# The ModelCamera

## Efficient acquisition of large-scale building interiors

*Voicu Popescu, Gleb Bahmutov,*
*Elisha Sacks, Mihai Mudure*

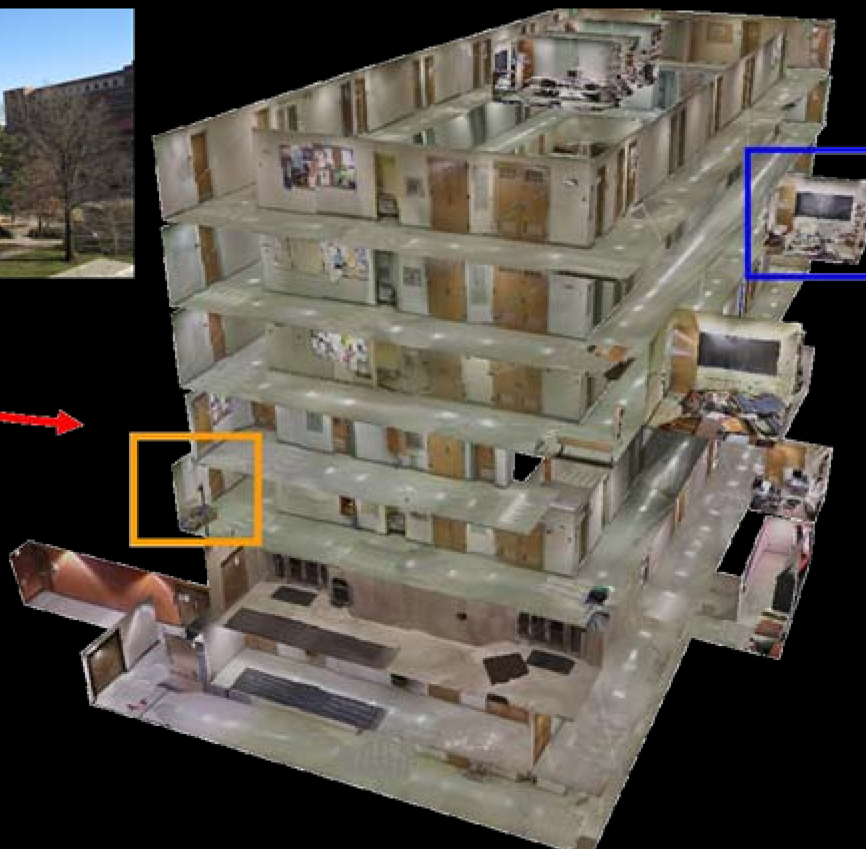
# Motivation

- Modeling—bottleneck for computer graphics applications
- Solutions restricted to *outside-looking-in*
- *Inside-looking-out* much harder
  - Sheer size of scene (area, range of depths)
  - Lack of control over scene
  - Occlusions

# Goal

- Develop a modeling system for building interiors that is *efficient* and *effective*
  - 100-room building in one week
  - Support for computer graphics applications—realistic virtual walkthroughs at interactive rates
  - Support for quantitative applications (i.e. physical simulation)

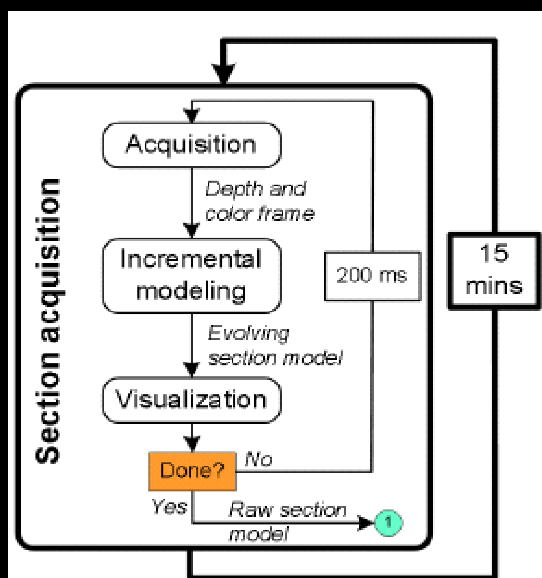
# We are almost there: ModelCamera modeling system

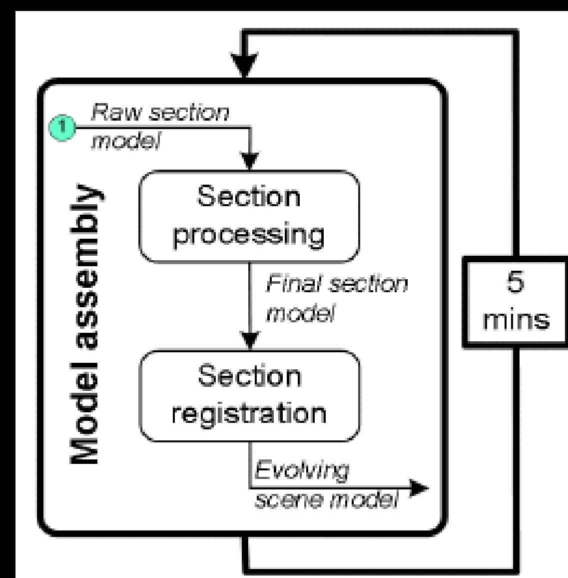


20 rooms & corridors
7 floors
1,450 $m^2$ of floor space
1 acquisition device
2 person team
40 hours

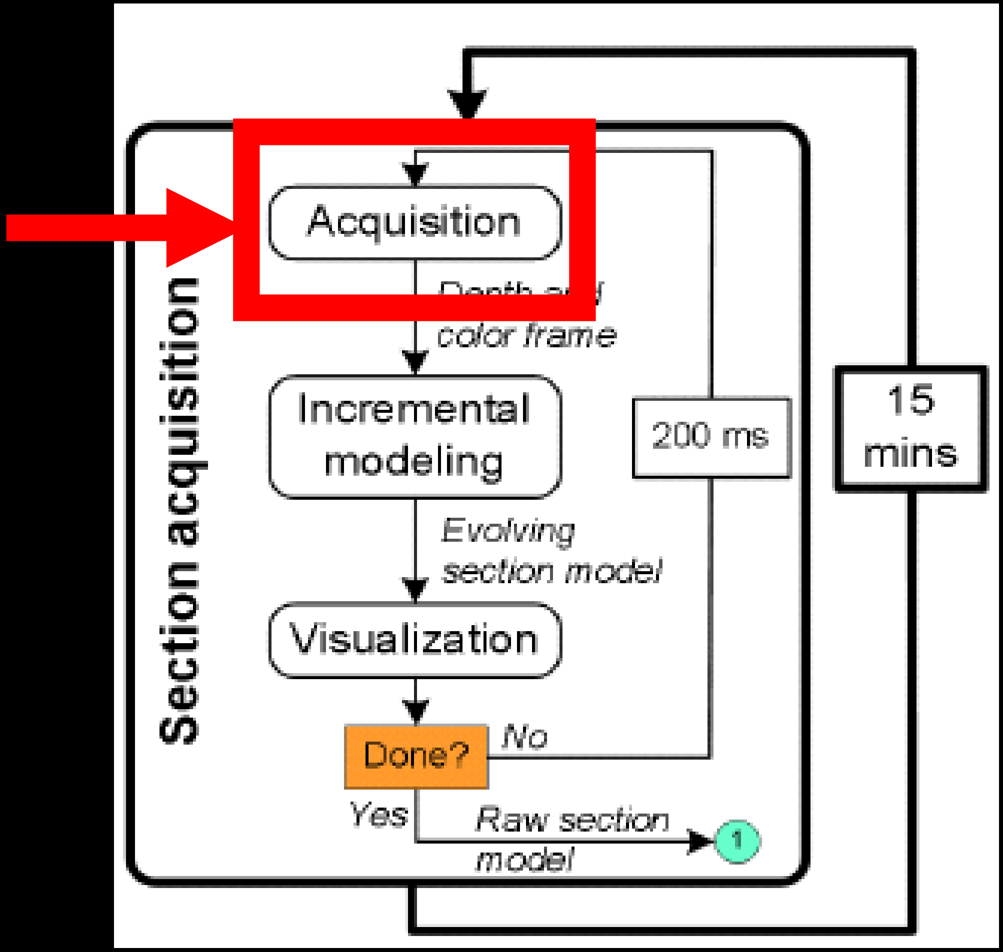
# System overview

- Sections (room or corridor segment) acquired at interactive rates
- Model assembled from sections

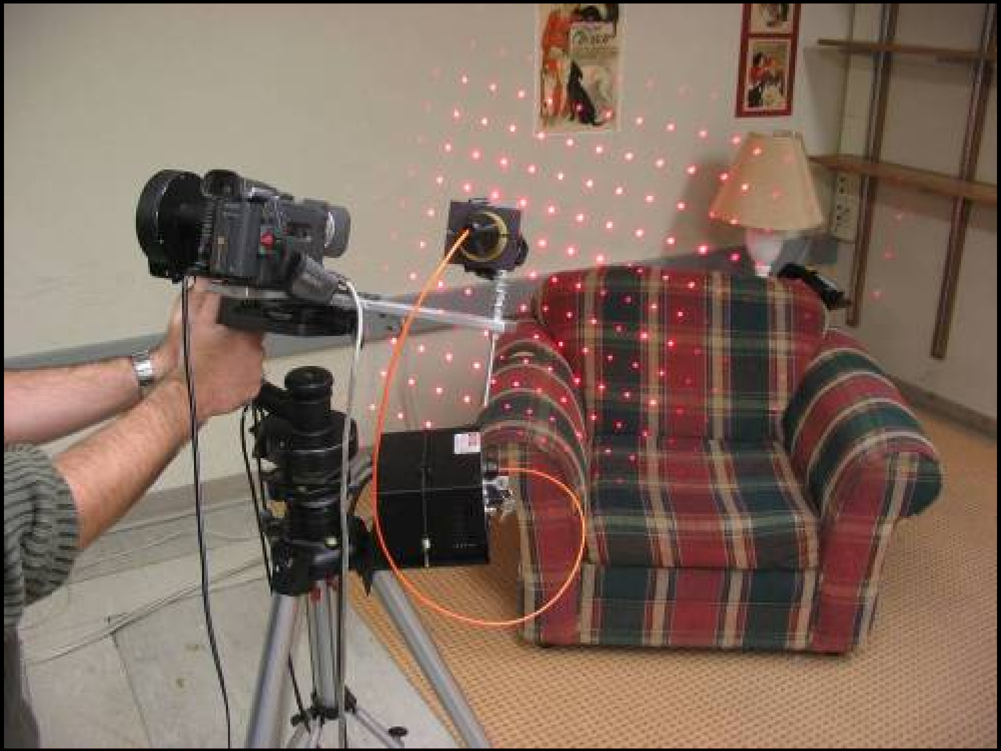


# Data acquisition

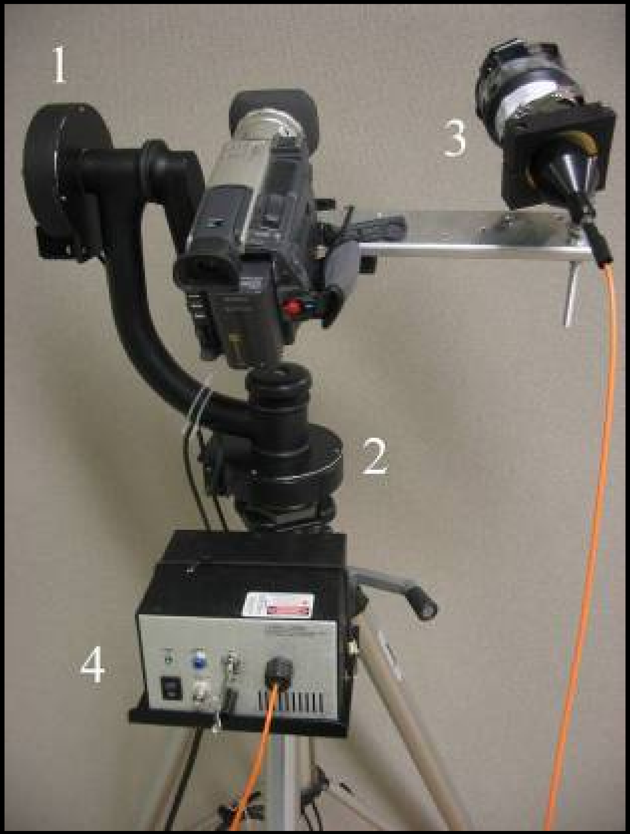


# Data acquisition: ModelCamera



- Custom structured-light device
  - 720x480 color samples
  - 11x11 depth samples
  - 15 fps
  - Parallax-free pan-tilt bracket

# ModelCamera prototype 4

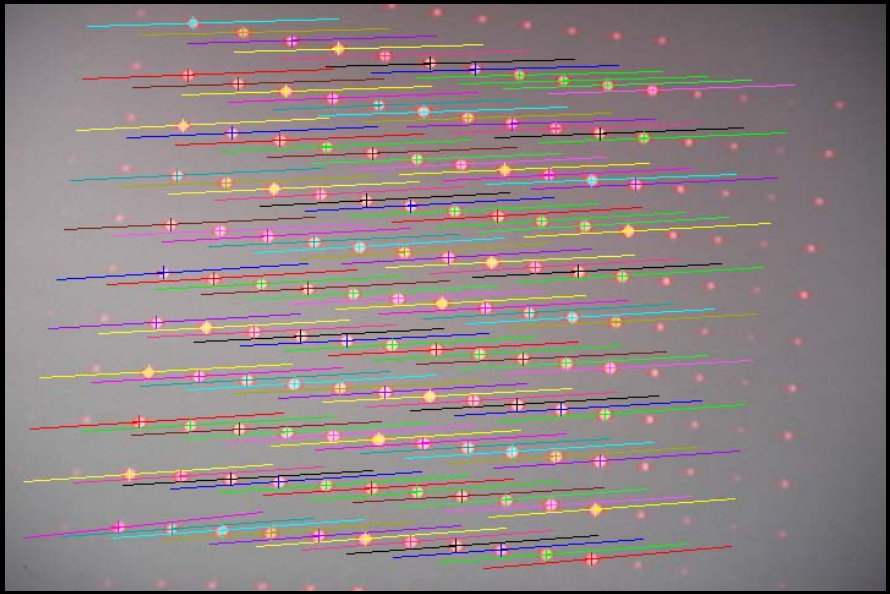


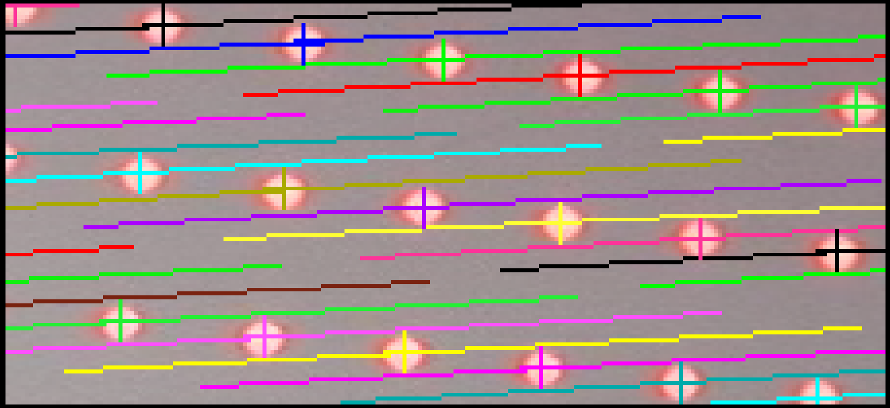

1 & 2: shaft encoders

3: diffraction grating

4: laser source

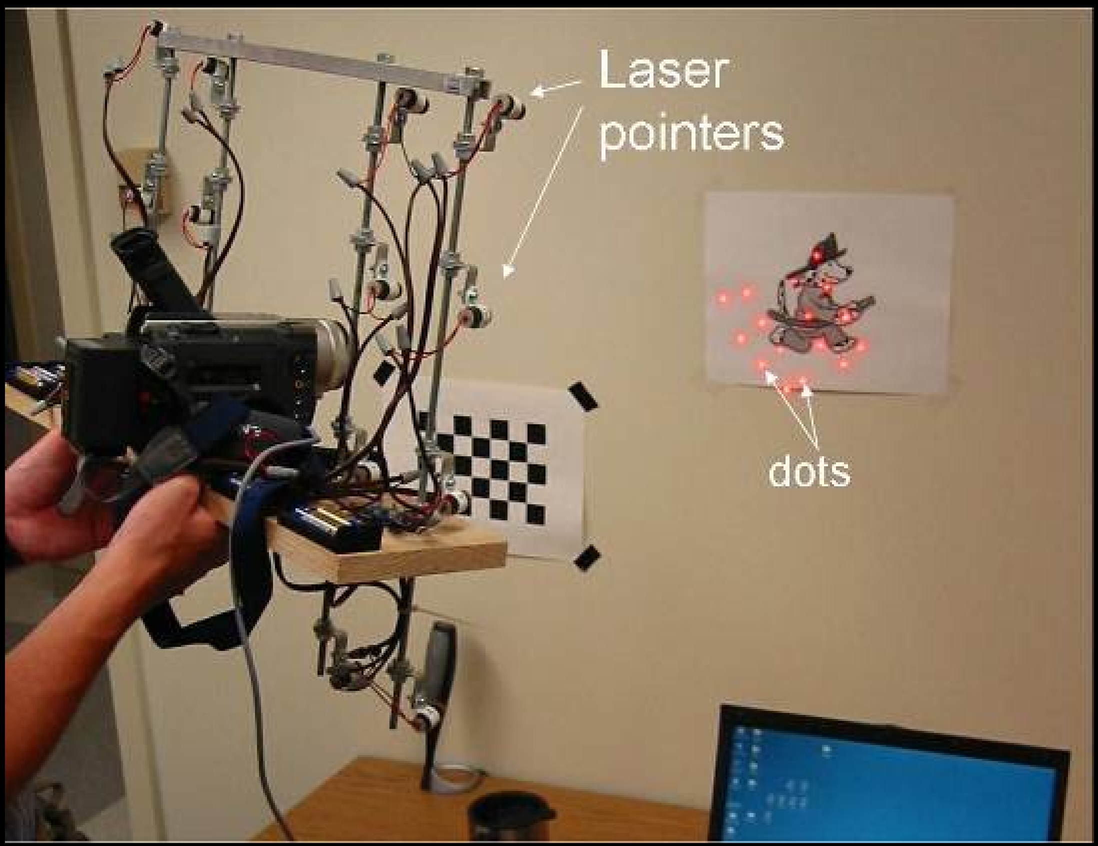
# Robust and efficient depth acq.

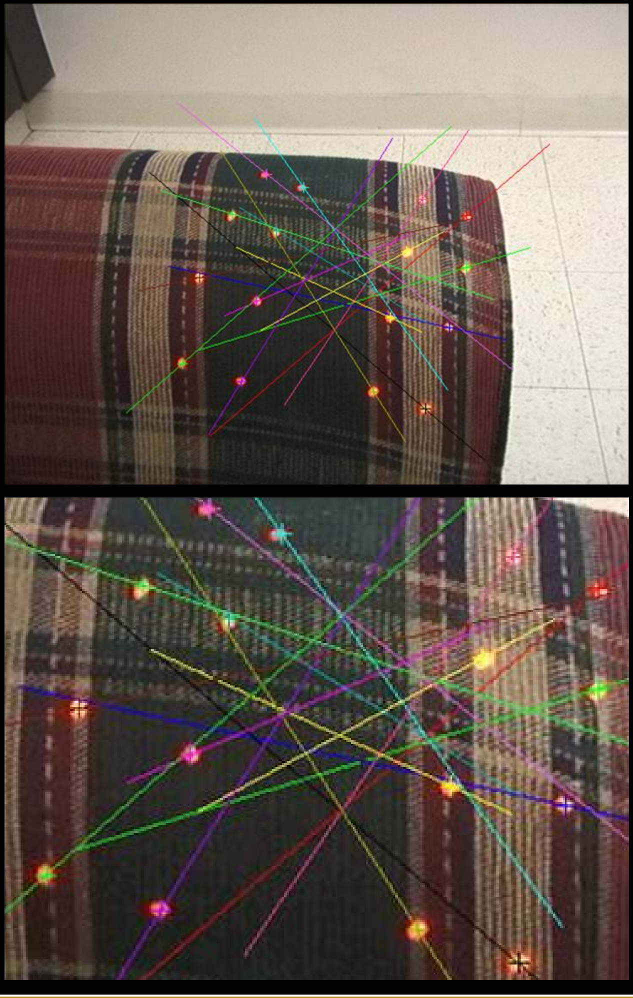




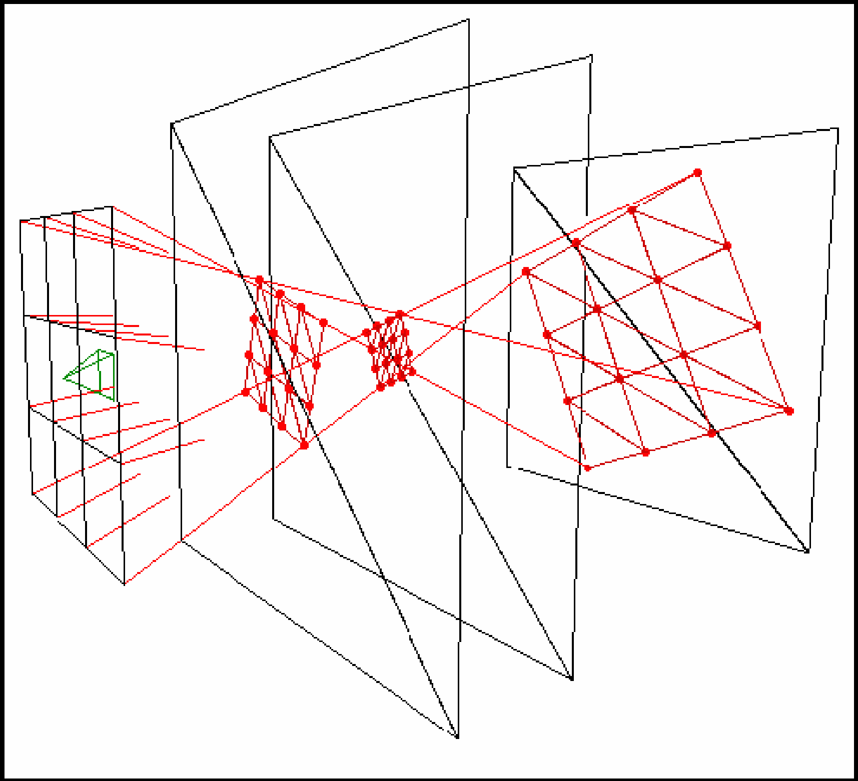
- Disjoint “epipolar” segments
  - Efficient 1D dot search
  - No dot confusion
- Optimal use of frame
- Large solid angle depth sampling

