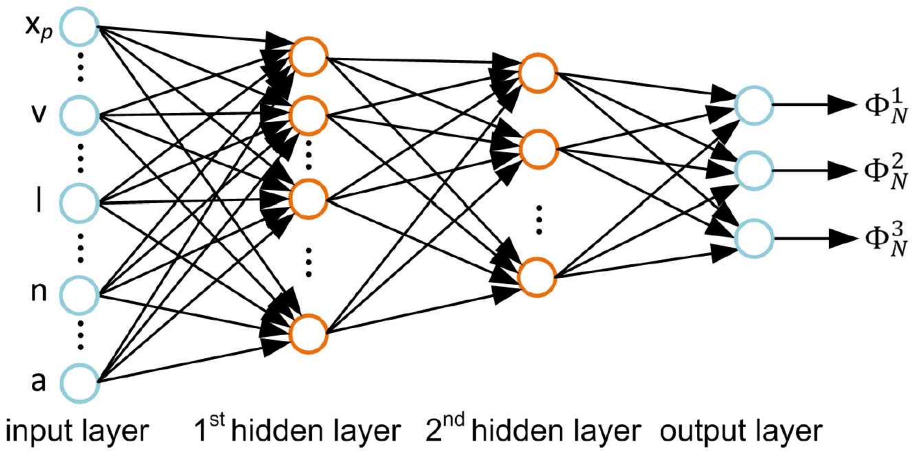
$$E = \sum_i ||\mathbf{y}^i - \Phi(\mathbf{x}_p^i, \mathbf{v}^i, \mathbf{l}^i, \mathbf{n}^i, \mathbf{a}^i)||^2$$

RRF Φ is determined by minimizing error, E

↓ + weight vector **w**

$$E(\mathbf{w}) = \sum_i ||\mathbf{y}^i - \Phi_N(\mathbf{x}_p^i, \mathbf{v}^i, \mathbf{l}^i, \mathbf{n}^i, \mathbf{a}^i, \mathbf{w})||^2$$

Radiance Regression Functions(RRF)

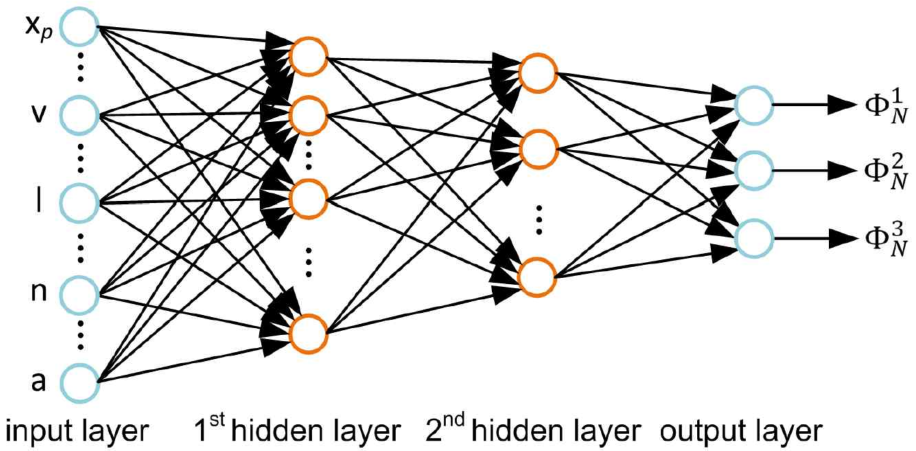


$$n_j^i = \sigma(z_j^i), \;\; z_j^i = w_{j0}^i + \sum_{k>0} w_{jk}^i n_k^{i-1}$$

n_j^i : node j in i-th layer w_{j0}^i : bias weight

w_{jk}^i : weight of the directed edge from node k to node j

Radiance Regression Functions(RRF)

