

Interactive appearance editing in RGB-D images

Stephan Bergmann[1], Tobias Ritschel[2], Carsten Dachsbacher[1]

[1]Karlsruhe Institute of Technology

[2]Max Planck Institute for Informatics

Motivation & Goals

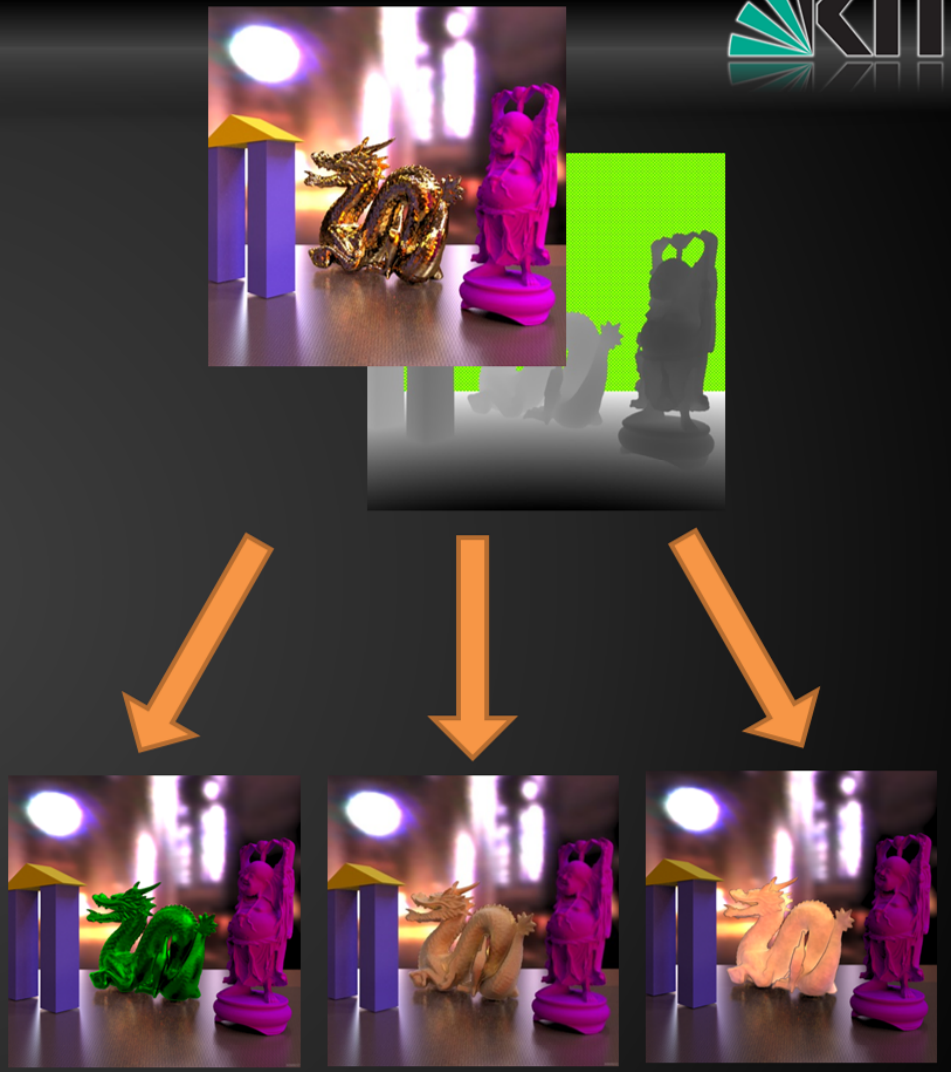
- Ubiquitous digital images
- Convincing editing is difficult & not easy to use

Motivation & Goals

- Ubiquitous digital images
- Convincing editing is difficult & not easy to use

Goals

- Change material appearance in photographs
  - color, roughness, translucency
- Single color/depth image as input source
- Artistic tool, plausible results
- No scene complete reconstruction
- Interactive response times → intuitive use



Previous Work



[Fang and Hart 2004]



[Khan et al. 2006]

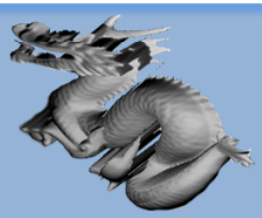
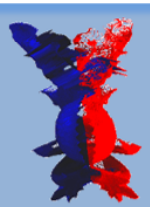


[Carroll et al. 2011]