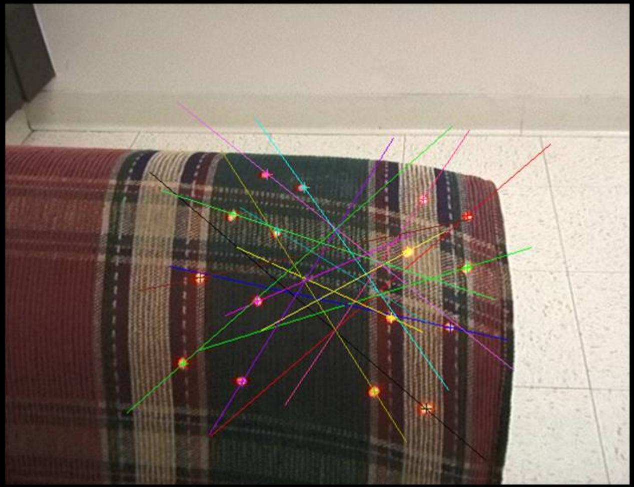
# ModelCamera prototype 1

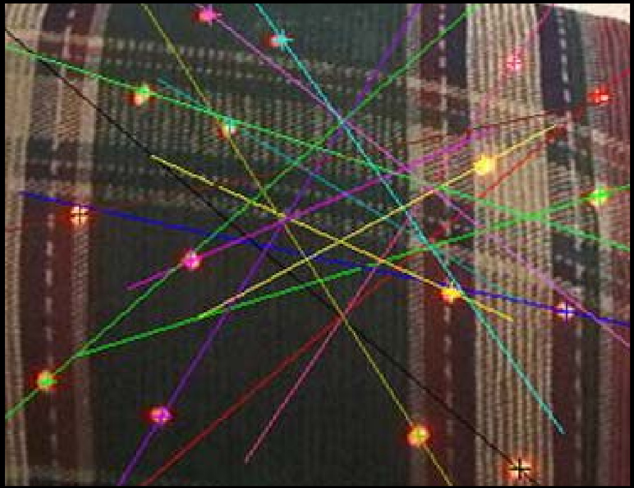


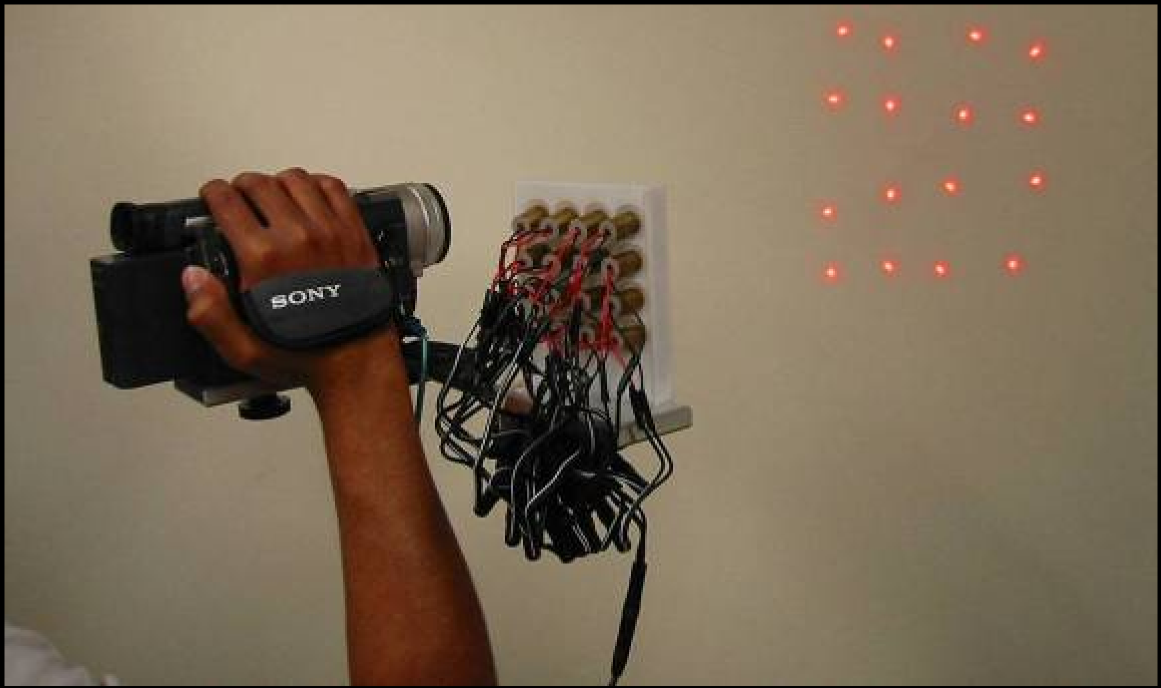
# ModelCamera prototype 1



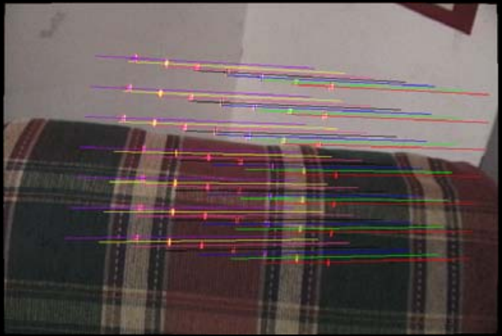


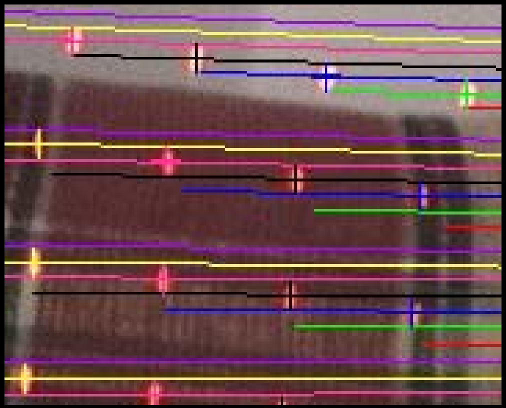


# ModelCamera prototype 2

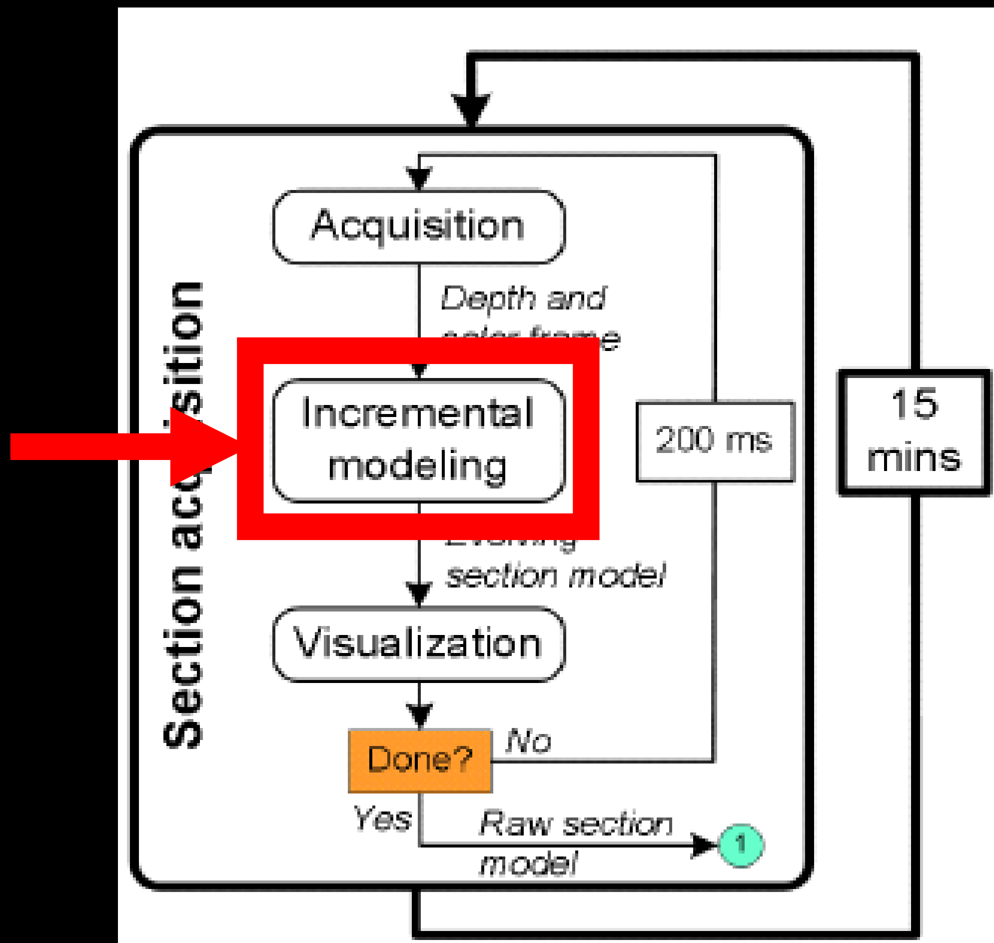


# ModelCamera prototype 3

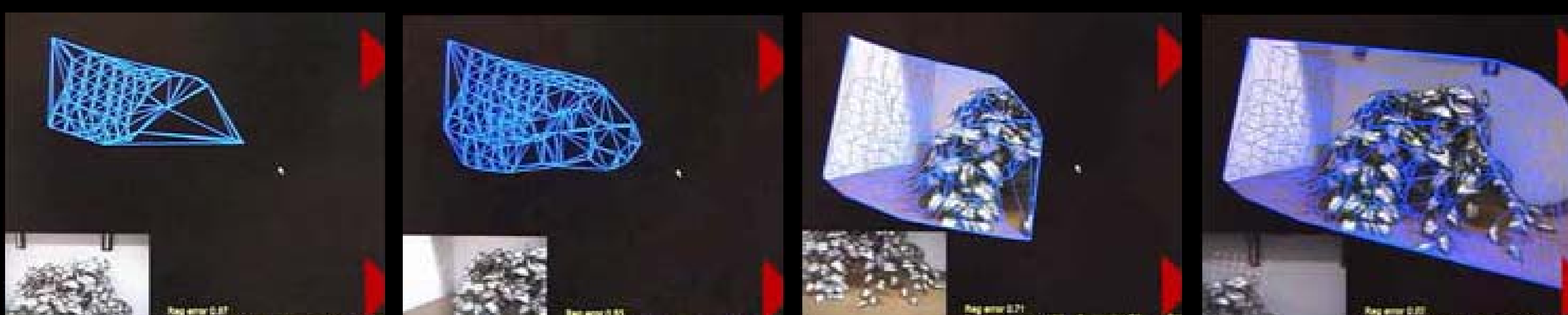


# Incremental modeling

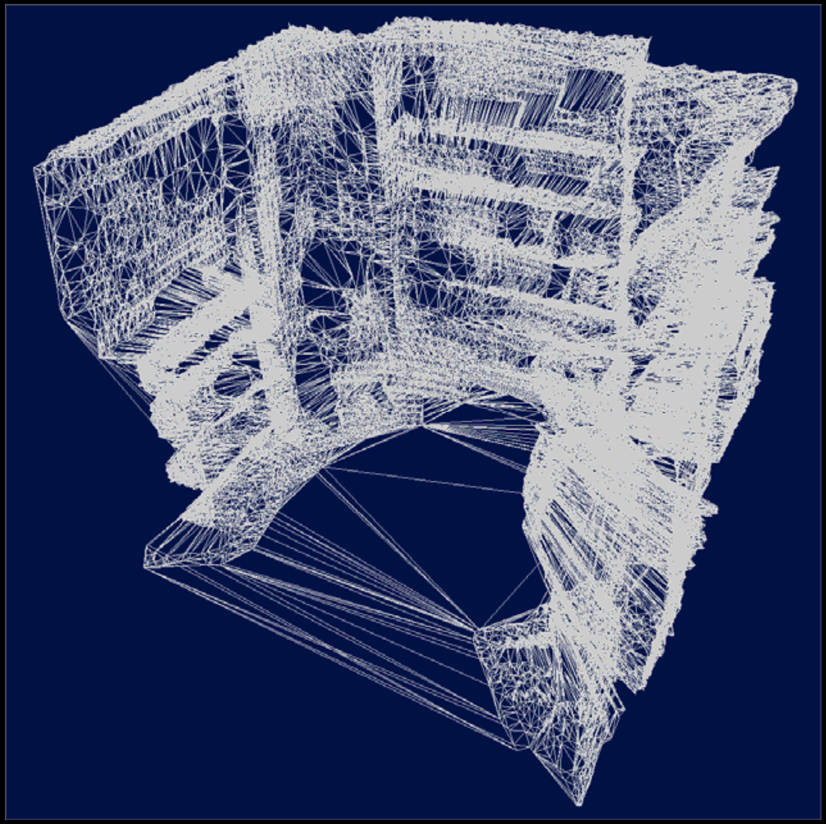


# Real-time incremental modeling: Depth Enhanced Panorama (DEP)

- *Input*: stream of dense-color & sparse depth frames with same acquisition viewpoint
- *Output*: evolving texture-mapped triangle mesh (DEP)



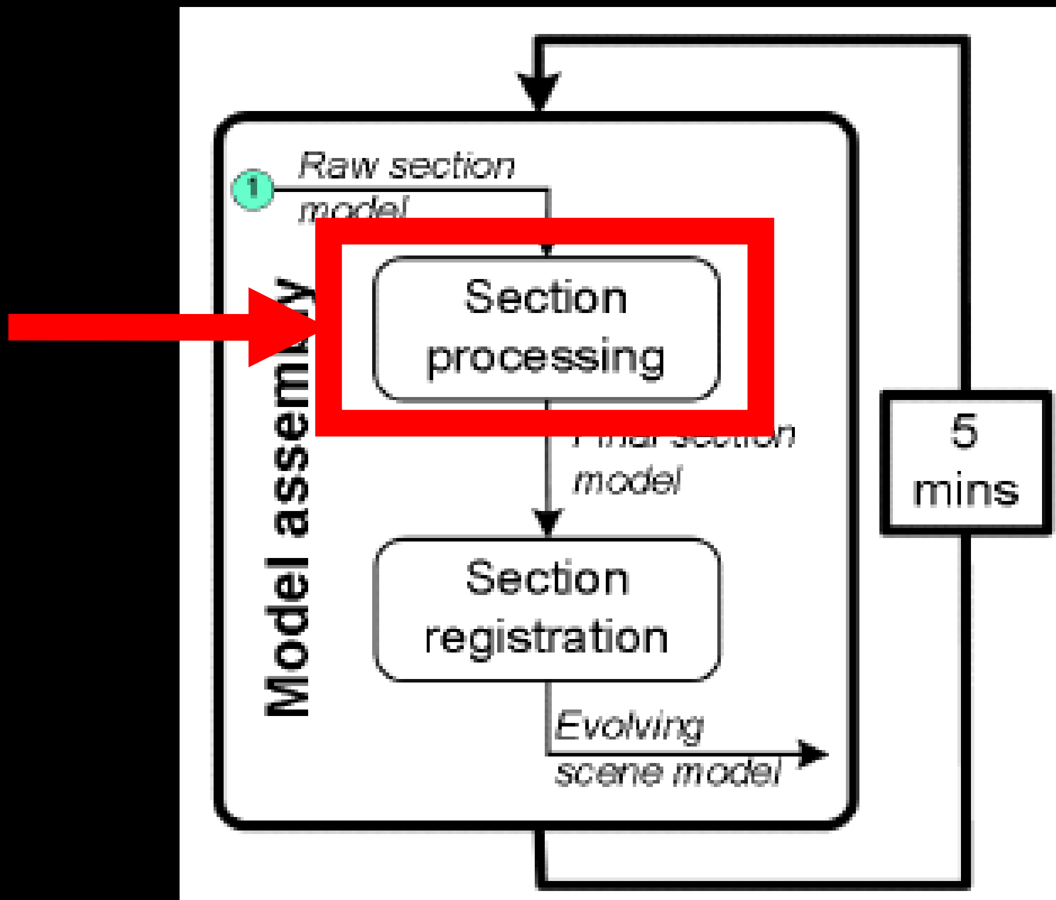
# DEP examples



# DEP examples

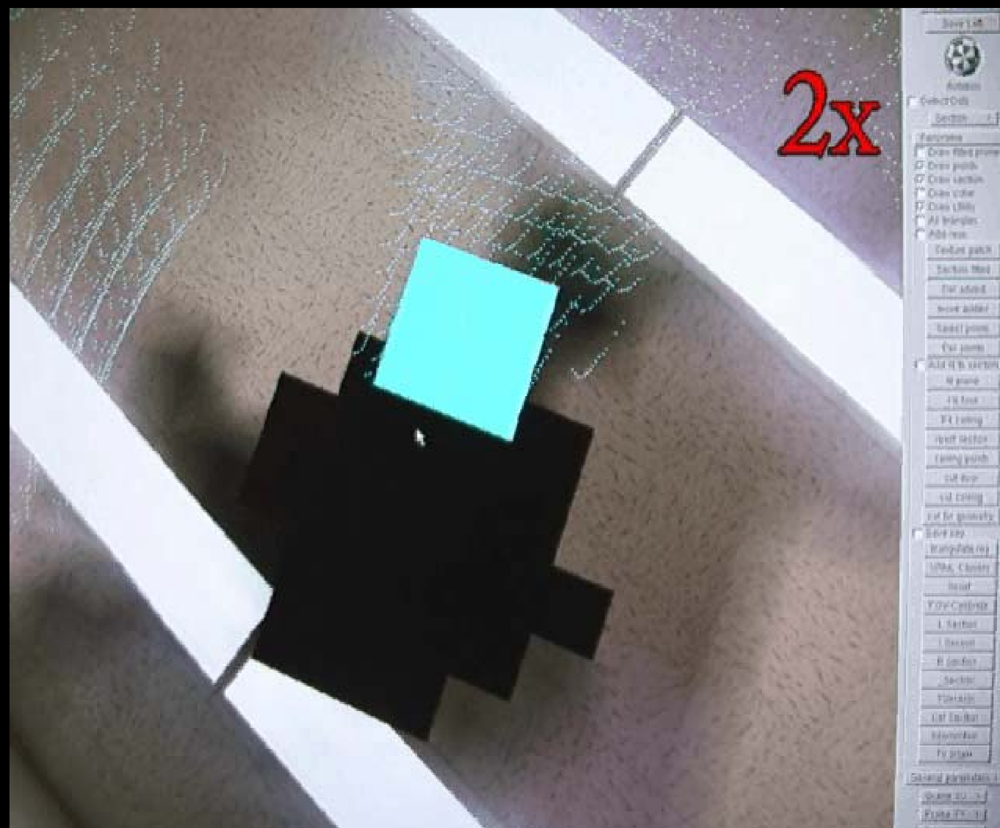
# Blending on the fly

# Section processing



# Final section model: proxy with embedded detail

# Proxy modeling



# Corridor sections occasionally need embedded geometric detail

Proxy only: water fountain appears flat

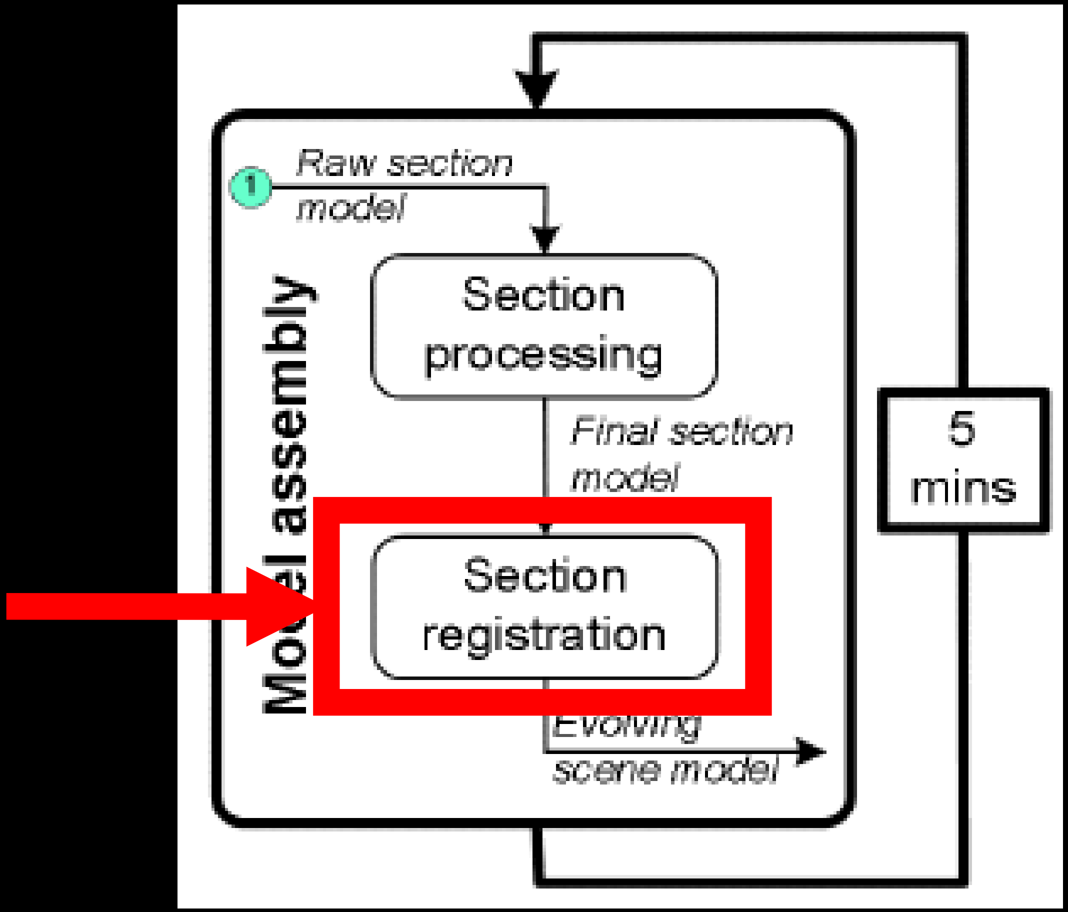
Operator selects DEP region where points should be kept

Water fountain triangle mesh

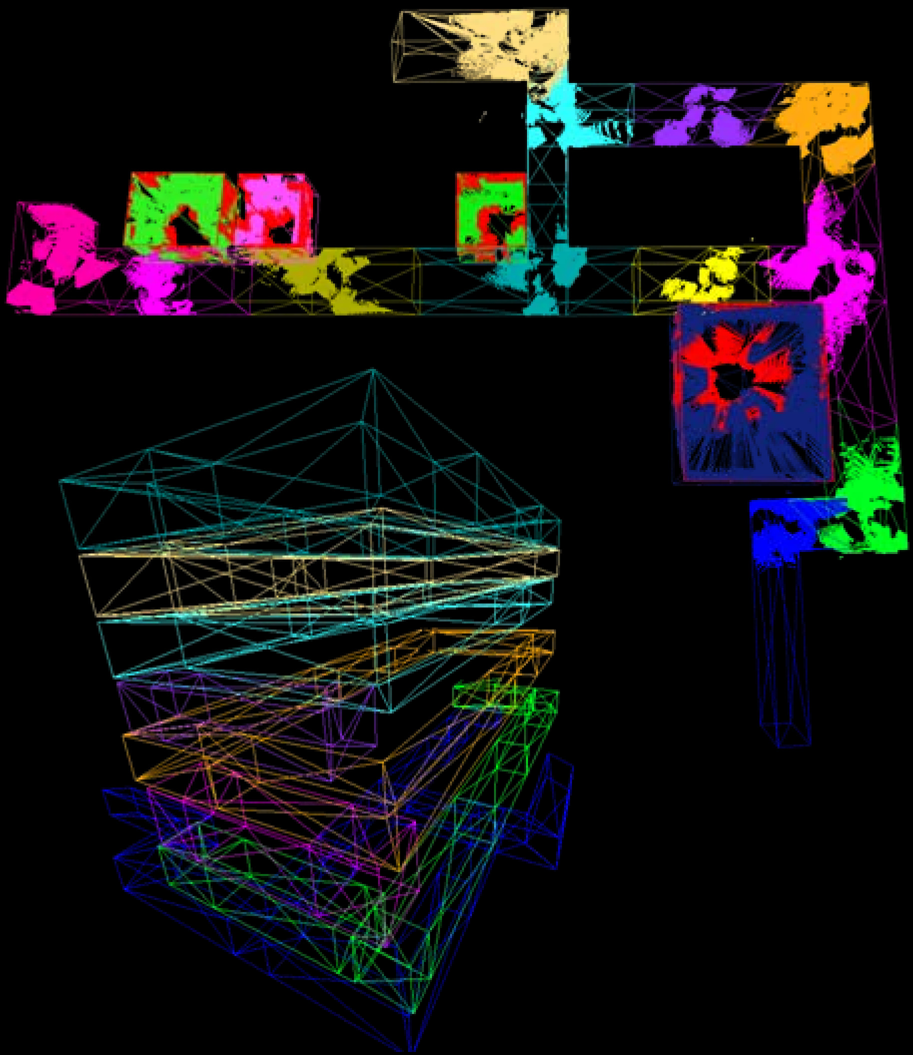
Correct parallax

# Corridor sections occasionally need embedded geometric detail

# Section registration



# Section registration



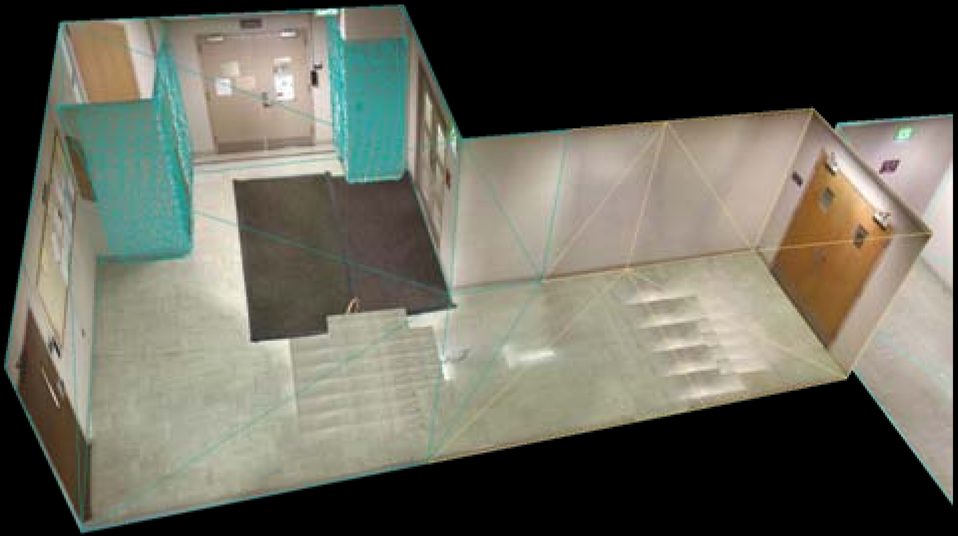
- Same-plane constraints provide at least 5 degrees of freedom
- Remaining degree of freedom resolved with operator input

# Results

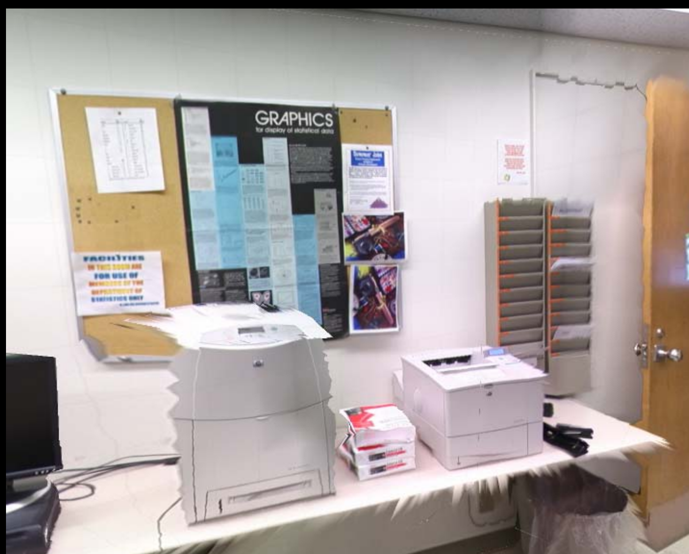
- 20 rooms & corridors of 7 floors
- 1,450$m^2$ floor space
- 2M triangles, 2GB of textures
- Corridor length error 2.5%
- 40 hours, single acquisition device, 2 person team

# Results

# Results

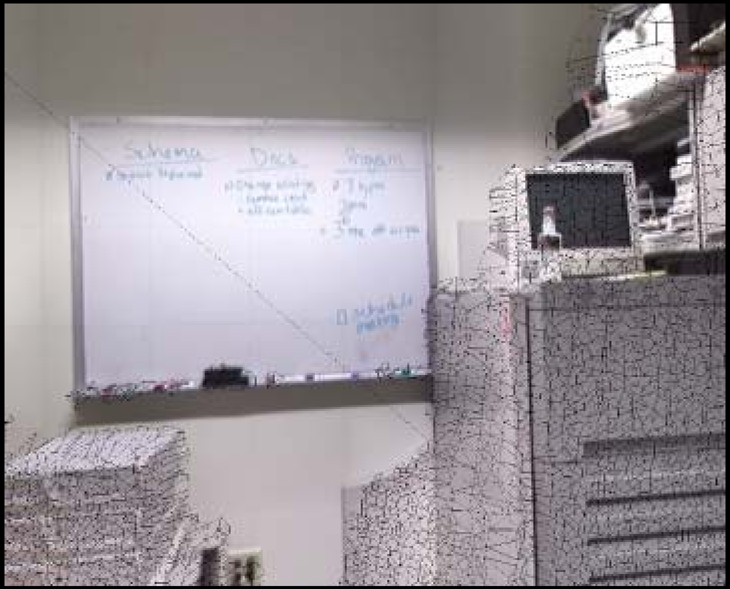


# Results




# Results

# Results



# Results

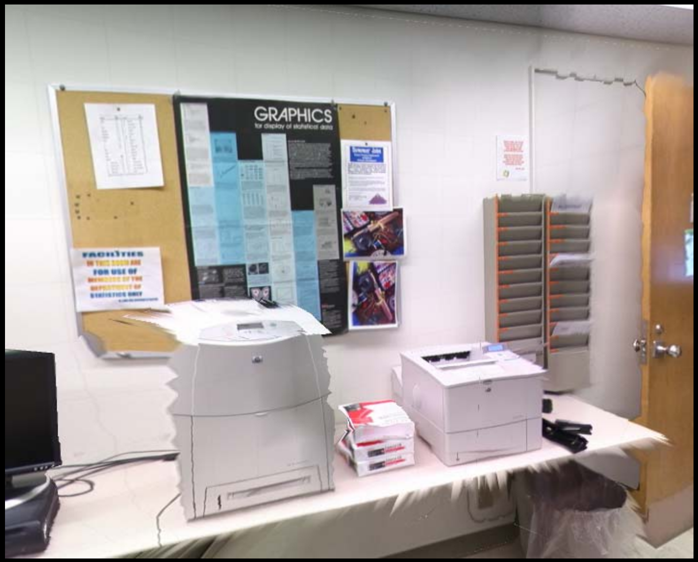
# Conclusions

- The ModelCamera system works because
  - Operator controls modeling in real time
  - Sparse depth is acquired efficiently and robustly
  - Sequences of dense color and sparse depth frames have great modeling power
  - System leverages simplifying assumptions that hold true in indoor environments

# Future work

- Take advantage of object, placement, and material repetition in large buildings
  - Model materials and light sources
- Modeling in parallel
  - 3 acquisition units served by same model assembly unit, wireless connectivity

# Future work




- Remove single acquisition viewpoint constraint

# Robust and Globally-Consistent 3D Reconstruction

PIs: Daniel G. Aliaga [*]
Mireille Boutin [x]
Students: Ji Zhang [+]
Jamie Gennis [*]

[*] Department of Computer Science
[+] Department of Mathematics
[x] Department of Electrical and Computer Engineering
Purdue University

# Motivation

- Acquiring 3D models of real-world environments is one of the great challenges of digital technology today
- However, images are discrete samples of the environment and the observations made with them are approximate, noisy, and may contain outliers
- Thus, a common challenge of approaches that reconstruct a 3D scene from photographs is to obtain a robust and globally-consistent solution

# Related Work

- Most all current formulations attempt to minimize pixel re-projection error in image space
  - Unfortunately, a small re-projection error does not necessarily imply a small structural error!