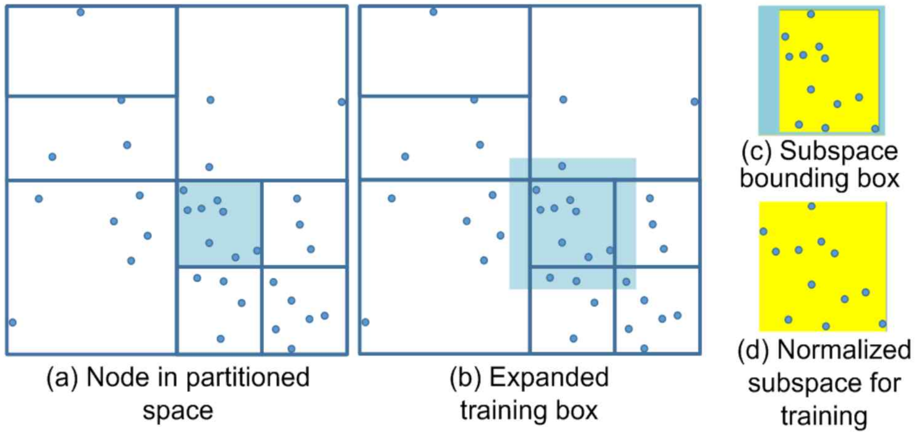


$$\sigma(z) = tanh(z) = 2/(1+e^{-2z}) - 1$$

$$\Phi_N = [\Phi_N^1, \Phi_N^2, \Phi_N^3] \qquad \mathbf{x} = [\mathbf{x}_p, \mathbf{v}, \mathbf{l}, \mathbf{n}, \mathbf{a}]$$

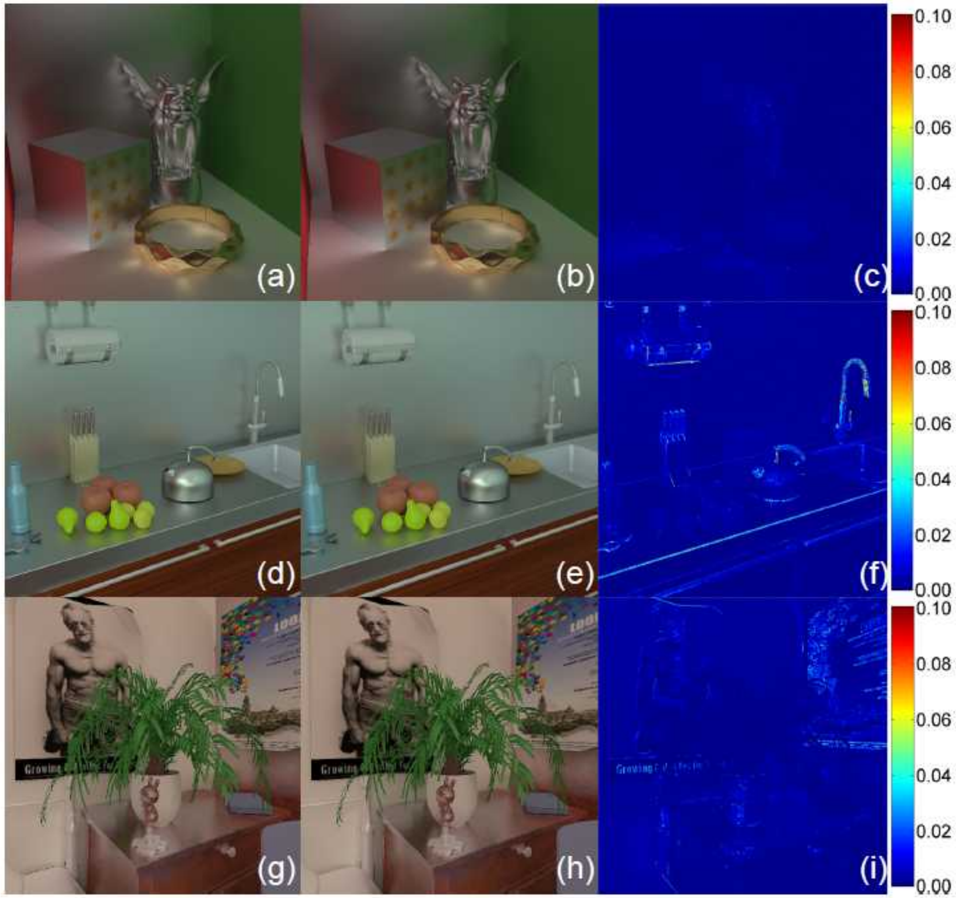
$$\Phi_N^i(\mathbf{x}, \mathbf{w}) = w_{i0}^3 + \sum_{j>0} w_{ij}^3 \sigma(w_{j0}^2 + \sum_{k>0} w_{jk}^2 \sigma(w_{k0}^1 + \sum_{l=1}^{9} w_{kl}^1 x_l))$$