Overview

Preparatory steps

Data acquisition → Back side reconstruction → Scene EM generation

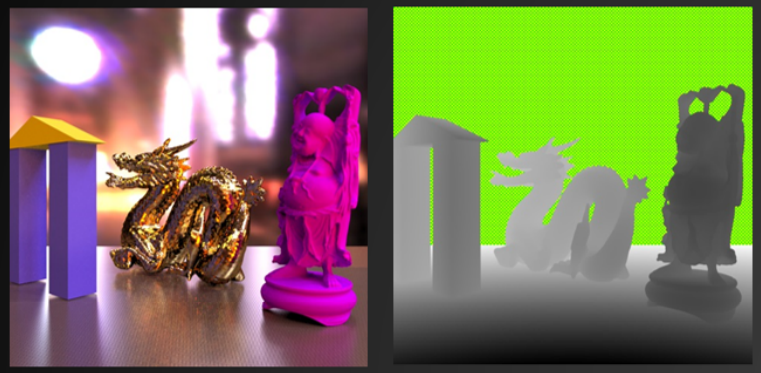





Editing

- Reshading
- Translucency

Data acquisition and preprocessing

- Color and depth image

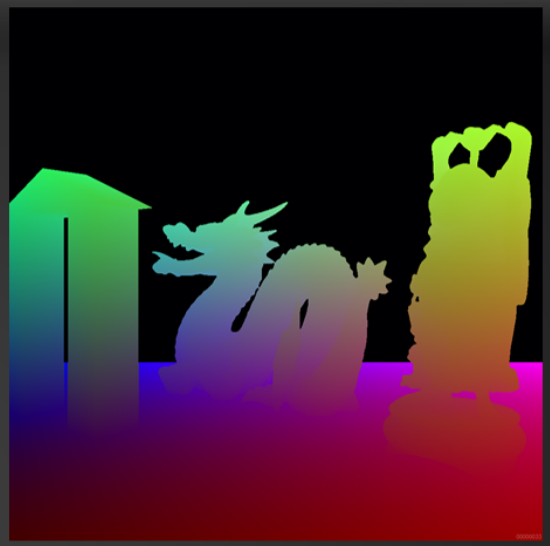


Data acquisition and preprocessing

- Color and depth image



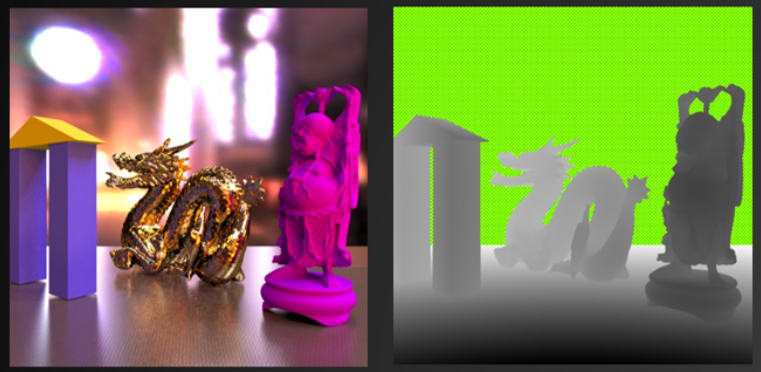




- Preprocessing
  - Determine world coordinates

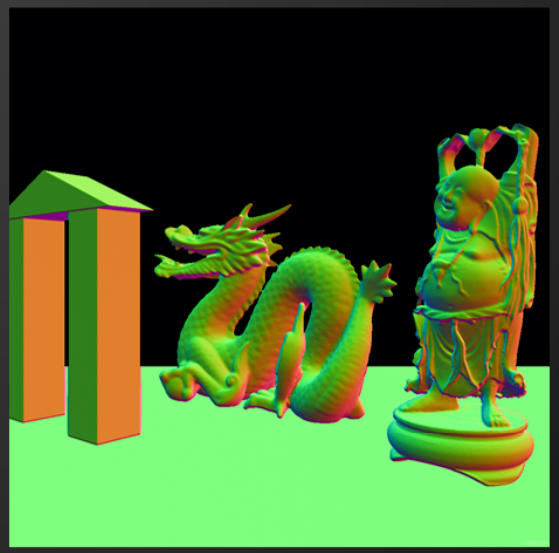
Data acquisition and preprocessing

- Color and depth image



- Preprocessing
  - Determine world coordinates
  - Determine normals





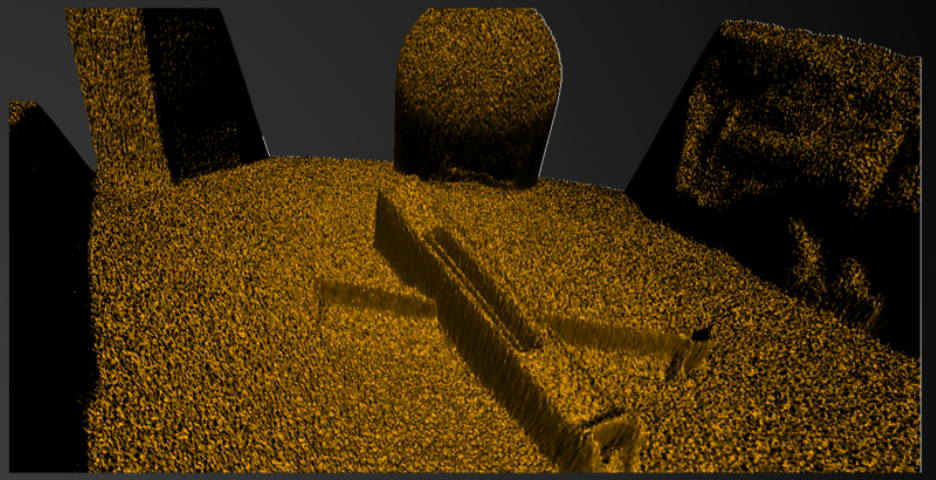
Data acquisition and preprocessing

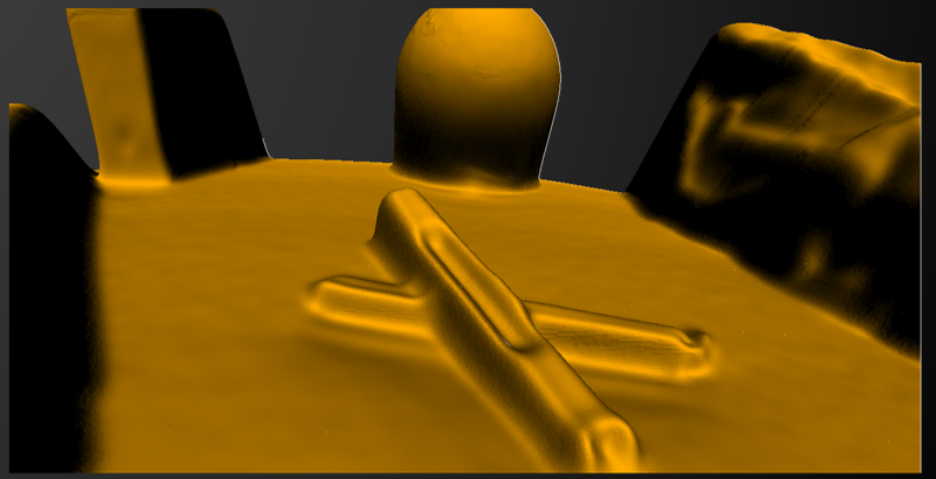
- Color and depth image





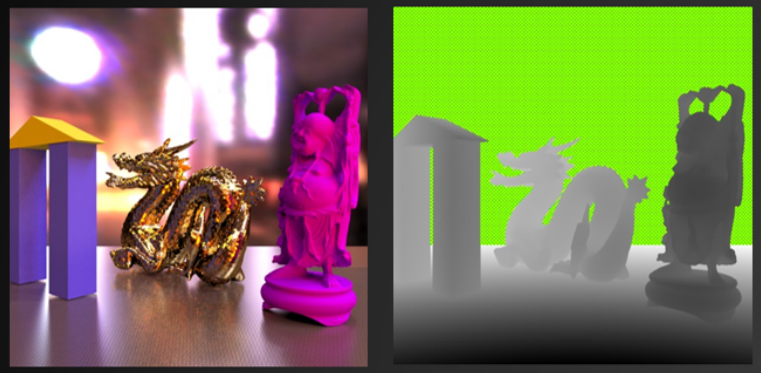
- Preprocessing
  - Determine world coordinates
  - Determine normals
  - Filter depth/normals to avoid artifacts later
    - Bilateral filter
    - [Han and Lee 2013]: High Quality Shape from a Single RGB-D Image under Uncalibrated Natural Illumination





Data acquisition and preprocessing

- Color and depth image



- Preprocessing
  - Determine world coordinates
  - Determine normals
  - Filter depth/normals to avoid artifacts later
    - Bilateral filter
    - [Han and Lee 2013]: High Quality Shape from a Single RGB-D Image under Uncalibrated Natural Illumination

- Segment labels



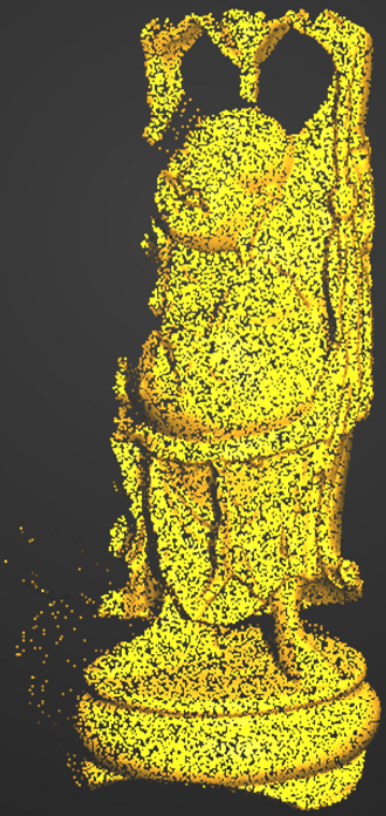
- We assume a binary label for each segment: diffuse or glossy

Back side reconstruction

- Assumption: back side is symmetrical to front side
- Steps to create back side:

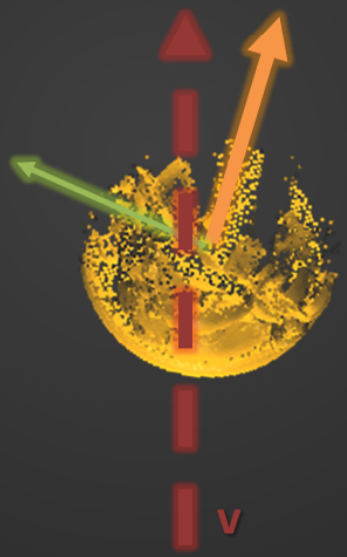
Back side reconstruction

- Assumption: back side is symmetrical to front side
- Steps to create back side:
  1. Consider point cloud of pixel centers in world coordinates



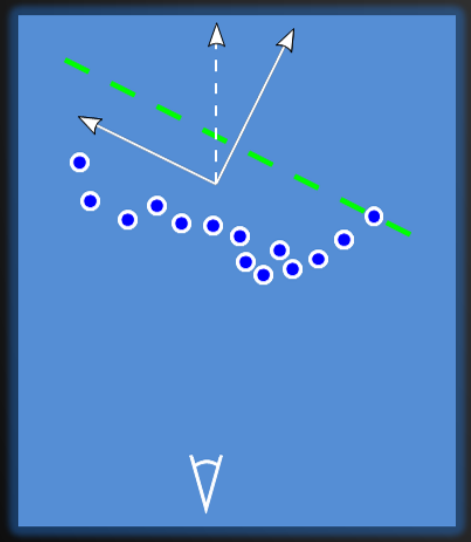
Back side reconstruction

- Assumption: back side is symmetrical to front side
- Steps to create back side:
  1. Consider point cloud of pixel centers in world coordinates
  2. Choose mirror plane normal using eigenvectors



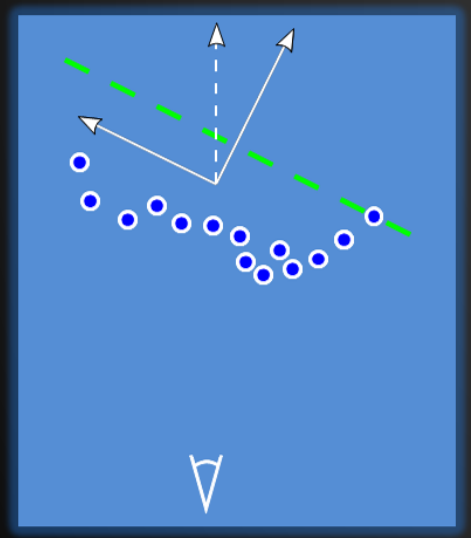
Back side reconstruction

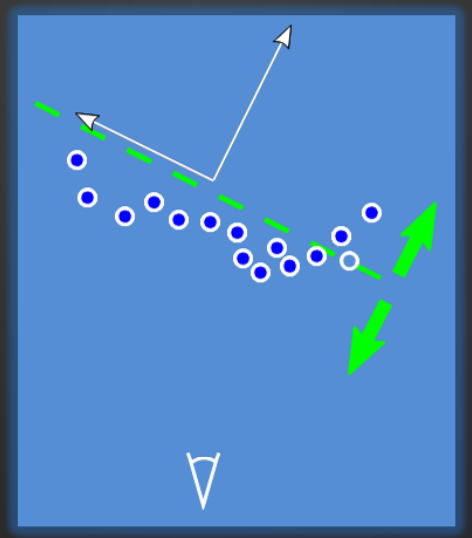
- Assumption: back side is symmetrical to front side
- Steps to create back side:
  1. Consider point cloud of pixel centers in world coordinates
  2. Choose mirror plane normal using eigenvectors



Back side reconstruction

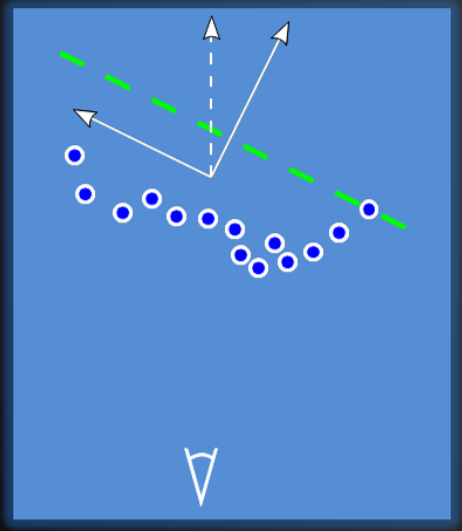
- Assumption: back side is symmetrical to front side
- Steps to create back side:
  1. Consider point cloud of pixel centers in world coordinates
  2. Choose mirror plane normal using eigenvectors
  3. Optimize plane position along that normal

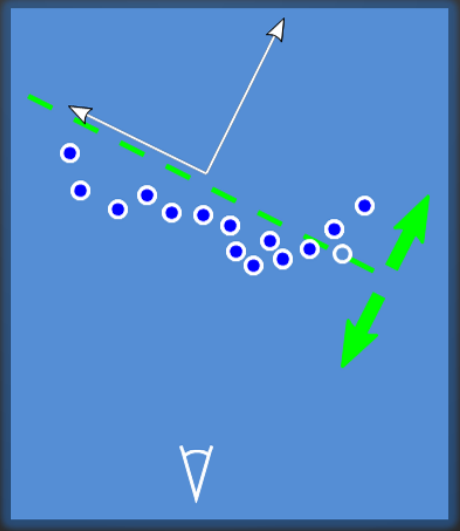


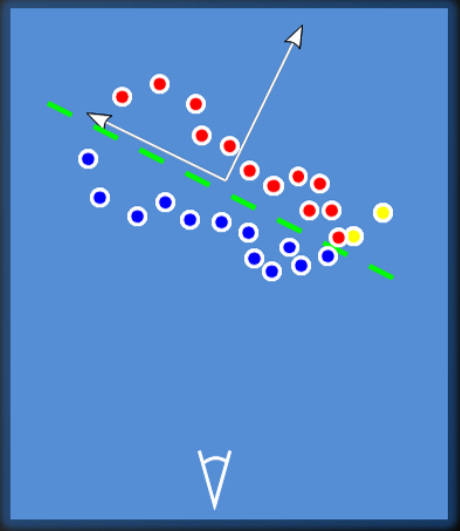


Back side reconstruction

- Assumption: back side is symmetrical to front side
- Steps to create back side:
  1. Consider point cloud of pixel centers in world coordinates
  2. Choose mirror plane normal using eigenvectors
  3. Optimize plane position along that normal
  4. Mirror the points