- To overcome this limitation, methods use “more images” but this only masks the problem and does not solve it
  - In fact, simultaneously solving for unknown camera rotation and camera translation is a mathematically ill-conditioned problem

# Objective

- Our goal is to eliminate the fundamentally ill-conditioned aspect of 3D reconstruction
- Reaching this goal has a *major implication for all 3D acquisition methods and applications* such as
  - Reconstruction and refinement of 3D geometry for telepresence, simulations, and 3D gaming
  - Global registration of a 3D scene for virtual reality and augmented reality applications
  - Large-scale geometry acquisition and modeling

# Approach

- Our novel formulation *eliminates* camera parameters completely from the reconstruction process
  - Thus, this removes the troublesome confusion between unknown camera parameters

- The result is dramatically better formulated reconstruction process that also requires less data and supports faster computations

# Standard 3D Formulation

corresponded 2D points $p_{ij}$

*(known)*

Reconstruction

camera centers $C_j$

scene points $P_i$

disparity $\lambda_{ij}$

camera translation $T_j$

camera rotation $R_j$

*(known)*

Degree 1 (linear) reconstruction…

# Standard 3D Formulation

corresponded 2D points $p_{ij}$

*(known)*

Reconstruction

camera centers $C_j$

scene points $P_i$

disparity $\lambda_{ij}$

camera translation $T_j$

camera rotation $R_j$

Degree 2 (non-linear) reconstruction…

*(and ill-conditioned!)*

# Our Novel 3D Formulation (I)

corresponded 2D points $p_{ij}$

*(known)*

→ **Reconstruction** →

camera centers $C_j$

scene points $P_i$

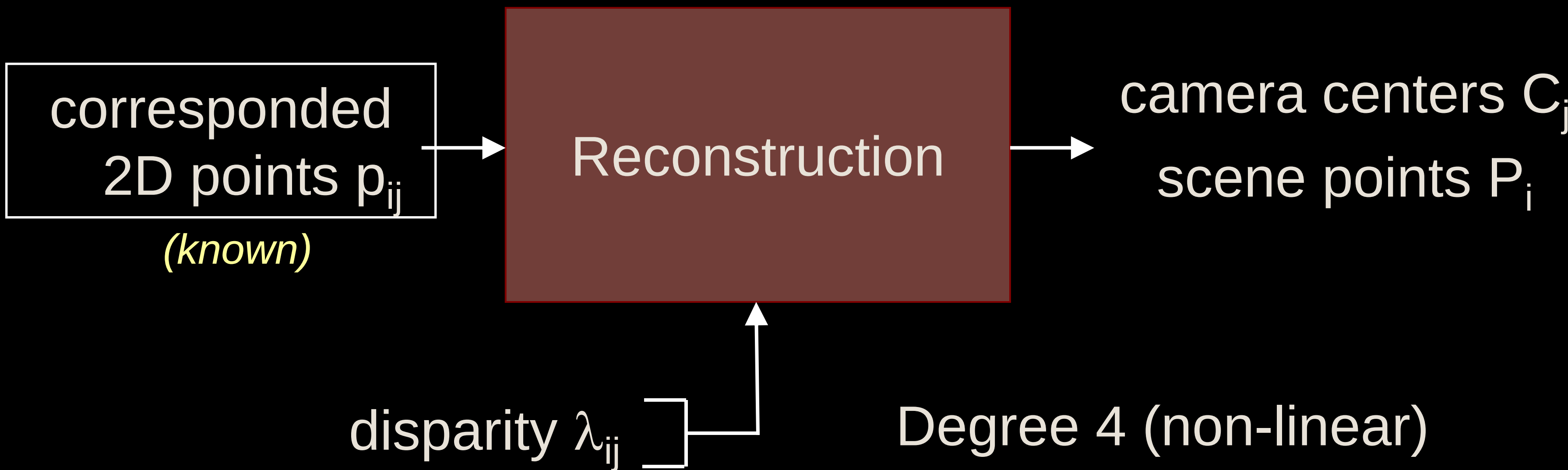
disparity $\lambda_{ij}$

camera translation $T_j$

Degree 2 (non-linear) reconstruction…

*(and well-conditioned!)*

# Our Novel 3D Formulation (II)

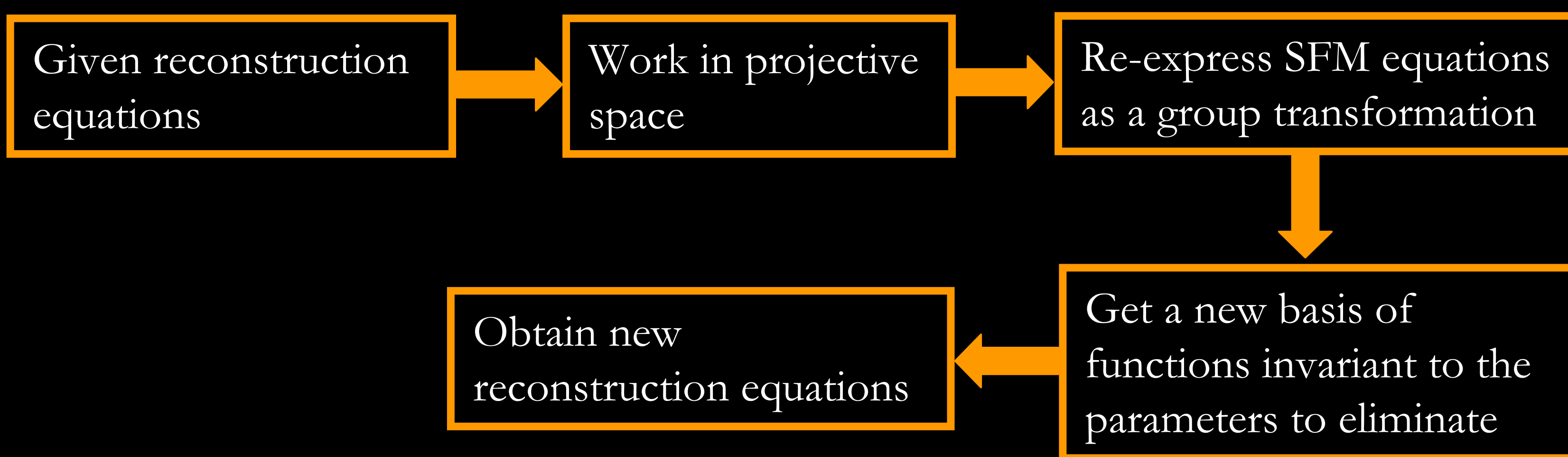



Degree 4 (non-linear) reconstruction…

*(and well-conditioned!)*

# Key Idea

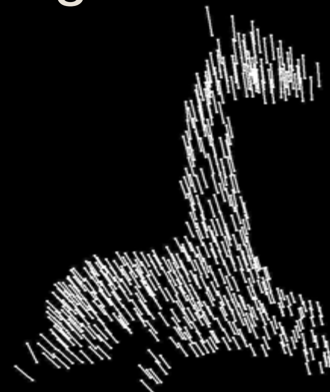
- Algebraically eliminate camera parameters from the reconstruction equations
  - General variable elimination is hard
  - However, by going to projective space and using invariant-based methods in the context of differential geometry, we are able to eliminate variables and maintain a low-degree polynomial formulation



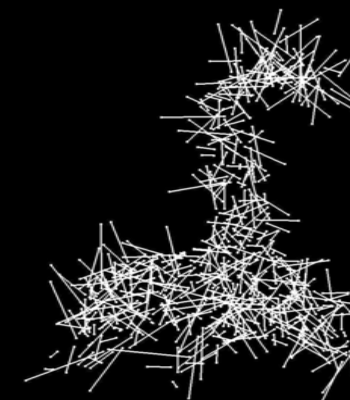
# Results: Structure Recovery

- Our method shows significant robustness to noise

Actual Model



Others (medium noise)



Others (large noise)

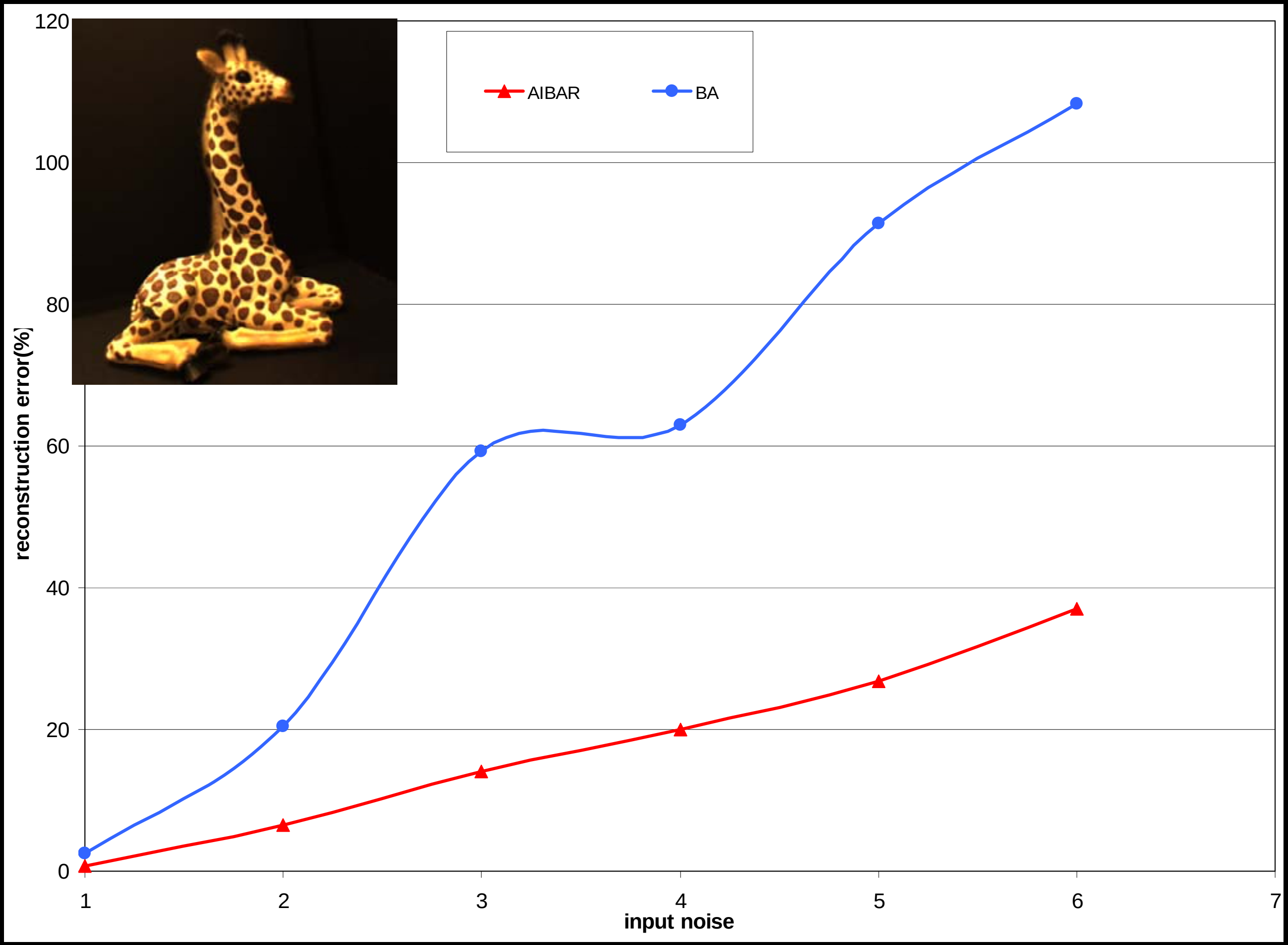
Ground Truth

Us (medium noise)

Us (large noise)

# Results: Structure Recovery

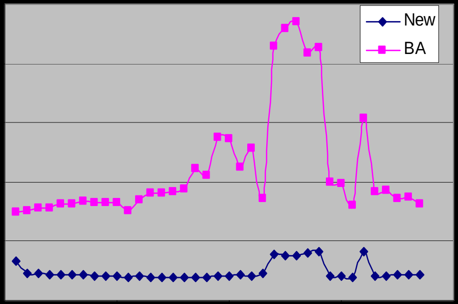
- Our method shows significant robustness to noise

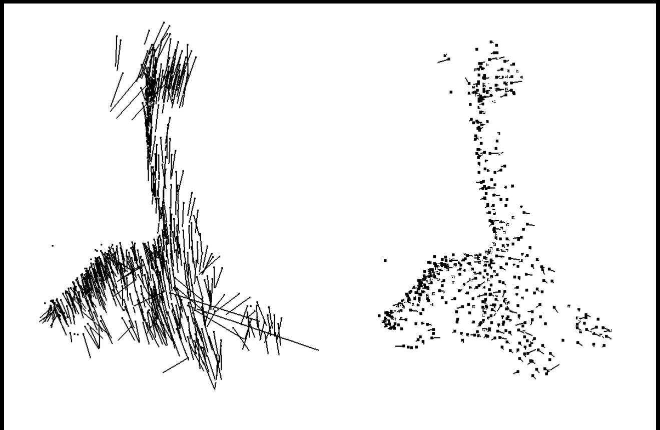


# Results: 3D Registration

- Registration improves by 10:1 on average

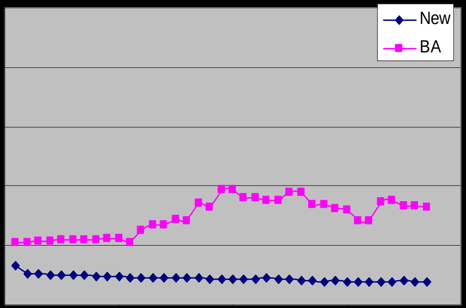
Re-projection error



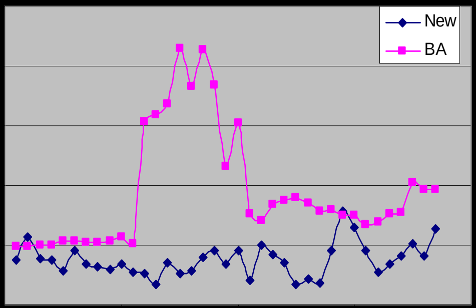



scene　　others　　us

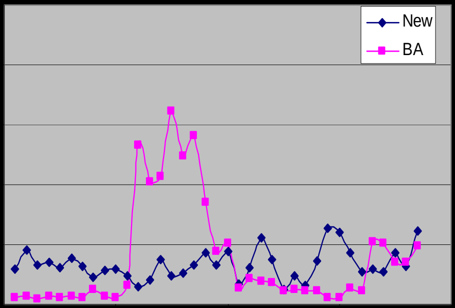
Scene point error




Camera center error



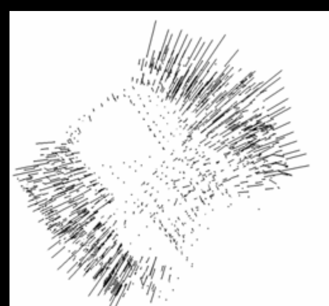

Camera rotation error



# Results: 3D Registration Initialization

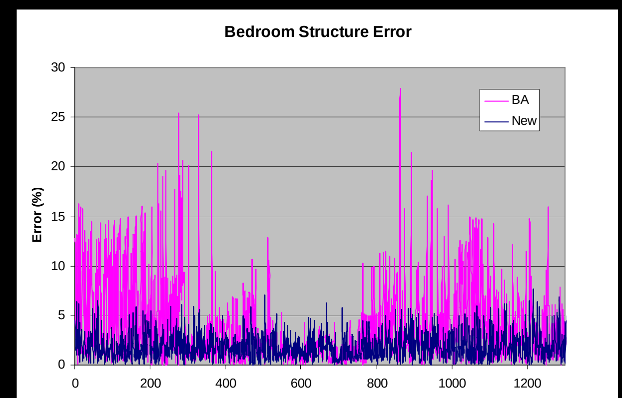
- Even at one-meter initial error, we converge at up order of magnitude less error and only needing a fraction of the data

scene



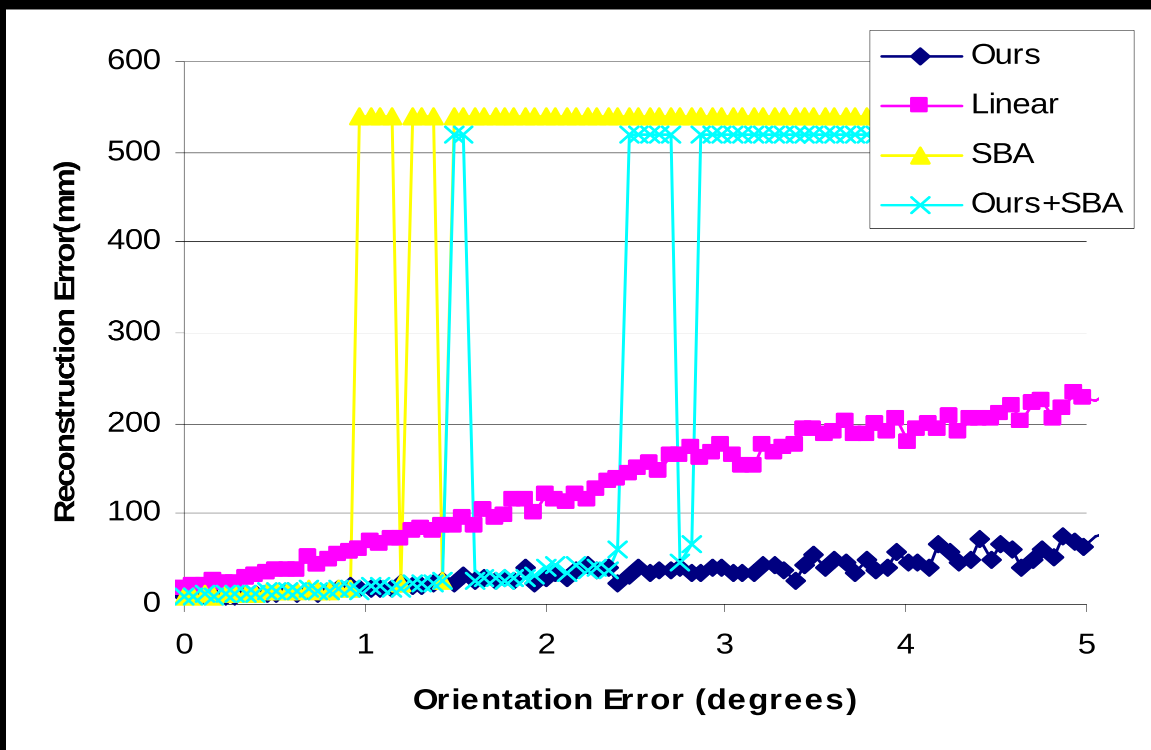
others

us



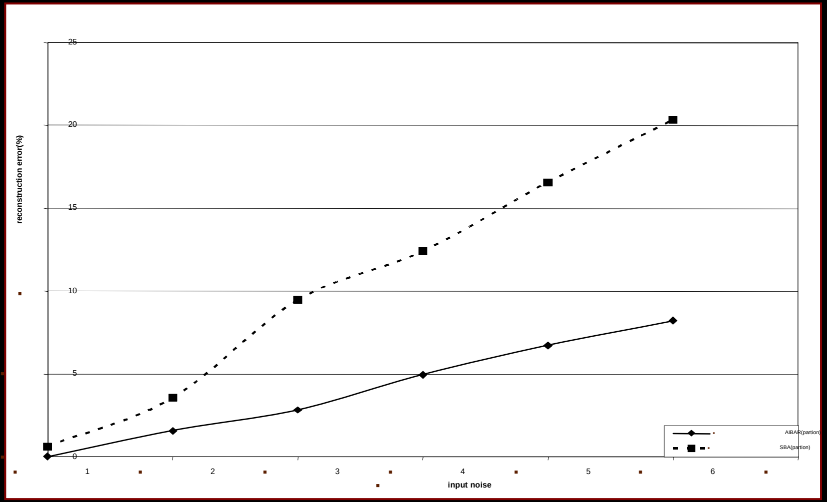
# Results: Ground Truth Comparison

- A ground-truth comparison shows our robustness
- A subsequent further optimization-pass of our results yields no improvement, indicating *our solution is already best*




# Results: Large Datasets

- Our formulation affords a *fast and linear-time method* when provided with camera-centers, enabling very fast reconstruction of large-datasets



# Conclusions

- We have presented novel formulations of the 3D reconstruction equations that improve robustness and global-consistency of the numerical computations
- Our approach serves to improve 3D registration, useful in virtual and augmented reality
- Our method serves to refine structure recovery from images, applicable to content creation
- Our technique leads to a fast reconstruction method for large-scale acquisition and modeling

# Future Work: Completely pose-free

corresponded 2D points $p_{ij}$

*(known)*

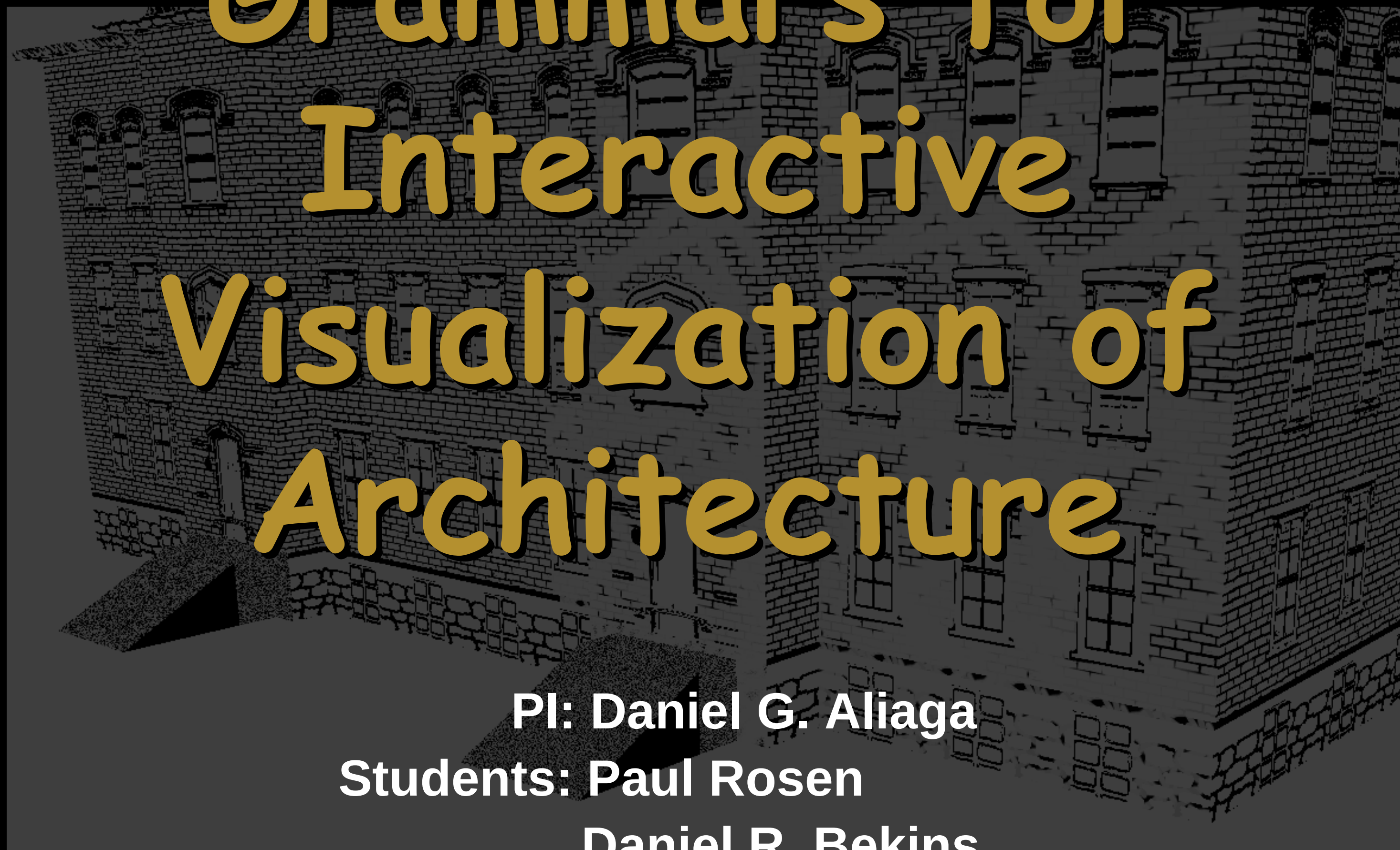
→ Reconstruction →

camera centers $C_j$

scene points $P_i$

Work in progress…

# Inferring Style Grammars for Interactive Visualization of Architecture




PI: Daniel G. Aliaga

Students: Paul Rosen

Daniel R. Bekins

Department of Computer Science

Purdue University

# Motivation

- Interactive visualization of architecture provides a way to see current structures as well as future structures and tentative changes to existing buildings
- A common design challenge is to require little effort by the user to alter observed architecture and buildings
- Thus, the system must at least semi-automatically and instantly infer the low-level details of the changes from only a few specifications on part of the user

# Procedural Modeling

- Procedural modeling is a powerful paradigm that can be coupled with interactive rendering to generate plausible details of a model without much user interaction
- These methods exhibit a high-degree of detail amplification, i.e. given only a small number of parameters, significant content can be generated
- Consequently, a small change in the parameters or rules can yield a drastically different result

# Related Work

- Forward-generating procedural models have been proposed for restricted arenas, e.g.:
  - L-system for plants
  - Shape grammars
  - Procedural methods for buildings and architecture

- However, all these methods assume the “rules” are provided *beforehand* by the user

# Approach

- Our inverse-modeling approach is to infer a grammar for creating architecture and buildings, thus enabling rapid visualization of new buildings “in the style” of others
  - We *parse* previously captured images to create a grammar
  - We *derive* new buildings using the data from the grammar

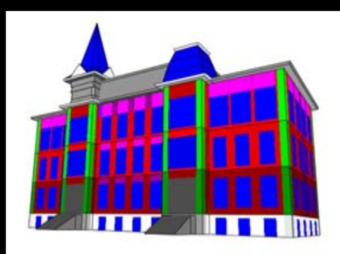
- This affords interactively rendering new buildings using
  - View-dependent texture mapping (e.g., photorealistic views)
  - Stylized procedural rendering (e.g., artistic views)

- The discovered redundancy within the images also allows us to
  - Fill-in occluded and poorly sampled areas
  - Equalize color and lighting between images and surfaces of the model

# Approach: Example

Photographs

(parse)



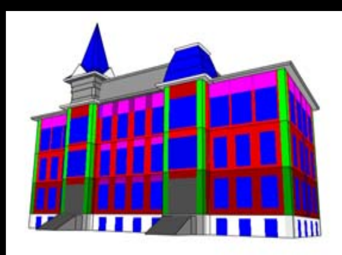
(derive)

Building

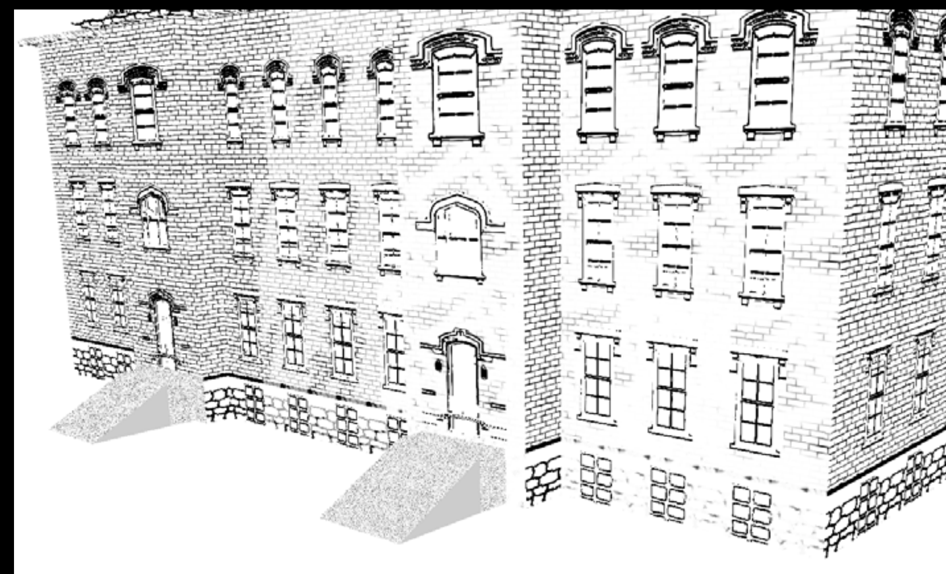
# Approach: Example

Photographs

(parse)



(derive)

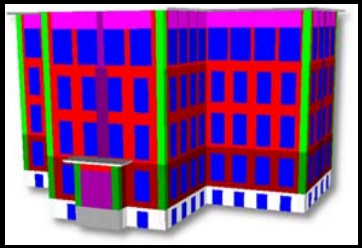


Building

# Approach: Example

Photographs

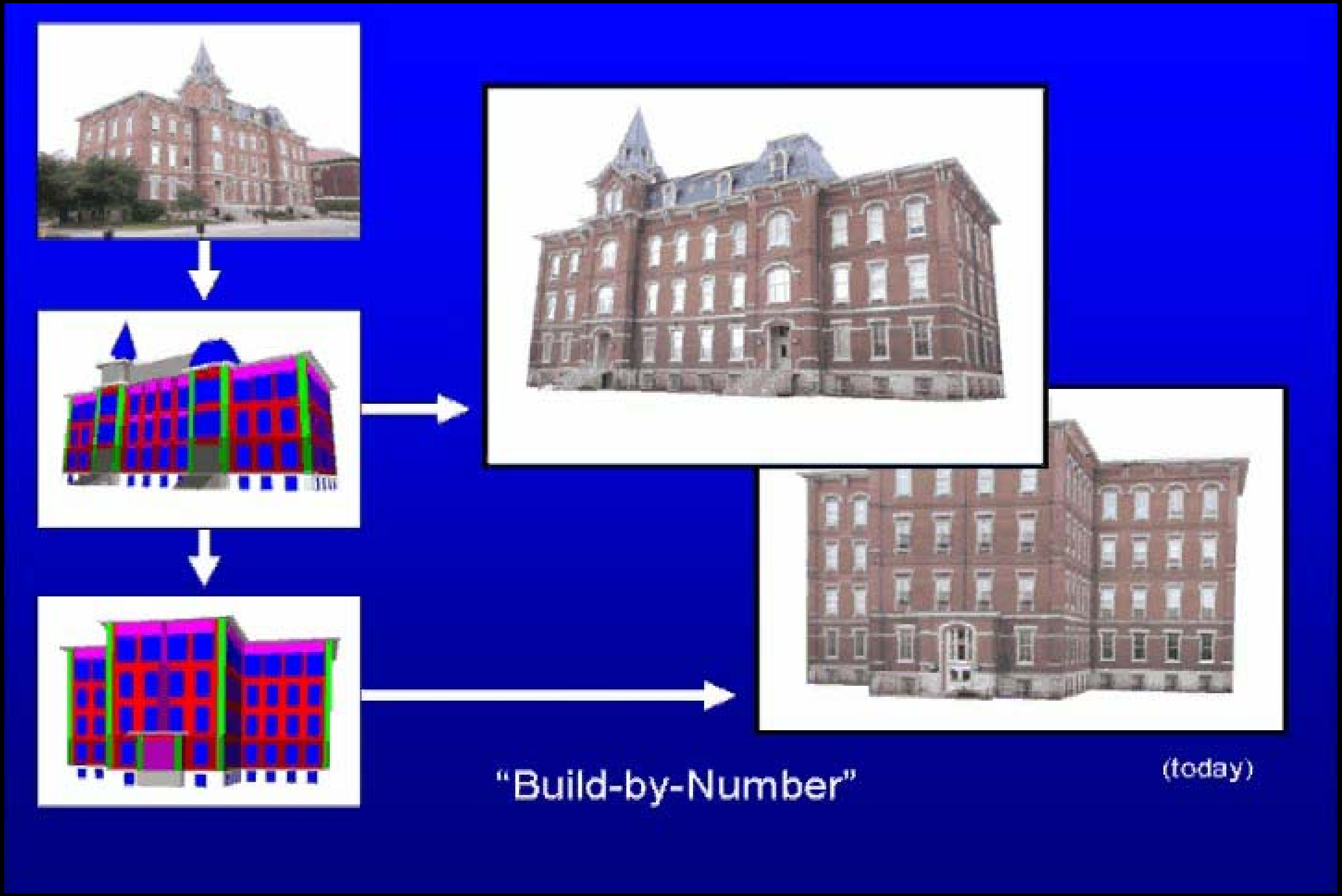
(parse)



