

# 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**

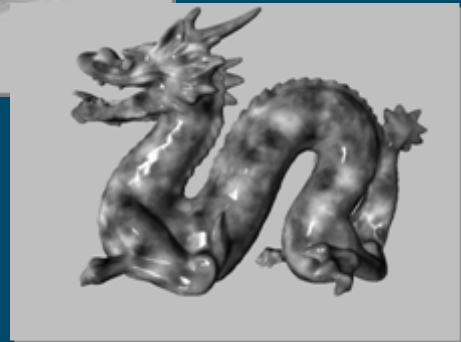
- **Soft shadows**  
27fps



- **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 (1<sup>st</sup> pass), then upsample to target resolution (2<sup>nd</sup> pass). [Montrym97]
  - # of original pixel shader invocation is reduced ( $\propto 1/r^2$ )
  - Blurs geometric discontinuities

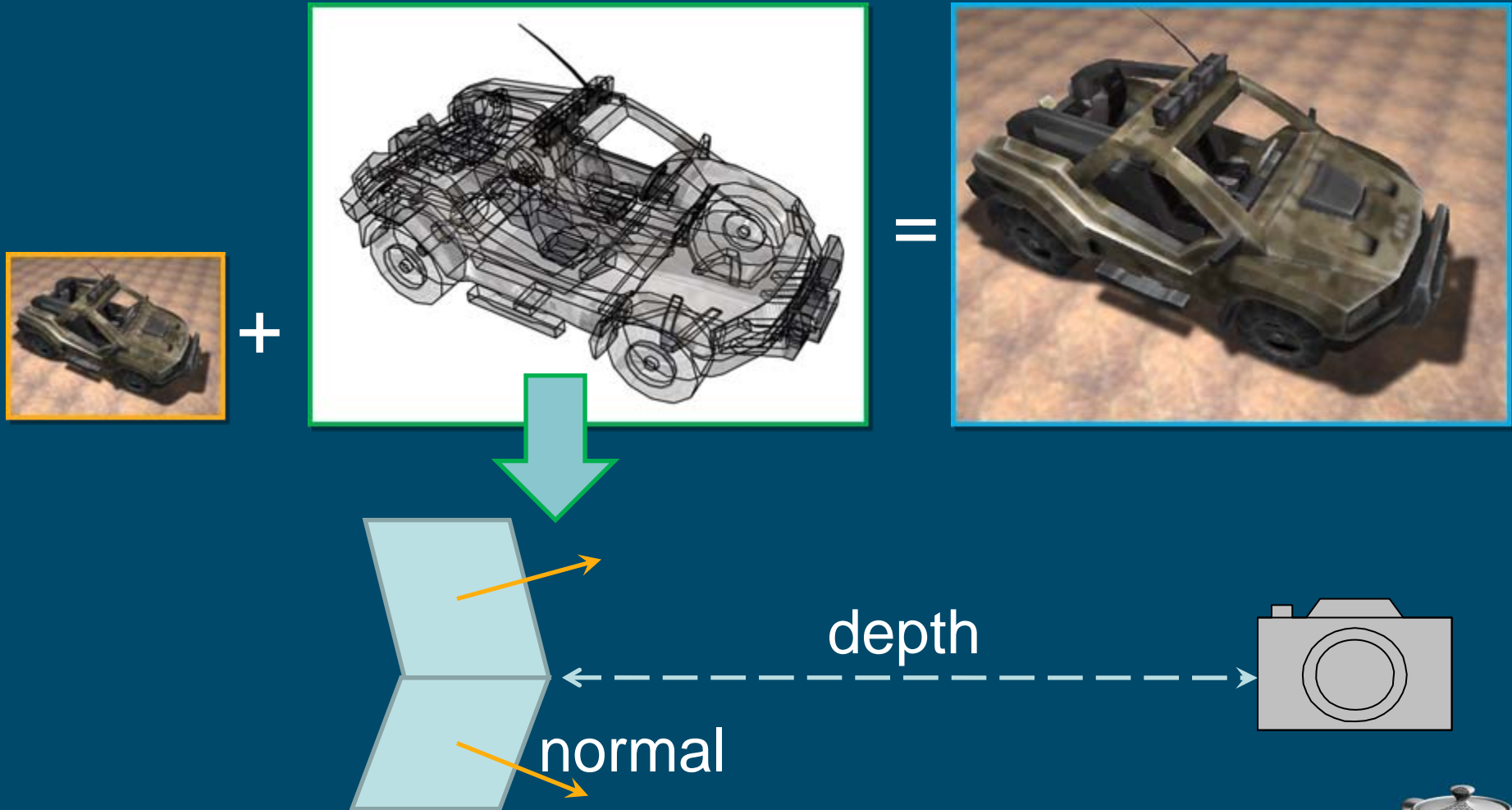
1<sup>st</sup> pass  
Original shader



2<sup>nd</sup> pass  
Upsample



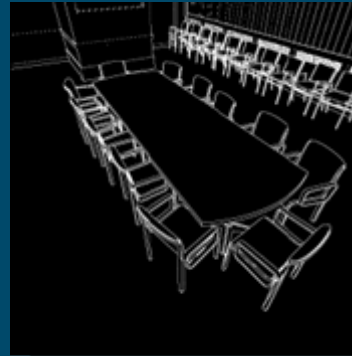
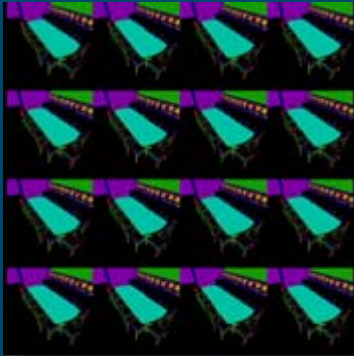
# Geometry-Aware?