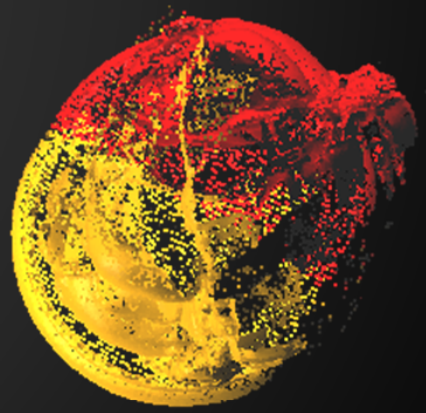






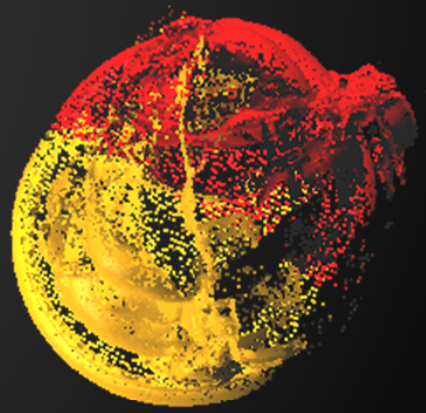
Back side reconstruction

- Assumption: back side is symmetrical to front side
- Steps to create back side:
  1. Consider point cloud of pixel centers in world coordinates
  2. Choose mirror plane normal using eigenvectors
  3. Optimize plane position along that normal
  4. Mirror the points



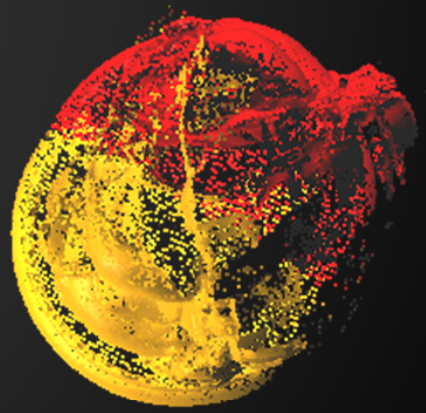
Back side reconstruction

- Assumption: back side is symmetrical to front side
- Steps to create back side:
  1. Consider point cloud of pixel centers in world coordinates
  2. Choose mirror plane normal using eigenvectors
  3. Optimize plane position along that normal
  4. Mirror the points
- Enforce minimal thickness constraint (for ray marching)



Back side reconstruction

- Assumption: back side is symmetrical to front side
- Steps to create back side:
  1. Consider point cloud of pixel centers in world coordinates
  2. Choose mirror plane normal using eigenvectors
  3. Optimize plane position along that normal
  4. Mirror the points
- Enforce minimal thickness constraint (for ray marching)
- Store back side in second layer in depth buffer

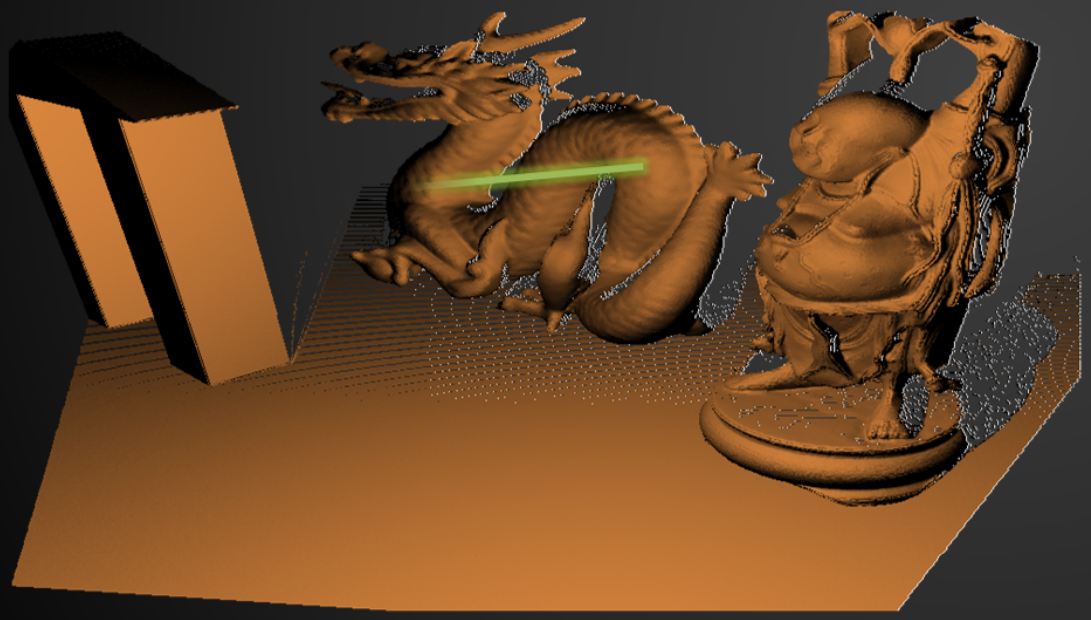


Environment map generation

- Need a way to evaluate $L_i(x, \omega)$ for reshading

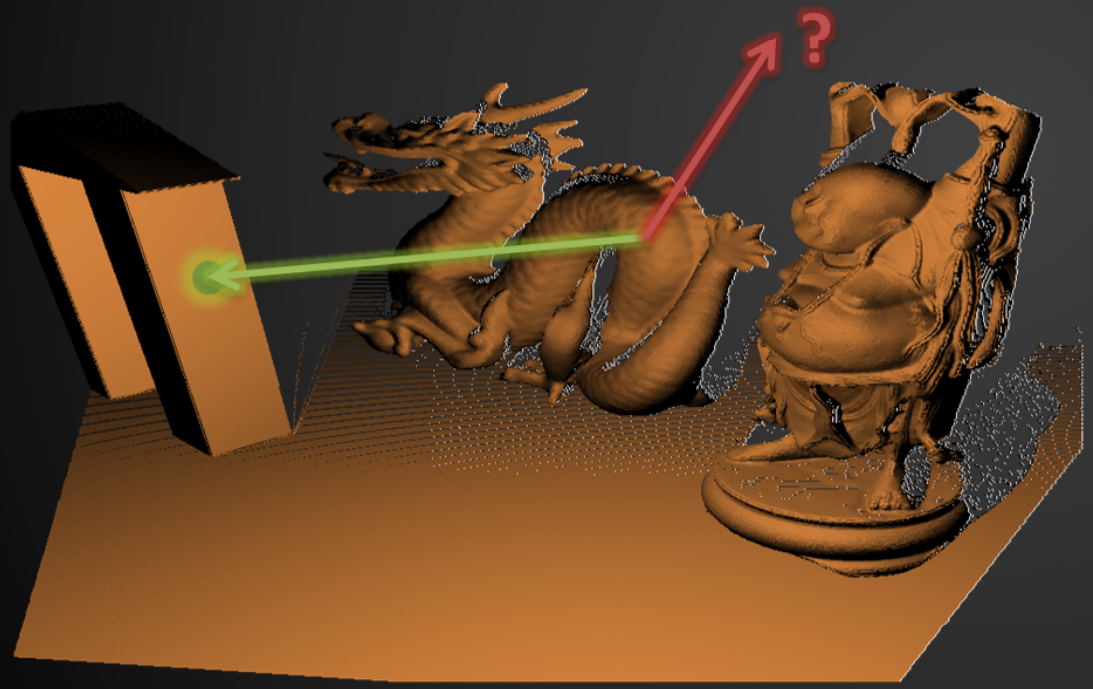
Environment map generation

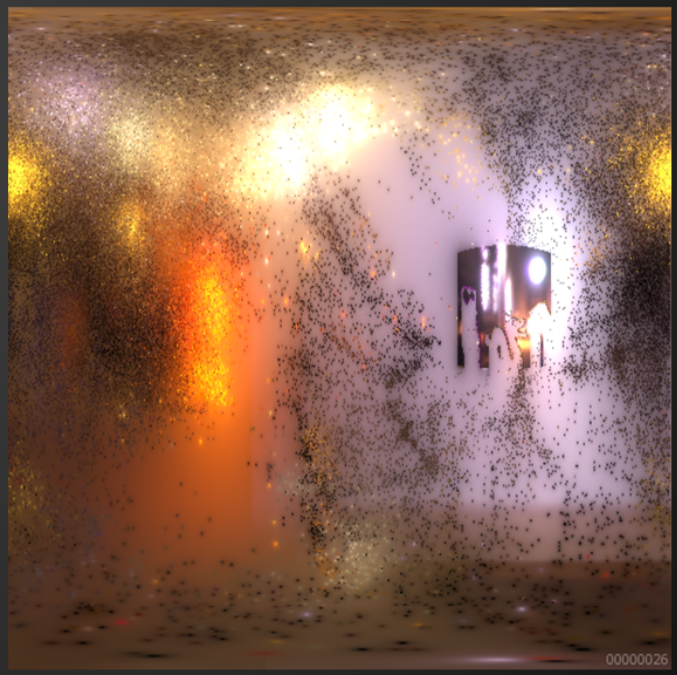
- Need a way to evaluate $L_i(x, \omega)$ for reshading
  - Evaluate using ray marching in depth buffer



Environment map generation

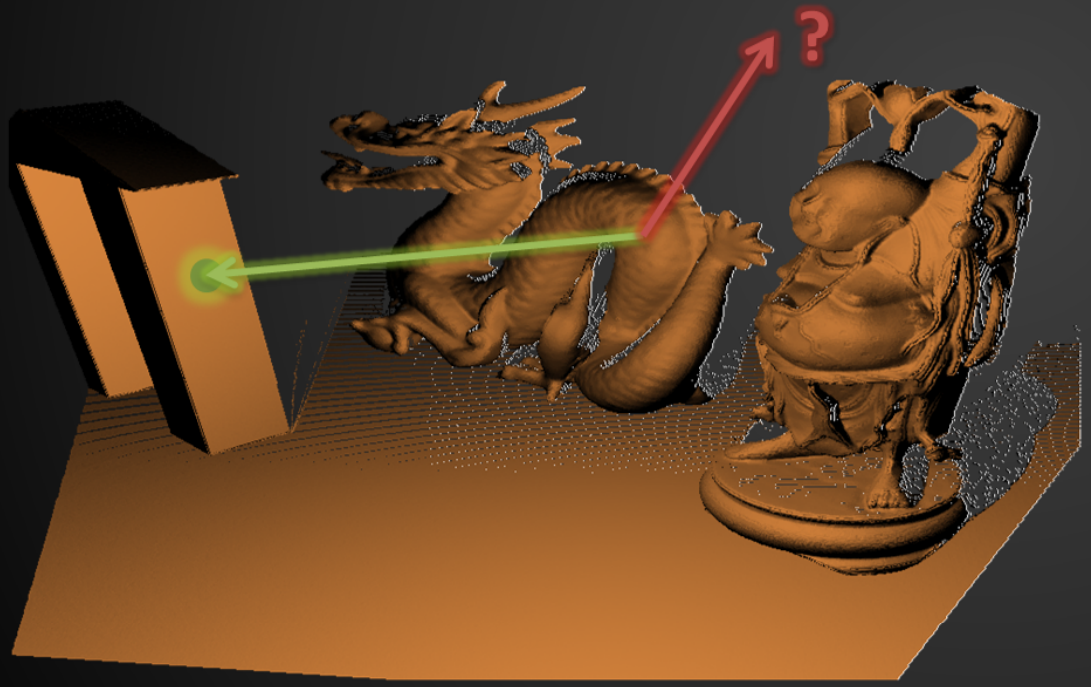
- Need a way to evaluate $L_i(x, \omega)$ for reshading
  - Evaluate using ray marching in depth buffer
  - Use an environment map when no hit in the depth buffer