(derive)

New Building

# Examples

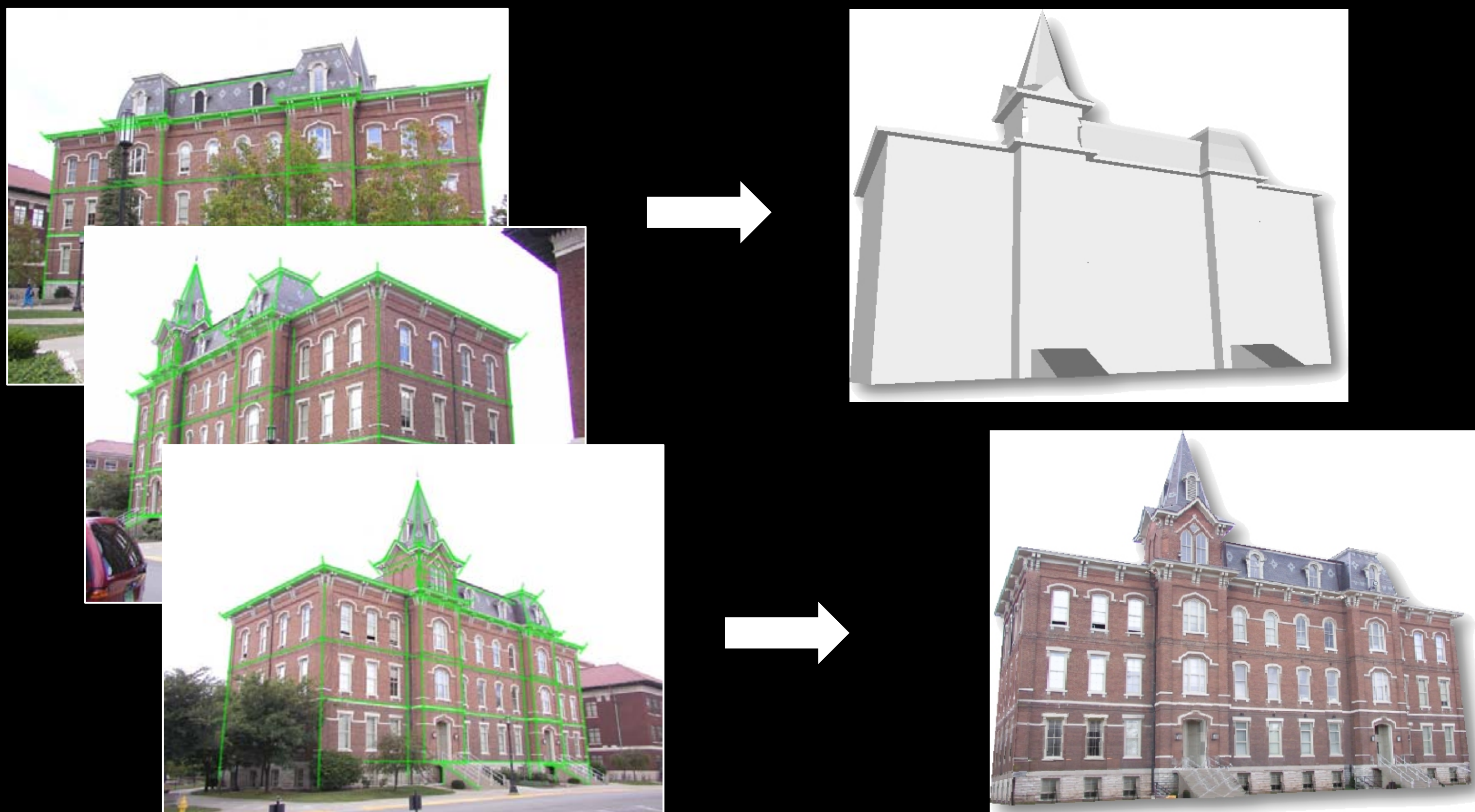


# Approach

- Building Specification
  - From photographs to model specification
- Parsing
  - Automatically parse the specification into a grammar
- Deriving
  - Given a new specification, automatically derive a new building
- Rendering
  - Draw the building using texture mapping or stylistic rendering

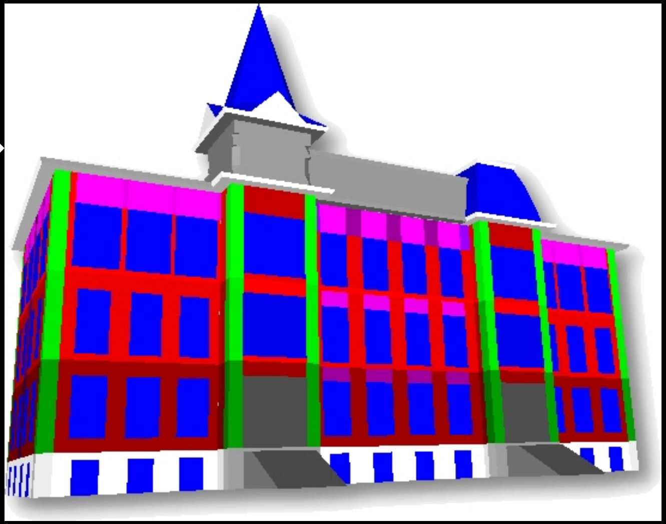
# Building Specification

1. Geometric model is recovered from a sparse image set.



# Building Specification

2. The model is subdivided into feature regions such as brick, windows, and doors. Identical or similar features are grouped together.

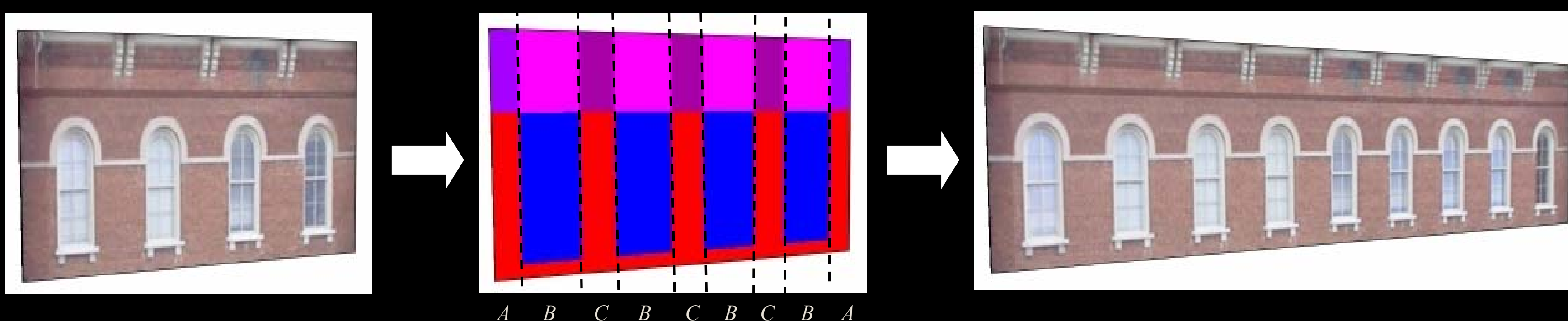


# Parsing and Deriving

- Building specification is parsed into a style grammar:
  - Model M -> (base) (ground) { $S_1$ … $S_n$ } (roof)
  - Floor S -> { $F_1$ … $F_M$ }
  - Face F -> { $C_1$ … $C_P$ }
  - Column C -> { $T_1$ … $T_R$ }

# Face Productions

- A face production contains symbols for each individual column and geometric information for determining precisely how many repetitions of each column to use when creating a novel face of arbitrary size.



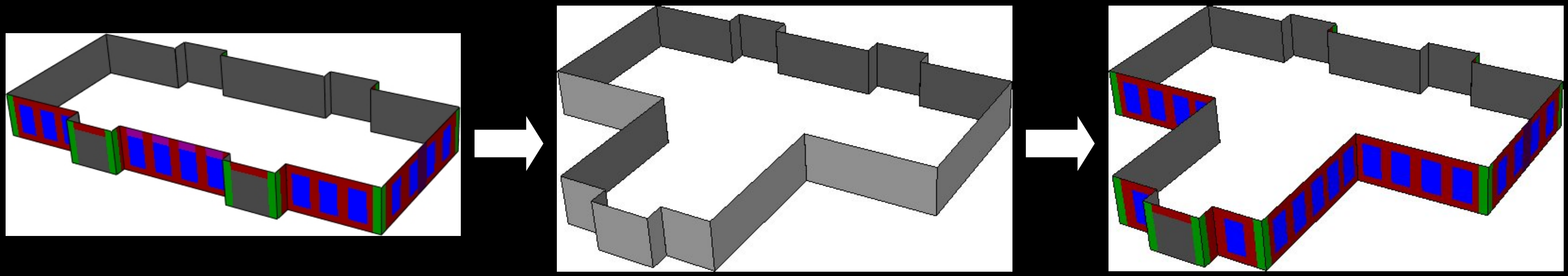

$$F = ABCBCBCBA$$

$$F \rightarrow A(BC)^{*}BA$$

$$F' = ABCBCBCBCBCBCBCBCBA$$

# Floor Productions

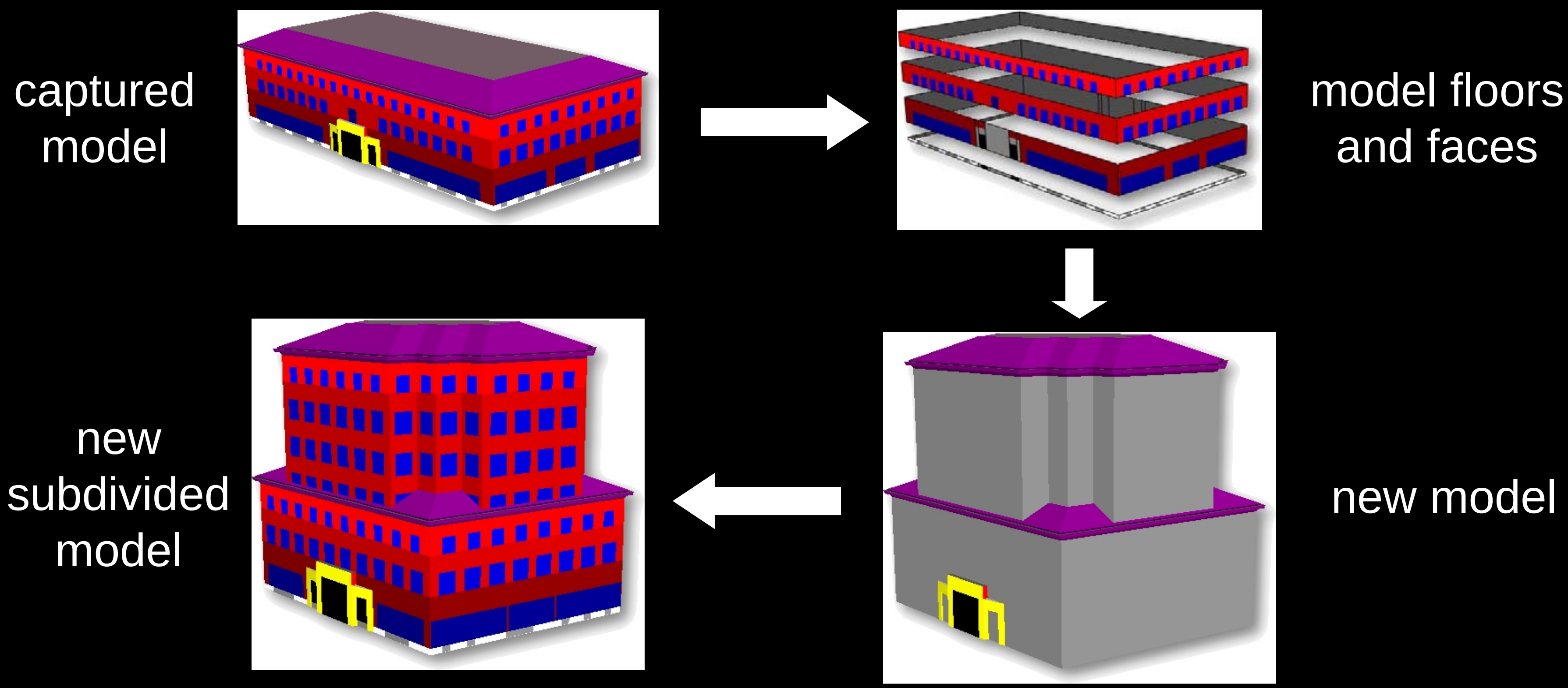
- A floor production rule is a description of the outward facing surface of a single floor wrapping around a model and serves to make new floors



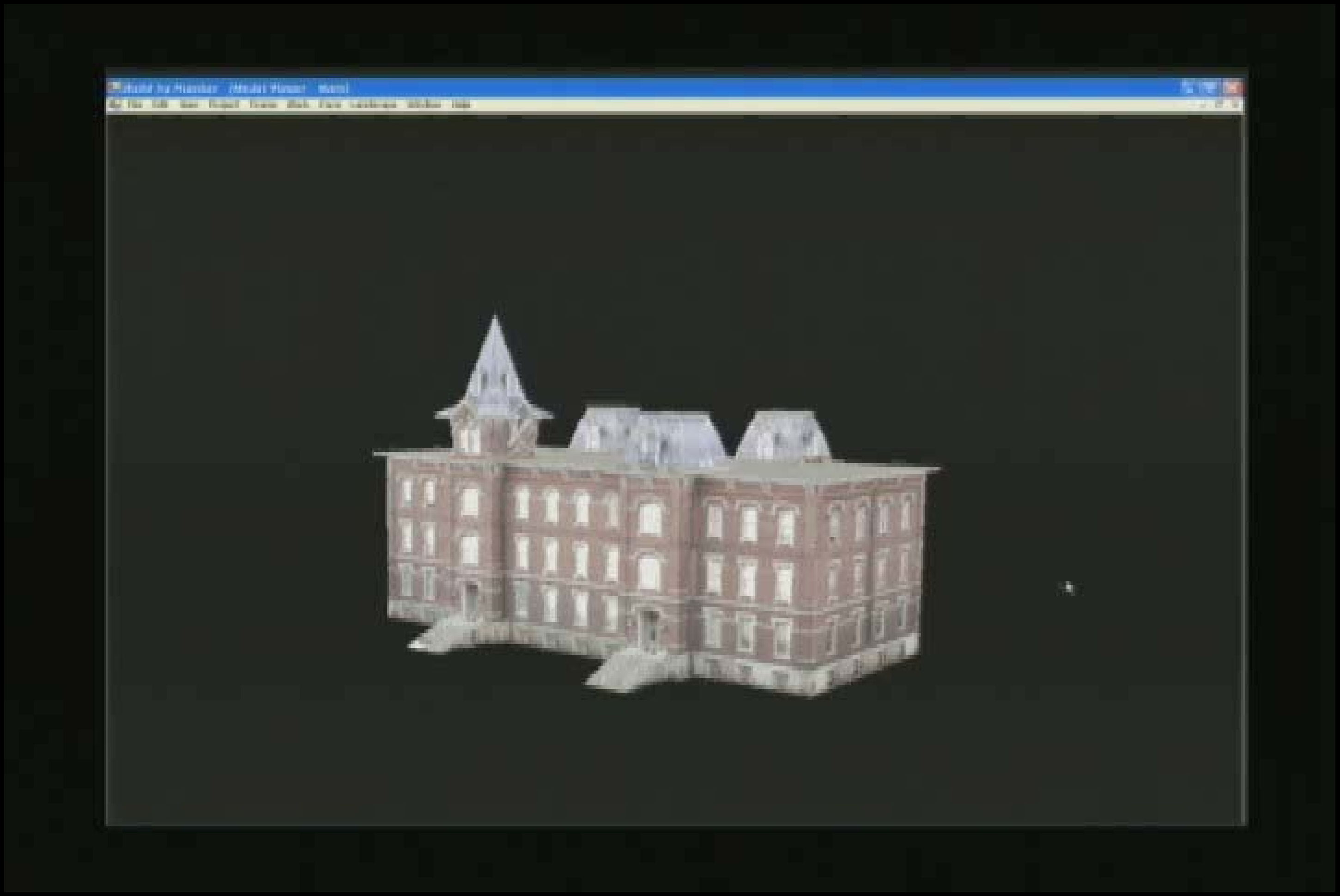
captured floor     new floor     new subdivided floor

# Model Productions

- A model production rule consists of all the aforementioned rules and enables subdividing a given collection of building blocks in a single operation

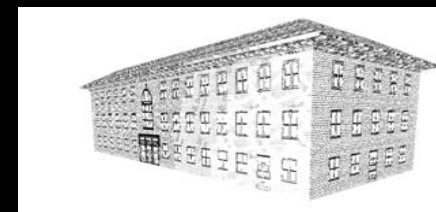


# Style Grammars



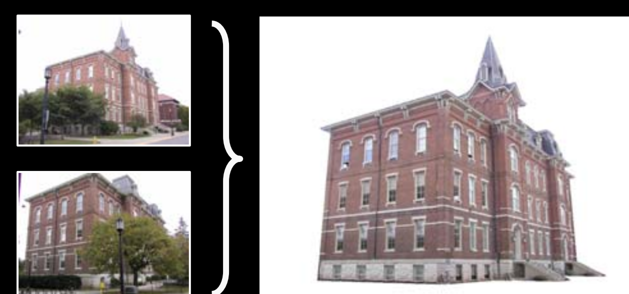
# Rendering

- Stylized Rendering
  - e.g., pen-and-ink drawings



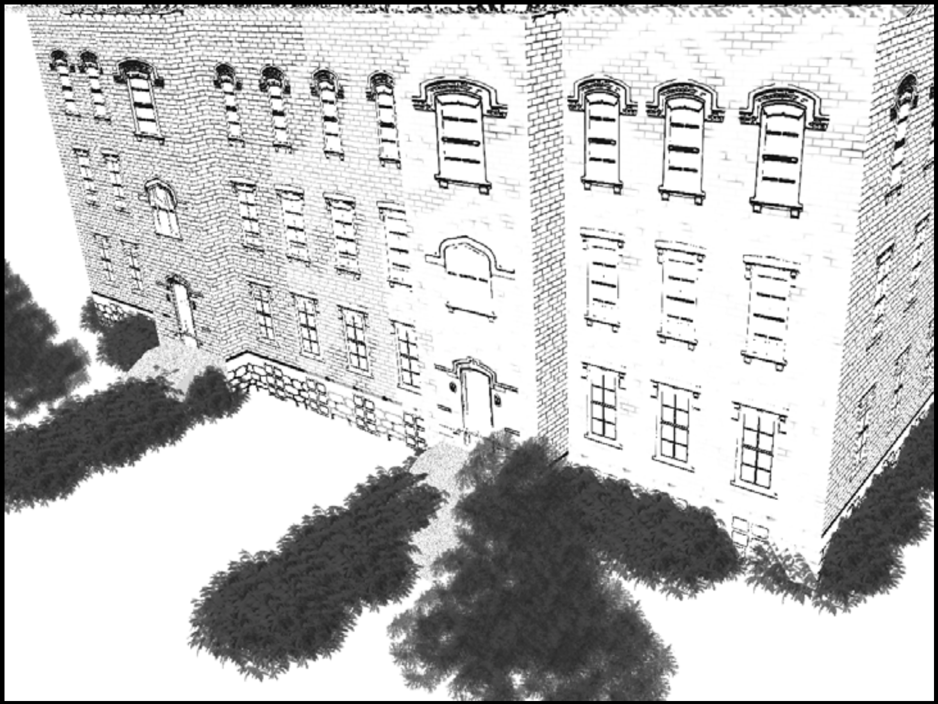
- Photorealistic Rendering
  - e.g, view-dependent texture mapping

- Occlusion-removal
  - Use redundancy within the samples to remove unwanted “occlusions”



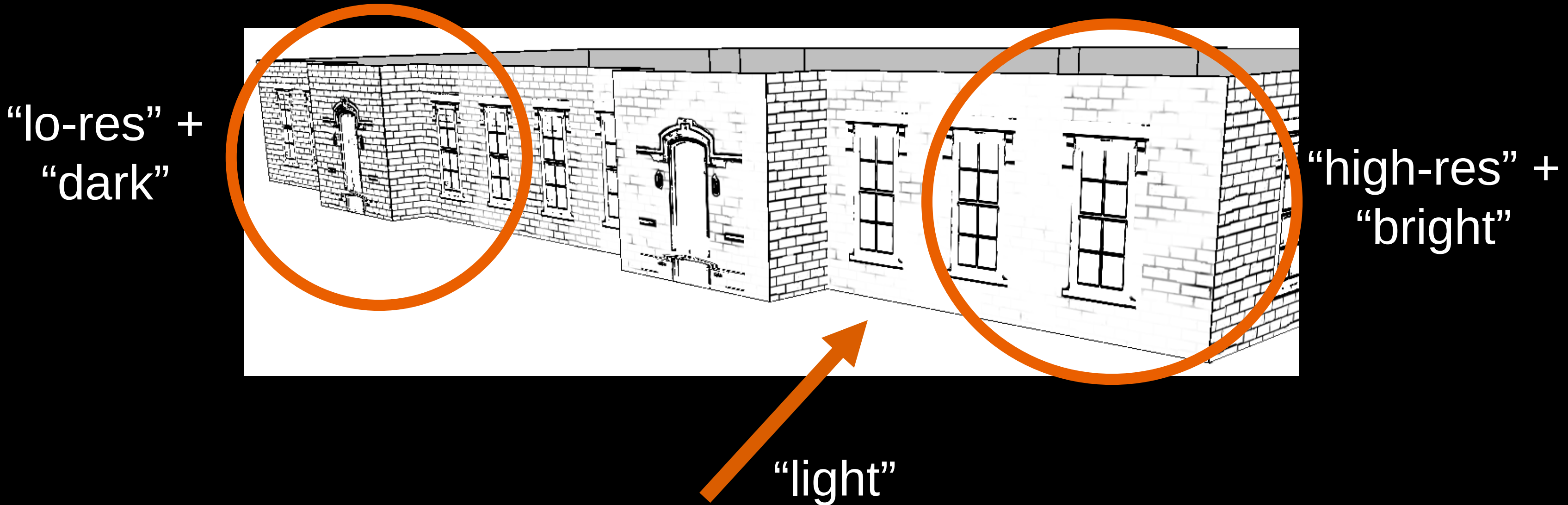
# Stylized Rendering

- The partitioning of the building features into a relatively small number of terminal types enables several forms of stylized rendering



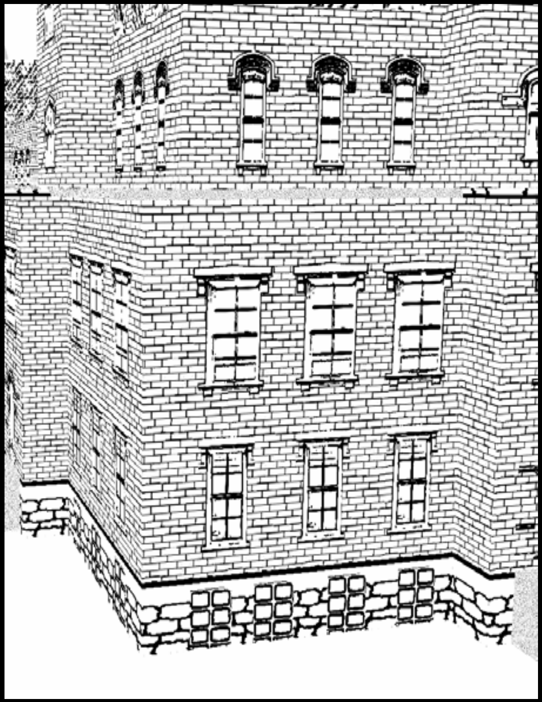
# Stylized Rendering

- From captured images a procedural version is inferred that spans both *resolution space* and *tonal space*

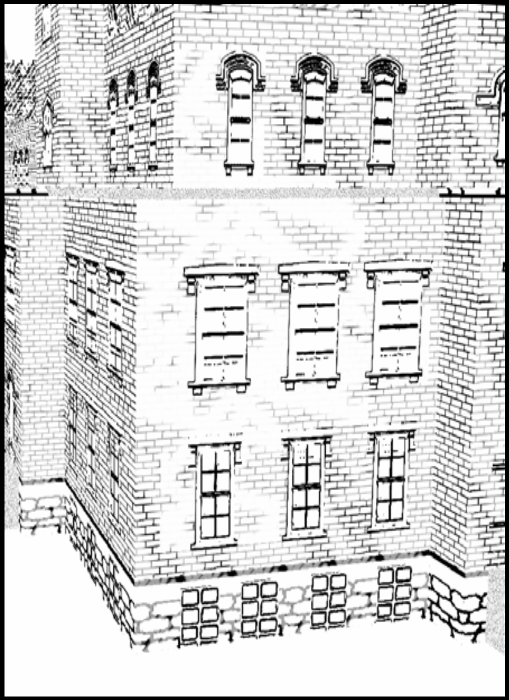


# Stylized Rendering

- From captured images a procedural version is inferred that spans both resolution space and tonal space



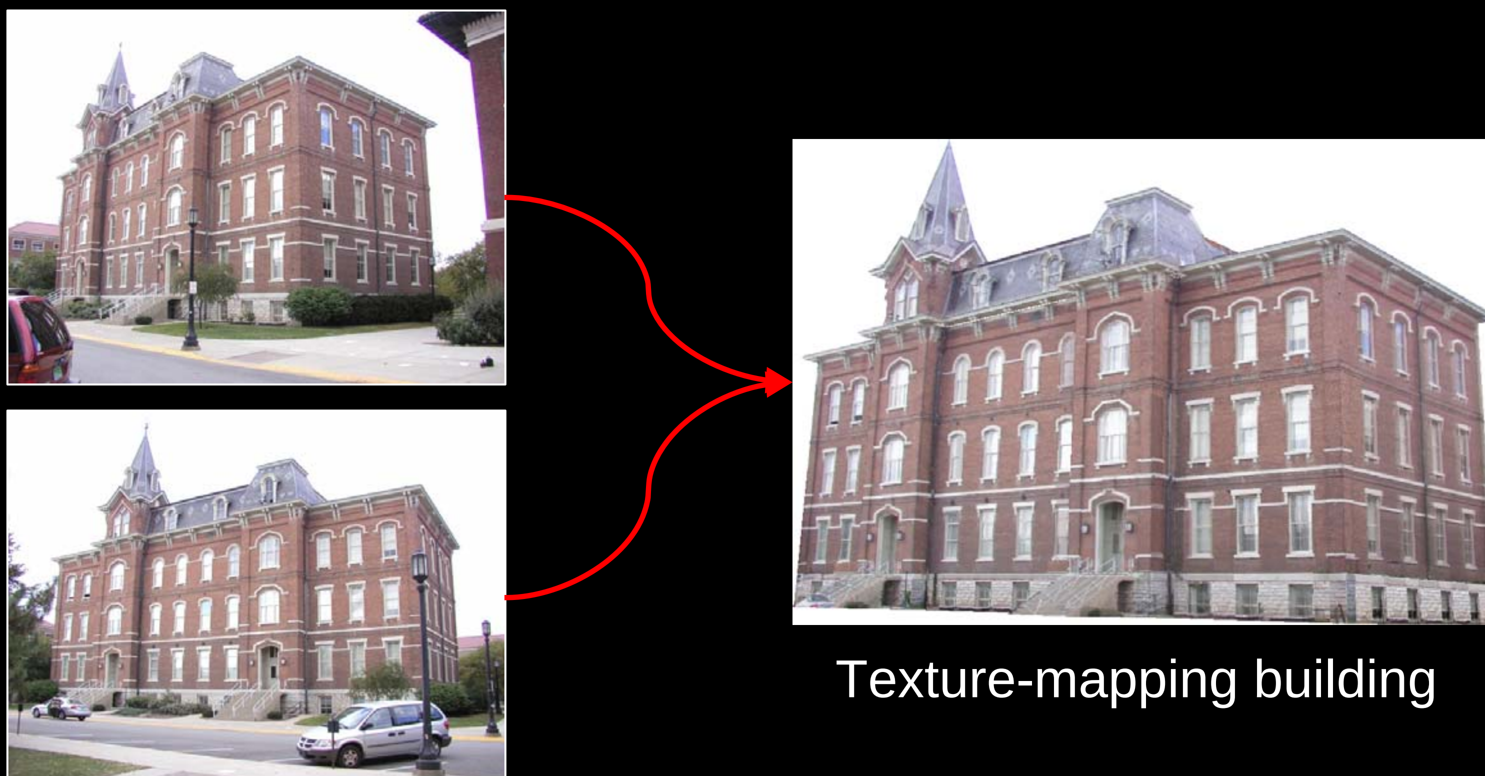
Constant shading



Diffuse shading model with stylization

# Projective Texture Mapping

- Novel views are rendered by blending the closest source views to the current view