# Related Work

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



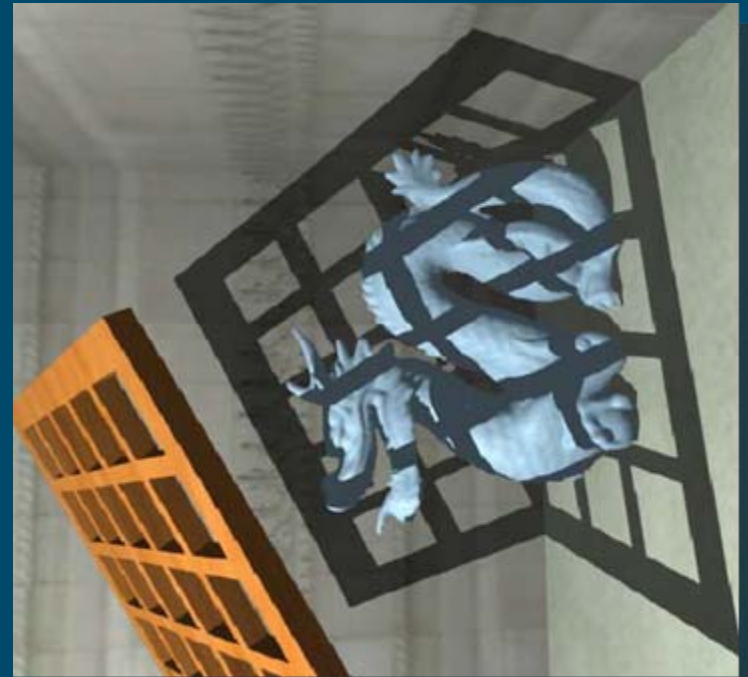
- **Image-based proxy accumulation** [Sloan07]



# 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**



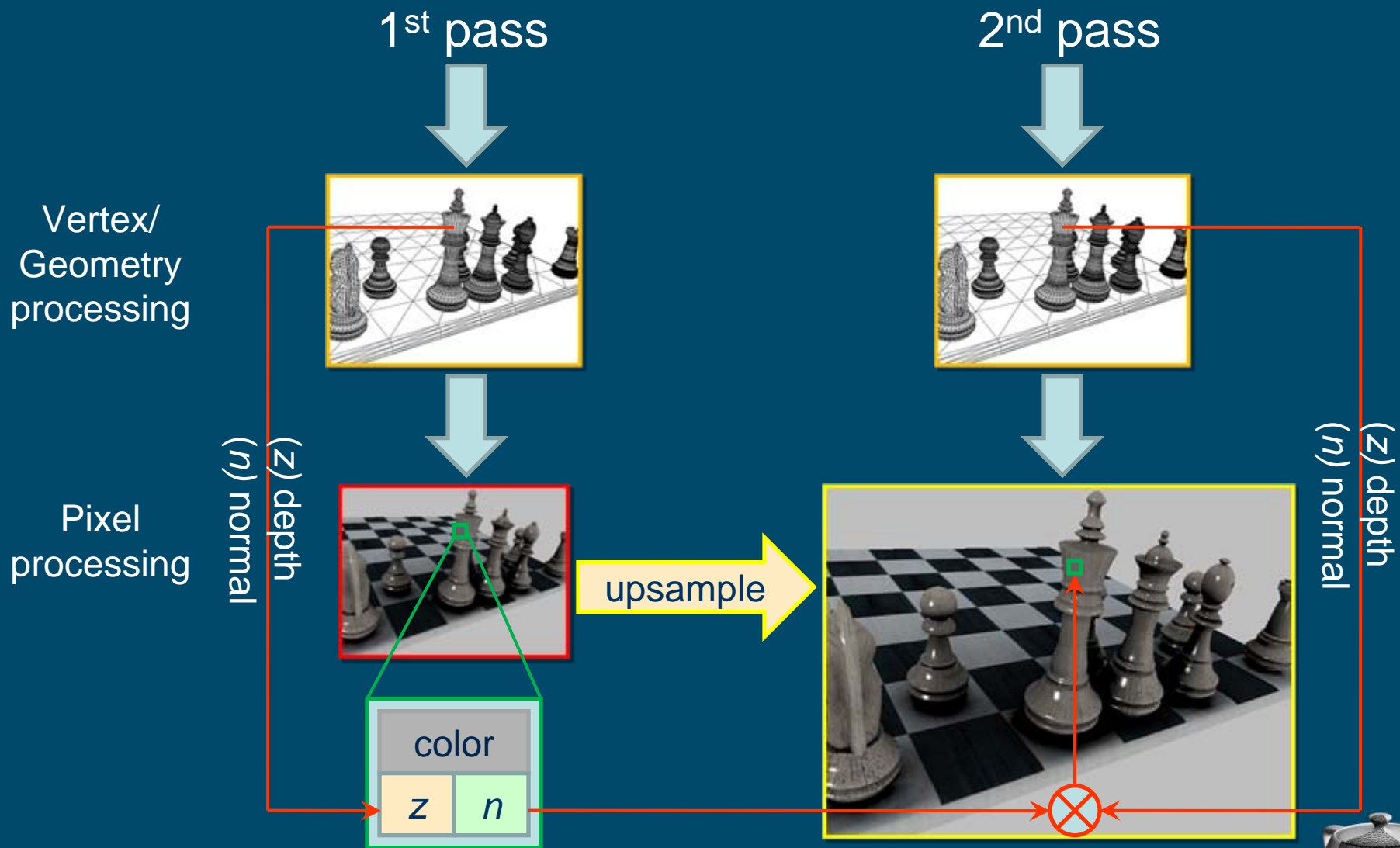


# Our Approach

- **Geometry-Aware Resizing**
  - Upsample according to geometric similarities between lo-res and hi-res buffers
  - Two-pass technique
    - 1<sup>st</sup> pass: Render geometry with the original pixel shader on low-res buffer, store *geometric info* (normal & depth) + color
    - 2<sup>nd</sup> pass: Render geometry at full resolution and use *geometry-aware kernel* to reconstruct the shading from the lo-res buffer