Environment map generation

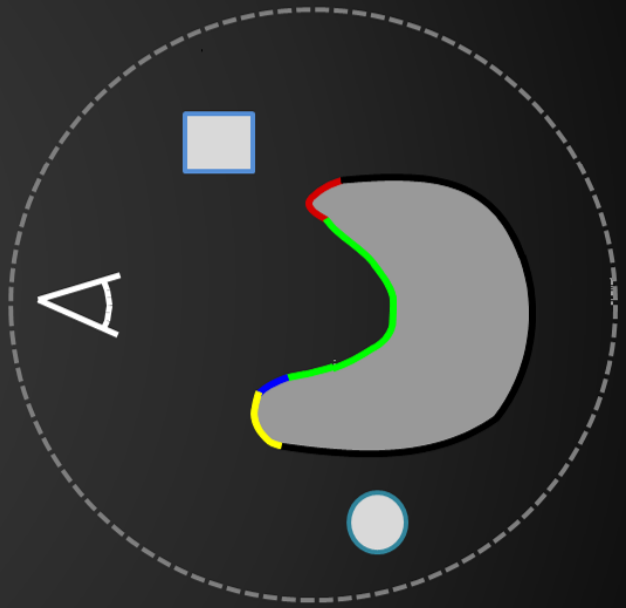
- Need a way to evaluate $L_i(x, \omega)$ for reshading
  - Evaluate using ray marching in depth buffer
  - Use an environment map when no hit in the depth buffer
- Build the EM from the surfaces of glossy objects





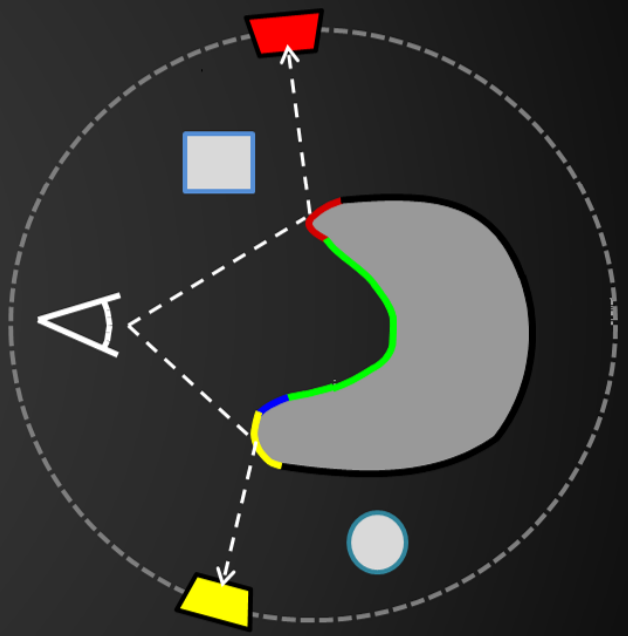
Environment map generation

- Steps:
  1. Estimate glossy reflectance of objects
  2. Iterate over objects & surface points, add color to sparse environment map
  3. Add distant pixel information
  4. Fill gaps in environment map



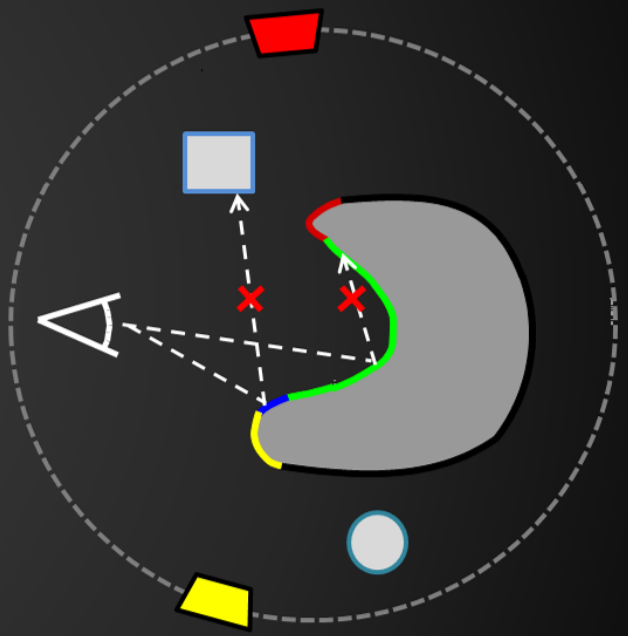
Environment map generation

- Steps:
  1. Estimate glossy reflectance of objects
  2. Iterate over objects & surface points, add color to sparse environment map
  3. Add distant pixel information
  4. Fill gaps in environment map



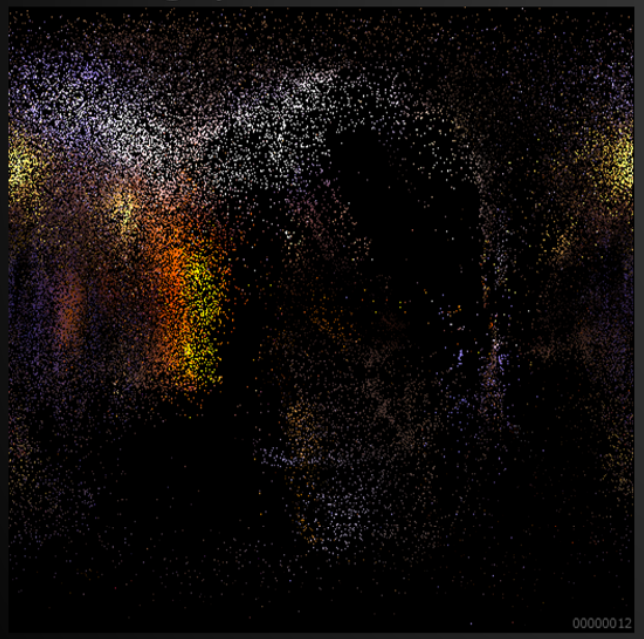
Environment map generation

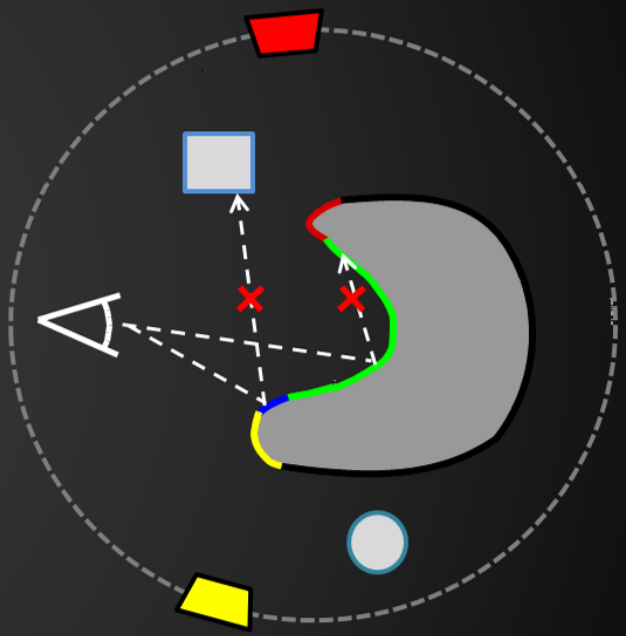
- Steps:
  1. Estimate glossy reflectance of objects
  2. Iterate over objects & surface points, add color to sparse environment map
  3. Add distant pixel information
  4. Fill gaps in environment map



Environment map generation

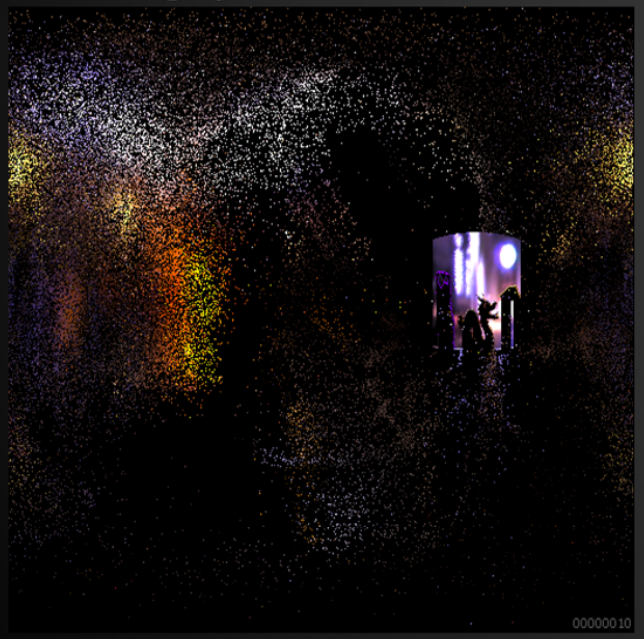
- Steps:
  1. Estimate glossy reflectance of objects
  2. Iterate over objects & surface points, add color to sparse environment map ✓
  3. Add distant pixel information
  4. Fill gaps in environment map

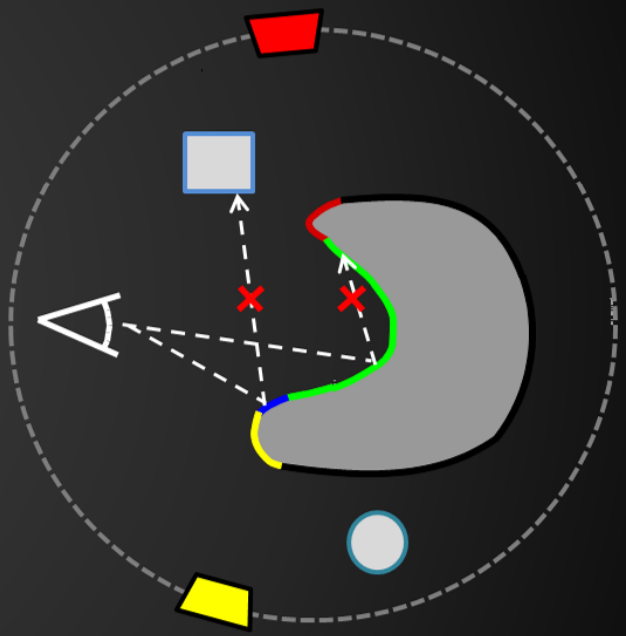




Environment map generation

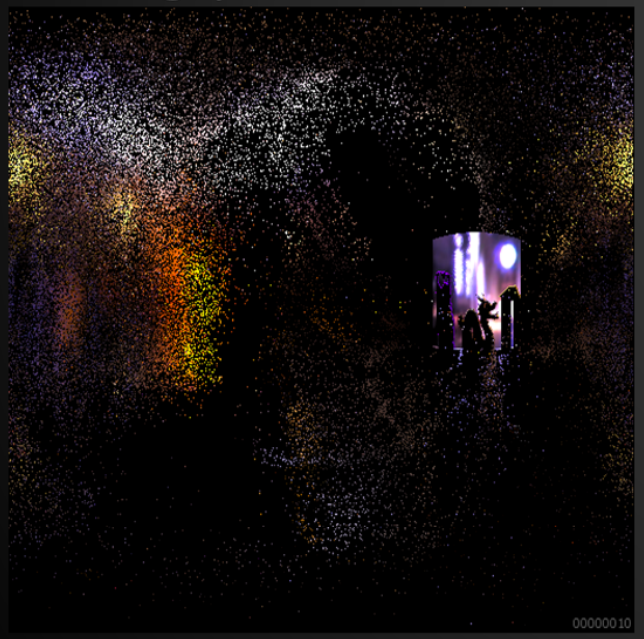
- Steps:
  1. Estimate glossy reflectance of objects
  2. Iterate over objects & surface points, add color to sparse environment map ✓
  3. Add distant pixel information ✓
  4. Fill gaps in environment map