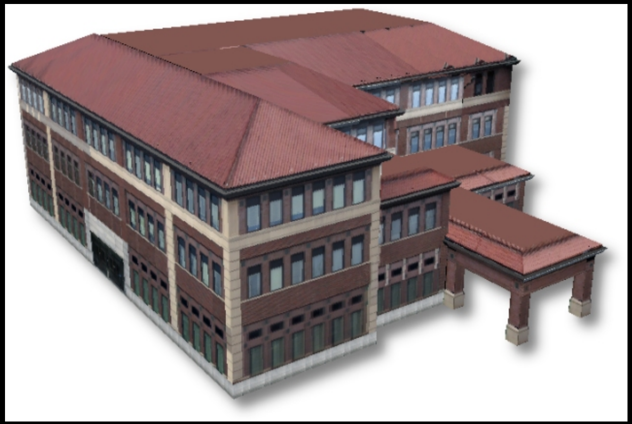
Source photographs

Texture-mapping building

# Projective Texture Mapping

- Novel views are rendered by blending the closest source views to the current view

Photograph



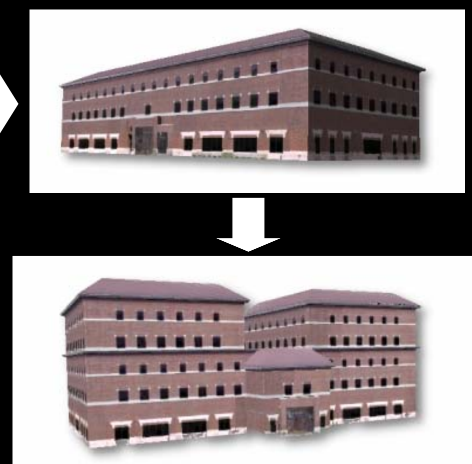
Novel building

Novel building plus landscaping

# Projective Texture Mapping

- Novel views are rendered by blending the closest source views to the current view

Photograph



Model editing

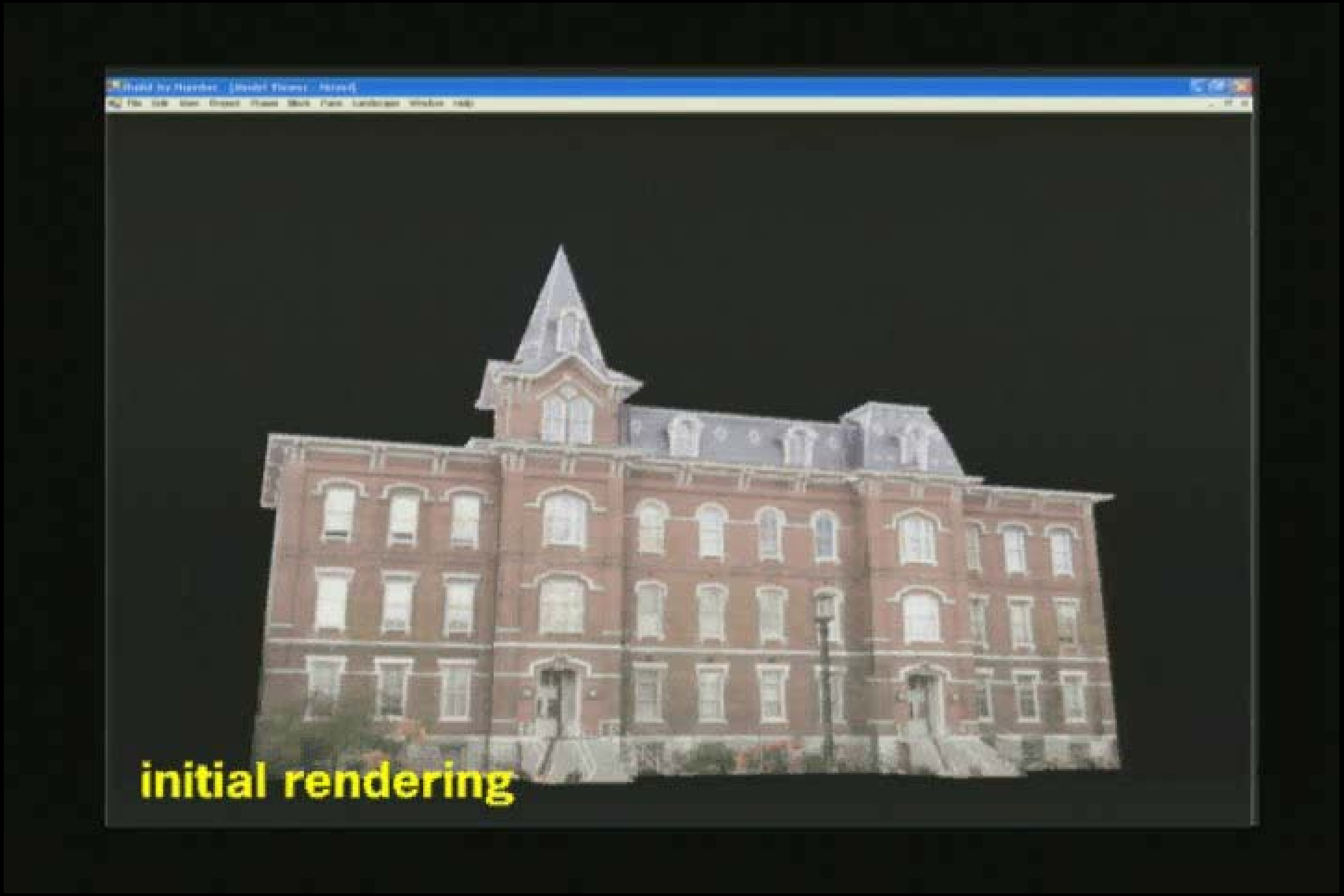
In-place viewing

# Occlusion Removal

- Our method uses the inferred production rules and multiple instances of each terminal type to replace parts of the building occluded in the original images.

basic rendering → occlusion-free → color equalized

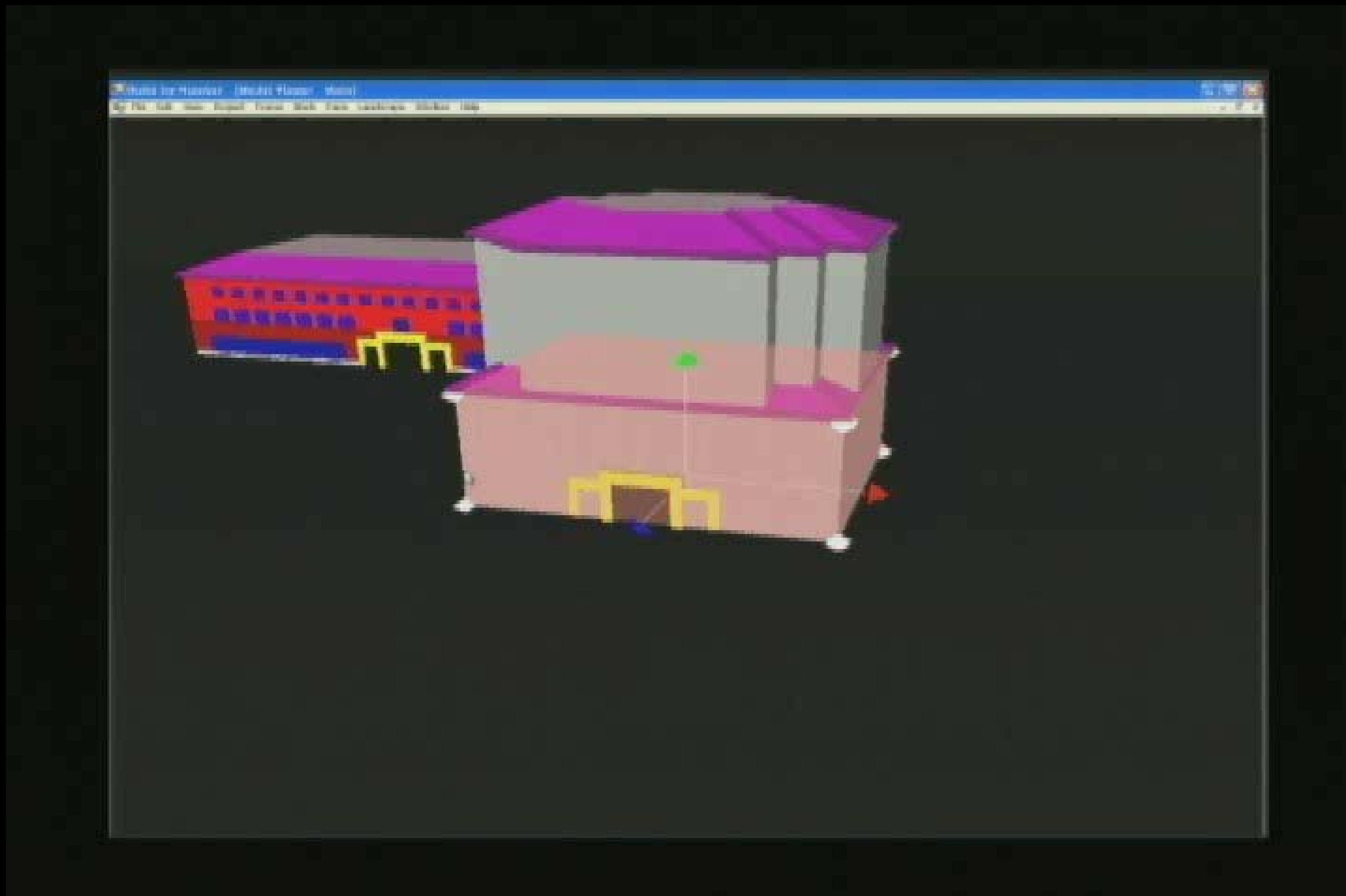
# Occlusion Removal

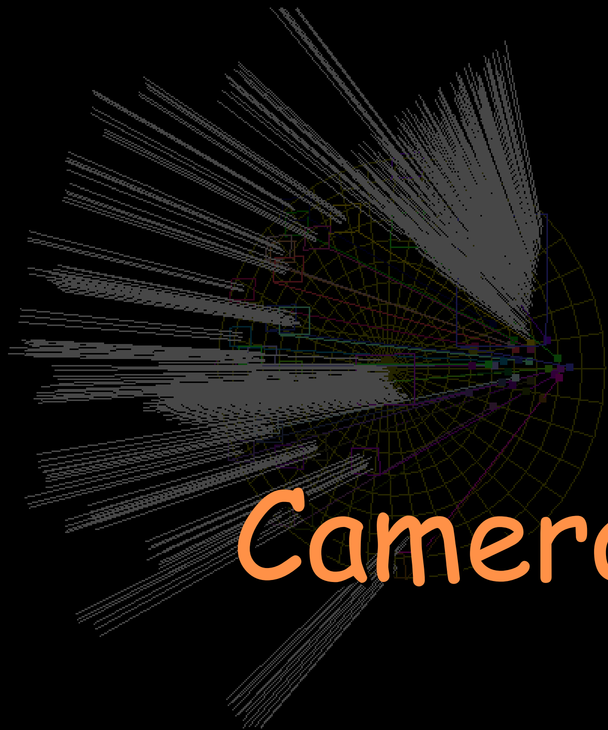


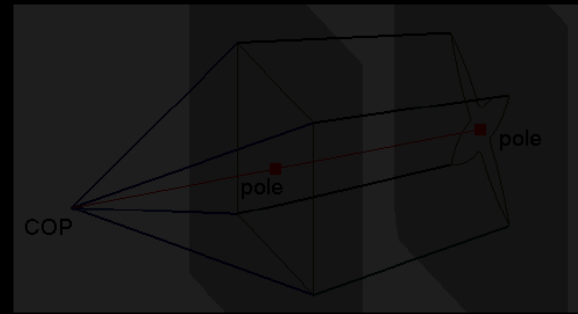
# Conclusions and Future Work

- Method to *infer from real-world images a grammar for creating buildings* and enabling the rapid sketching of novel architectural structures in the style of the original
- Supports *stylized rendering* for sketching and intuitive viewing
- Supports photorealistic *projective texture map rendering*
- *Occlusion-free views* can be generated despite no such viewing existing in the original images
- Future Work
  - Discover higher-level patterns and styles
  - Extend grammar to include an entire urban space (work in progress)

# Examples

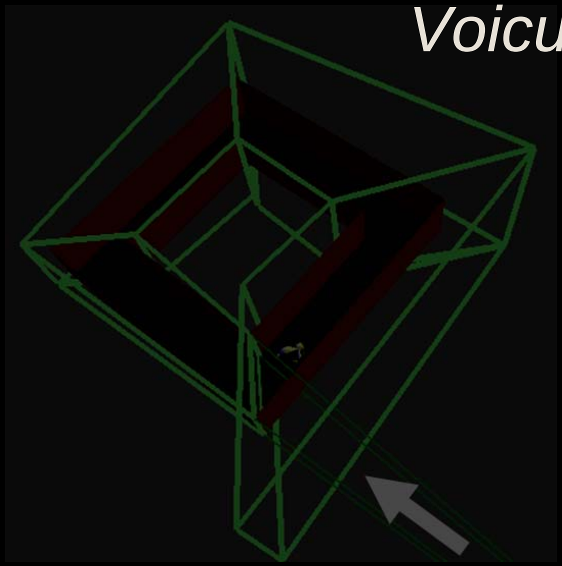


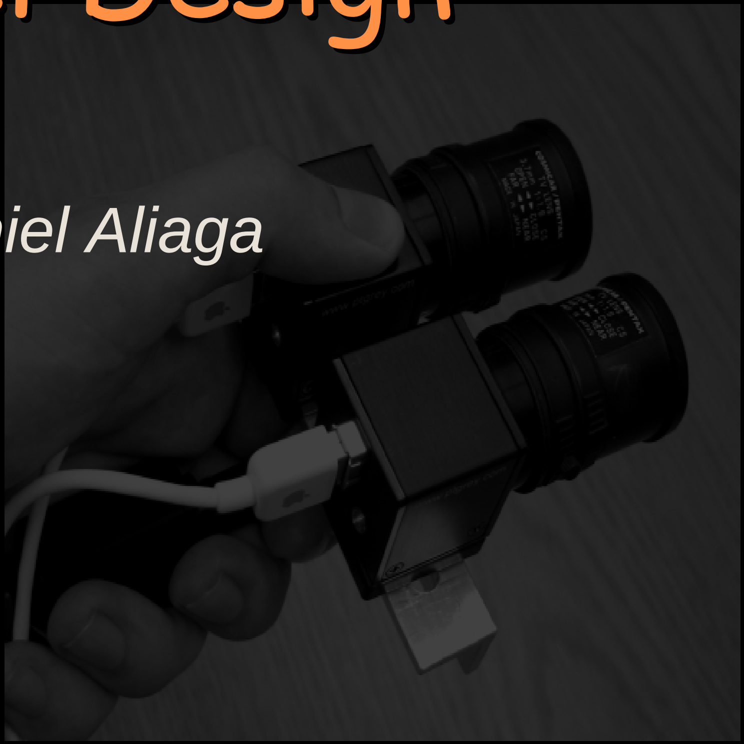





# Camera Model Design

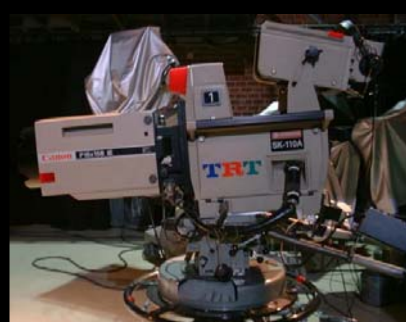
*Voicu Popescu, Daniel Aliaga*





# Motivation

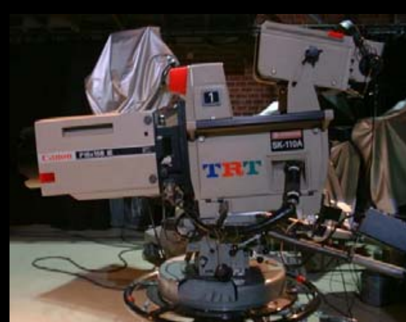
- Camera models are essential infrastructure in graphics, visualization and vision
  - Definition: a camera model is a mapping from pixels to rays
- The dominant model is the planar pinhole camera
  - Approximates human eye well
  - Has simple software and hardware implementations



# Motivation

- Camera models are essential infrastructure in graphics, visualization and vision
- The dominant model is the planar pinhole camera
- Planar pinhole camera model is very restrictive
  - Limited field of view
  - All rays must pass through common point (pinhole)



# Field of view limitation has been overcome

# Little has been done to remove pinhole limitation

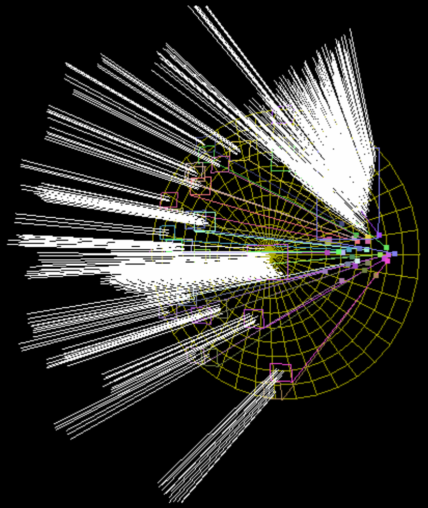
- Apprehension vis a vis non-pinhole cameras based on misconceptions
  - “images produced are not useful”
  - “images can only be produced at the cost of substantial computational effort”

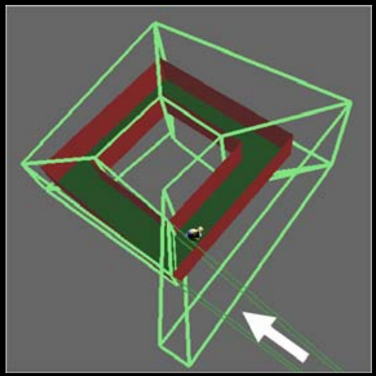
# We propose a novel paradigm for graphics, visualization, and vision

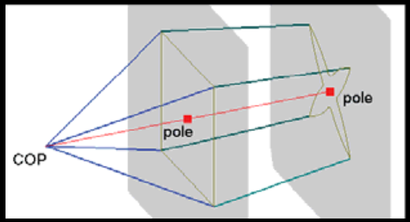
- Camera model design paradigm
  - ***Instead*** of using one of a few off-the-shelf camera models
  - Design non-pinhole camera model for each application
  - Optimize for data set at hand
- Approach: two steps
  - Choose rays of interest—*effectiveness*
  - Devise algorithm for fast projection—*efficiency*

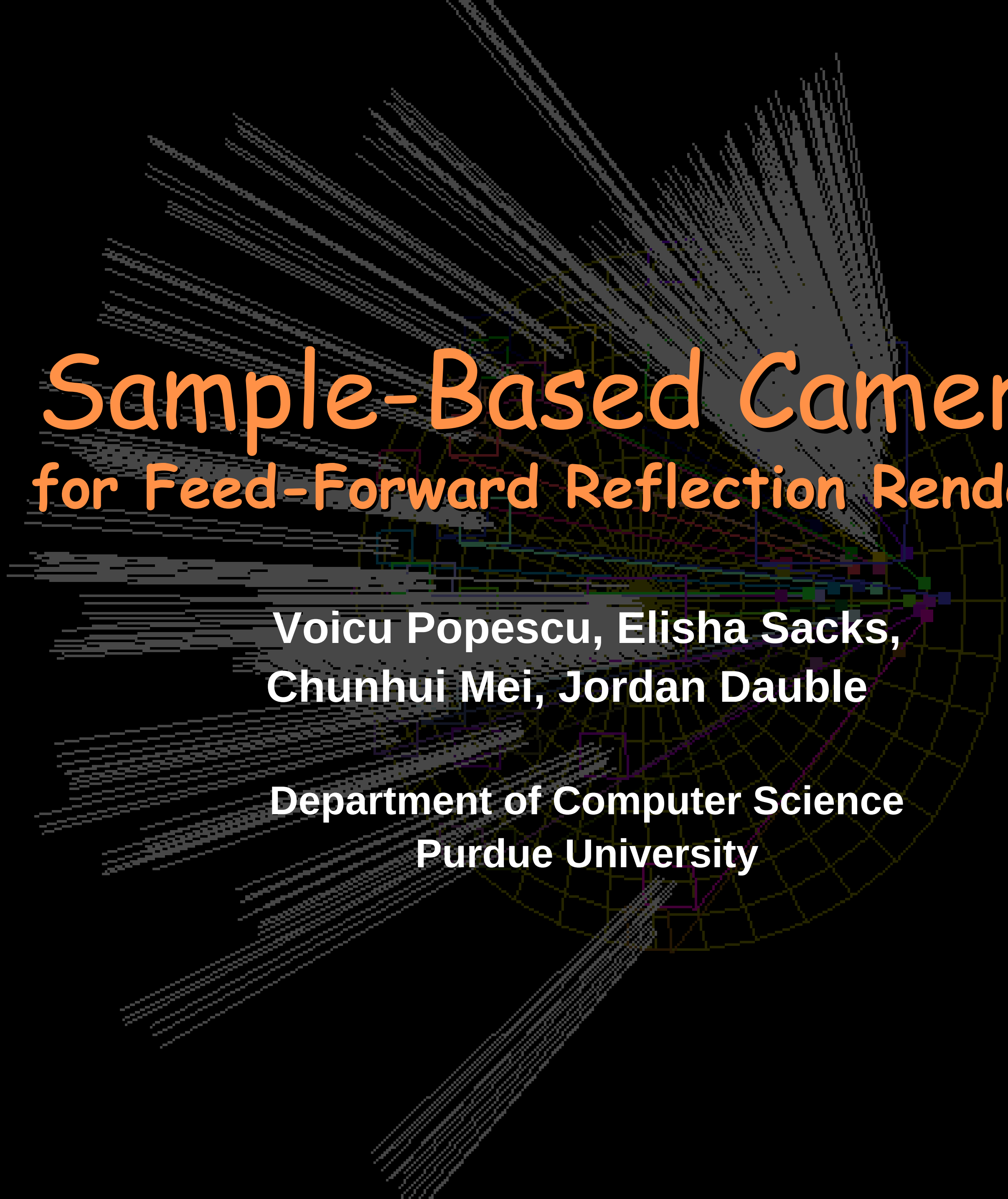
# Camera Model Design

- Applied successfully to 4 challenging problems
  - Sample-Based Cameras for feed forward reflection rendering
  - Occlusion Cameras for disocclusion-error-free reference images
  - Graph Camera for comprehensive single-image visualization
  - Occlusion-Resistant Camera Model Design







# Sample-Based Cameras
## for Feed-Forward Reflection Rendering


**Voicu Popescu, Elisha Sacks, Chunhui Mei, Jordan Dauble**

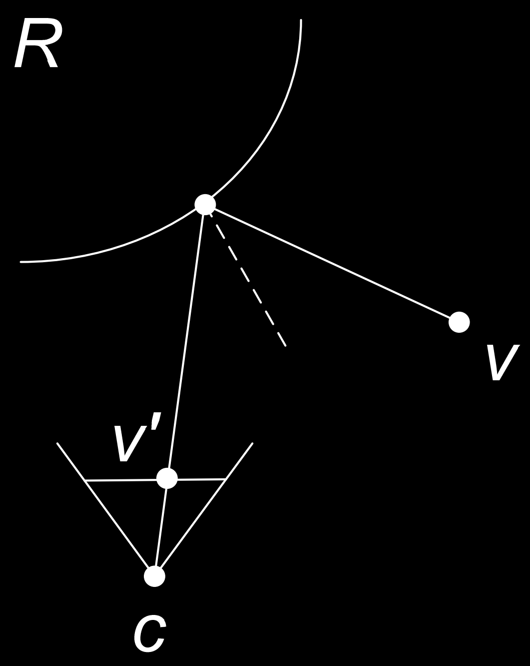
**Department of Computer Science**
**Purdue University**

# Motivation

- Reflections are important
  - Encountered frequently
  - Convey surface properties
  - Convey relative position of objects

# Motivation

- Reflections are important
- Reflections are challenging
  - No closed form projection of reflected points



# Motivation

- Reflections are important
- Reflections are challenging
  - No closed form projection of reflected points
  - Current interactive methods employ drastic approximations

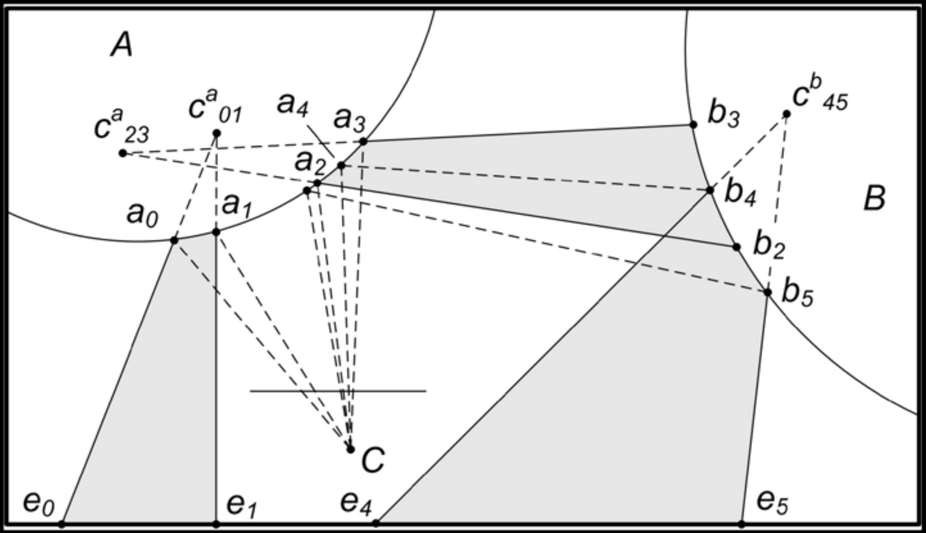
Prior art, 60 fps.

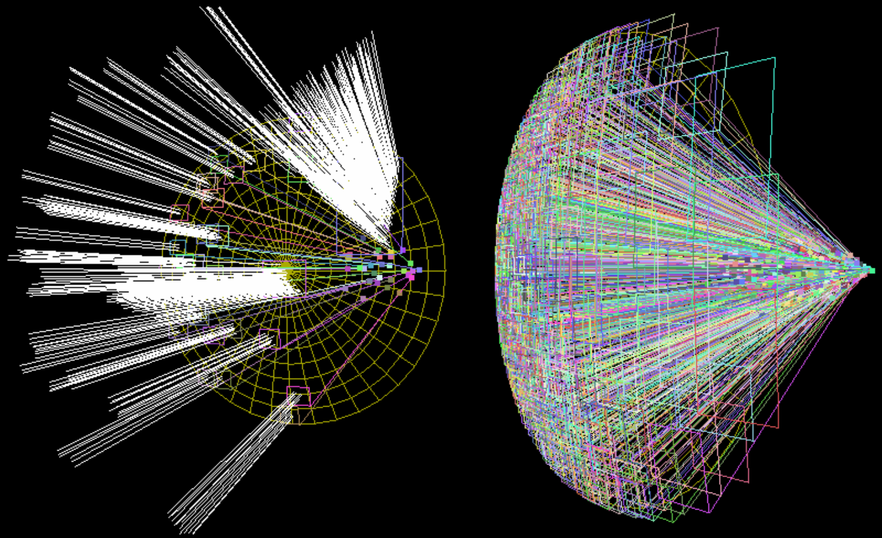
Our method, 60 fps.

