EGSR 2008

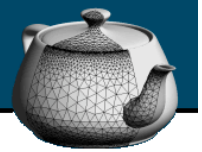
Geometry-Aware Framebuffer Level of Detail

Lei Yang **Pedro V. Sander**

Hong Kong University of Science and Technology

Jason Lawrence

University of Virginia



Motivation

- **Expensive procedural shading effects**
  - **Heavy pixel shader workload**
  - **Examples**
    - **Soft shadows**
      27fps
    - **Ambient Occlusion**
      3.2fps
    - **Procedural noise texture**
      120fps
    - **…**

Motivation

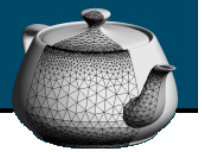
- A method for reducing pixel workload
  - General
  - Lightweight
  - No preprocessing
  - Smoothly adjustable tradeoff between speed/quality

Dynamic Resizing

- Render scene to low-res buffer (1st pass), then upsample to target resolution (2nd pass). [Montrym97]
  - # of original pixel shader invocation is reduced ($\propto 1/r^2$)
  - Blurs geometric discontinuities

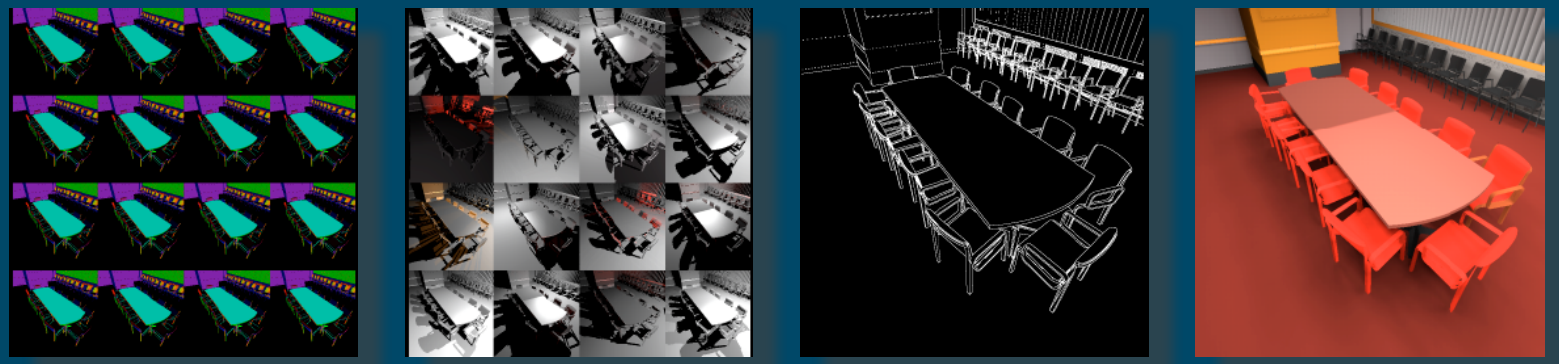
1st pass
Original shader

2nd pass
Upsample

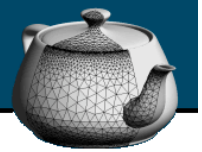


Related Work

- **Interleaved sampling** [Segovia06, Laine07]



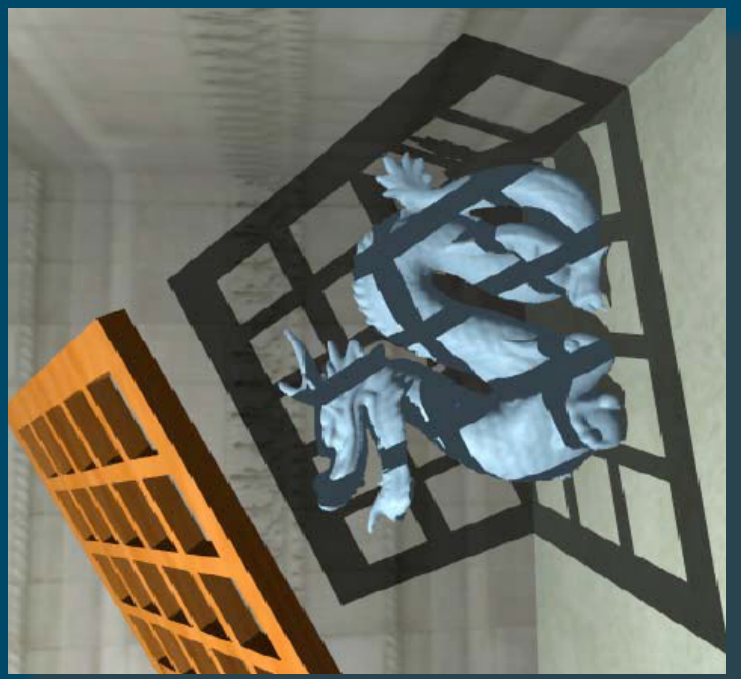
- **Image-based proxy accumulation** [Sloan07]

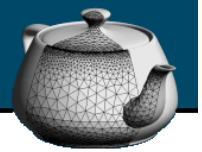


Related Work

- **Edge-and-Point render cache**

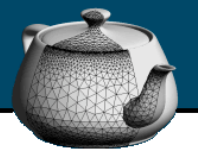
  [Bala03, Velázquez-Armendáriz06]