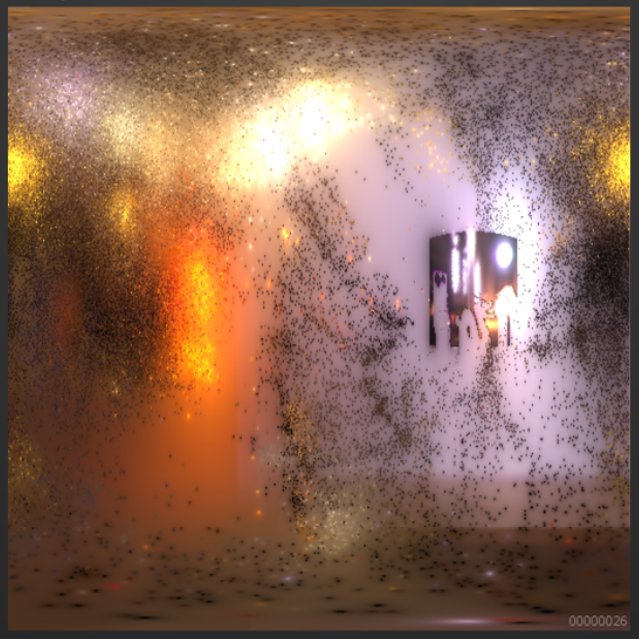


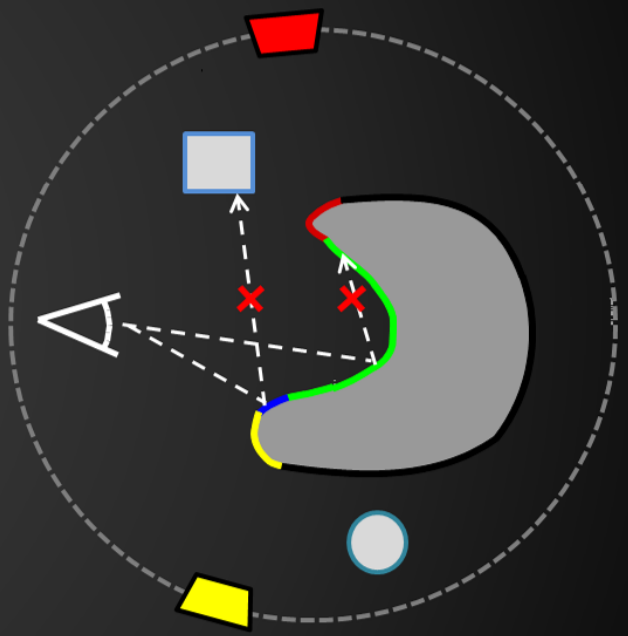


Environment map generation

- Steps:
  1. Estimate glossy reflectance of objects
  2. Iterate over objects & surface points, add color to sparse environment map ✓
  3. Add distant pixel information ✓
  4. Fill gaps in environment map ✓





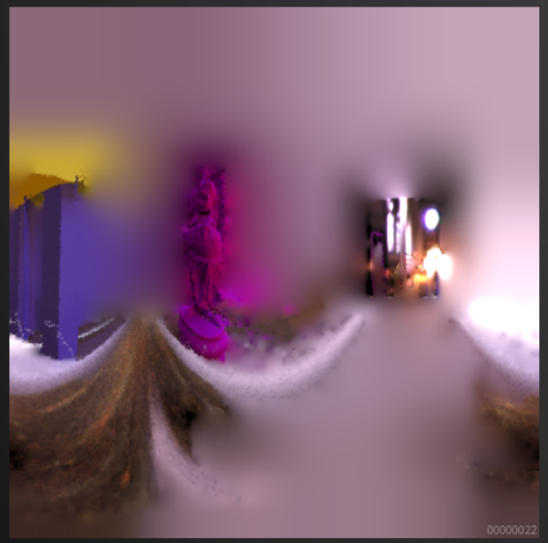


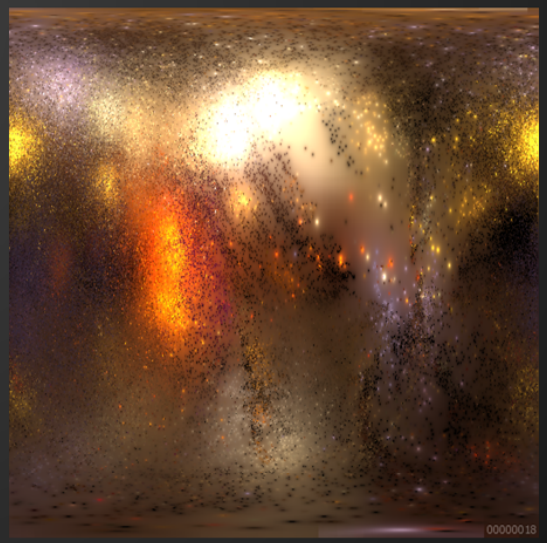
Reflectance estimation

- Need to estimate glossy reflectance to improve EM
- For each object:
  - Build two maps, incoming and reflected light
  - Calculate the mean colors m_S and m_G
  - Then divide to get the reflectance estimation $\frac{m_S}{m_G}$

Reflectance estimation

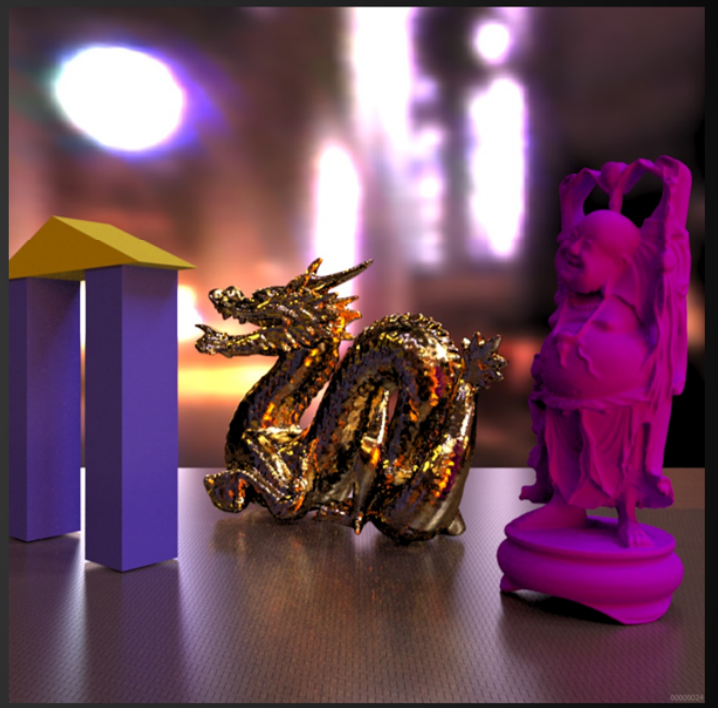
- Need to estimate glossy reflectance to improve EM
- For each object:
  - Build two maps, incoming and reflected light
  - Calculate the mean colors m_S and m_G
  - Then divide to get the reflectance estimation $\frac{m_S}{m_G}$





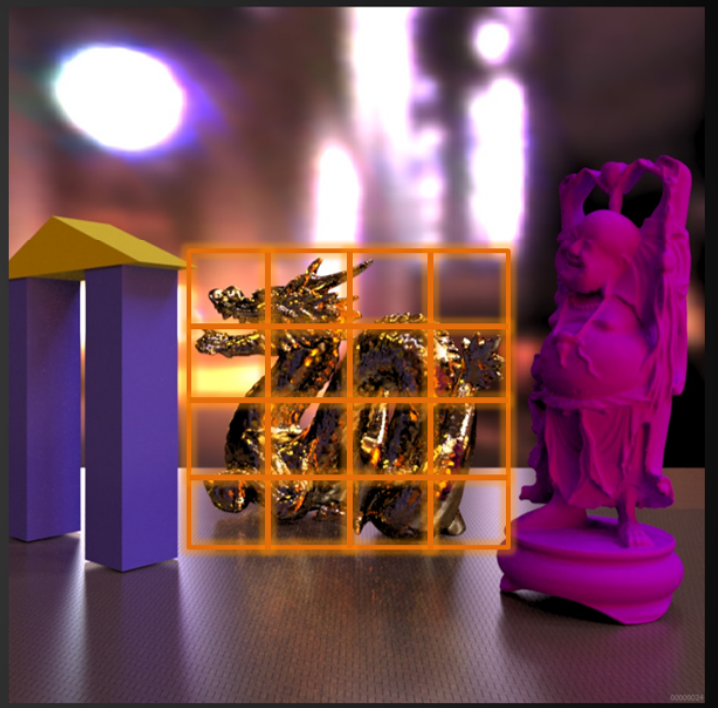
Reshading

- It is now possible to evaluate $L_I(x, \omega)$:
  - Start with ray marching in the depth buffer
    - Hit: Use pixel color at hit position
    - No hit: Access EM with ray direction



Reshading

- It is now possible to evaluate $L_I(x, \omega)$:
  - Start with ray marching in the depth buffer
    - Hit: Use pixel color at hit position
    - No hit: Access EM with ray direction
- Reshading:
  1. Iterate over visible pixels of object