Handling scene complexity

Input Space Partitioning	RRF Combination
	$$s^{+}(\mathbf{x}_p, \mathbf{v}, \mathbf{l}_1, \ldots, \mathbf{l}_K) = \sum_{k=1}^{k=K} c_k \Phi_N(\mathbf{x}_p, \mathbf{v}, \mathbf{l}_k, \mathbf{n}, \mathbf{a}, \mathbf{w})$$ $\mathbf{l}_K$: position of k-th light source c_k : color of the k-th light
For complex scenes, expanding the neural network becomes infeasible for real-time rendering → By decomposing the space and fitting a separate RRF to the training data of each region	Linear combination of respective point source RRF.

Results

Path-tracing RRF Difference b/w results



Results

Scene	RRF Size	FPS	Dir. Shading	Tree Trav.	RRF Eval.
CornellBox	5.64MB	61.9fps	5.33ms	2.39ms	8.43ms
Plant	66.77MB	32.6fps	5.10ms	2.52ms	23.05ms
Kitchen	33.12MB	36.5fps	15.341ms	2.37ms	9.62ms
Sponza	24.81MB	60.8fps	6.75ms	2.16ms	7.54ms
Bedroom	109.09MB	69.1fps	2.61ms	2.44ms	9.40ms

Results

Limitation of RRF

- Long time for preprocessing
- Dimensionality of the input vector should not be too high
- It provides a good approximation only of the indirect illumination near sampled viewing directions and light positions.

2. Image-Based Bidirectional Scene Reprojection

Motivation

- Existing upsampling strategies only reuse information from previous frames

 - smooth shading interpolation
 - Higher, more stable framerate

Key Idea

- Temporal direction: Bidirectional
  - upsamples rendered content by reusing data from both temporal directions(forward and backward)
- Data access: Gather
  - Simply involves texture lookups(index) into a previously rendered image
- Correspondence domain: Source
  - Performs reprojection using only image buffers without rasterization

Image-based Interpolation

F_t : the framebuffer of the I-frame rendered at time t

I-frames: Rendered frames

B-frames: interpolated frames (bidirectionally predicted)

Between successive I-frames F_t , F_{t+1} , there are n-1 B-frames, corresponding to times t+α

$$\alpha \in \left\{\frac{1}{n}, \dots, \frac{n-1}{n}\right\}$$

$\pi_{t \to t'}$: the transformation that maps the surface point $\bar{p}_t$ at time t into the clip space of time t′

$p = (p_x, p_y)$: 2D coordinates of a pixel in clip space

$\bar{p} = (p_x, p_y, Z[p])$: 3D coordinates of geometry rasterization, Z: depth buffer

