Lecture 23: Image-based Modeling and Lighting

Fall 2004
Kavita Bala
Computer Science
Cornell University

Announcements
• Projects: Email short (1-2 paragraphs) update on project by next Tuesday
  – Plans till project due date

Complexity
• Lighting: many lights, environment maps
  – Global illumination, shadows
• Materials: BRDFs, textures
• Geometry: Level-of-detail, point-based representations
• All: image-based rendering

Image-based Modeling
• Extract geometry + textures from pictures
  – Create a simple model
• Reuse geometry and texture to render scene from new viewpoints
• Façade system:
  –Debevec and Malik, UCB

Match Edges
• Start from photograph of building
• User creates rudimentary model with cubes
• Marks corresponding edges between model and photographs
• Enough correspondences to reconstruct camera and model parameters

Use blocks
• Easy to specify
• Fewer interactions
• Easy to determine final surface
Find best fit

- Project model representation to camera
- Create error term
- Minimize error
- Iterative method updates model representation and looks for best fit
- Also solves for camera parameters

Extract textures

- Extract textures from original photograph
- Project them onto the surface

Example: Extract Geometry

Extract textures

- Extract textures from original photograph
- Project them onto the surface

View-dependent texture mapping

Example: Final Model w/ Textures
Used in the Matrix

Image Based Lighting

- Real Scene

- Goal: place synthetic objects on table

IBM Discussion

- Cons:
  - Small geometric details not included in model
  - Features in textures not part of model
  - Fair amount of manual input required!

- Pros:
  - Effective and useful
  - RealViz, ...
  - Open area of research

Cons of IBM

- Reasonable image quality but
  - Lighting is baked in: to undo lighting need material properties
  - Geometry is fixed
Image Based Lighting

- Use renderer - compute effects of synthetic objects on local scene

Image Based Lighting

- Render into the scene

Image Based Lighting

- Render synthetic objects
Image Based Lighting
• Effect of local scene on real scene

Discussion
• Good results for special effects
• But, estimating reflectances problematic

Image Based Lighting
• Add differences to image

Inverse global illumination
• Goal: recover BRDF per patch
• But account for GI in an interior environment
• Idea: solve for Ward BRDF model
  – specular parameters uniform per patch
  – diffuse interreflection handled trivially
  – specular interreflection (rare) handled by simple iterative algorithm

Handling interreflection
• In diffuse case, all is easy
  – each patch lit by sources and other patches
    \[ D_i = E_i + \rho_d \sum_j D_j F_{ij} \]
• Specular case is more tricky
  – illum. depends on specular parameters elsewhere
  – assuming diffuse dominates, iteratively solve for a correction for specular illumination
  – (note: must observe a highlight on every surface)
    \[ L_i = \left( \frac{F_i}{\rho_s} \right) + \rho_s K(\alpha_i, \Theta_i) I_i \]
IGI inputs

40 high-dynamic-range images

IGI results

Re-rendered (bottom) to match input (top)

Summary of IBM, IBL

• Very powerful

• Many interesting research areas to be solved

Image-based Approaches

• Goal: Realism!

• Image is input and rendering primitive
Comparison

• No depth
  – QuickTime VR (simplest, 2D panoramas)
  – Lumigraph/Light Field (4D arrays)

• Reconstructed Depth
  – Plenoptic Modeling (2.5 D)

• IBM vs. IBR
  – Some manual user input ok
  – Simple geometry recovered with user assistance
  – Complex geometry represented as texture

Conclusions

• IBR: promising approach to handle complexity

• Benefits:
  – No labor-intensive modeling
  – High geometric and material complexity
  – Rendering time constant: proportional to image size, independent of scene complexity

• Disadvantages:
  – Quality
  – Not-quite automatic

• Exam: Everything till today