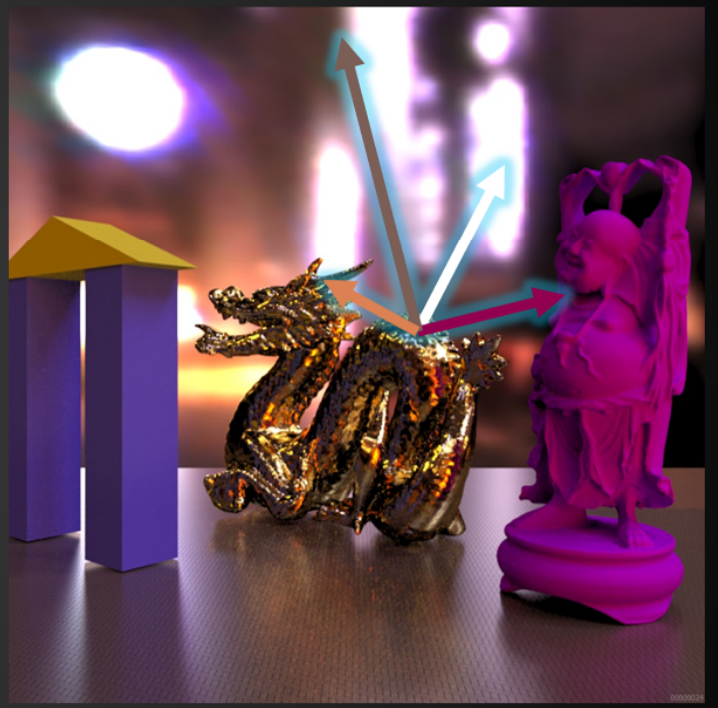
Reshading

- It is now possible to evaluate $L_I(x, \omega)$:
  - Start with ray marching in the depth buffer
    - Hit: Use pixel color at hit position
    - No hit: Access EM with ray direction
- Reshading:
  1. Iterate over visible pixels of object
     a) Generate rays in pos. Hemisphere



Reshading

- It is now possible to evaluate $L_I(x, \omega)$:
  - Start with ray marching in the depth buffer
    - Hit: Use pixel color at hit position
    - No hit: Access EM with ray direction
- Reshading:
  1. Iterate over visible pixels of object
     a) Generate rays in pos. Hemisphere
     b) evaluate $L_I(x_j, \omega_i)$ for each ray, add weighted (MC-Integration)



Reshading

- It is now possible to evaluate $L_I(x, \omega)$:
  - Start with ray marching in the depth buffer
    - Hit: Use pixel color at hit position
    - No hit: Access EM with ray direction
- Reshading:
  1. Iterate over visible pixels of object
     a) Generate rays in pos. Hemisphere
     b) evaluate $L_I(x_j, \omega_i)$ for each ray, add weighted (MC-Integration)



Reshading

- It is now possible to evaluate $L_I(x, \omega)$:
  - Start with ray marching in the depth buffer
    - Hit: Use pixel color at hit position
    - No hit: Access EM with ray direction
- Reshading:
  1. Iterate over visible pixels of object
     a) Generate rays in pos. Hemisphere
     b) evaluate $L_I(x_j, \omega_i)$ for each ray, add weighted (MC-Integration)
  2. Apply smoothing filter to pixels



Reshading

- It is now possible to evaluate $L_I(x, \omega)$:
  - Start with ray marching in the depth buffer
    - Hit: Use pixel color at hit position
    - No hit: Access EM with ray direction
- Reshading:
  1. Iterate over visible pixels of object
     a) Generate rays in pos. Hemisphere
     b) evaluate $L_I(x_j, \omega_i)$ for each ray, add weighted (MC-Integration)
  2. Apply smoothing filter to pixels



- Possible to evaluate arbitrary BRDF (limited by EM detail/geometry)

Translucency

- Uses Dipole diffusion approximation from Jensen et al. 2001
- Overview:
  1. Select sample points on object surface (uniform distribution)
  2. Gather irradiance at sample points
     (using uniform sampling and raymarching/EM lookup)
  3. Iterate over all surface points and evaluate dipole diffusion approximation

Translucency

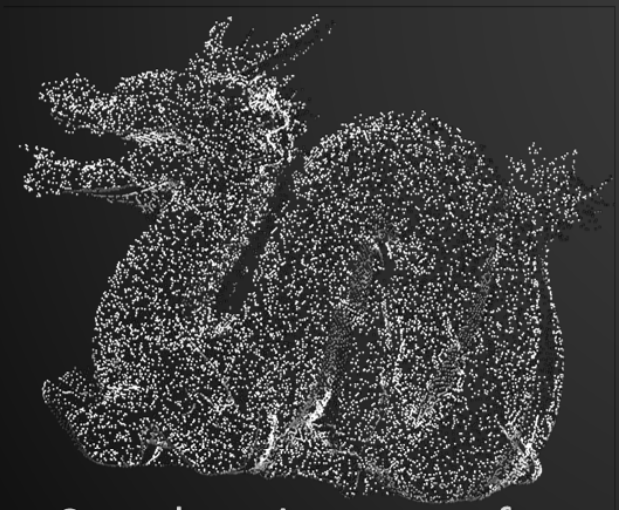
- Uses Dipole diffusion approximation from Jensen et al. 2001
- Overview:
  1. Select sample points on object surface (uniform distribution) ✓
  2. Gather irradiance at sample points
     (using uniform sampling and raymarching/EM lookup)
  3. Iterate over all surface points and evaluate dipole diffusion approximation



Sample points on surface

Translucency

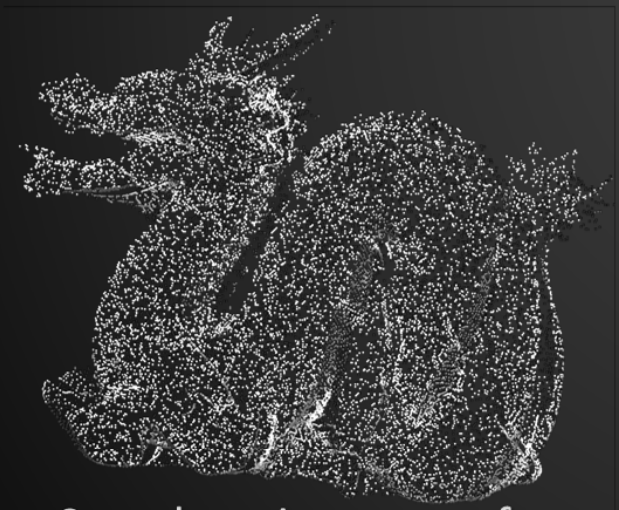
- Uses Dipole diffusion approximation from Jensen et al. 2001
- Overview:
  1. Select sample points on object surface (uniform distribution) ✓
  2. Gather irradiance at sample points
     (using uniform sampling and raymarching/EM lookup) ✓
  3. Iterate over all surface points and evaluate dipole diffusion approximation



Sample points on surface

Translucency

- Uses Dipole diffusion approximation from Jensen et al. 2001
- Overview:
  1. Select sample points on object surface (uniform distribution) ✓
  2. Gather irradiance at sample points
     (using uniform sampling and raymarching/EM lookup) ✓
  3. Iterate over all surface points and evaluate dipole diffusion approximation ✓