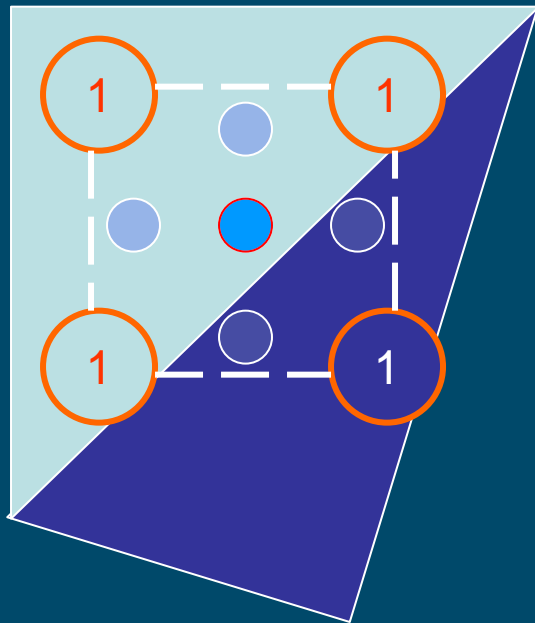


# Geometry-Aware Resizing

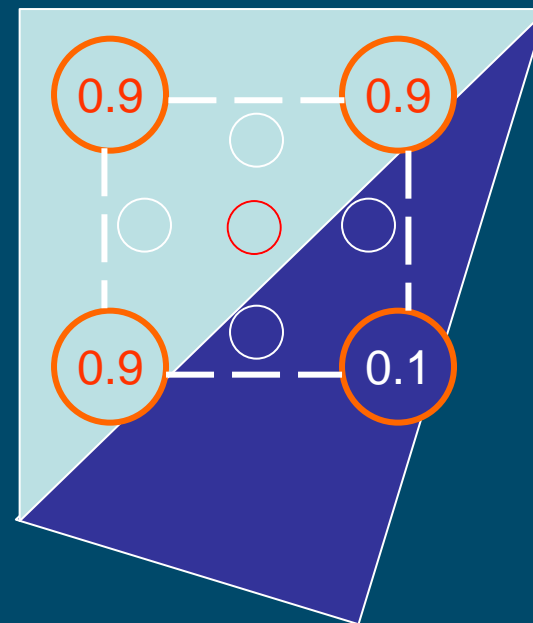


# Geometry-Aware Reconstruction

Bilinear



Bilateral



**Weight samples based on geometric similarity**



# Joint Bilateral Filter

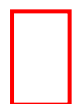
$$c_i^H = \frac{\sum c_j^L f(\hat{x}_i, x_j) g(|n_i^H - n_j^L|, \sigma_n) g(|z_i^H - z_j^L|, \sigma_z)}{\sum f(\hat{x}_i, x_j) g(|n_i^H - n_j^L|, \sigma_n) g(|z_i^H - z_j^L|, \sigma_z)}$$

- 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



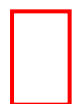
Spatial filter: bilinear / biquadratic / bicubic / Gaussian