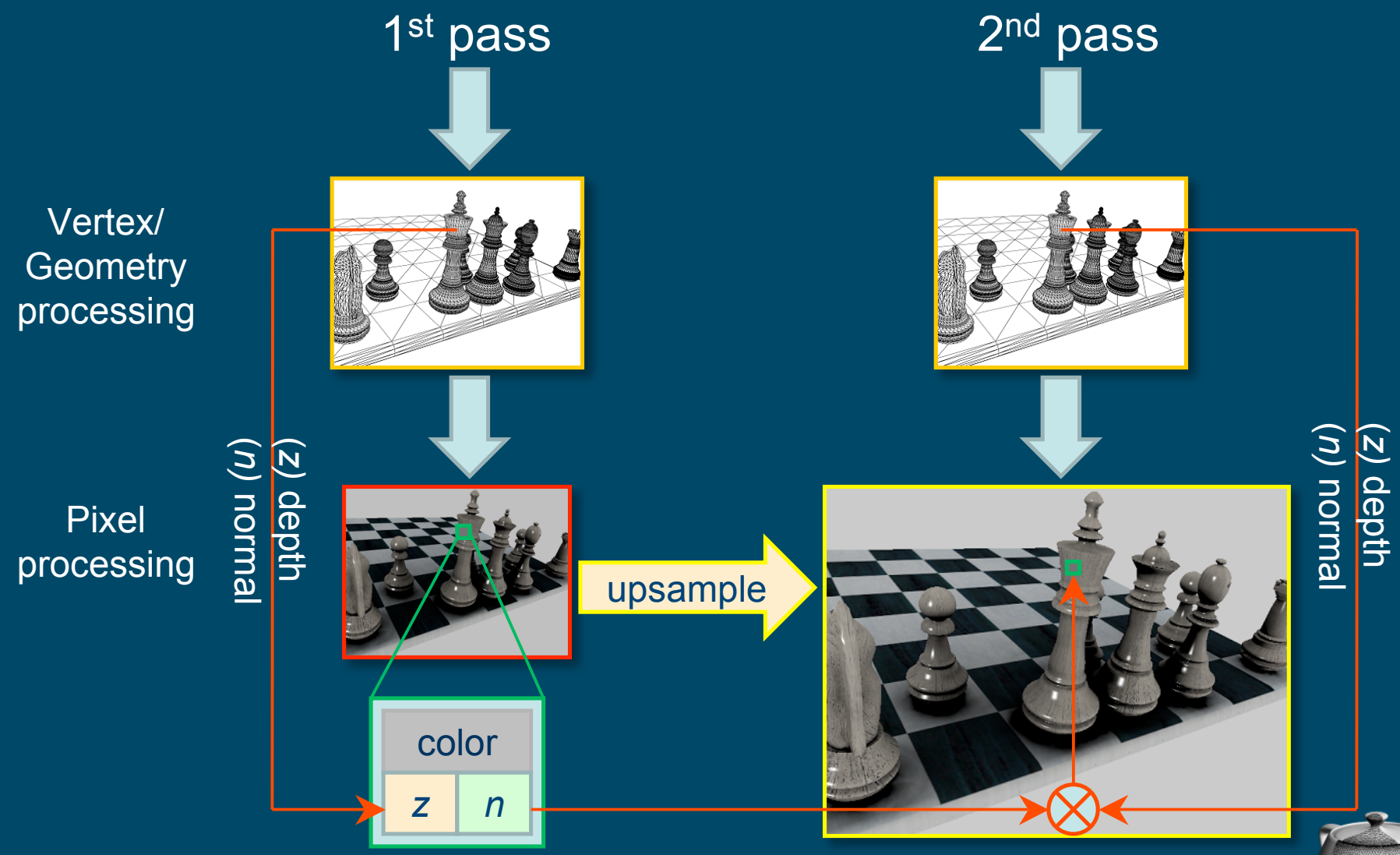


Overview

- Geometry-Aware Resizing
- Fine-Grained Resizing
- Automatic Framerate Control
- Results and Demo
- Discussions and Conclusion

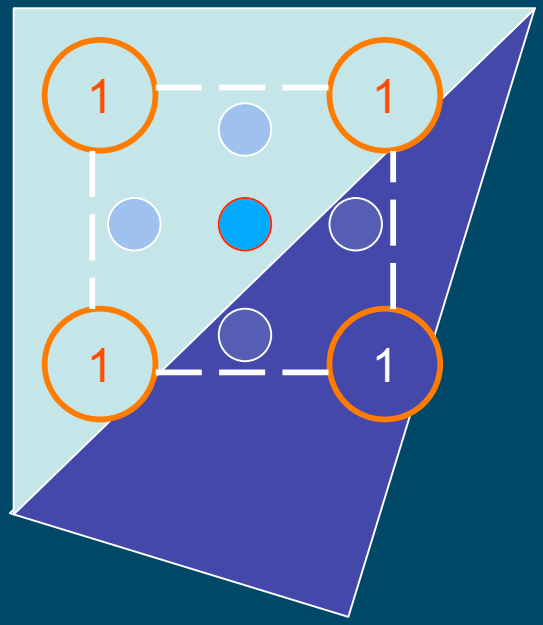


Geometry-Aware Resizing

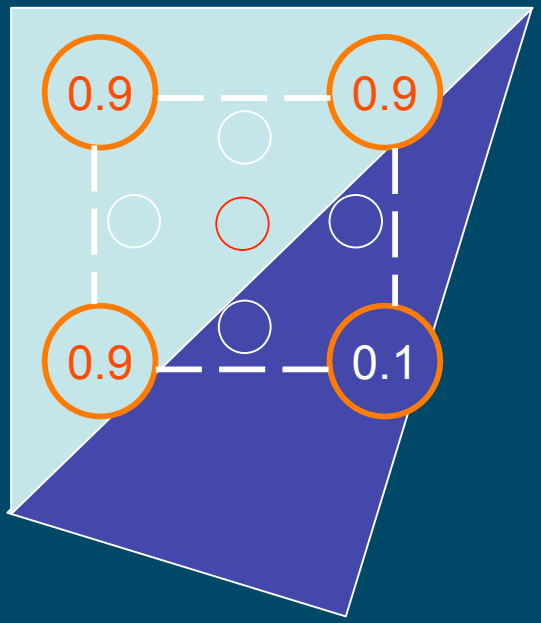


Geometry-Aware Reconstruction

Bilinear



Bilateral



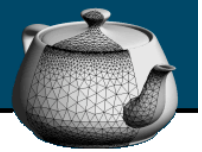
Weight samples based on geometric similarity

Joint Bilateral Filter

$$c_i^H = \frac{\sum c_j^L \; f(\hat{x}_i, x_j)\; g(|n_i^H - n_j^L|, \sigma_n)\; g(|z_i^H - z_j^L|, \sigma_z)}{\sum f(\hat{x}_i, x_j)\; g(|n_i^H - n_j^L|, \sigma_n)\; g(|z_i^H - z_j^L|, \sigma_z)}$$

Color sample *j* from the low-res buffer

Filter weight of sample *j*

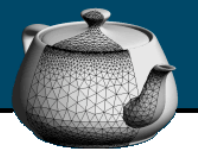


Joint Bilateral Filter

$$c_i^H = \frac{\sum c_j^L \; f(\hat{x}_i, x_j) \; g(|n_i^H - n_j^L|, \sigma_n) \; g(|z_i^H - z_j^L|, \sigma_z)}{\sum f(\hat{x}_i, x_j) \; g(|n_i^H - n_j^L|, \sigma_n) \; g(|z_i^H - z_j^L|, \sigma_z)}$$