Prior art, 1 fps.

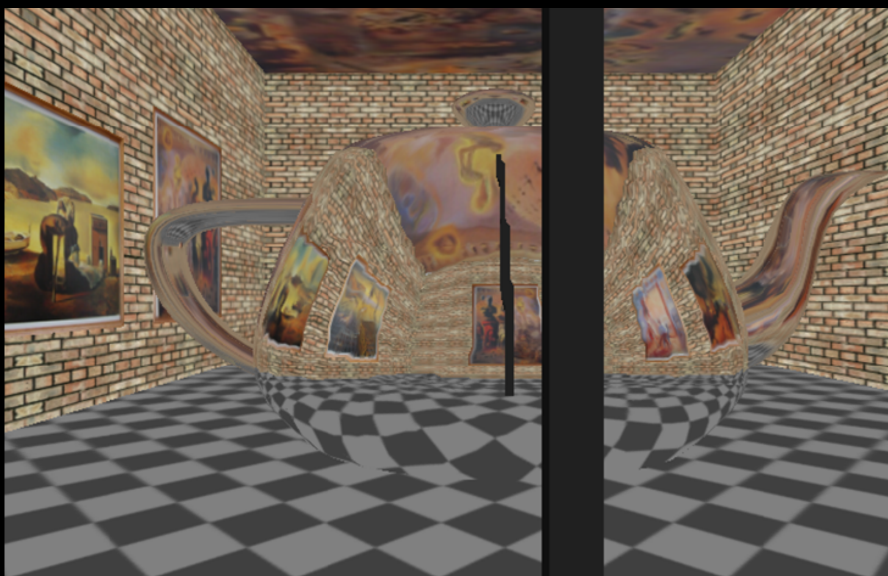
# Approach

- Solve the problem of projected reflected vertices
  - Take advantage of ray coherence
  - Design a camera that efficiently projects reflected vertices
- Sample-Based Camera (SBC)
  - A set of trees with simple cameras at their leafs



# Results: complex reflectors



# Results: second order reflections

Our method, 24 fps.

Prior art, 0.5 fps.

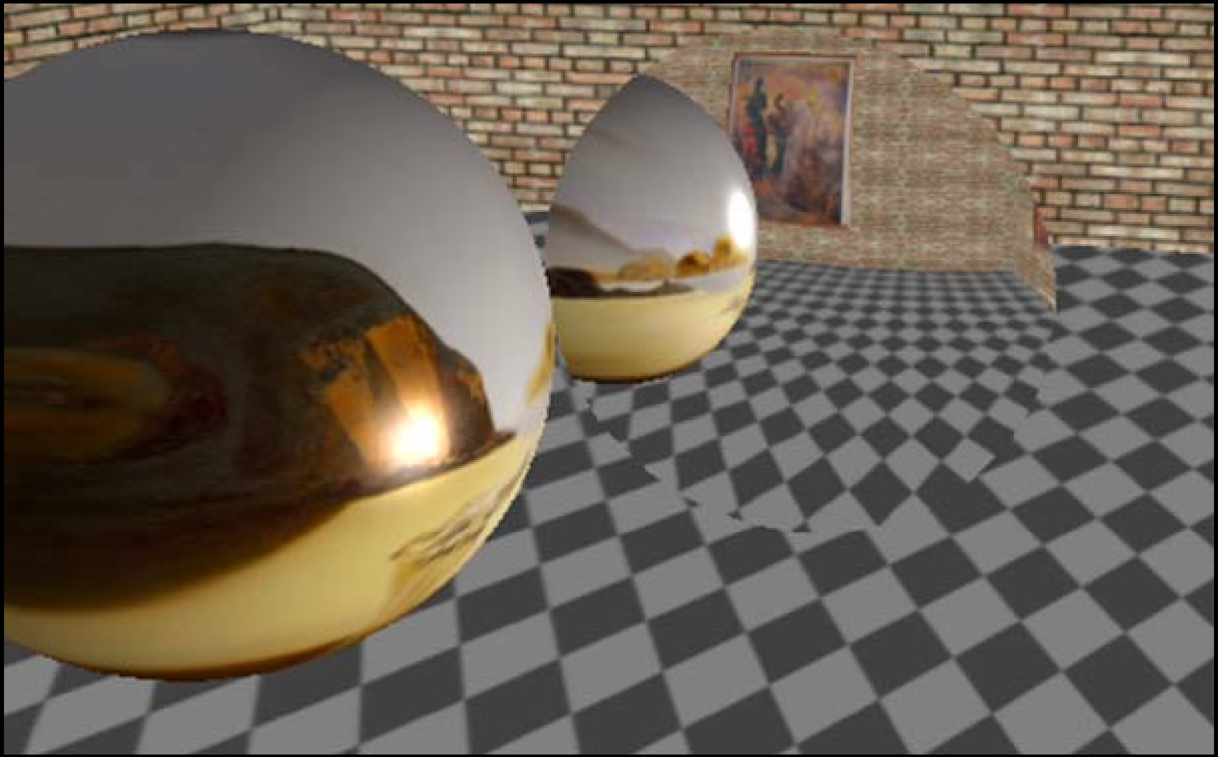
# Results: complex reflective materials

Fresnel effect

Attenuation with distance effect

Both effects combined

# Results: correct view-dep. lighting

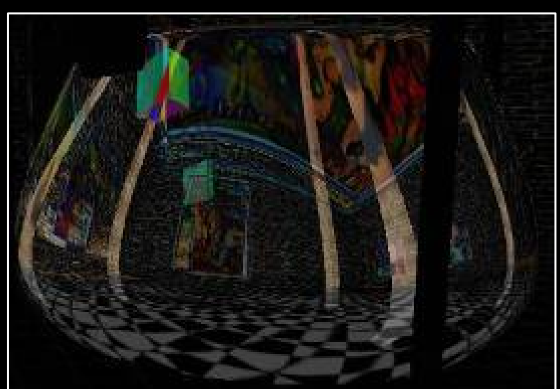


Highlight on reflected object correctly occurs at different location.

# Results: pixel accurate reflections

Prior art, 60 fps.

Our method, 60 fps.



Error images

# Future: extension to refraction

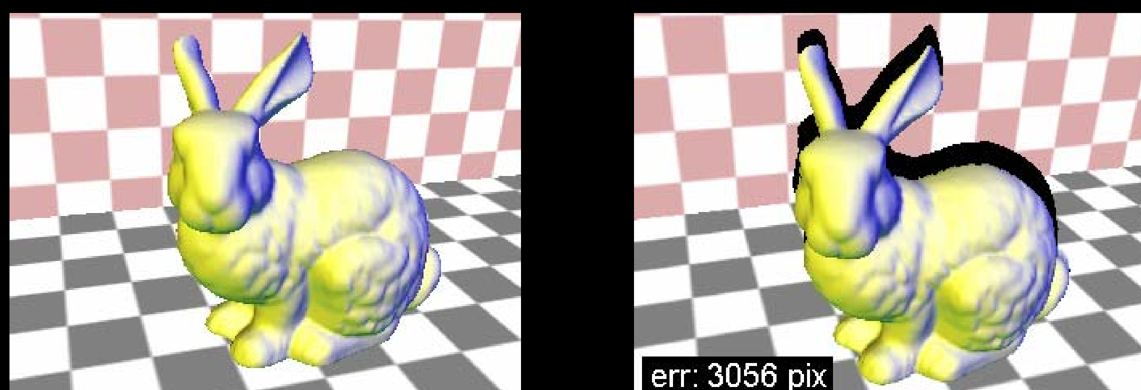
# Occlusion Cameras

## for Disocclusion-Error-Free Reference Images

*Voicu Popescu, Daniel Aliaga, Chunhui Mei,*
*Elisha Sacks, Paul Rosen*

# Motivation

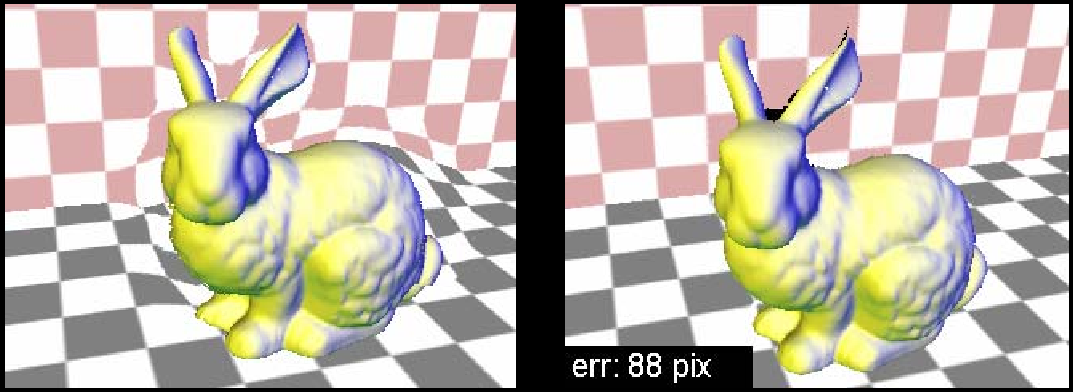
- Images are a powerful modeling and rendering primitive
  - Finite number of samples
  - Photorealistic
- Limitation: disocclusion errors
  - An image does not suffice to render the scene from a novel view




Regular reference produces disocclusion errors

# Approach

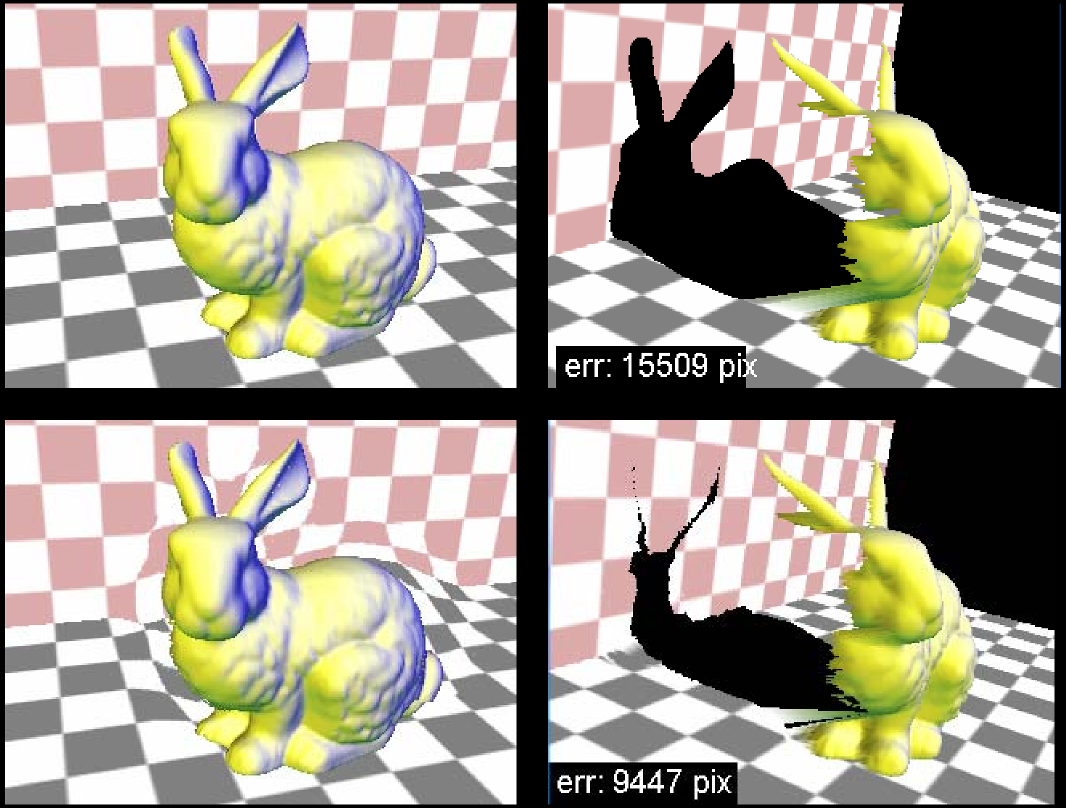
- Camera model design
  - Rays of interest include rays that circumvent the occluder to gather barely hidden samples needed in nearby views




Occlusion camera reference image alleviates disocclusion errors

# Occlusion cameras

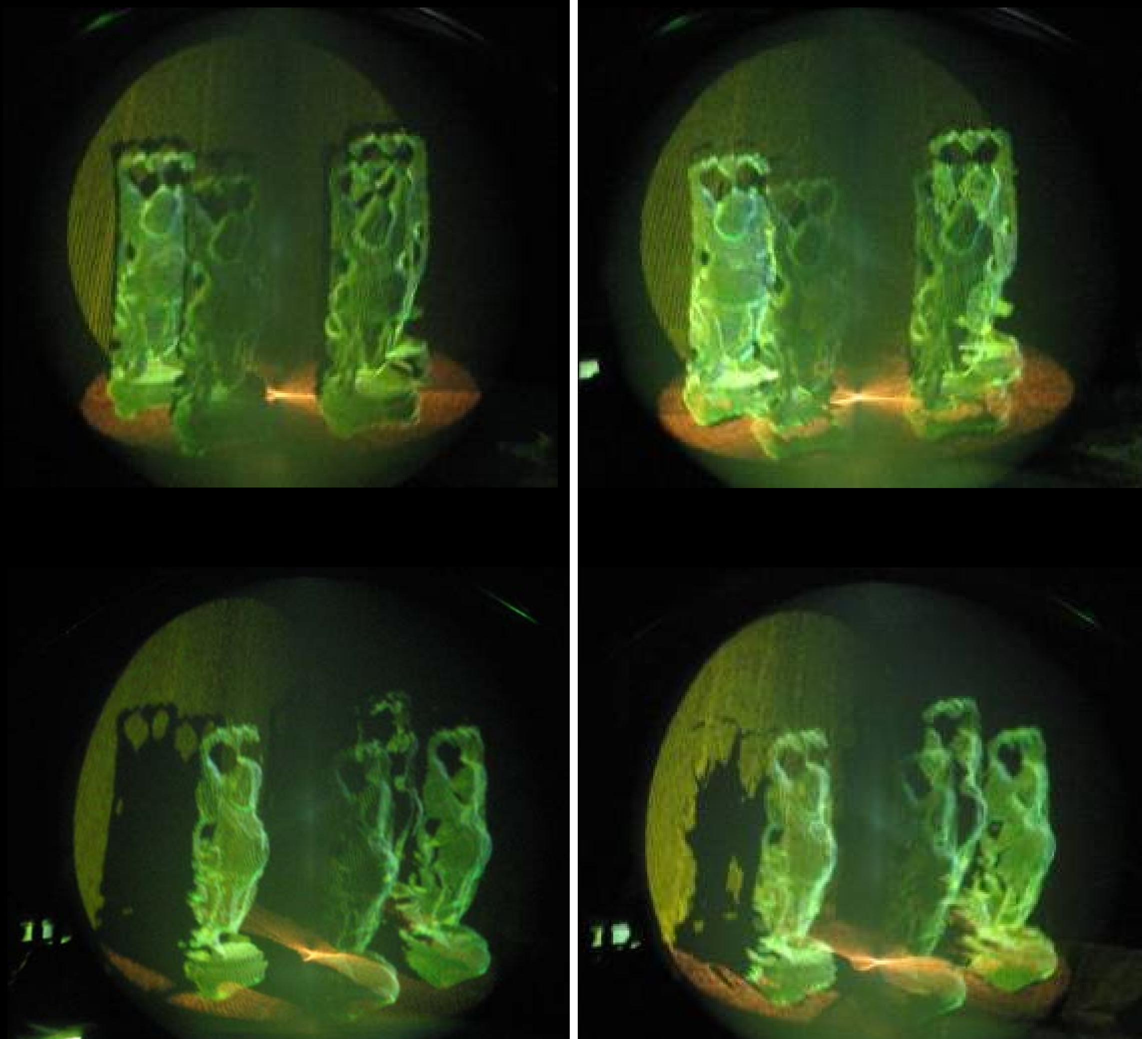
- Rays are 3D distorted at occluder edge
  - Hidden samples close to edge are “pulled out”
  - The occluder “shadow” is shrunk



# Application to 3D image rendering

# Application to 3D image rendering

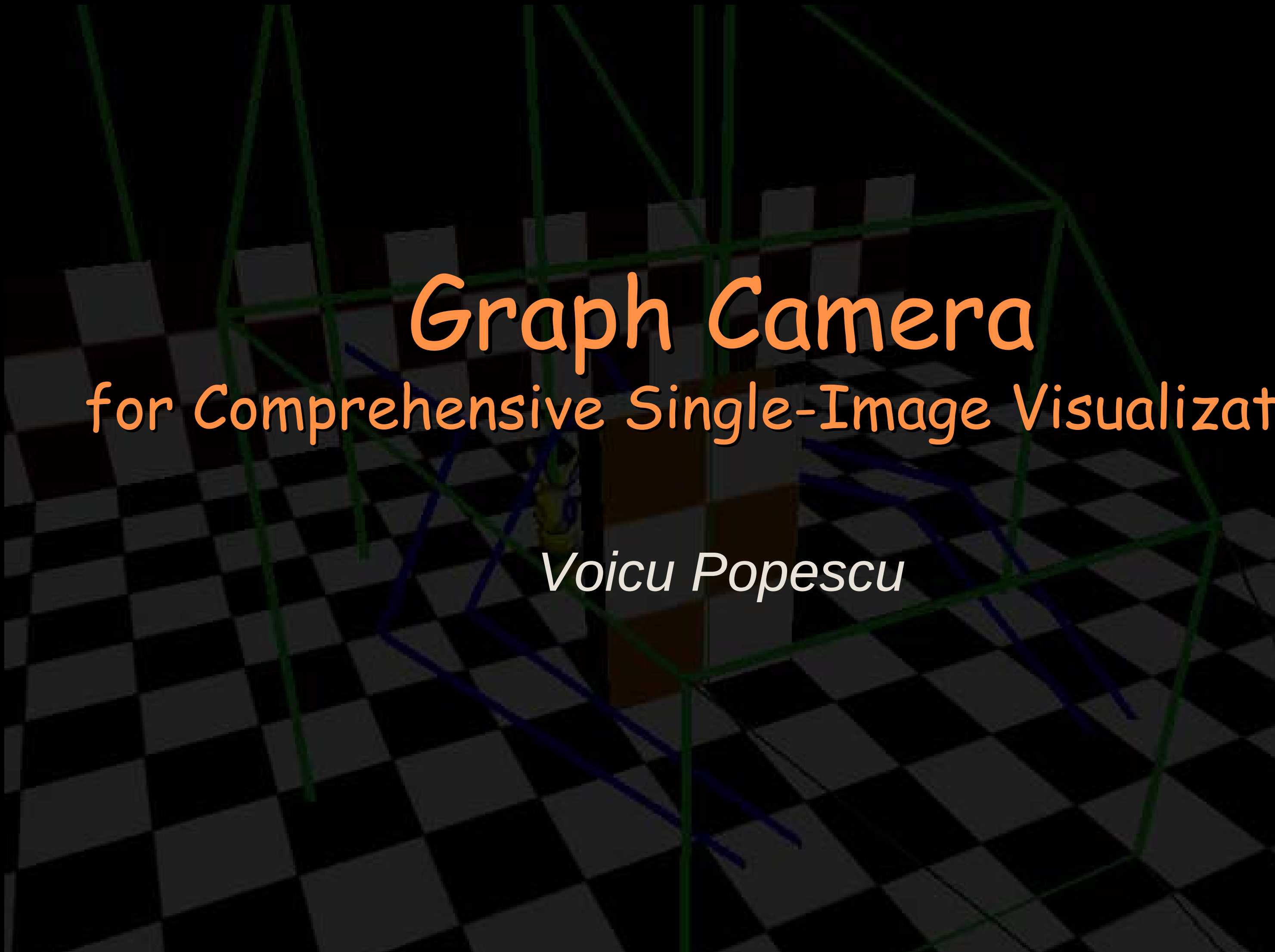
Regular reference image



Occlusion camera reference image

# Future

- Compression
- Soft shadows
- Antialiasing



# Graph Camera

## for Comprehensive Single-Image Visualization

*Voicu Popescu*

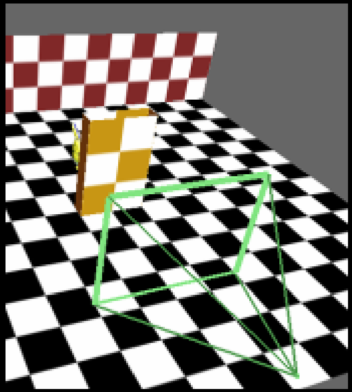
# Motivation

- Navigation is challenging in graphics applications
  - Unintuitive interfaces
  - Unreliable trackers
  - Bulky head-mounted displays
- Navigation reduces information assimilation efficiency
- Navigation cannot support more than one region of interest
  - User needs to be in more than one place at the same time
- Not all applications require the actual experience of locomotion in the virtual environment

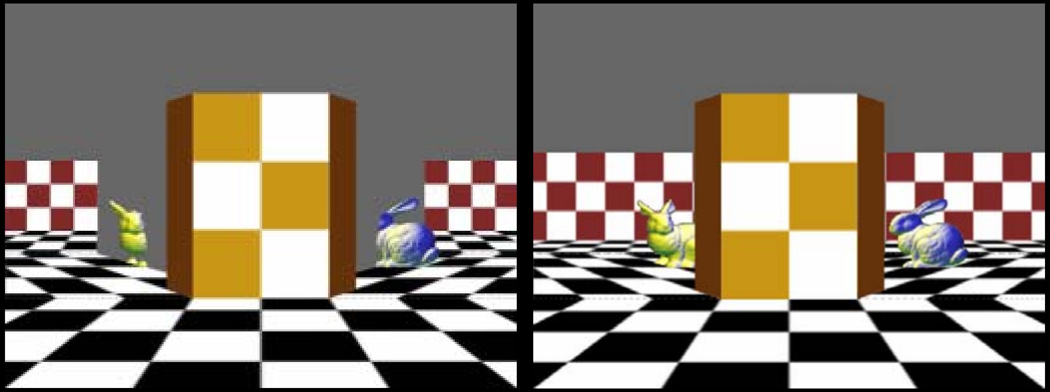
# Approach

- Avoid navigation by visualization with comprehensive non-pinhole camera image
- Design camera that simultaneously captures all points of interest
- Graph camera
  - Start with pinhole
  - Fold, split, and splice ray subsets