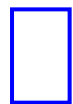


# Joint Bilateral Filter

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



Color sample  $j$  from the low-res buffer

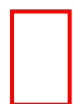


Range filter 1: Gaussian of the *normal* distance



# 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

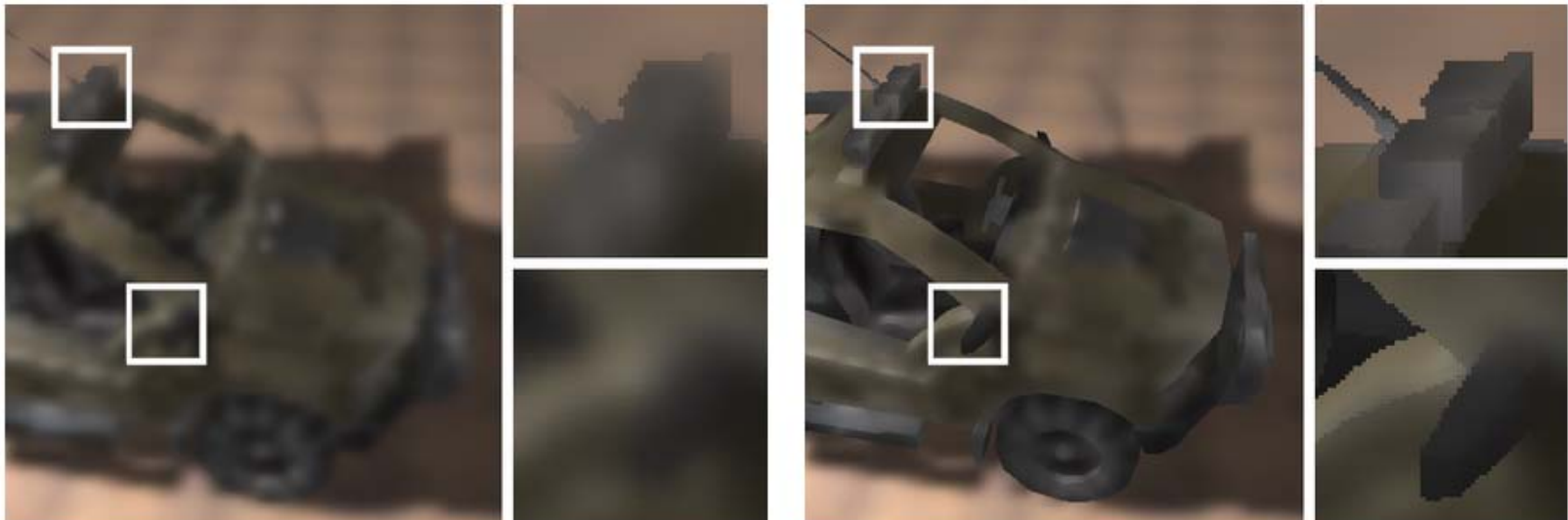


Range filter 2: Gaussian of the **depth** distance



# Joint Bilateral Filter

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



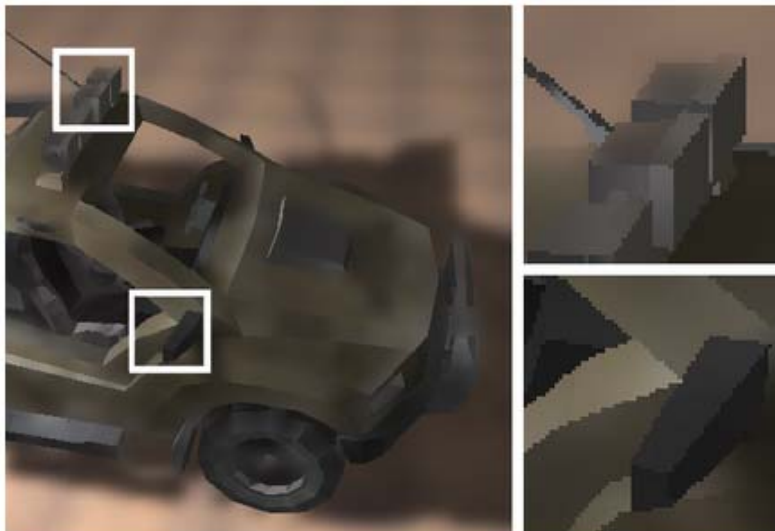
Large  $\sigma_z$ , large  $\sigma_n$

Small  $\sigma_z$ , large  $\sigma_n$



# 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  $\sigma_z$ , small  $\sigma_n$



Small  $\sigma_z$ , small  $\sigma_n$



# 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:  
1<sup>st</sup> pass (at low-res)
  - Inexpensive / spatially high-freq computation:  
2<sup>nd</sup> pass (at full-res)



# Fine-Grained Resizing

1<sup>st</sup> pass

2<sup>nd</sup> pass



+

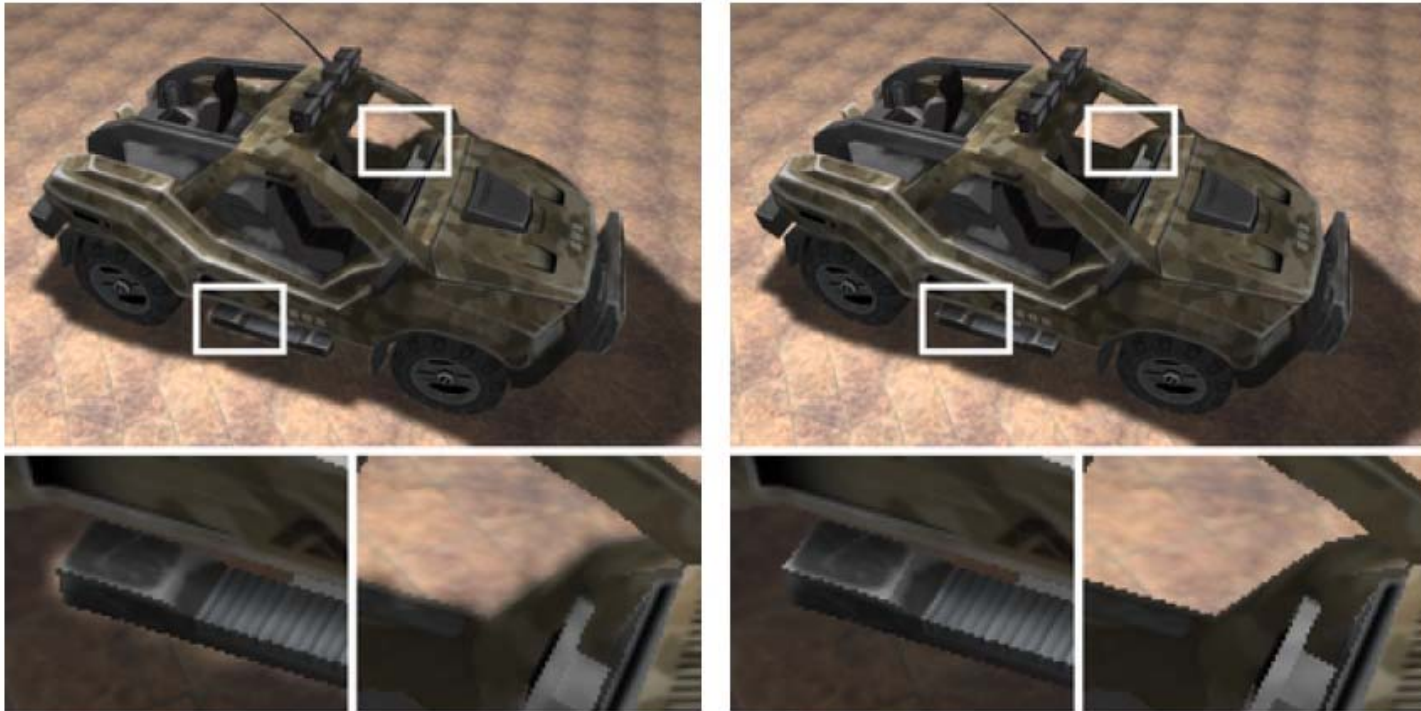


=



# Comparison: Bilinear vs. Bilateral

- Fine-grained resizing + Bilinear upsample?