Color sample *j* from the low-res buffer

Spatial filter: bilinear / biquadratic / bicubic / Gaussian

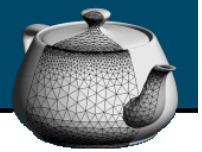


Joint Bilateral Filter

$$c_i^H = \frac{\sum c_j^L \, f(\hat{x}_i, x_j) \, g(|n_i^H - n_j^L|, \sigma_n) \, g(|z_i^H - z_j^L|, \sigma_z)}{\sum f(\hat{x}_i, x_j) \, g(|n_i^H - n_j^L|, \sigma_n) \, g(|z_i^H - z_j^L|, \sigma_z)}$$

Color sample *j* from the low-res buffer

Range filter 1: Gaussian of the ***normal*** distance

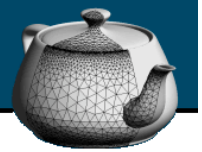


Joint Bilateral Filter

$$c_i^H = \frac{\sum c_j^L \, f(\hat{x}_i, x_j) \, g(|n_i^H - n_j^L|, \sigma_n) \, g(|z_i^H - z_j^L|, \sigma_z)}{\sum f(\hat{x}_i, x_j) \, g(|n_i^H - n_j^L|, \sigma_n) \, g(|z_i^H - z_j^L|, \sigma_z)}$$

Color sample *j* from the low-res buffer

Range filter 2: Gaussian of the ***depth*** distance

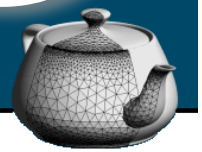


Joint Bilateral Filter

$$c_i^H = \frac{\sum c_j^L \, f(\hat{x}_i, x_j) \, g(|n_i^H - n_j^L|, \sigma_n) \, g(|z_i^H - z_j^L|, \sigma_z)}{\sum f(\hat{x}_i, x_j) \, g(|n_i^H - n_j^L|, \sigma_n) \, g(|z_i^H - z_j^L|, \sigma_z)}$$

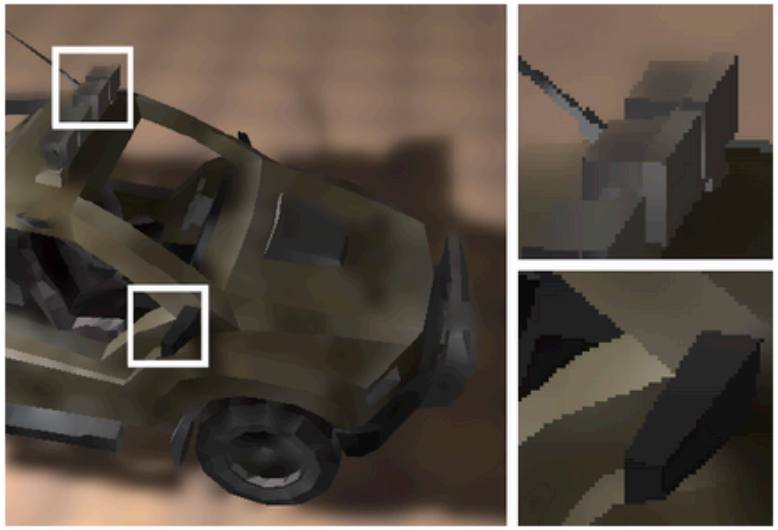
Large σ_z, large σ_n

Small σ_z, large σ_n



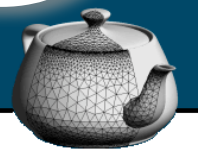
Joint Bilateral Filter

$$c_i^H = \frac{\sum c_j^L \, f(\hat{x}_i, x_j) \, g(|n_i^H - n_j^L|, \sigma_n) \, g(|z_i^H - z_j^L|, \sigma_z)}{\sum f(\hat{x}_i, x_j) \, g(|n_i^H - n_j^L|, \sigma_n) \, g(|z_i^H - z_j^L|, \sigma_z)}$$



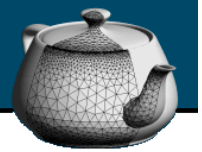
Large σ_z, small σ_n

Small σ_z, small σ_n



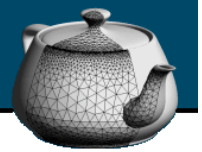
Overview

- Geometry-Aware Resizing
- Fine-Grained Resizing
- Automatic Framerate Control
- Results and Demo
- Discussions and Conclusion



Fine-Grained Resizing

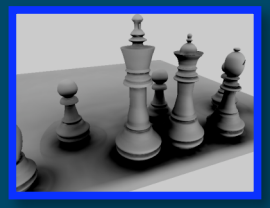
- Resize only expensive & spatially smooth computations
- Break up the original shader
  - Expensive & spatially smooth computation: 1st pass (at low-res)
  - Inexpensive / spatially high-freq computation: 2nd pass (at full-res)