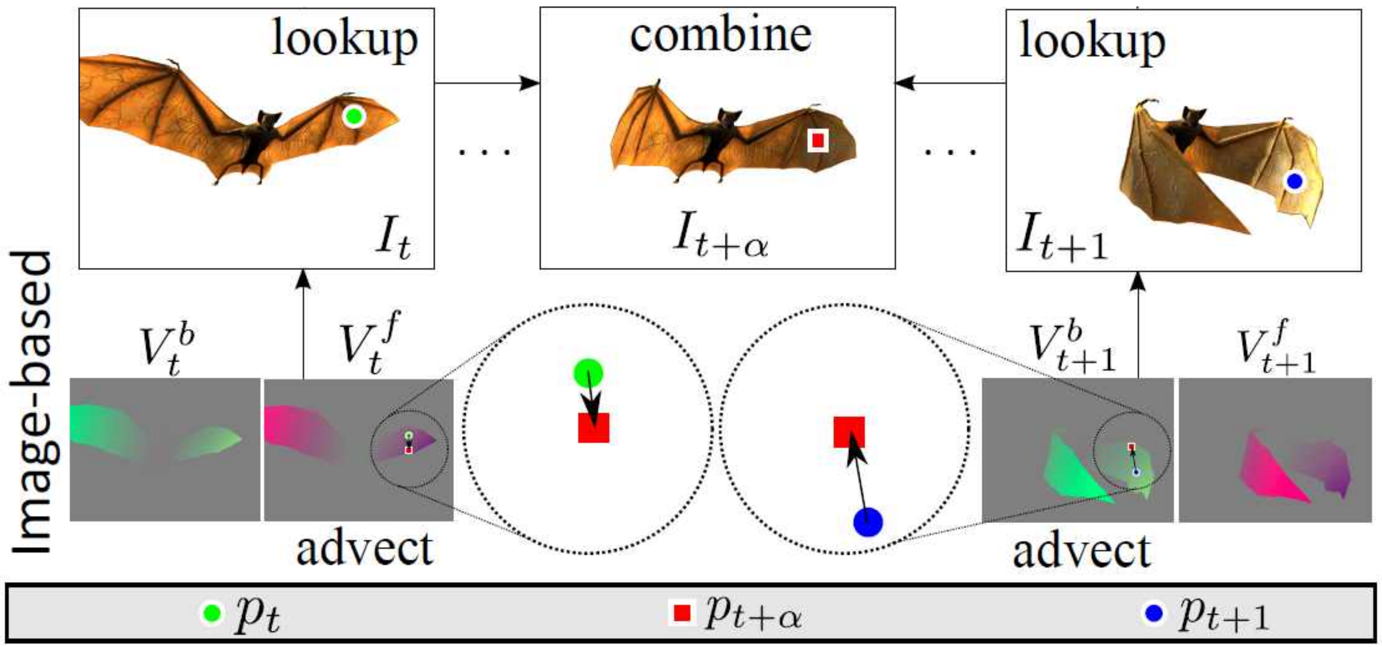
Image-based Interpolation

- To render the full 3D scene at I-frames using conventional methods and then insert interpolated B-frames between these to achieve a higher framerate.
- Its algorithm reconstructs B-frames at uniformly spaced time locations in the interval between t and t+1.
- The idea is to augment the I-frame buffers with information about the 3D scene flow between adjacent I-frames.

Image-based Interpolation

$V_t^f[p] = \pi_{t \to t+1}(\bar{p}_t) - \bar{p}_t$: forward flow field (encodes the motion of the scene at each pixel between I-frames [t,t+1])

$V_{t+1}^b[p] = \pi_{t+1 \to t}(\bar{p}_{t+1}) - \bar{p}_{t+1}$: backward flow field(between I-frames [t+1,t])



http://research.microsoft.com/en-us/um/people/hoppe/proj/bireproj/

Image-based Interpolation

- Assumptions:
  1. The motion between t and t+1 is linear
  2. The motion flow field is continuous and smooth
- Given $p_{t+\alpha}$, find p_t in field V_t^f such that