Bilinear

Geometry-Aware



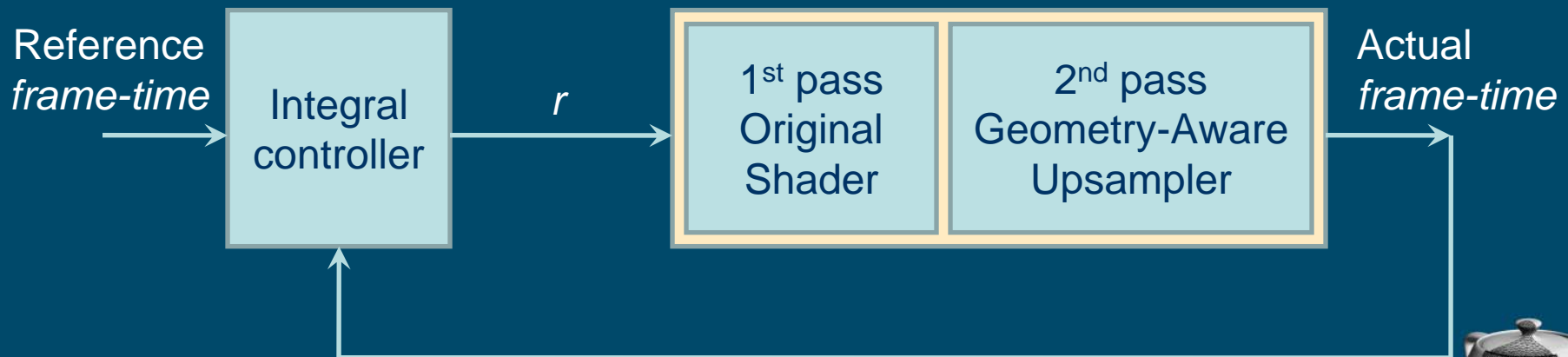
# Overview

- 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





# Controller Formulation

$$t \approx Kp + C$$

1. Pixel processing time  $\propto$  # of pixels
2. Pixel-bound

$$p \propto wh/r^2$$

Constant screen coverage

$$t \approx K'(1/r^2) + C$$

$$\Delta t \approx K' \Delta(1/r^2)$$



# AFC implementation

- Limit the range of  $\Delta t$ ,  $\Delta r$  and  $r$
- Experimentally determine  $K'$  with the maximum screen coverage



# Overview

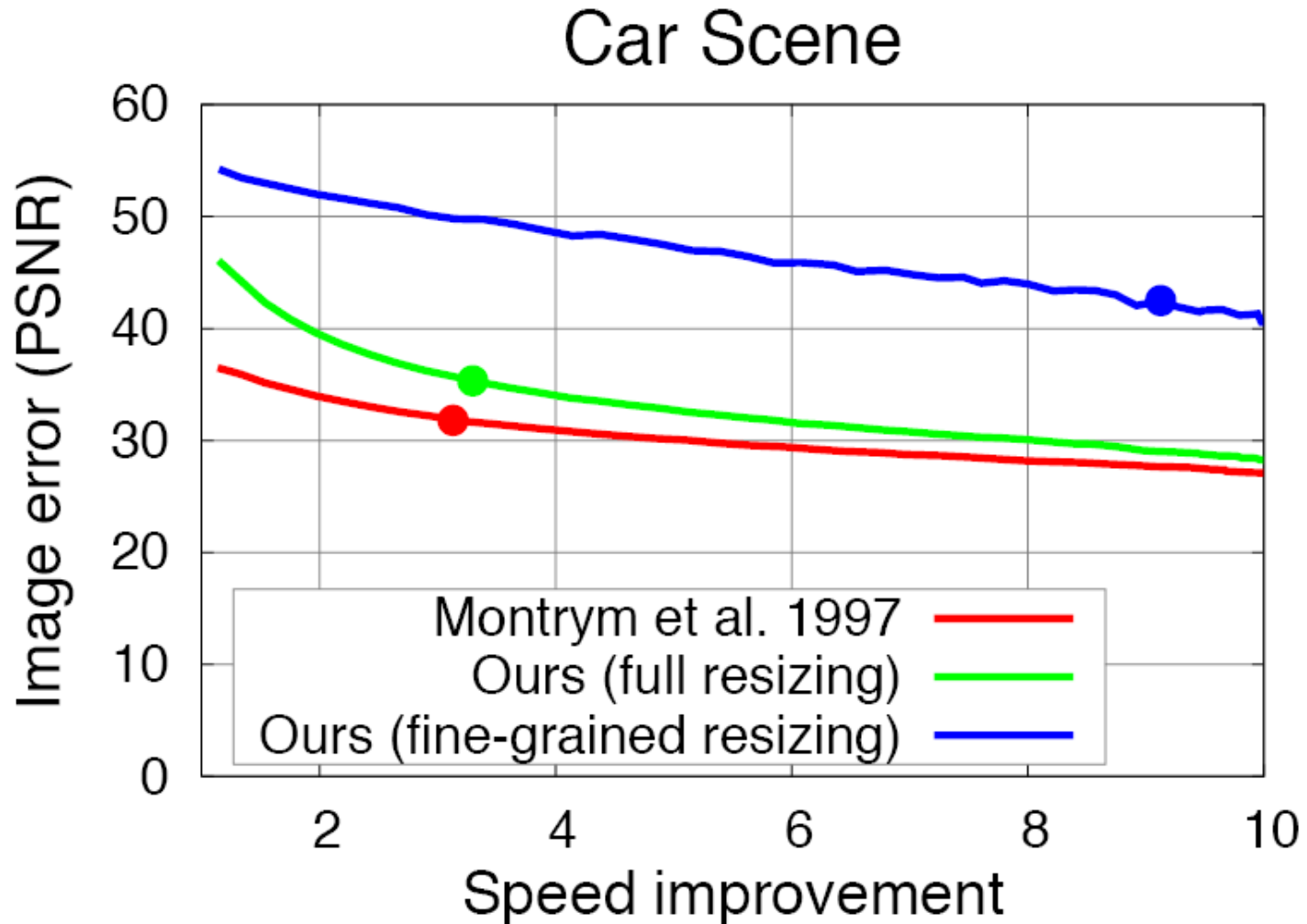
- Geometry-Aware Resizing
- Fine-Grained Resizing
- Automatic Framerate Control
- **Results and Demos**
- Discussions and Conclusion