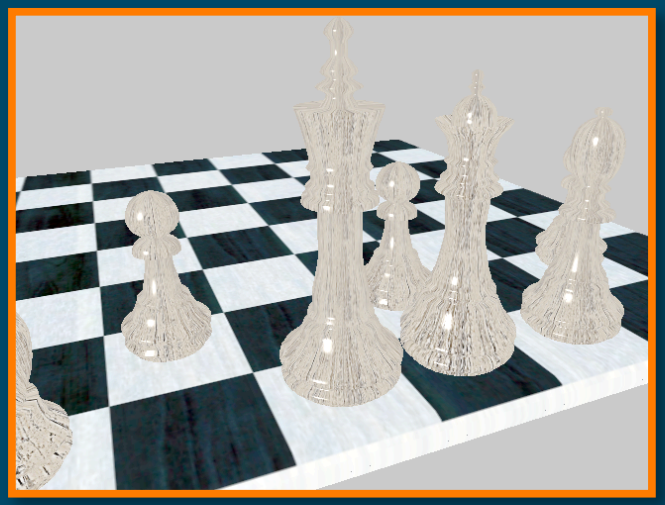
Fine-Grained Resizing

1st pass

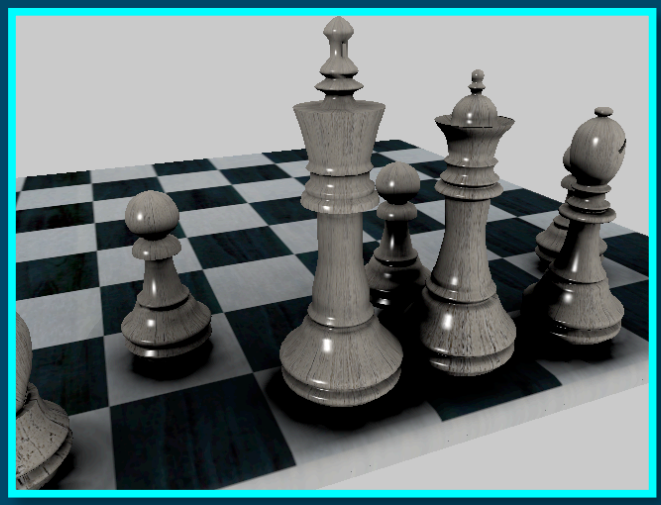
2nd pass

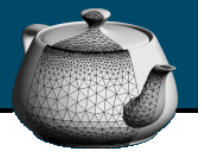


+



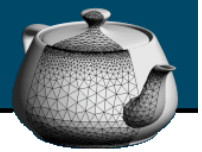
=





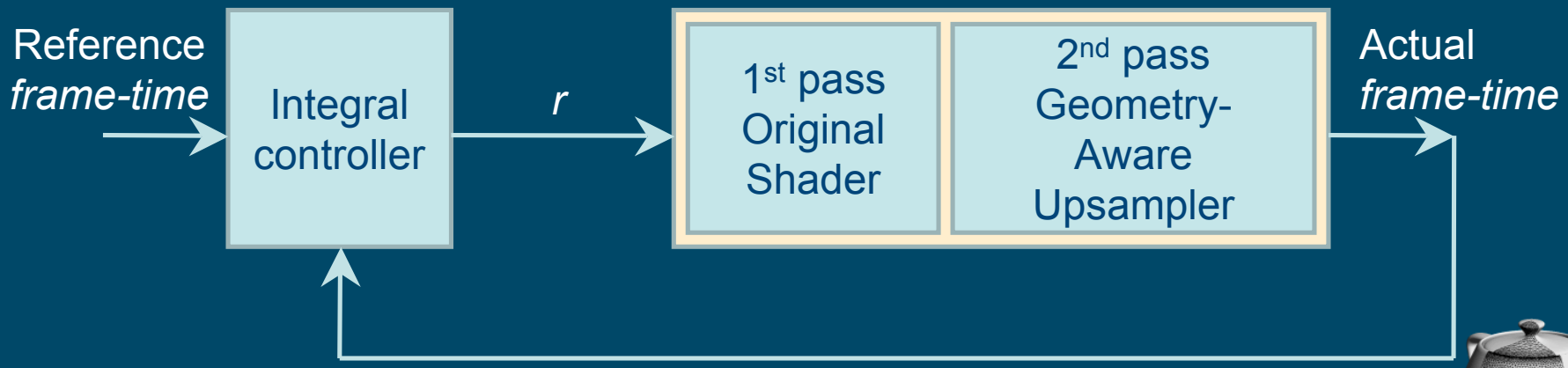
Overview

- Geometry-Aware Resizing
- Fine-Grained Resizing
- **Automatic Framerate Control**
- Results and Demo
- Discussions and Conclusion



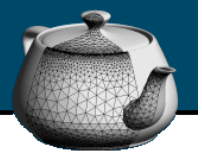