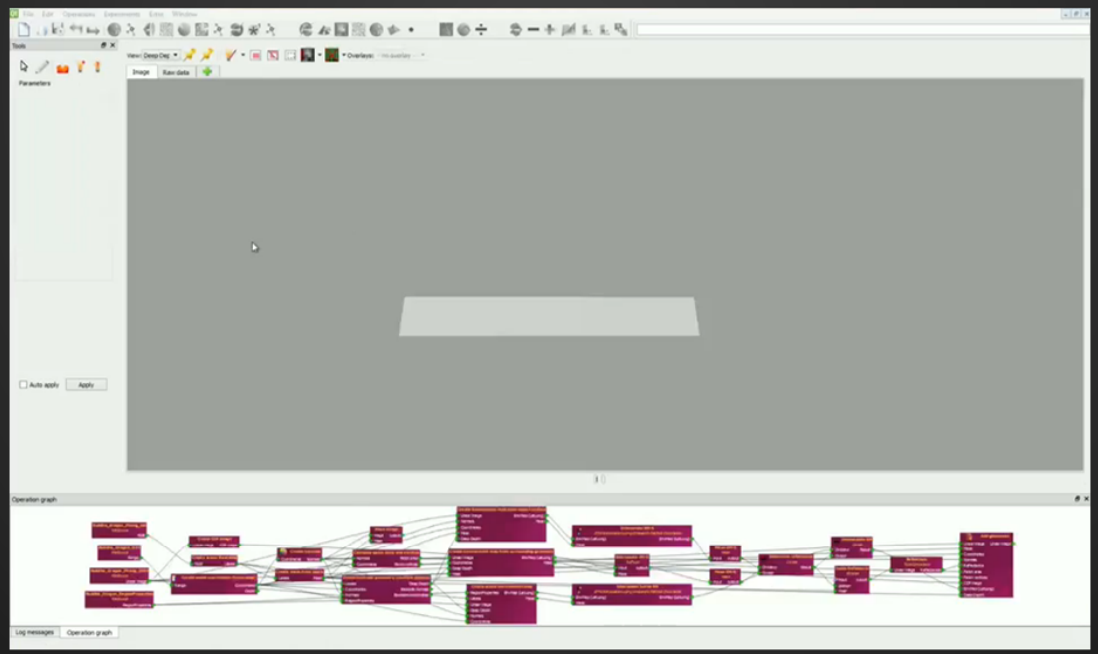


Sample points on surface

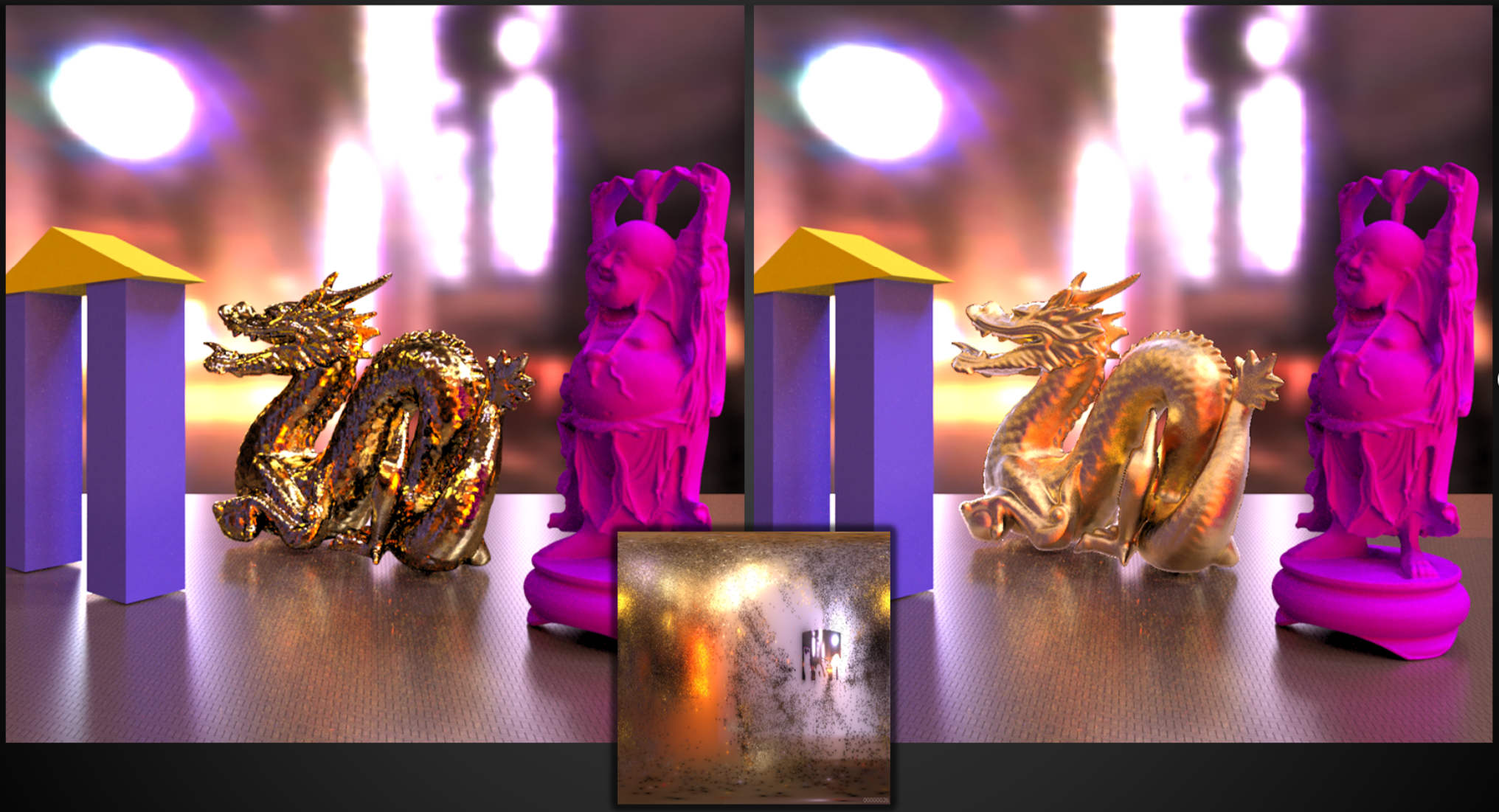


Results

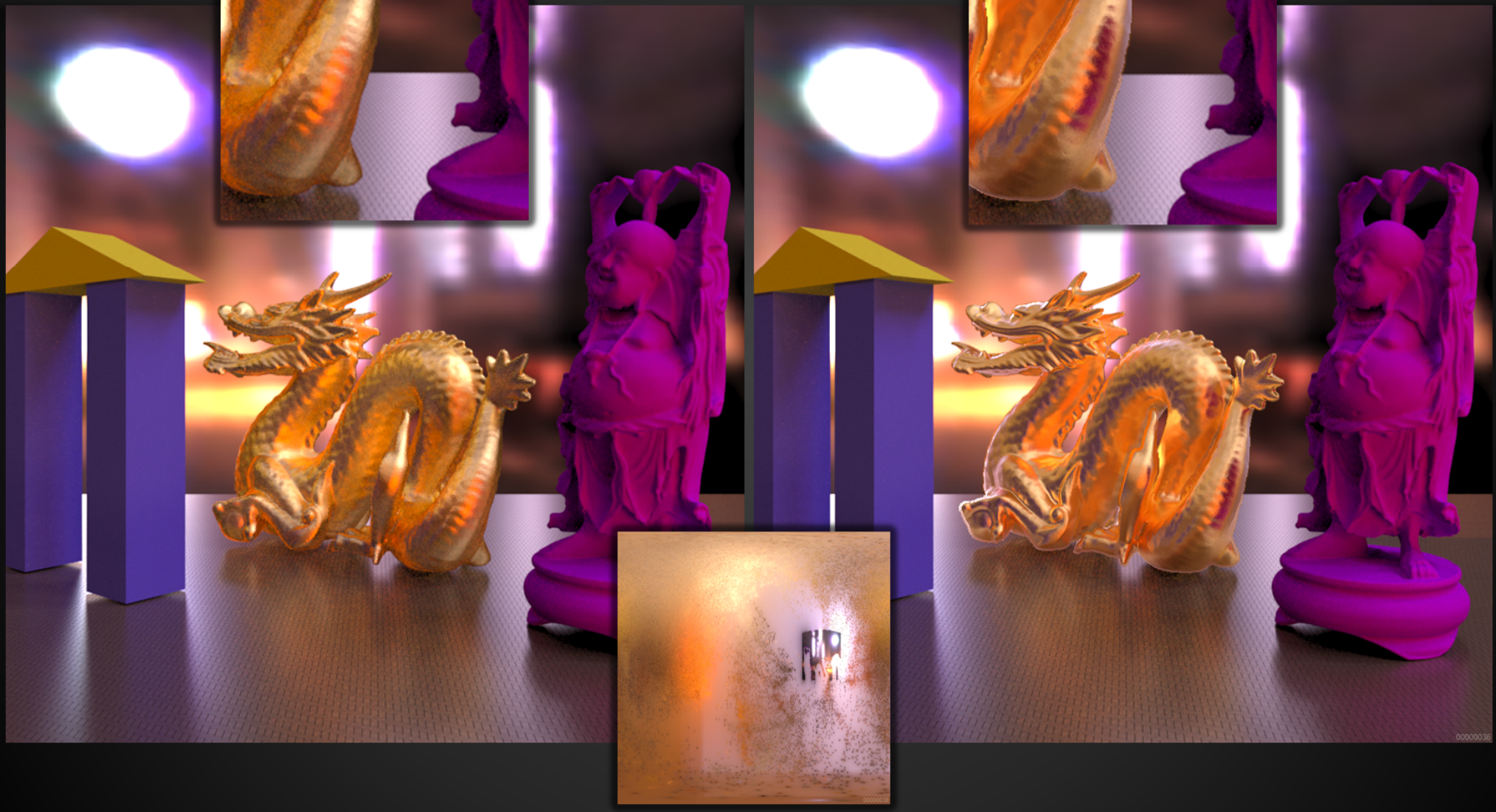
- Implementation on CPU & GPU
  - Enables interactive work with progressive updates
- Synthetic test images rendered with LuxRender
- Photos captured with digital camera and laser scanner for depth
  - Two pass capture for glossy objects