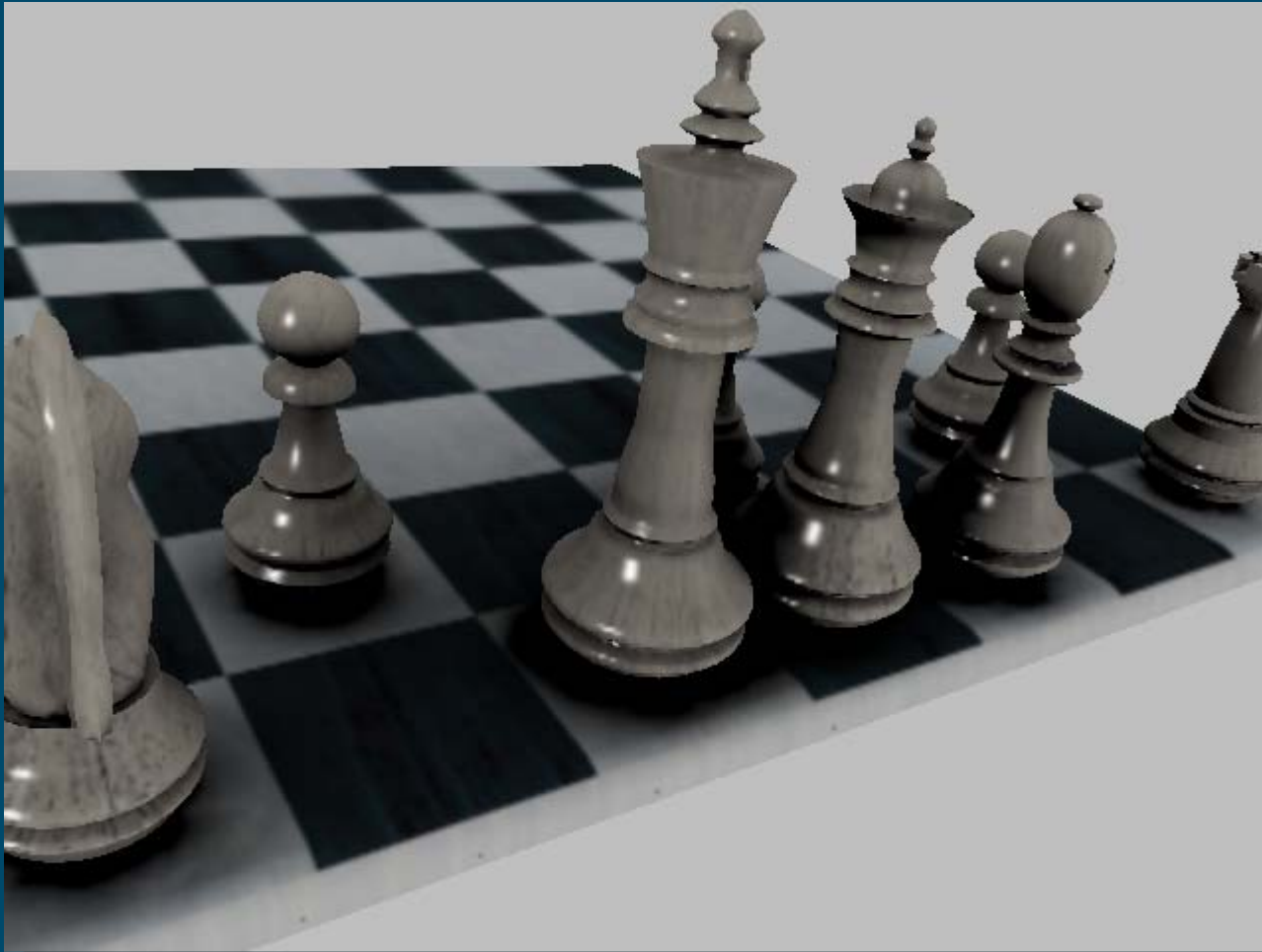
# Results – Car



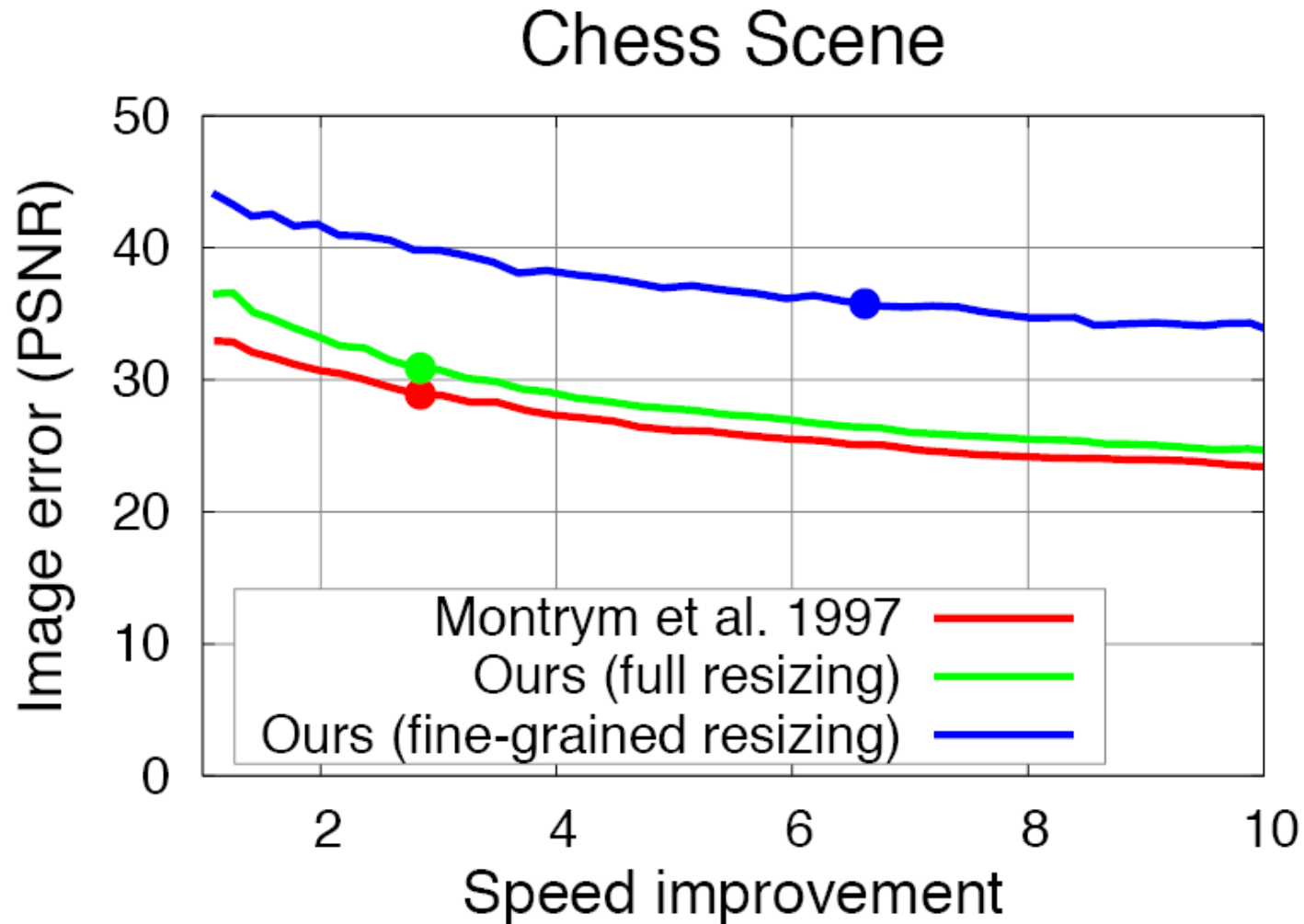
# Results – Car (con't)