Automatic Framerate Control

- **Dynamically select resizing factor *r* to maintain a *constant* framerate**
- **Use a feedback control mechanism**
  - **Input: previous *frame-time***
  - **Output: *r***
  - **Integral controller**

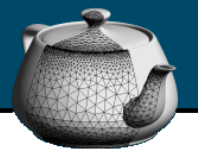


Overview

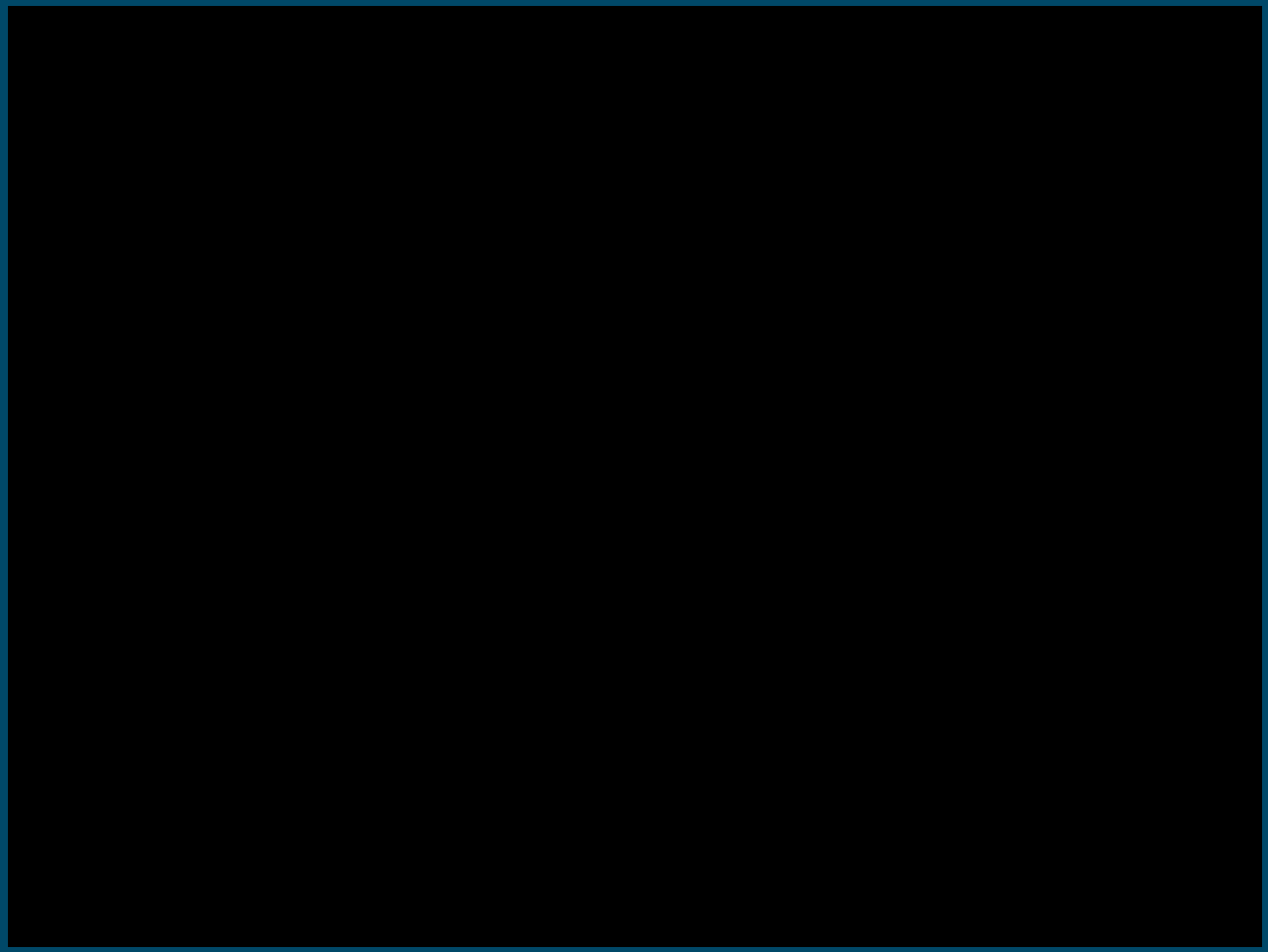
- Geometry-Aware Resizing
- Fine-Grained Resizing
- Automatic Framerate Control
- Results and Demos
- Discussions and Conclusion