$$p_t + \alpha V_t^f[p_t] = p_{t+\alpha}$$

  - Same for p_{t+1} (in reverse)
- An inverse-mapping problem

Motion flow

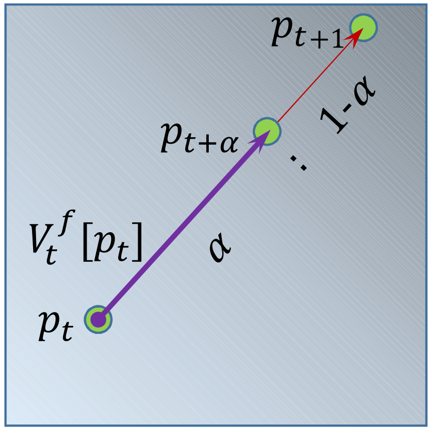


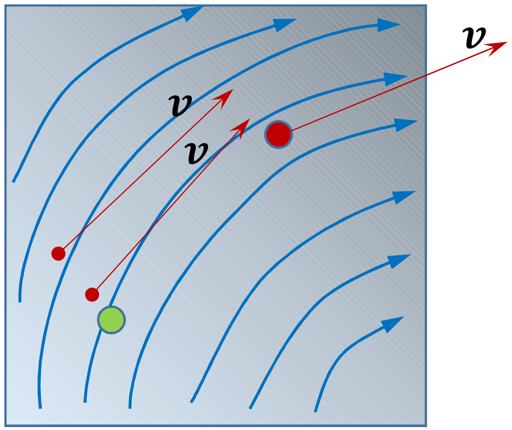
Image-space

Image-based Interpolation

$p_{t,0} = p_{t+\alpha}$ $\quad$ $p_{t,i} = p_{t+\alpha} - d_i^f$, where $d_i^f = \alpha V_t^f[p_{t,i-1}].xy$

- Iterative search
  1. Initialize vector $\boldsymbol{v}$ with the motion flow $\alpha V_t^f[p_{t+\alpha}]$
  2. Attempt to find p_t using $\boldsymbol{v}$
  3. Update $\boldsymbol{v}$ with the motion flow at current p_t estimate
  4. Repeat 2-3

Motion flow



Iterative reprojection

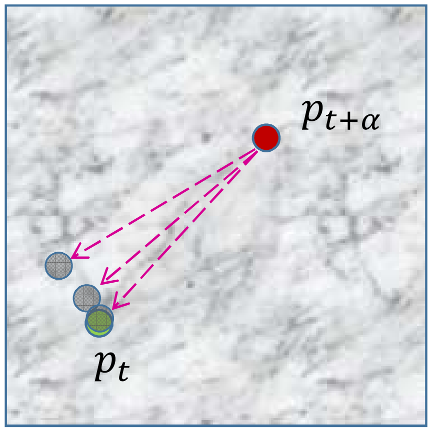


Image-based Interpolation

Iterative search – forward direction

$$z^f = Z_t[p_{t,m}] + \alpha V_t^f[p_{t,m}].z$$ Clip-space depth

$$e^f = \| \underbrace{p_{t,m} + \alpha V_t^f[p_{t,m}].xy}_{\text{final estimated point}} - \underbrace{p_{t+\alpha}}_{\text{initial estimated point}} \|$$ Screen-space error

final estimated point initial estimated point

Iterative search – backward direction

$$p_{t+1,0} = p_{t+\alpha}$$

$$p_{t+1,i} = p_{t+\alpha} - d_i^b$$ $$d_i^b = (1-\alpha)V_{t+1}^b[p_{t+1,i-1}].xy$$

$$z^b = Z_t[p_{t+1,m}] + (1-\alpha)V_{t+1}^b[p_{t+1,m}].z$$ Clip-space depth

$$e^b = \|p_{t+1,m} + (1-\alpha)V_{t+1}^b[p_{t+1,m}].xy - p_{t+\alpha}\|$$ Screen-space error

Image-based Interpolation

Visibility and shading

Case 1: $e^f, e^b < \epsilon_1$(tolerance error) and similar depths, or $\left|z^f - z^b\right| < \epsilon_2$

Blended color is

$$e^f < e^b \quad \rightarrow \quad (1-\alpha)I_t\left[p_{t,m}\right] + \alpha I_{t+1}[p_{t,m} + V_t^f[p_{t,m}].xy]$$

$$e^b \geq e^f \quad \rightarrow \quad (1-\alpha)I_t\left[p_{t+1,m} + V_{t+1}^b[p_{t+1,m}].xy\right] + \alpha I_{t+1}[p_{t+1,m}]$$