# Results – Chess

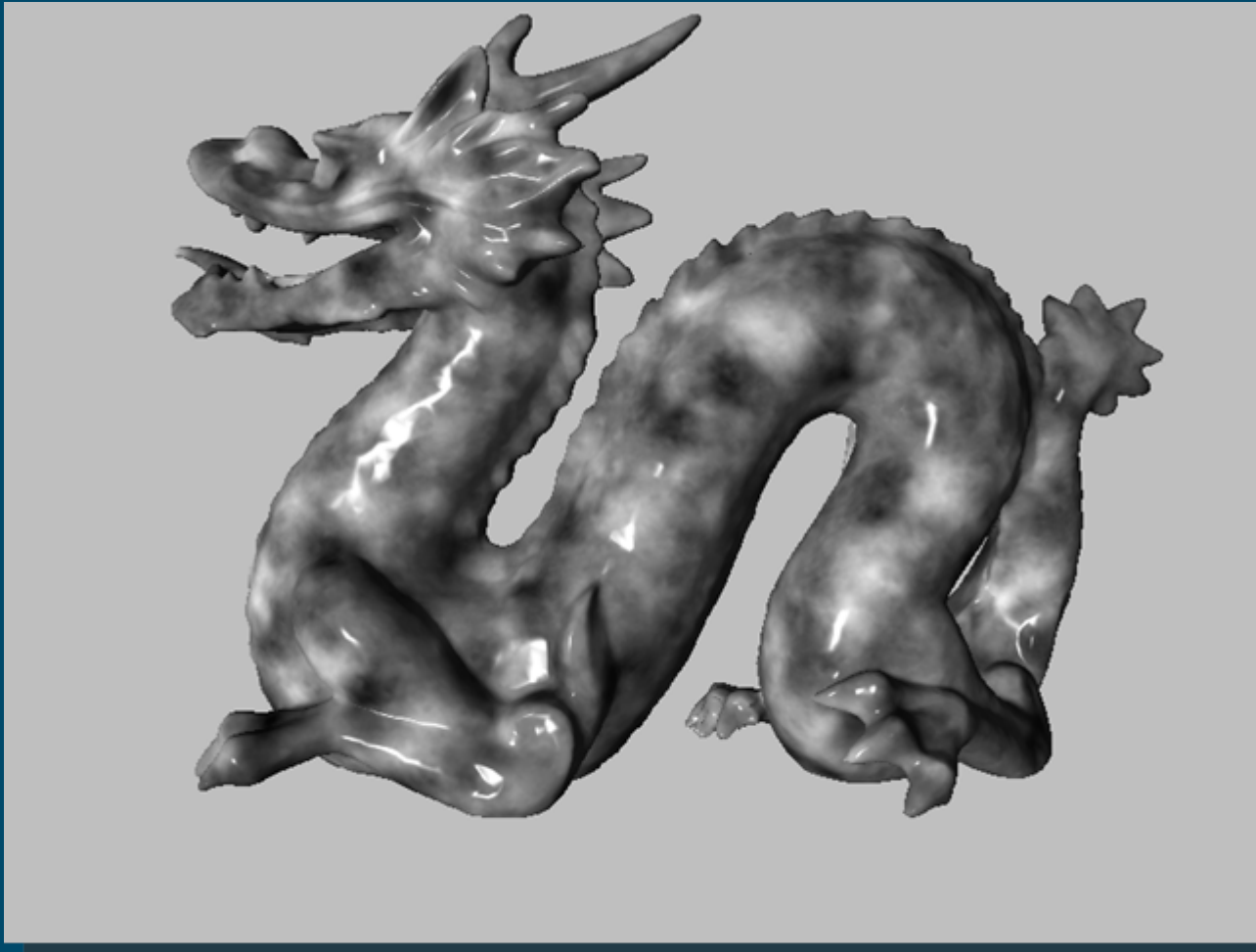


# Results – Chess (con't)

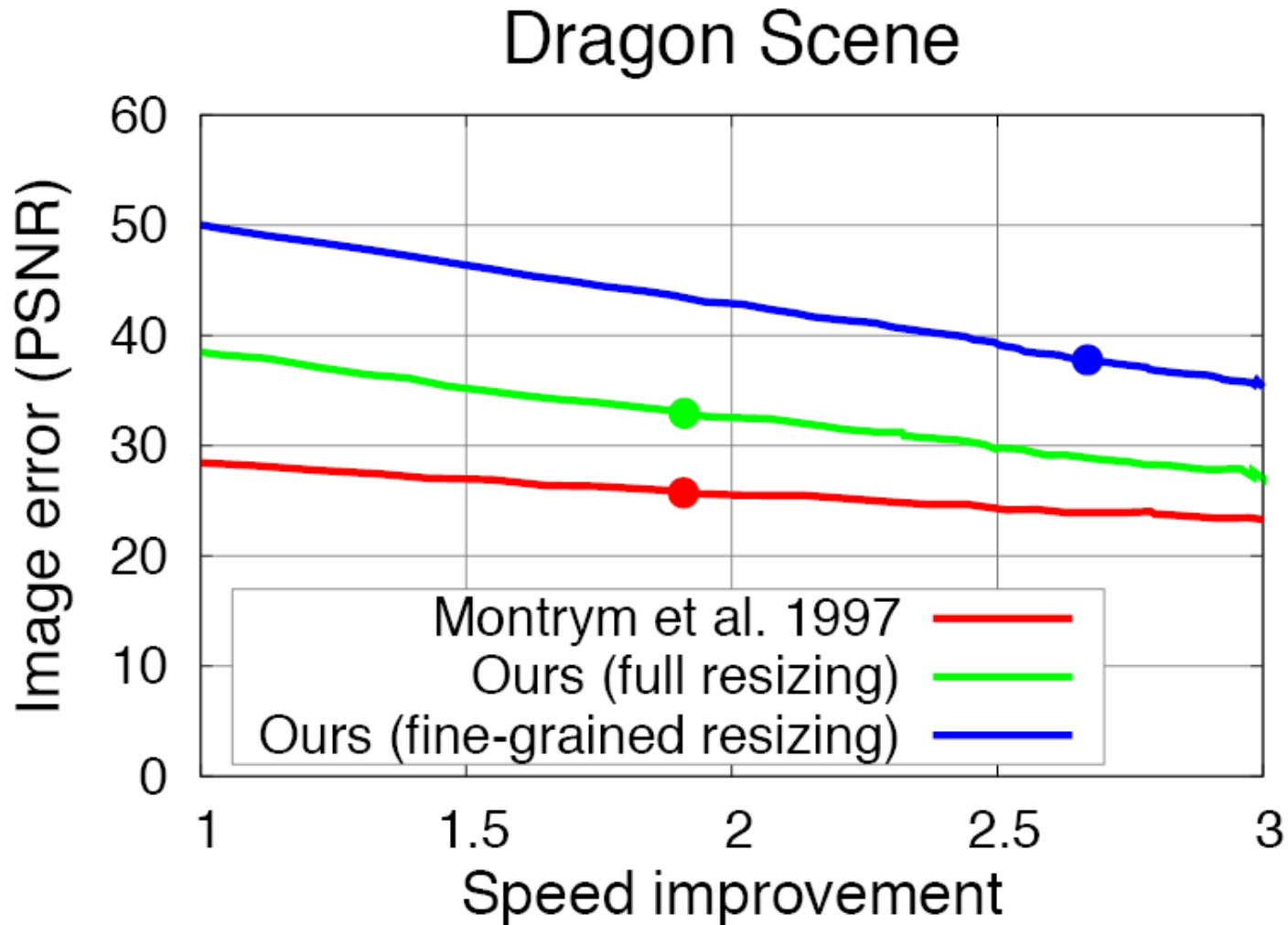




# Results – Dragon

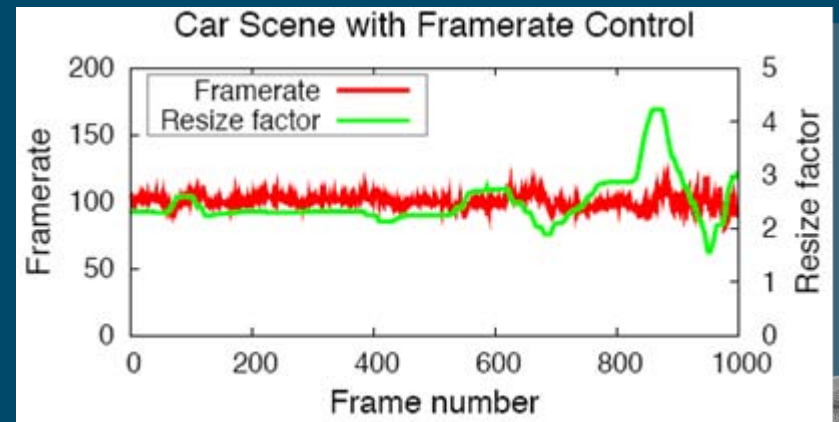
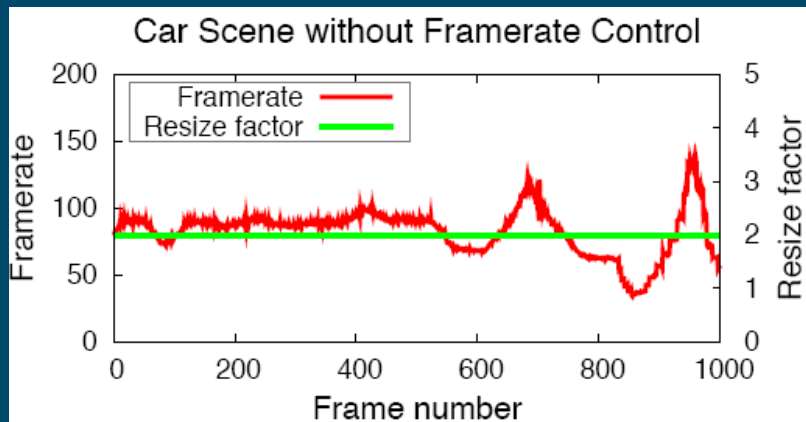


# Results – Dragon (con't)



# AFC results

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



# 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 3<sup>rd</sup> 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?