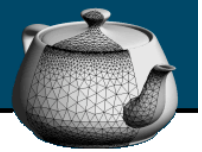


Results – Car

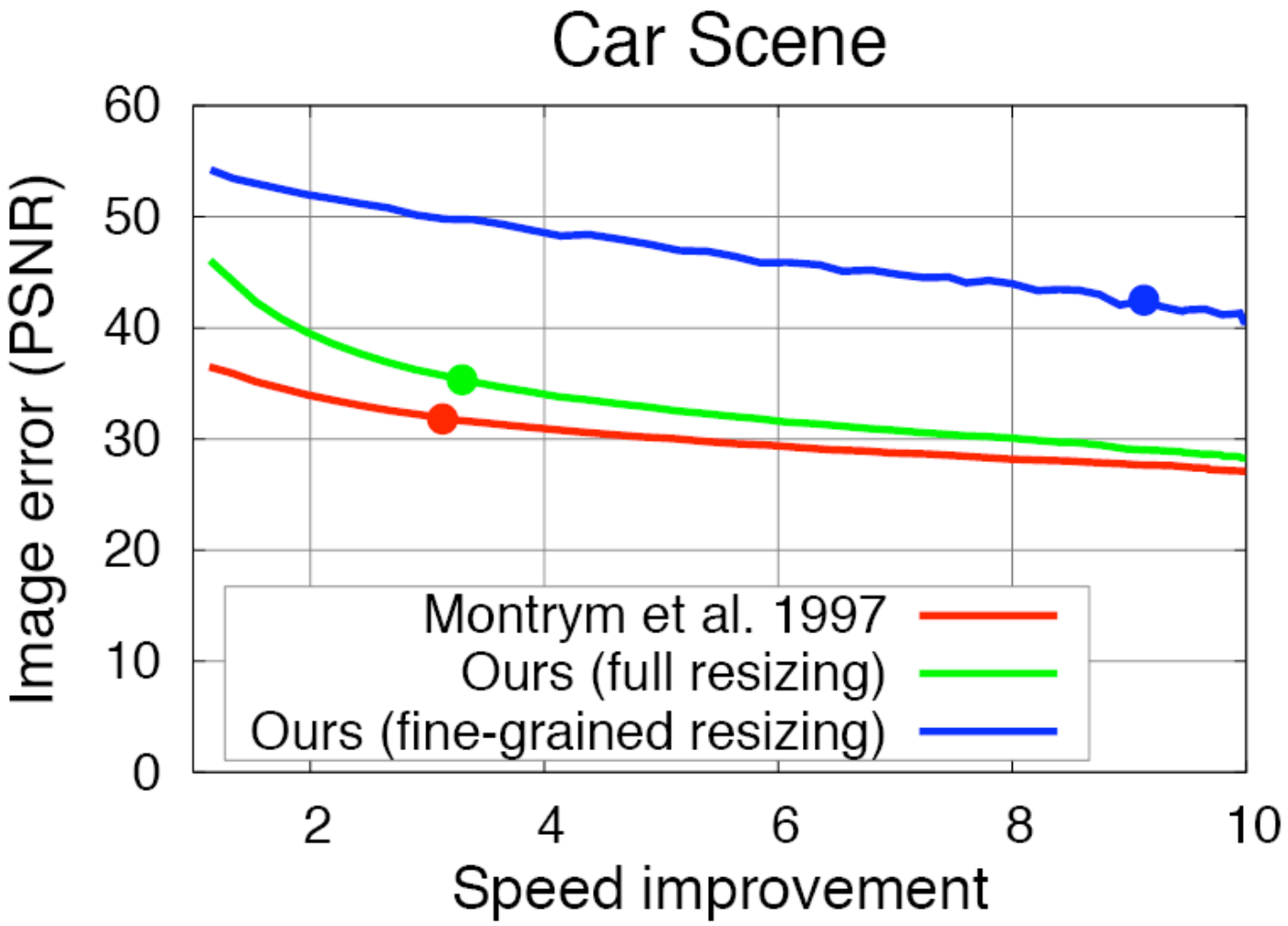


Results – Car

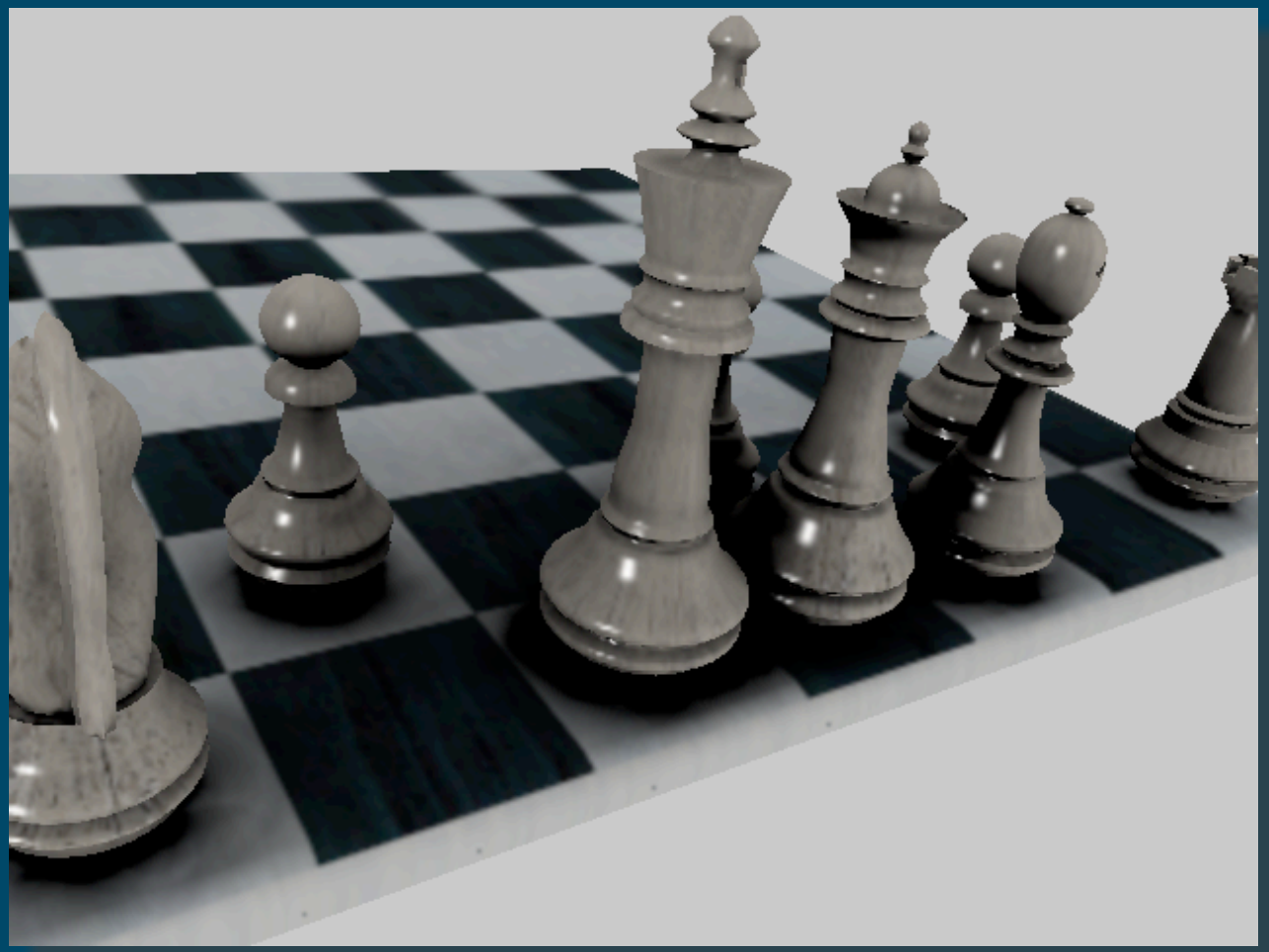


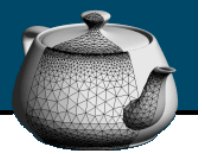


Results – Car (con't)