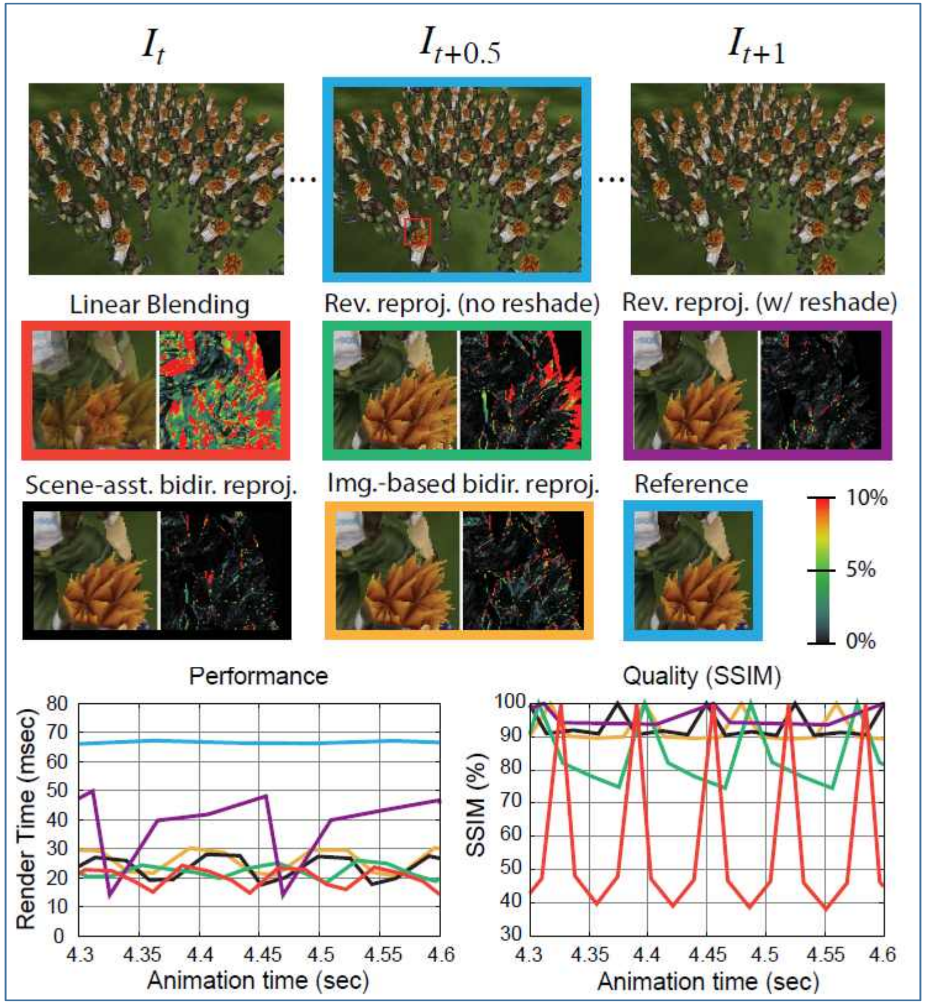
Case 2: $e^f, e^b < \epsilon_1$(tolerance error) and different depths