Results – Reshading



Results – Reshading



Results – Reshading

Original



Changed

Results – Reshading

Original

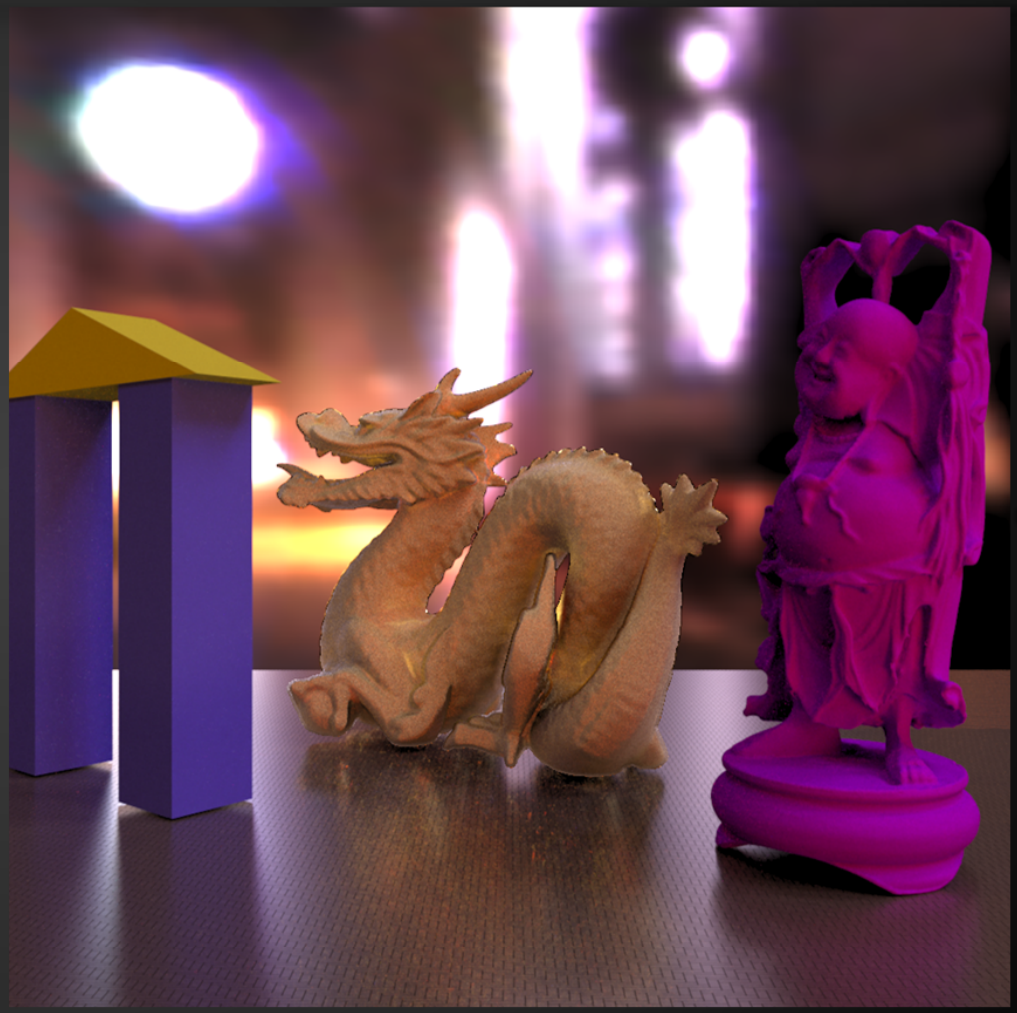


Changed

Results – Translucency

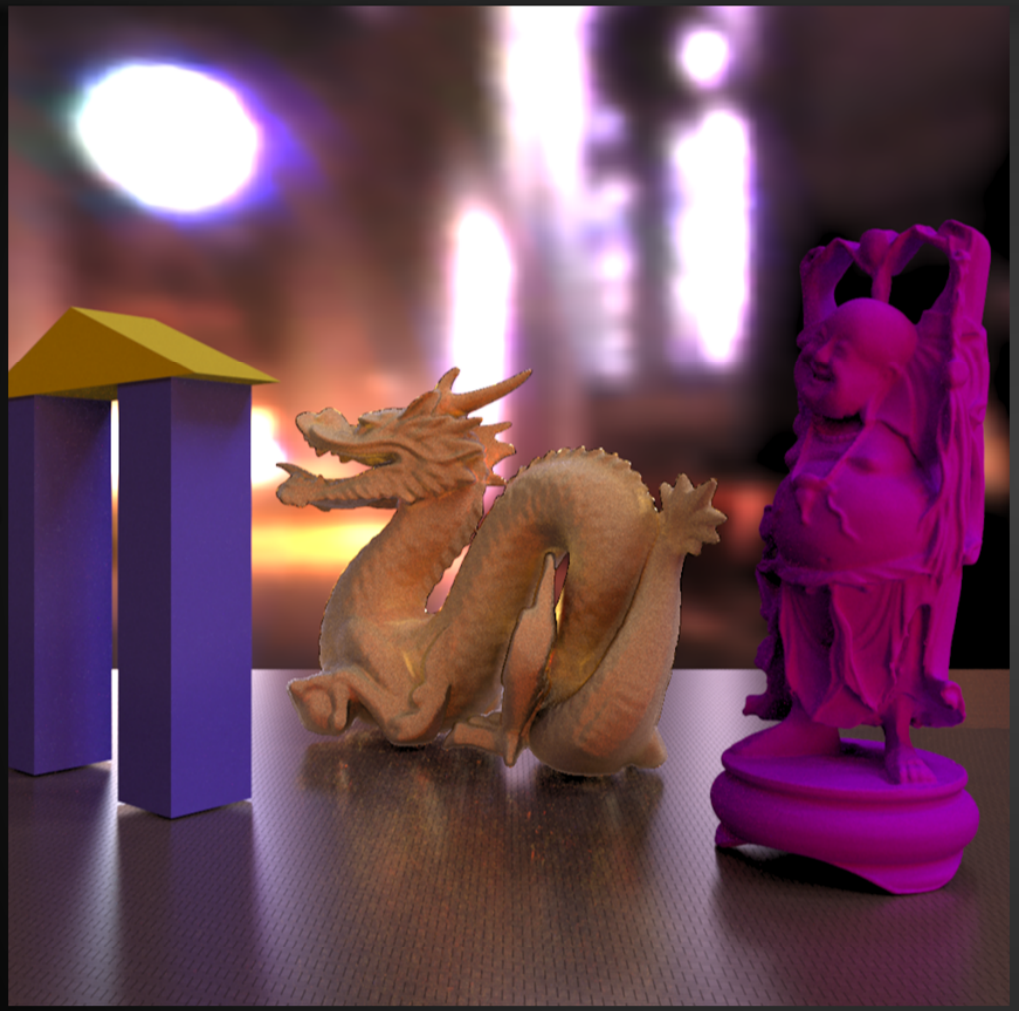


Original

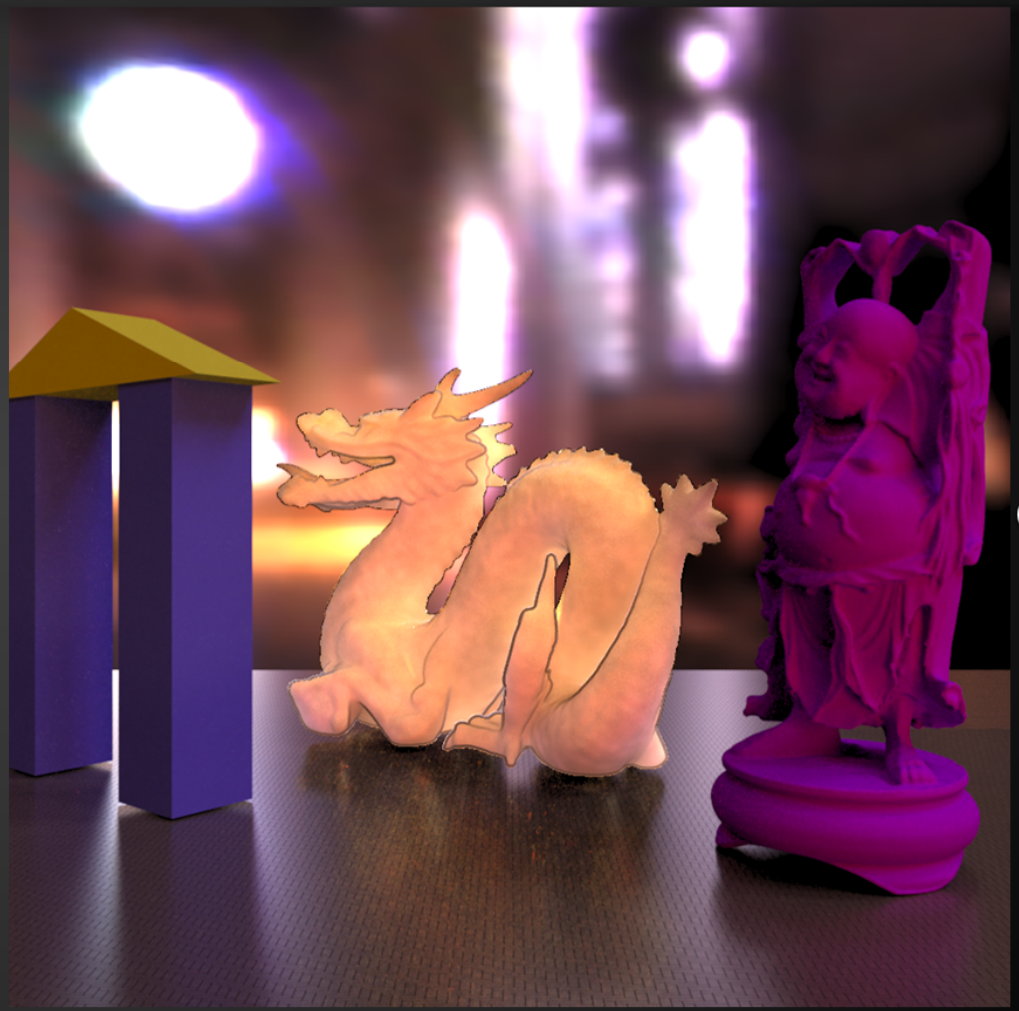


Changed

Results – Translucency



"Original"



Changed

Conclusion

- Interactive material editing in digital images
- Color and depth input
- Based upon ray marching in depth buffer and generation of an environment map

- Acknowledgements
  - Part of this work funded by Software Campus Grant FKZ 01IS12051
  - Dragon and Happy Buddha Model courtesy of Stanford Computer Graphics Laboratory
  - Grace Cathedral environment map used in the rendering from [Debevec 1998]

Conclusion

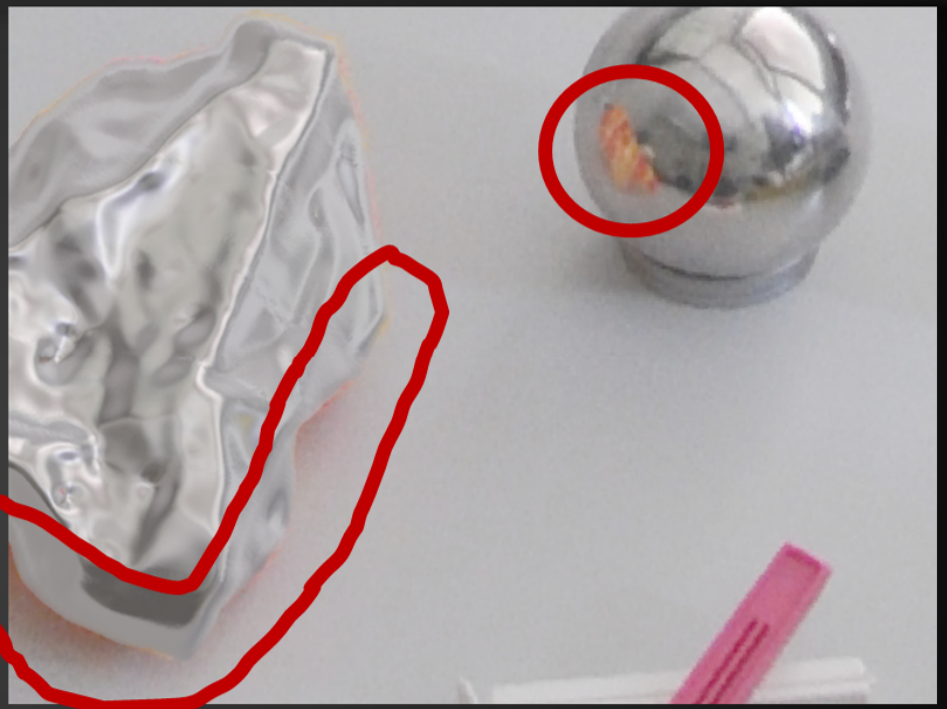
- Interactive material editing in digital images
- Color and depth input
- Based upon ray marching in depth buffer and generation of an environment map

- Future work
  - BRDF estimation

- Acknowledgements
  - Part of this work funded by Software Campus Grant FKZ 01IS12051
  - Dragon and Happy Buddha Model courtesy of Stanford Computer Graphics Laboratory
  - Grace Cathedral environment map used in the rendering from [Debevec 1998]

Conclusion

- Interactive material editing in digital images
- Color and depth input
- Based upon ray marching in depth buffer and generation of an environment map



- Future work
  - BRDF estimation
  - Indirect lighting effects

- Acknowledgements
  - Part of this work funded by Software Campus Grant FKZ 01IS12051
  - Dragon and Happy Buddha Model courtesy of Stanford Computer Graphics Laboratory
  - Grace Cathedral environment map used in the rendering from [Debevec 1998]

Bibliography

- **Carroll et al. 2011**: CARROLL R., RAMAMOORTHI R., AGRAWALA M.: Illumination decomposition for material recoloring with consistent interreflections. ACM Transactions on Graphics 30, 4 (2011), 43:1–43:10.
- **Debevec 1998**: DEBEVEC, P.: Rendering Synthetic Objects Into Real Scenes: Bridging Traditional and Image-Based Graphics With Global Illumination and High Dynamic Range Photography, Proceedings of SIGGRAPH '98 (1998), 189-198
- **Fang and Hart 2004**: FANG H., HART J.: Textureshop: Texture synthesis as a photograph editing tool. ACM Transactions on Graphics 23, 3 (2004), 354–358.
- **Han and Lee 2013**: HAN Y., LEE J.: High Quality Shape from a Single RGB-D Image under Uncalibrated Natural Illumination vol. 1, pp. 1617–1624, 2013.
- **Jensen et al. 2001**: JENSEN H. W., MARSCHNER S. R., LEVOY M., HANRAHAN P.: A practical model for subsurface light transport. SIGGRAPH'01 (2001), 511–518
- **Jensen et al 2002:** JENSEN H. W., BUHLER J.: A rapid hierarchical rendering technique for translucent materials. ACM Transactions on Graphics (Proc. of SIGGRAPH 2002) (2002).
- **Khan et al. 2006**: KHAN E., REINHARD E., FLEMING R. W., BÜLTHOFF H.: Image-based material editing. ACM Transactions on Graphics 25, 3 (2006), 654–663.