# Example: seeing around occluder

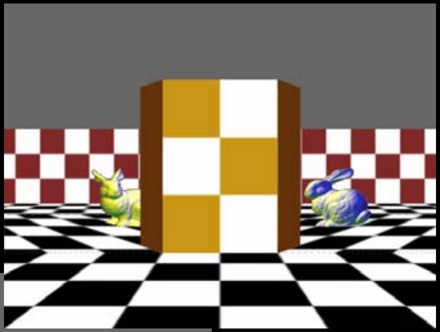


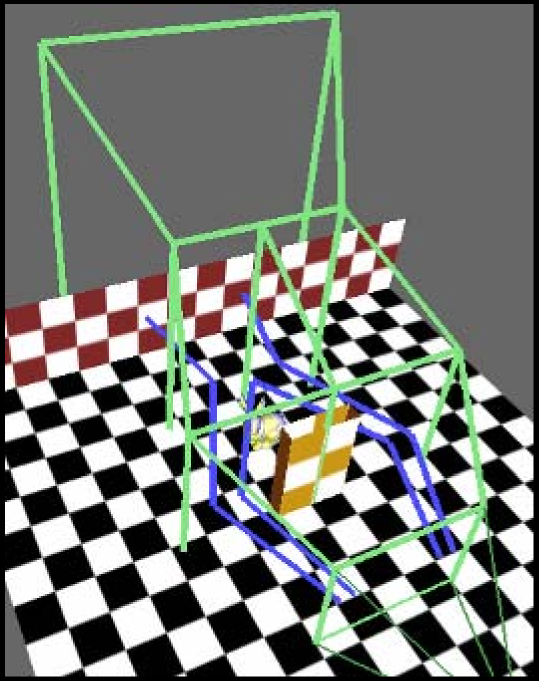
Bunny hidden behind vertical block

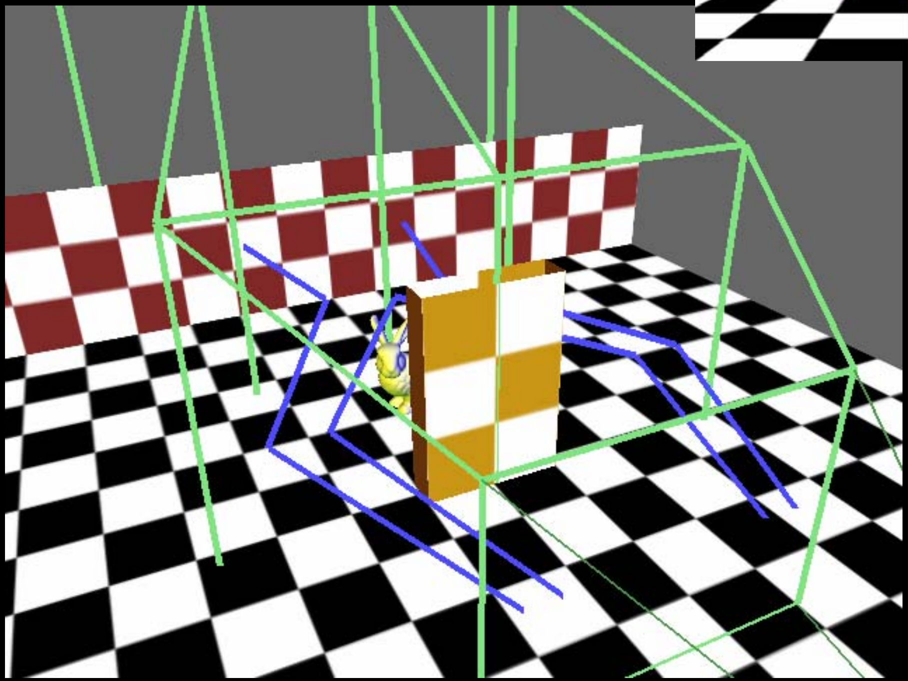


Visible in graph camera images

# Example: seeing around occluder

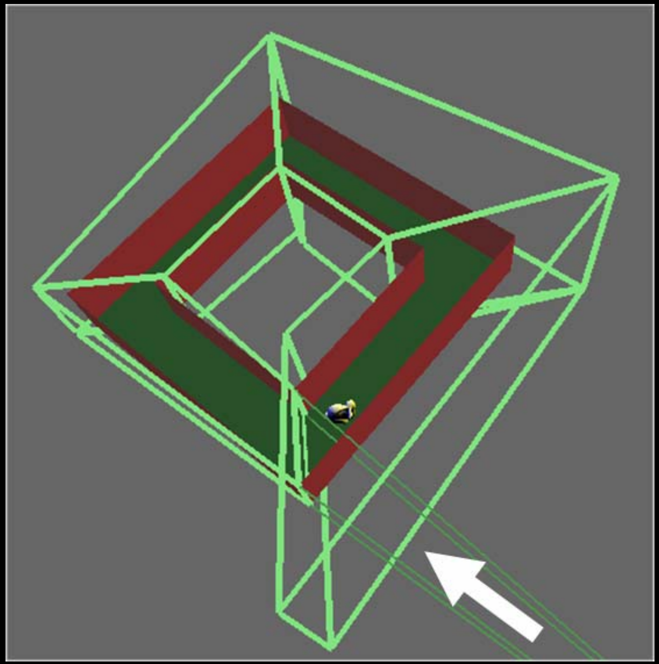


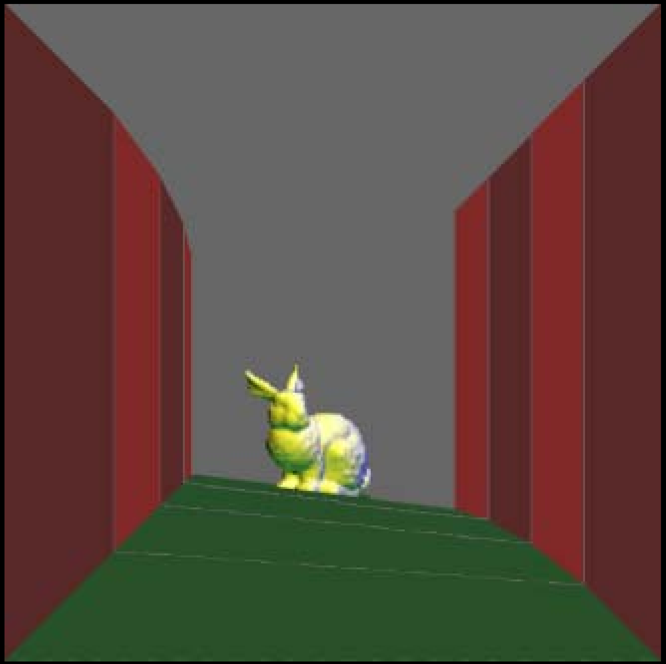




Visualization of graph camera model

# Example: seeing around corners



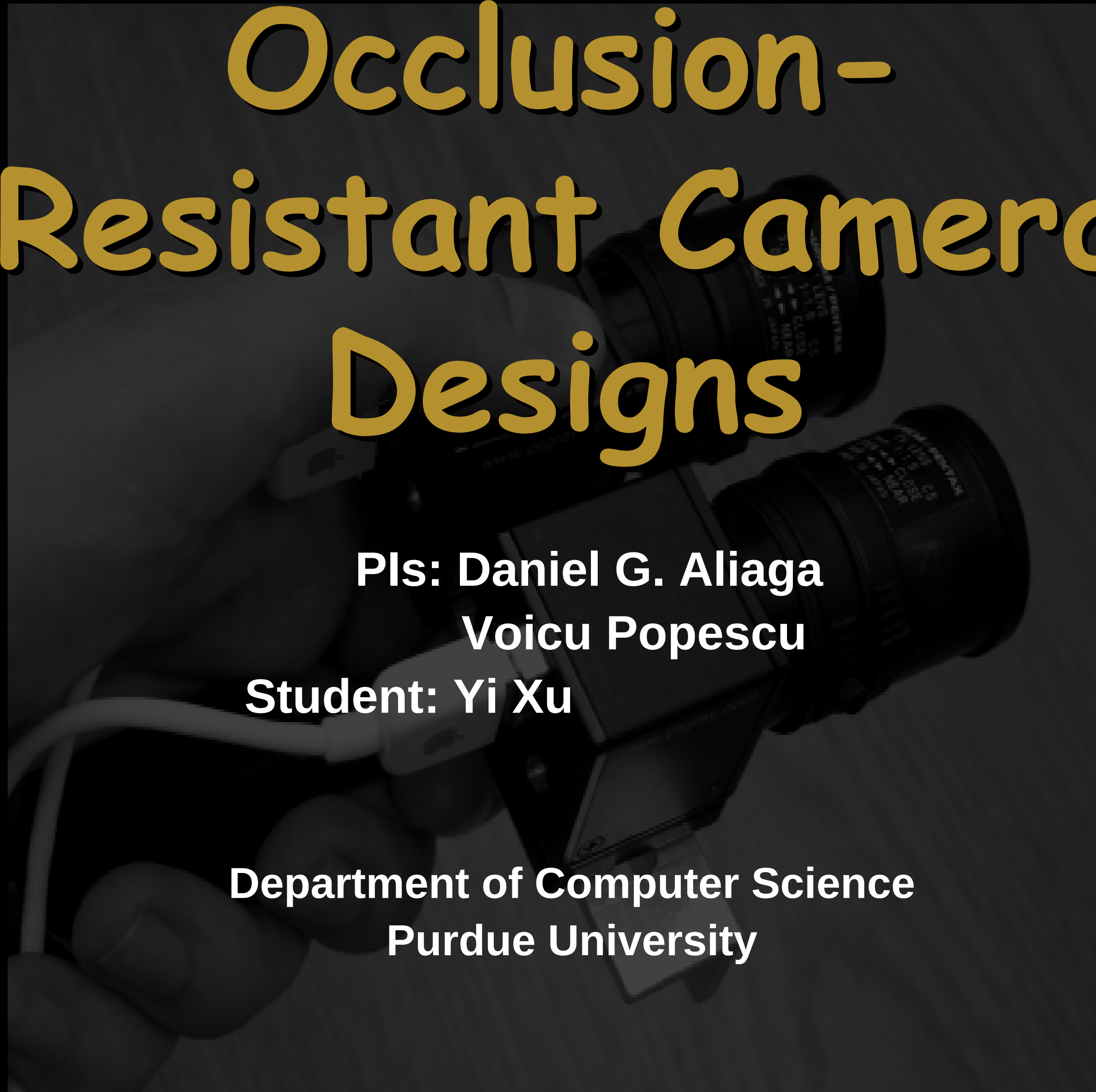


Graph camera model and graph camera image

# Conclusions and Future Work

- First successful steps towards robust acquisition in active environments
- Clear line-of-sight is achieved at a low-level abstracted away from the operator
- Our results both analyze and confirm the viability of our designs
- Future work:
  - Address effects due to strong illumination changes
  - Incorporate with a large-scale acquisition effort
  - Add real-time visualization to LCD screen on a portable camera

# Occlusion-Resistant Camera Designs



PIs: Daniel G. Aliaga
Voicu Popescu
Student: Yi Xu

Department of Computer Science
Purdue University

# Motivation

- Geometric modeling, image-based rendering, and computer vision use cameras to create models of real-world environments
- They require a clear line-of-sight between the acquisition device and the targeted scene
- Unfortunately, many compelling environments are in active use and thus dynamic occluders are frequently interposed between camera and scene

# Objective

- To design a new camera-based acquisition device unaffected by moving occluders in the scene
  - e.g., an “X-ray” vision camera
- This permits more valid samples to be acquired faster and without having to worry about moving occluders

# Related Work

- Single-view camera designs
  - Cannot easily distinguish scene motion from camera motion
  - Simultaneous structure and motion recovery is, in fact, mathematically ill-posed for planar cameras
- Multi-view camera designs
  - Depend on fragile correspondence computations
  - Or, require a large and pre-installed infrastructure
- Segmentation algorithms
  - Manual and time consuming
  - Most assume static cameras and/or complex camera rigs

# Approach

- We propose a family of occlusion-resistant cameras (ORCs) that appear temporarily stationary despite undergoing continuous camera motion

- Our ORC designs combine the benefits of a stationary camera with those of a moving camera yielding:
  - Real-time moving occluder segmentation
  - Clear line-of-sight samples to the surfaces, including those *behind* moving occluders

- All designs are intended to be embedded into the camera body so that in addition to “focus” and “zoom”, our cameras can “filter” moving objects

# Key Idea

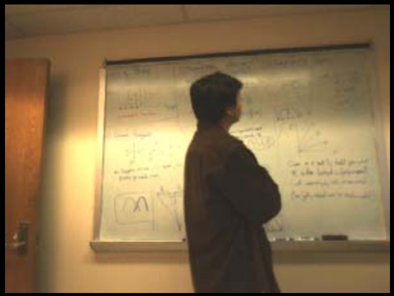


image at $t_0$

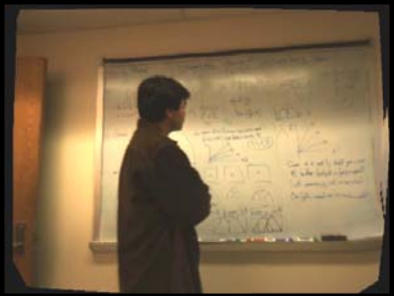
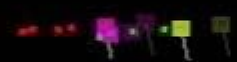


image at $t_1$

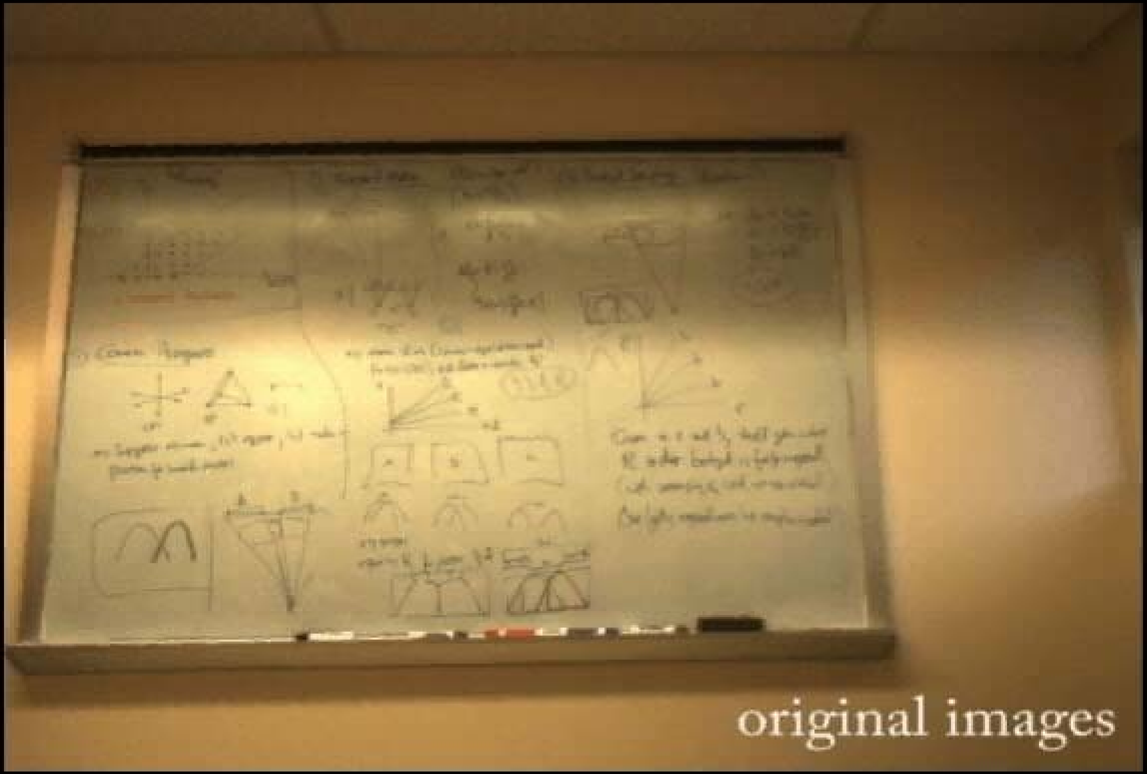
difference image

- Exploit that a camera-based device acquires discrete space and time samples of the environment
- Thus, by creating a camera that obtains the same viewpoint at two nearby instances in time and while continuously moving, the *background appears static* and only *moving objects are displaced*!

# Key Idea: Camera Motion

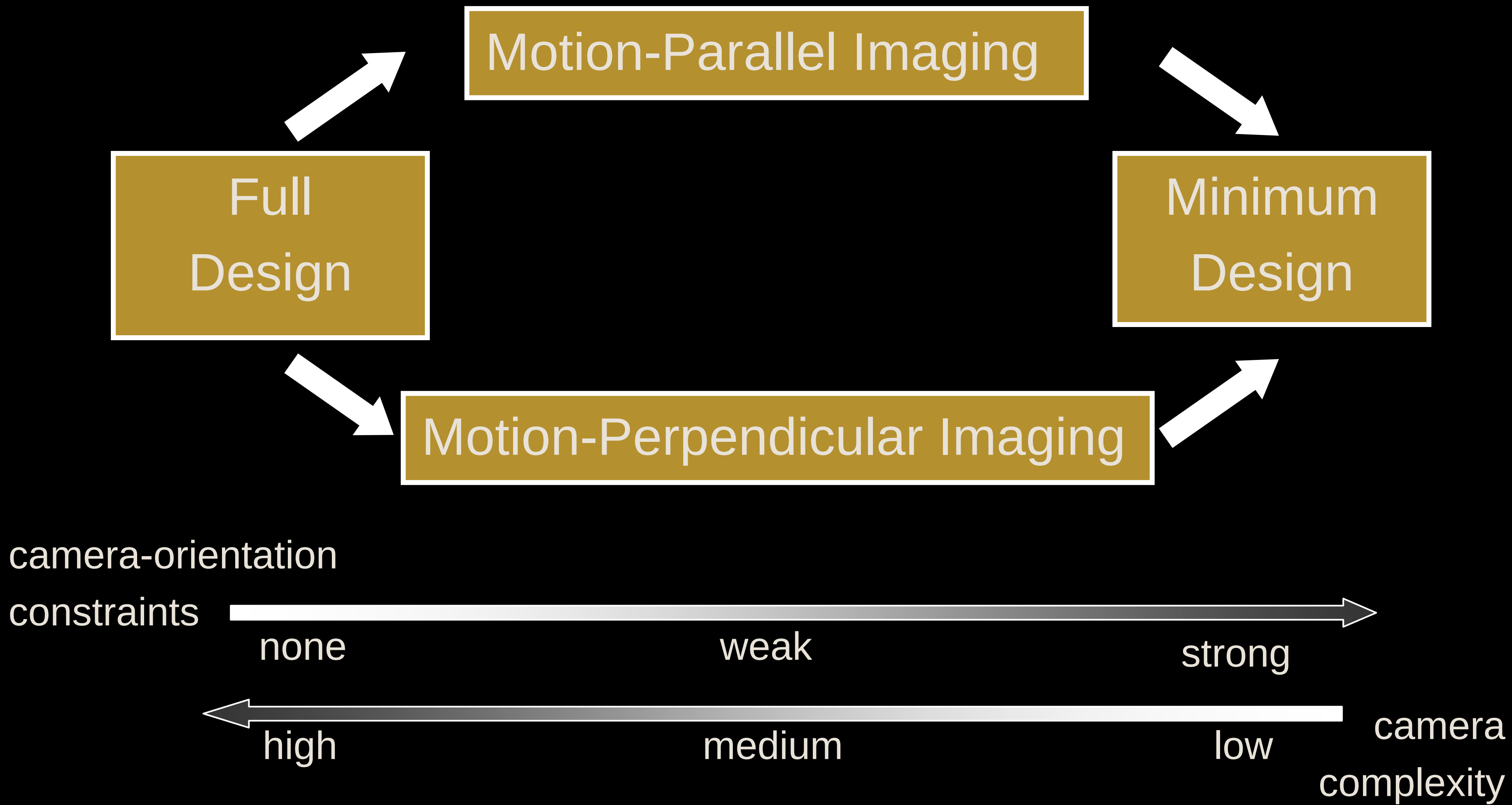


# Key Idea: Examples

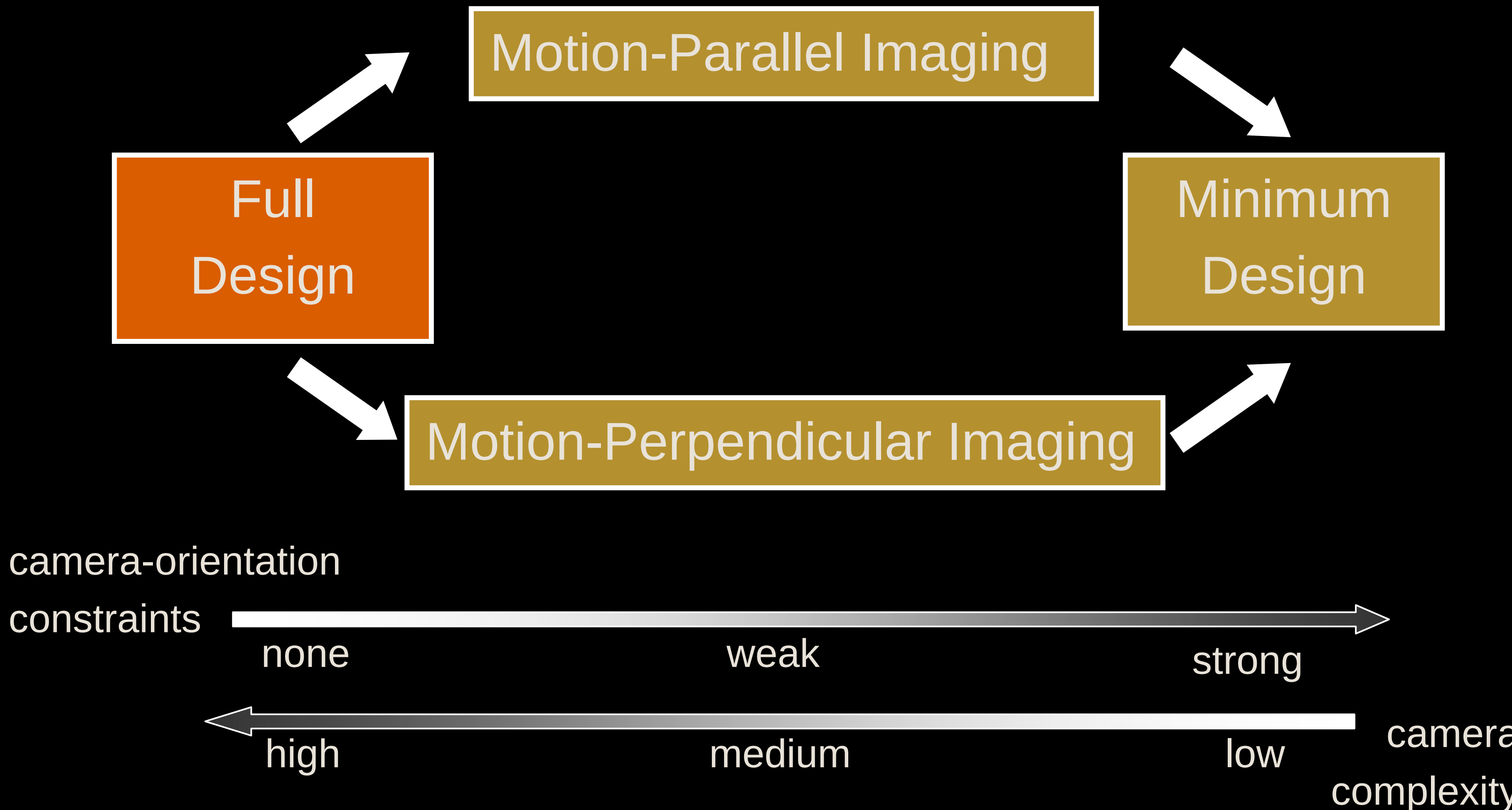


# Key Idea: Examples

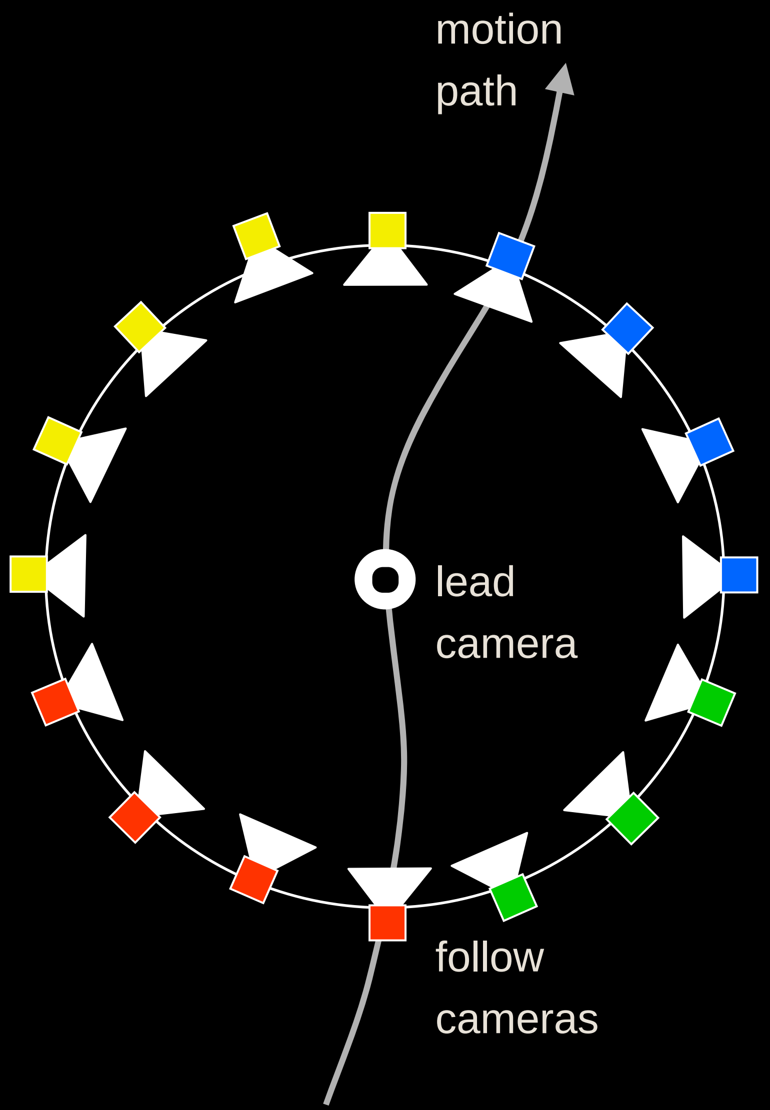
# Family of ORC Designs



# Family of ORC Designs

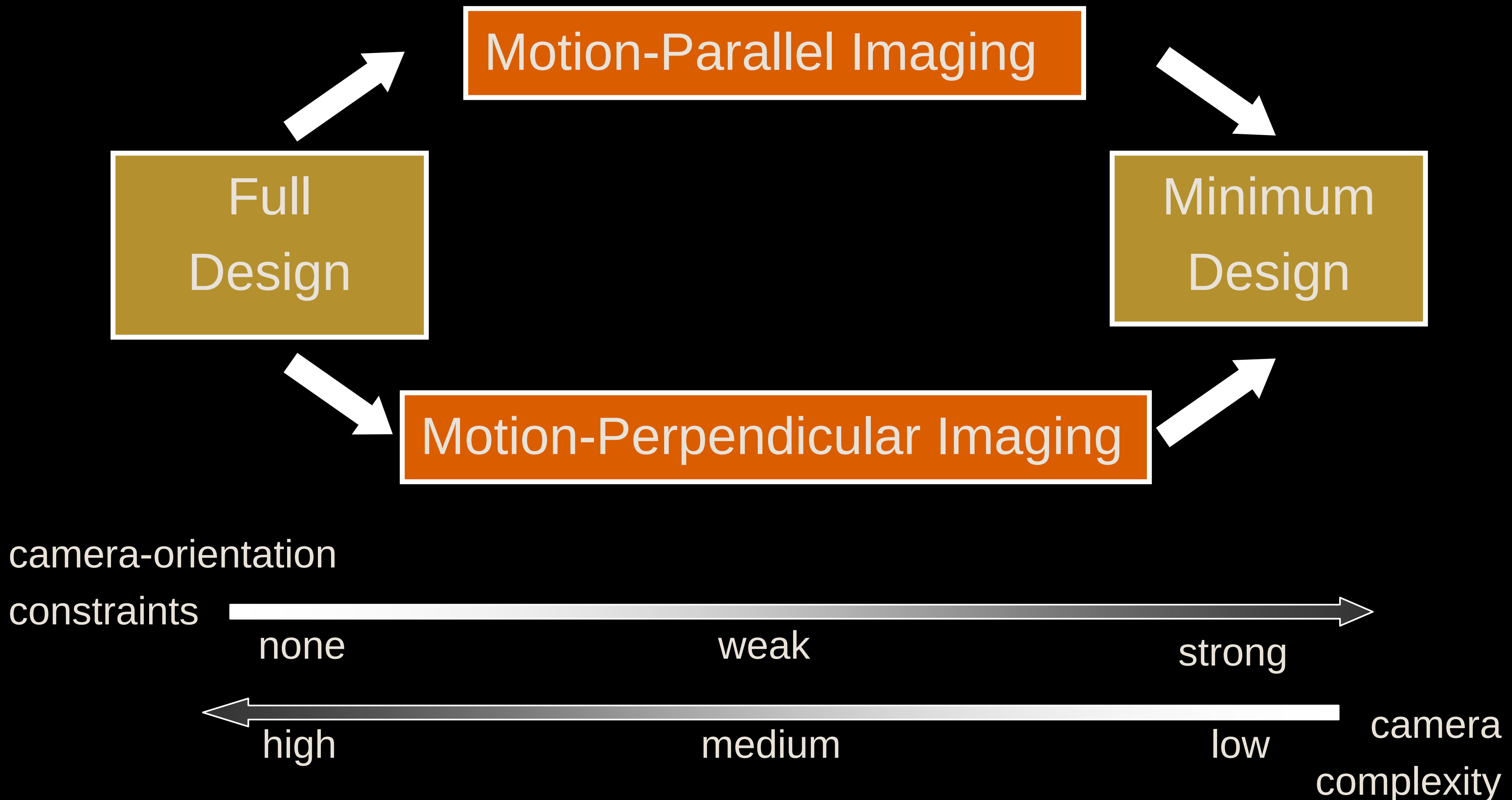


# Full Design

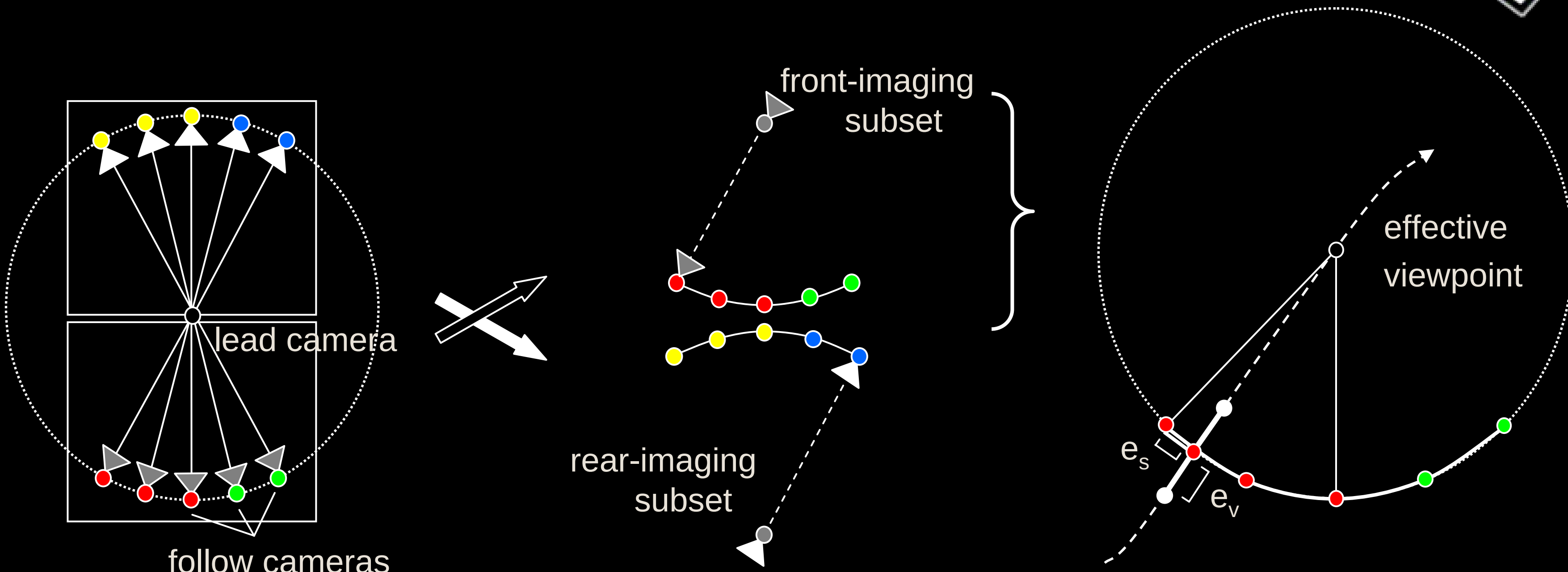



- A sphere of “follow cameras” surrounds a central “lead camera”
  - Viewpoint of a lead camera on a motion path will be reached again by a follow camera at a later and nearby time instant
- Advantages:
  - Arbitrary imaging directions
  - Arbitrary camera orientations
- Disadvantage:
  - Follow cameras appear in the FOV of the lead and follow cameras