Select the color closest to the camera

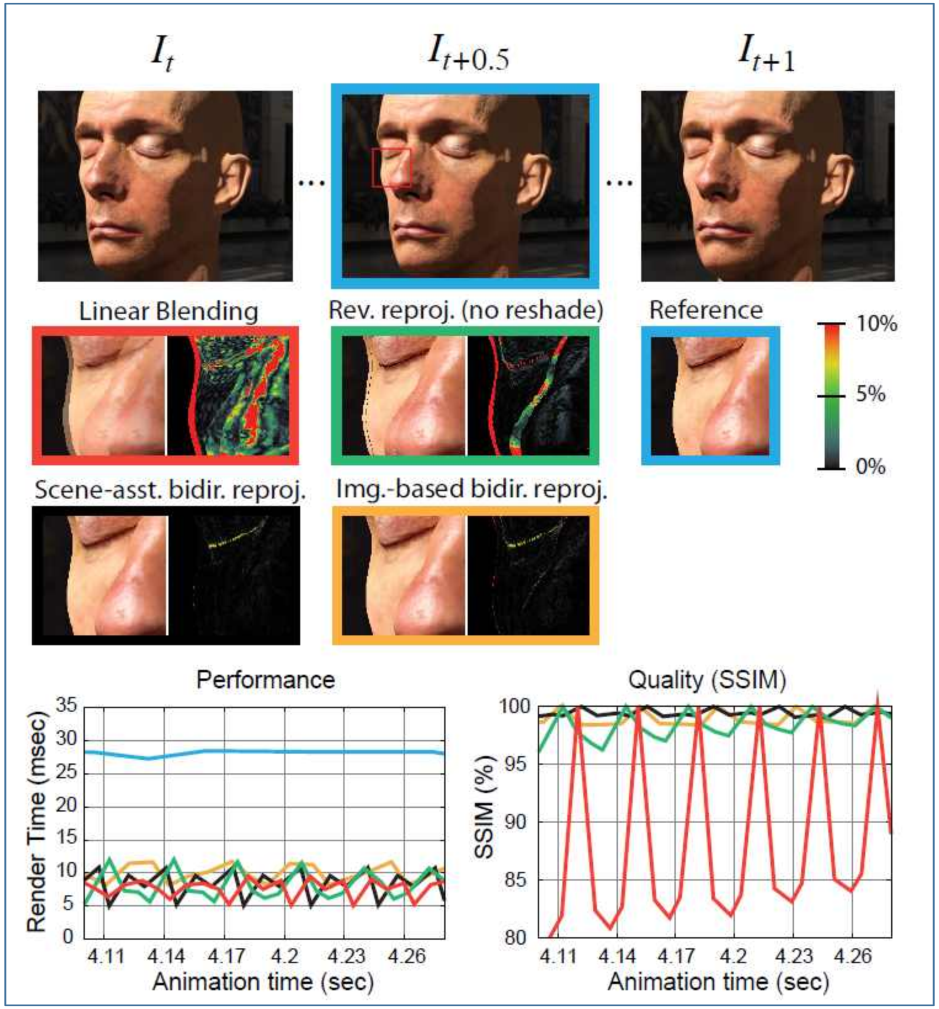
Additional Search Initializations

	Dual initialization	Latest-frame initialization
Forward search	$p'_{t,0} = p_{t+\alpha} + \alpha V^{b}_{t+1}[p_{t+\alpha}]$	$p'_{t,0} = p_{t+\alpha} - d^{f}_{0}$ $d^{f}_{0} = \frac{\alpha}{\alpha'} d'^{f}_{i}$
Backward search	$p'_{t+1,0} = p_{t+\alpha} + (1-\alpha)V^{f}_{t}[p_{t+\alpha}]$	$p'_{t+1,0} = p_{t+\alpha} - d^{b}_{0}$ $d^{b}_{0} = \frac{1-\alpha}{1-\alpha'} d'^{b}_{i}$

Result



I_t
$I_{t+0.5}$
I_{t+1}
Linear Blending
Rev. reproj. (no reshade)
Rev. reproj. (w/ reshade)
Scene-asst. bidir. reproj.
Img.-based bidir. reproj.
Reference
10%
5%
0%
Performance
Render Time (msec)
Animation time (sec)
Quality (SSIM)
SSIM (%)
Animation time (sec)



I_t
$I_{t+0.5}$
I_{t+1}
Linear Blending
Rev. reproj. (no reshade)
Reference
10%
5%
0%
Scene-asst. bidir. reproj.
Img.-based bidir. reproj.
Performance
Render Time (msec)
Animation time (sec)
Quality (SSIM)
SSIM (%)
Animation time (sec)

http://research.microsoft.com/en-us/um/people/hoppe/proj/bireproj/

Result

Result

Limitation of Image-Based Bidirectional Scene Reprojection

- Cannot express dynamic shading effects well (highlights, transparency etc.)
- Prone to make errors in the interpolated B-frames wherever the local search fails

THANK YOU

Do you have question or comment?

Quiz

Q1. In first paper, what is the number of attributes in RRF **represented by neural network**
+ the number of **hidden layer**?

a. 7 b. 8 c. 9 d. 10

Q2. In second paper, which is **correct** according to **temporal direction**?

a. Bidirection b. One-directional c. Random directional d. All-directional