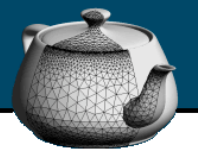


Results – Chess

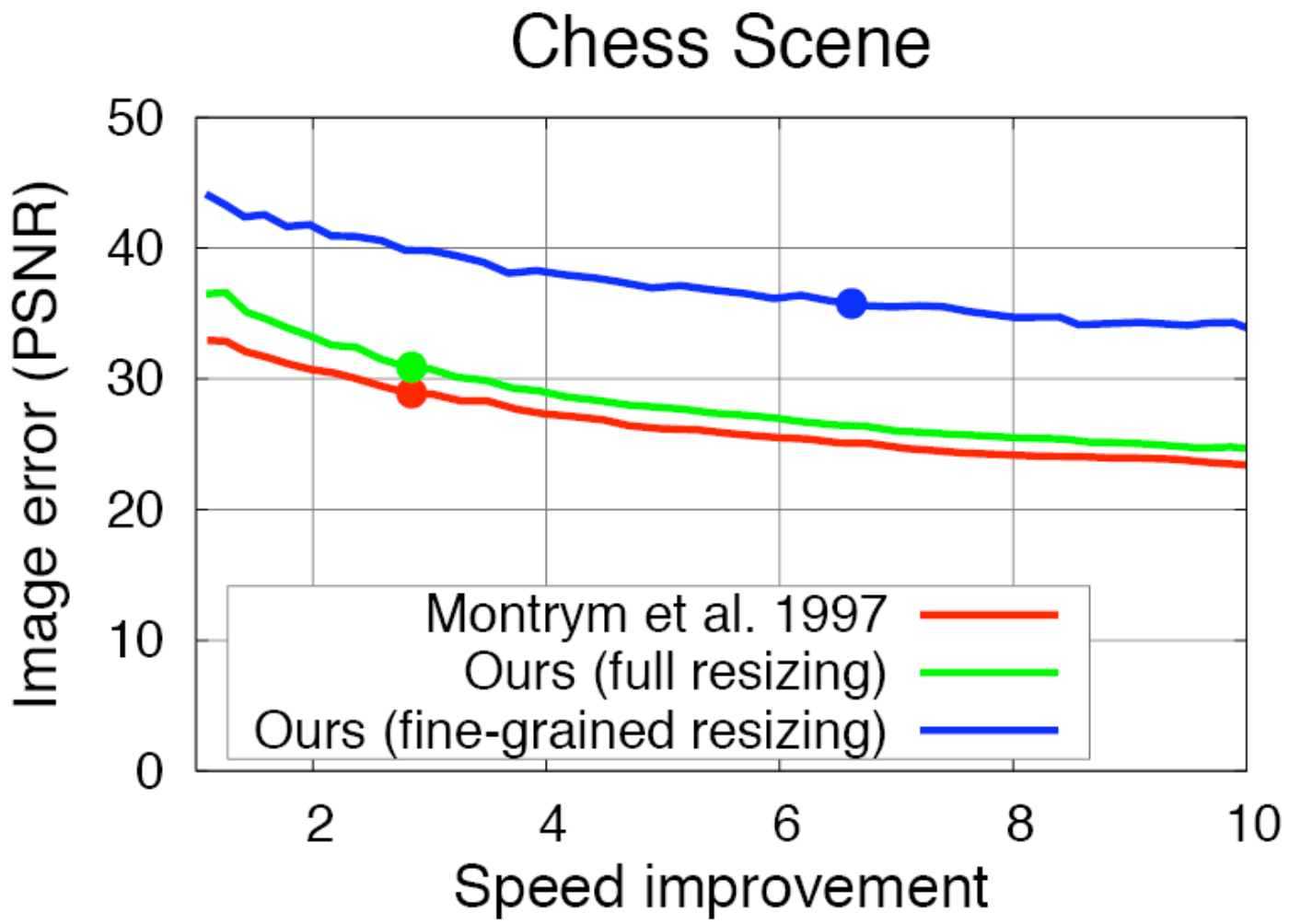




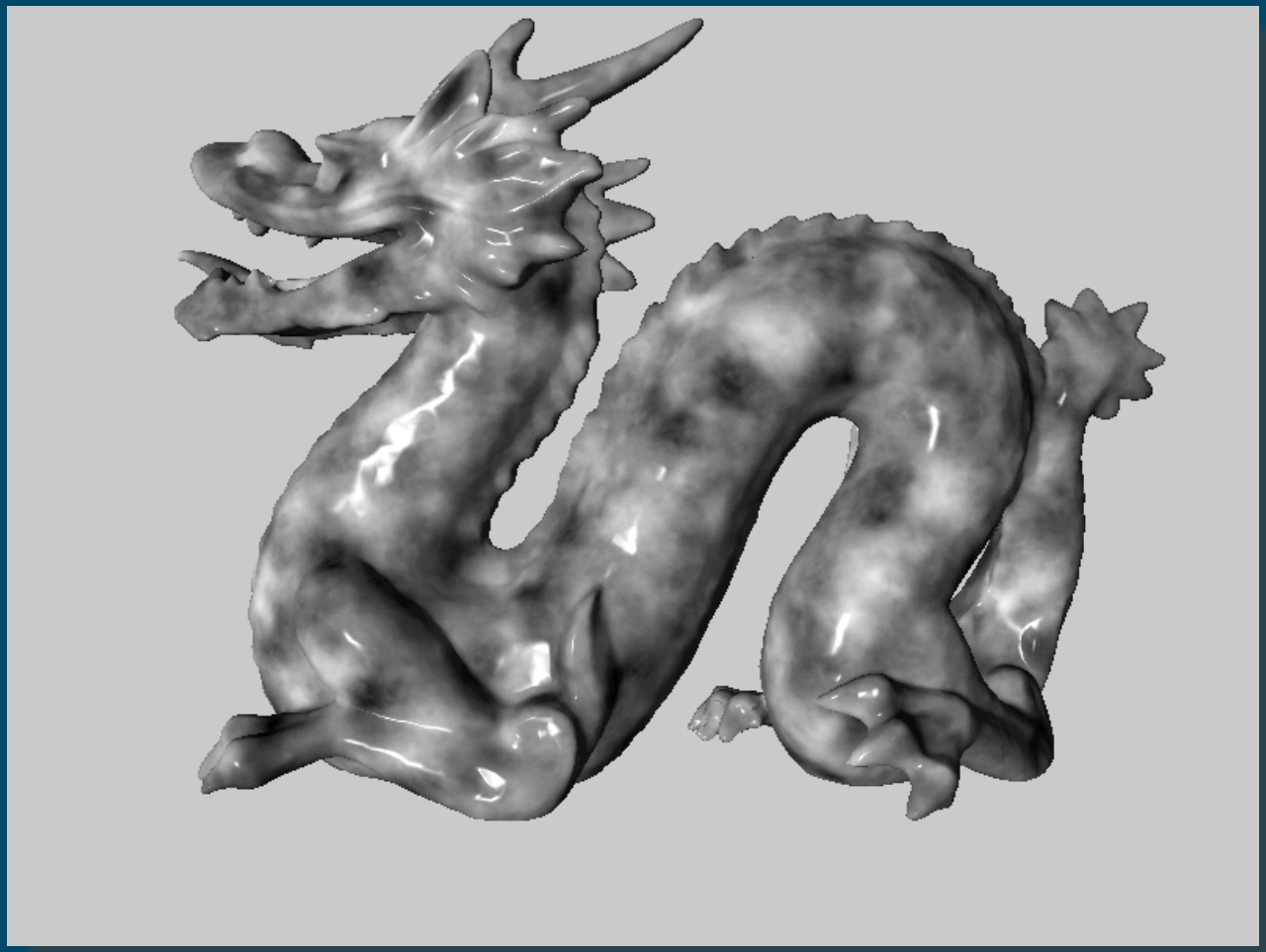
Results – Chess

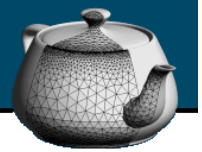


Results – Chess (con't)

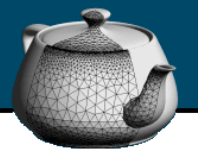


Results – Dragon

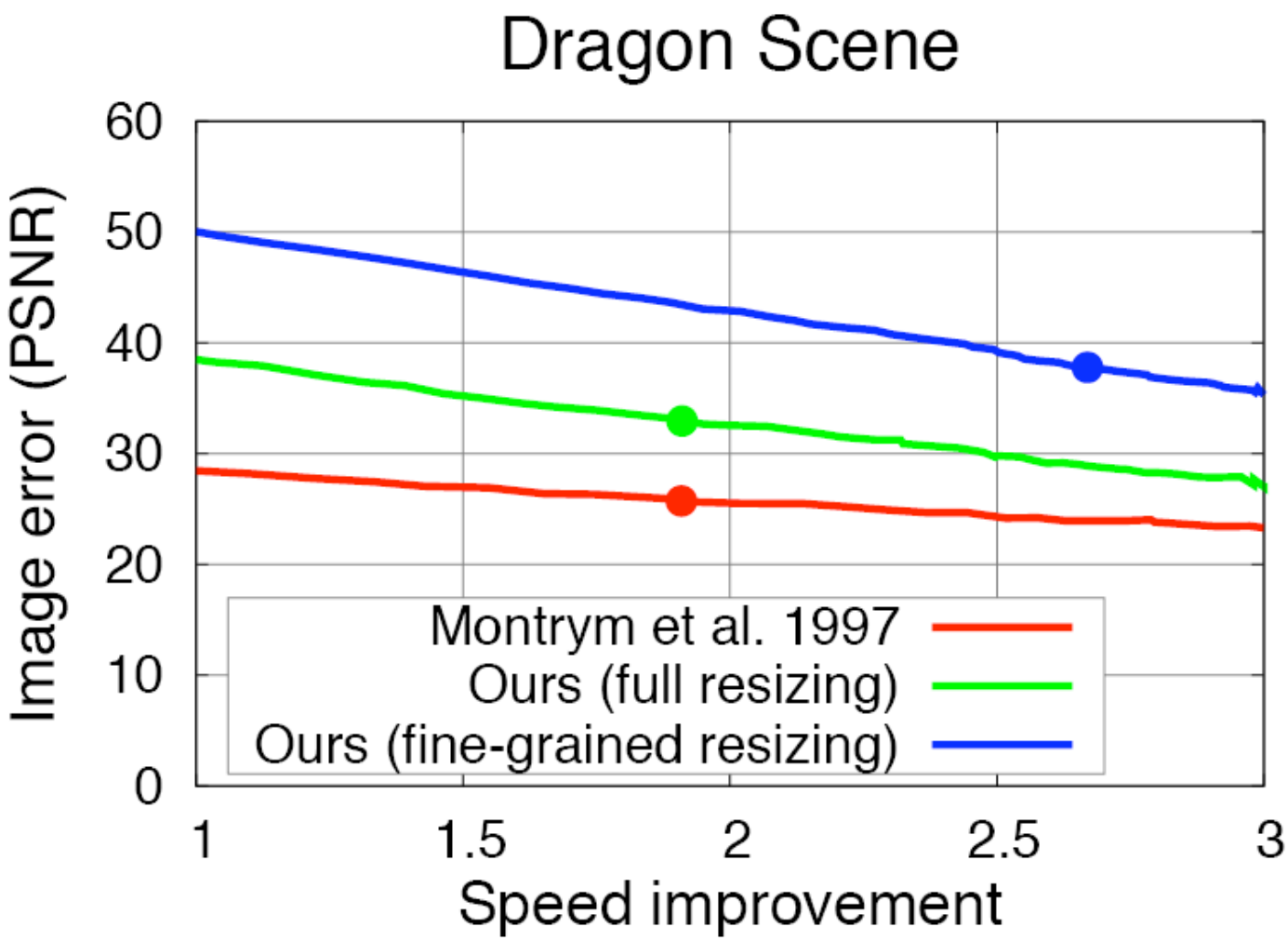




Results – Dragon

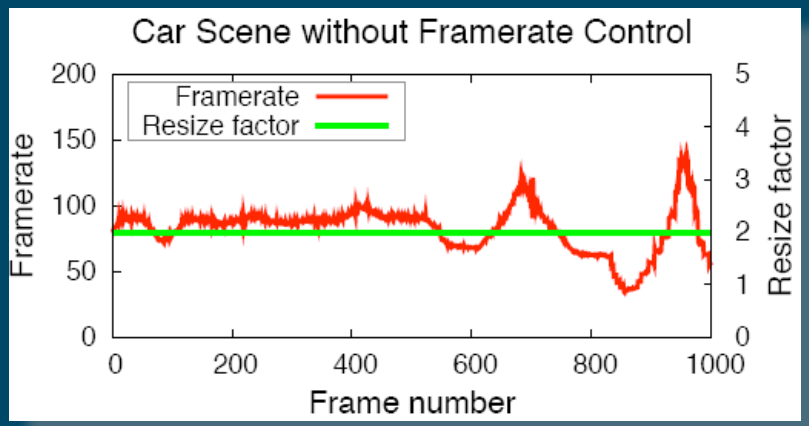


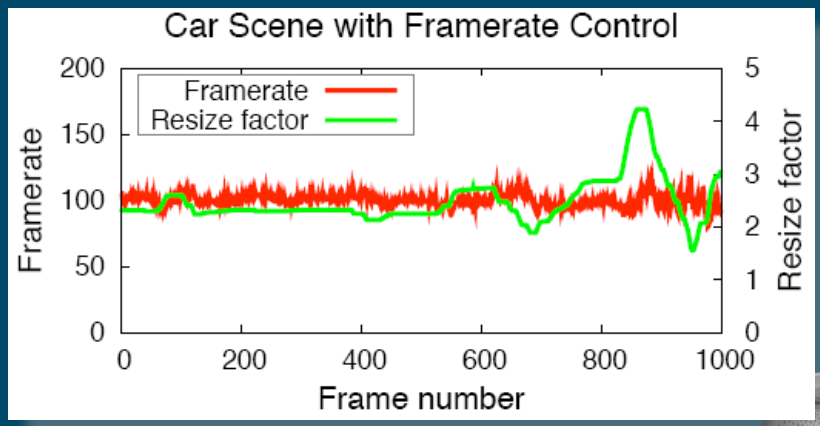
Results – Dragon (con't)



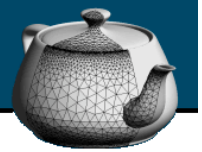
AFC results

- **Experimental data:**
  - **Over 1000 frames**
  - **Various outside disturbances**
    - **View changes**
    - **Screen coverage changes**
    - **Shader workload changes**



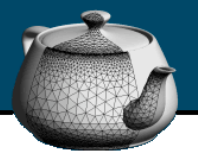


AFC results



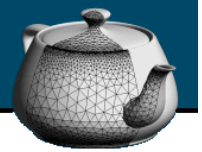
Overview

- Geometry-Aware Resizing
- Fine-Grained Resizing
- Automatic Framerate Control
- Results and Demo
- Discussions and Conclusion



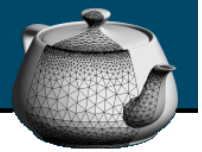
Limitations

- Resizing high frequency signal
  - Popping and flickering artifacts (aliasing)
- Undersampled fine geometry
  - Missing details around regions with high depth/normal complexities
  - Recompute missing samples in a 3rd pass?
- Added geometry processing overhead



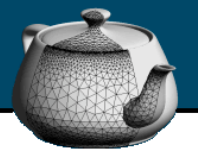
Practical Advantages

- Multiple shader / objects
  - Sharing the same resized buffer
  - Sharing the reconstruction pass
  - Allow unified AFC
- Easy to apply
  - Mainly an added reconstruction pass



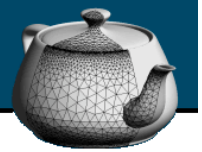
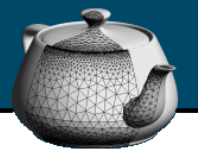
Conclusion

- A general approach for reducing shading costs
- Respect geometric discontinuities better than conventional resizing
- Allow continuous adjustment of error/performance tradeoff
- Automatic framerate control
- Straightforward to incorporate into existing systems



Future Work

- Multi-resolution resizing
- Automated selection of resized elements
- Resize for super-sample anti-aliasing
- Obtain a Bosnia-Herzegovina visa ☺



Questions?