# Family of ORC Designs

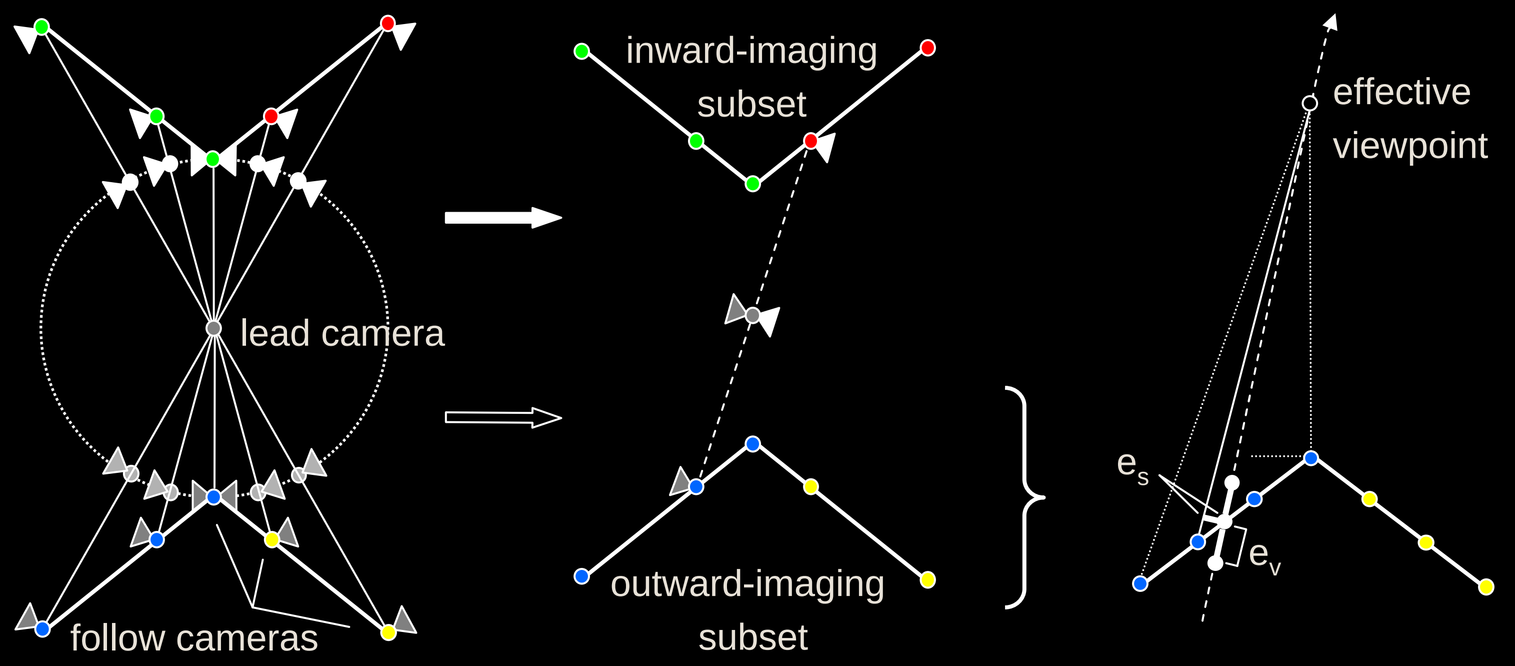


# Motion-Parallel Imaging Design



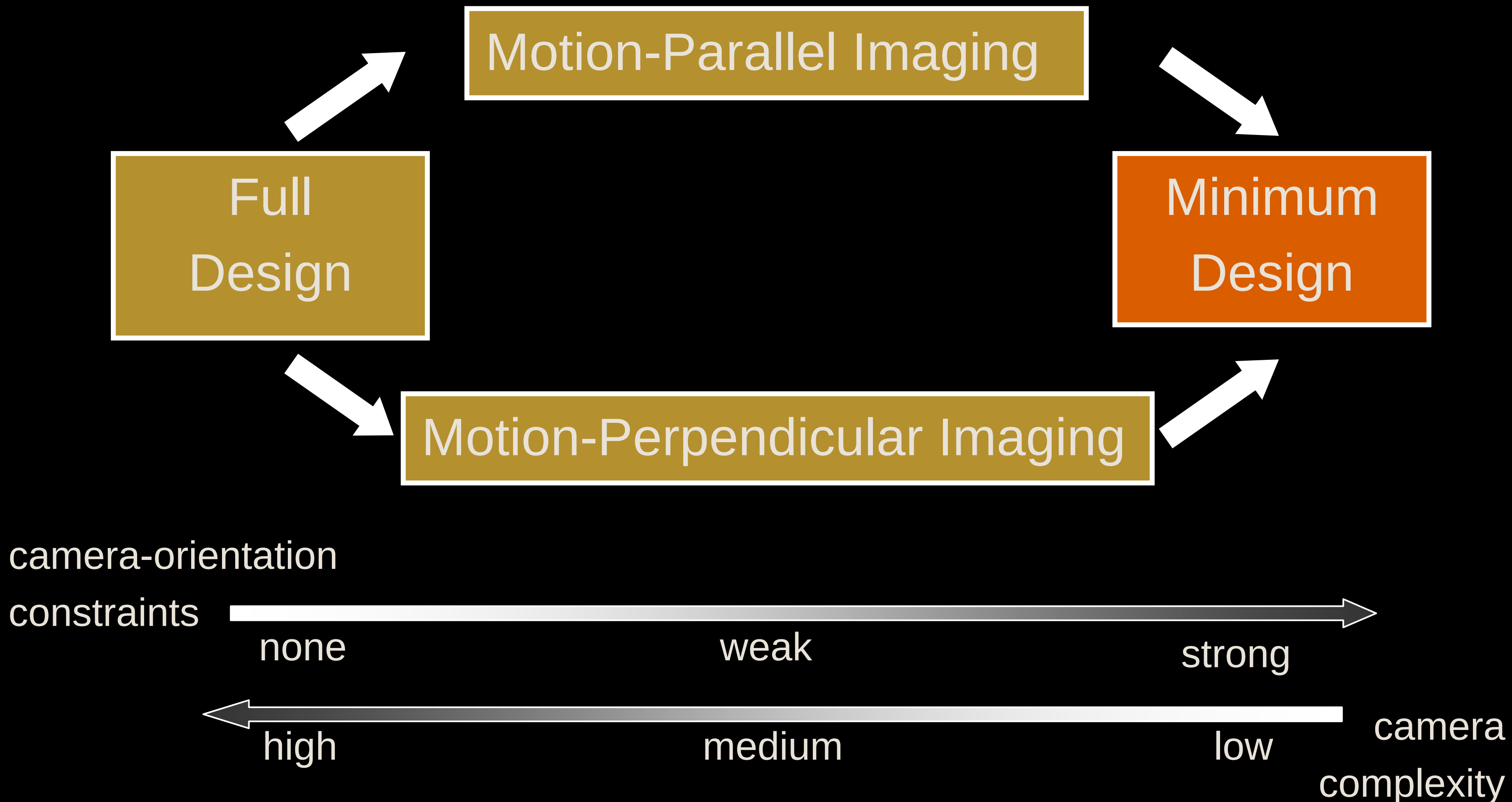

- For camera orientations approximately aligned with the motion path, a design for motion-parallel imaging reduces to a partial sphere of cameras
  - Two partial spheres are needed for front- and rear-imaging
  - Both surfaces can be compacted to a single plane of omnidirectional cameras

# Motion-Perpendicular Imaging Design

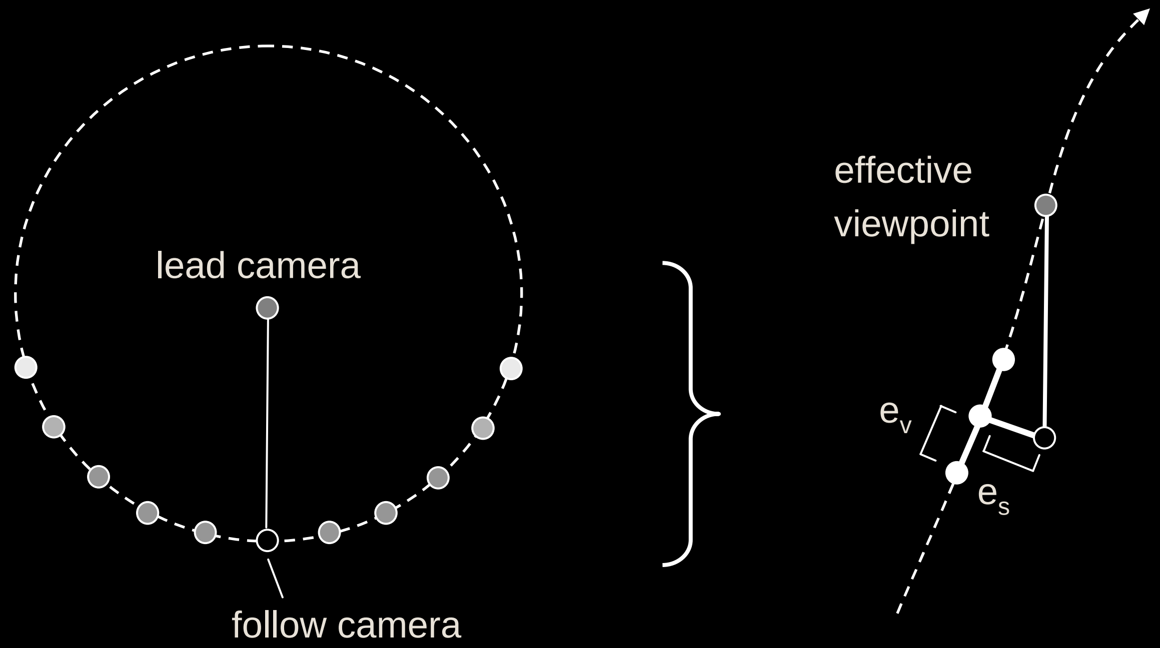



- For camera orientations approximately aligned with the motion path, a design for motion-perpendicular imaging reduces to a cone of cameras
  - Two cones are needed for inward- and outward-imaging
  - For one cone, the usage of lead camera and follow cameras harmlessly "swaps"

# Family of ORC Designs

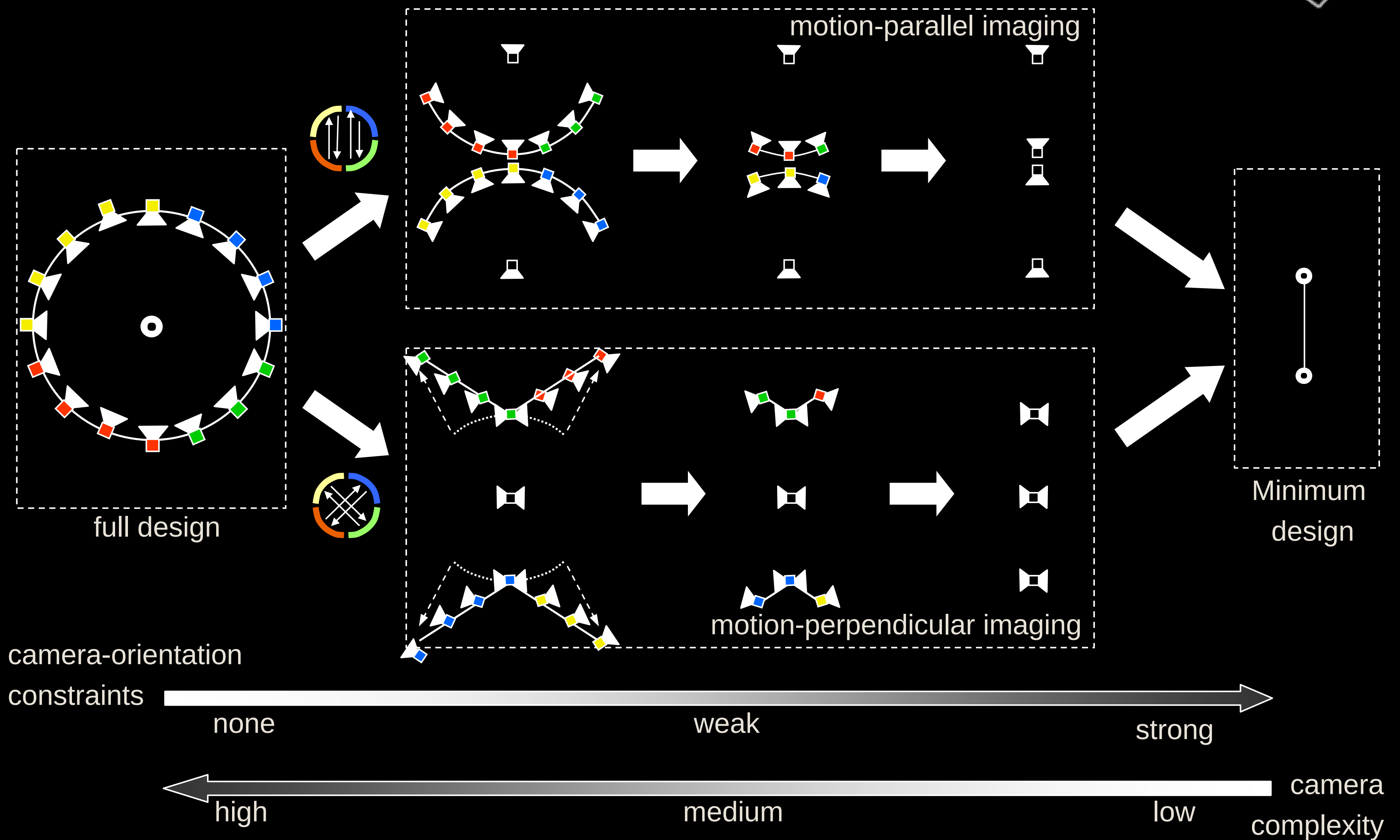


# Minimum Design



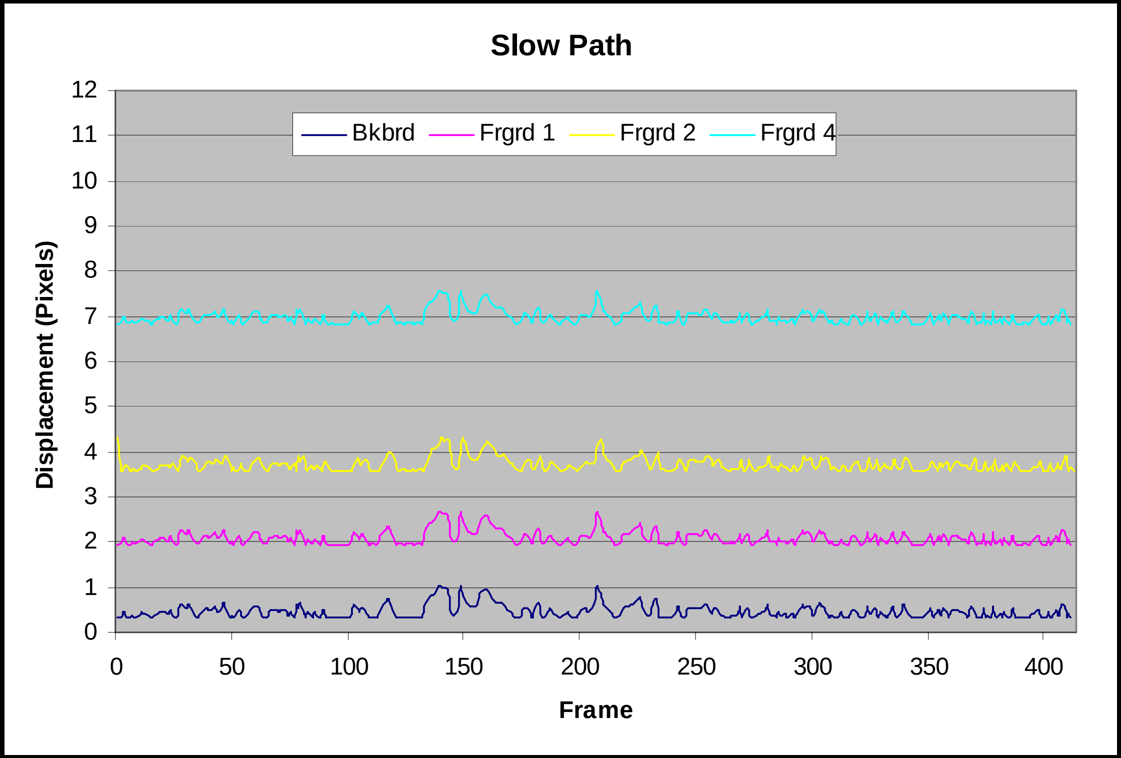

- For camera orientations aligned with the motion path, the design reduces a linear configuration of two cameras
  - Cameras can be omnidirectional
  - Depending on motion, both cameras exchange roles as follow or lead camera

# Family of ORC Designs



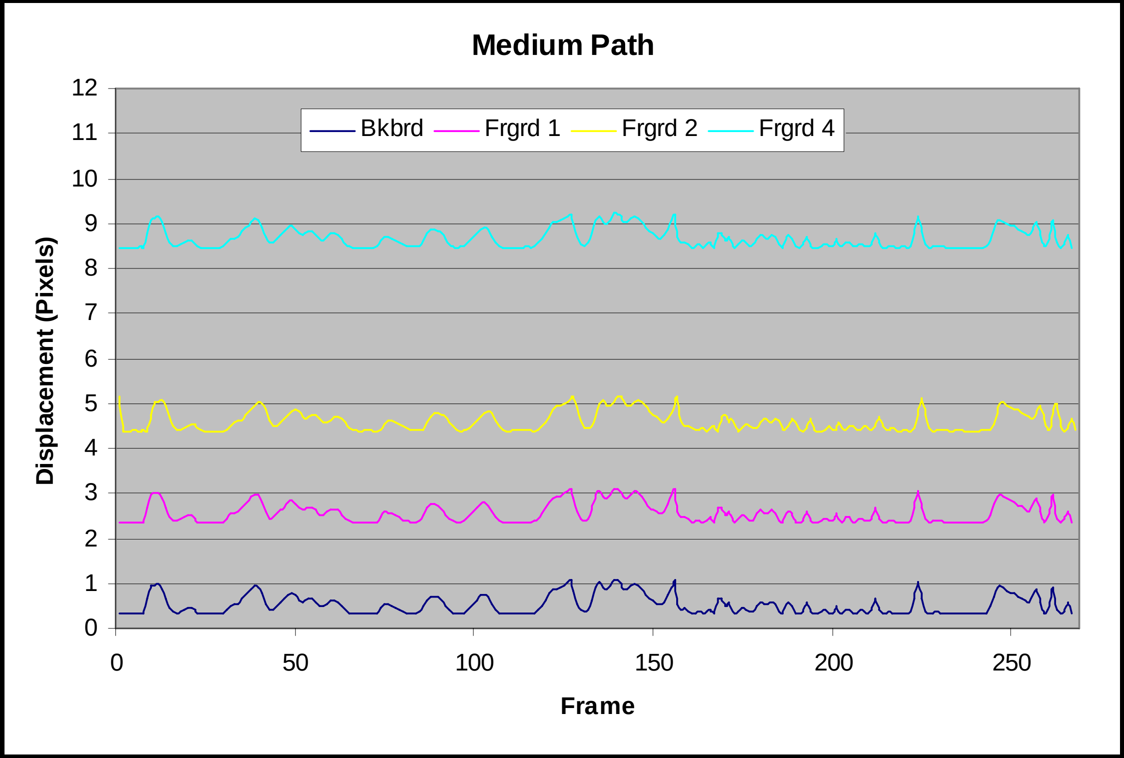
# Design Performance

- We can “tune” the parameters of each design so as to yield a stationary background and displacement of moving objects to within desired tolerances



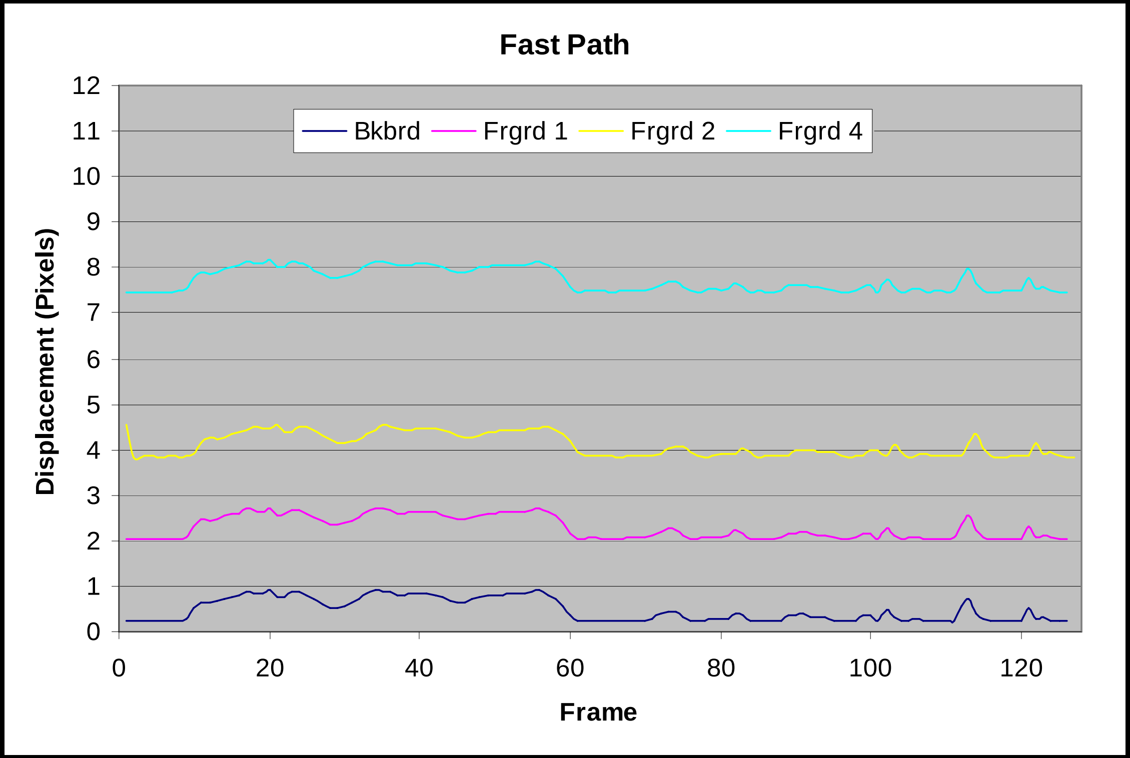
# Design Performance

- We can "tune" the parameters of each design so as to yield a stationary background and displacement of moving objects to within desired tolerances



# Design Performance

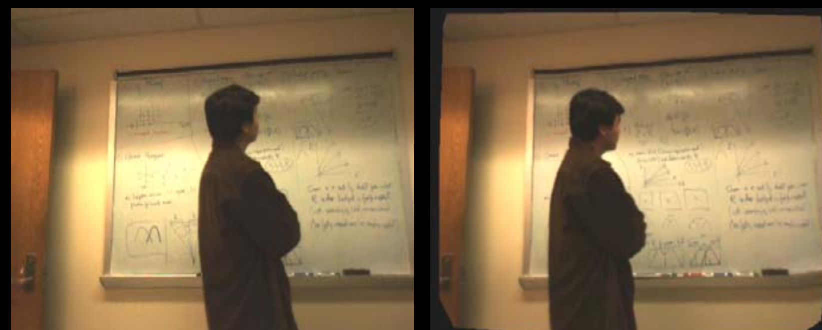
- We can "tune" the parameters of each design so as to yield a stationary background and displacement of moving objects to within desired tolerances



# Foreground Segmentation

- Continuous occluder motion

Prototype



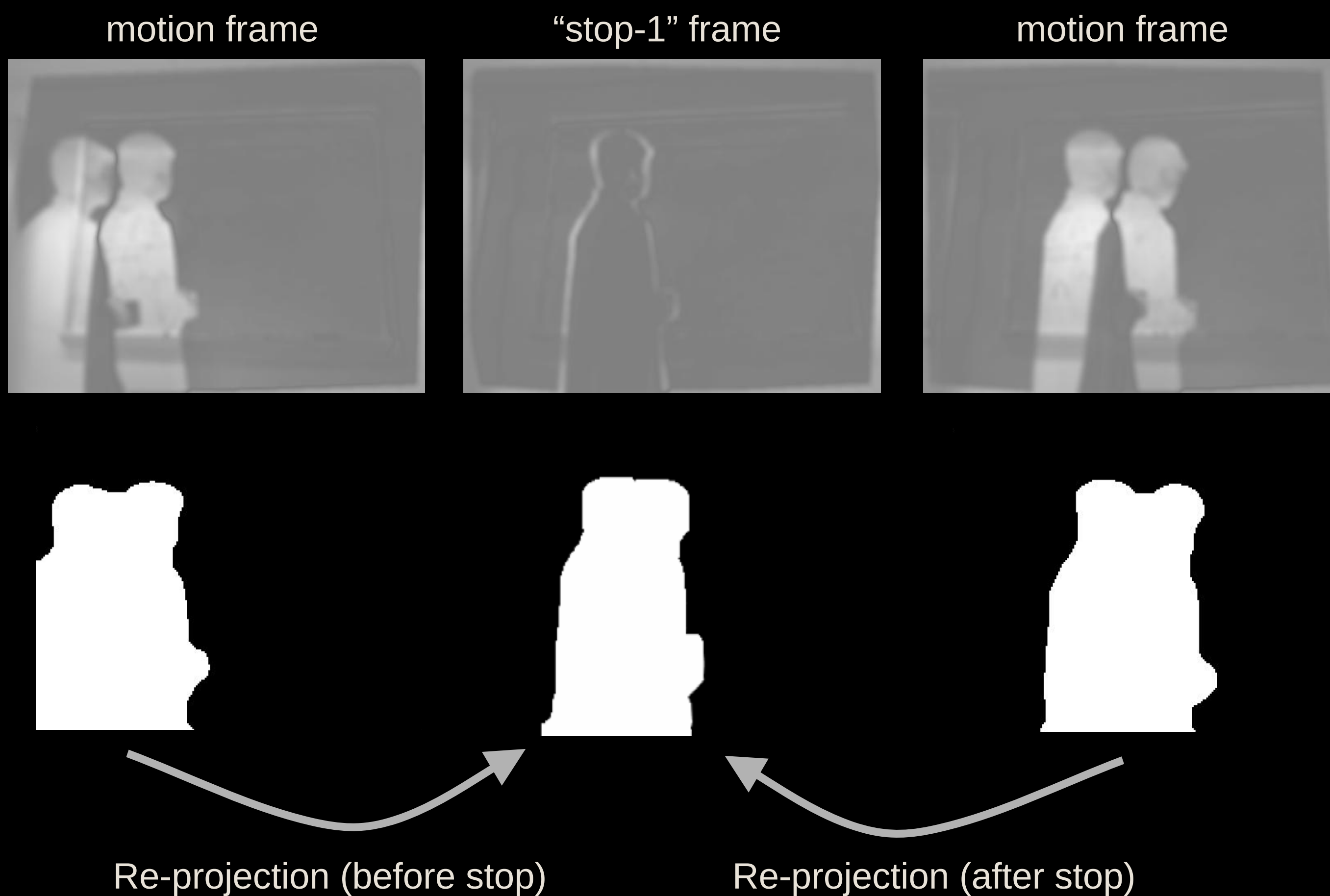
Source images

Image difference

Thresholded

# Foreground Segmentation

- Move-stop-move occluder motion



# Foreground Segmentation

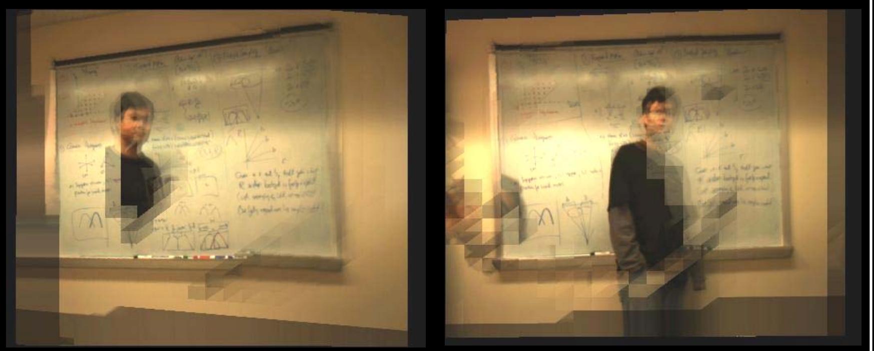
- We can exploit the redundancy over the image sequence to refine (offline) the segmentation and yield separate image sequences for foreground and background

# Foreground Segmentation

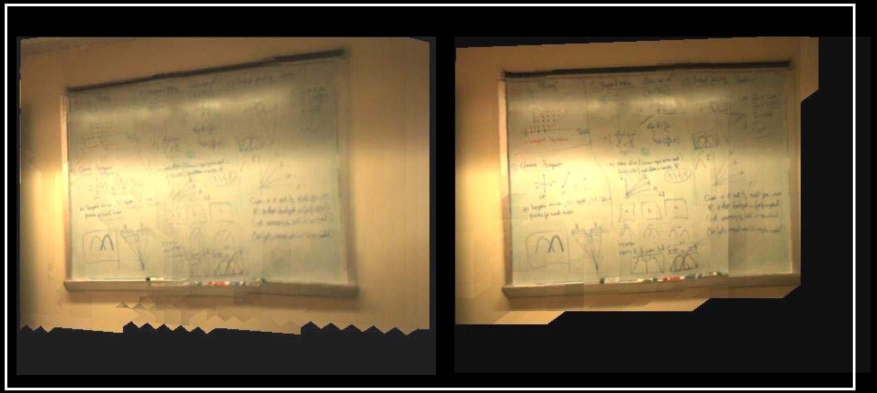
- We can exploit the redundancy over the image sequence to refine (offline) the segmentation and yield separate image sequences for foreground and background

# Background Reconstruction

- We can obtain images of the background scene despite moving occluders



Naïve rendering



Our rendering

# Background Reconstruction

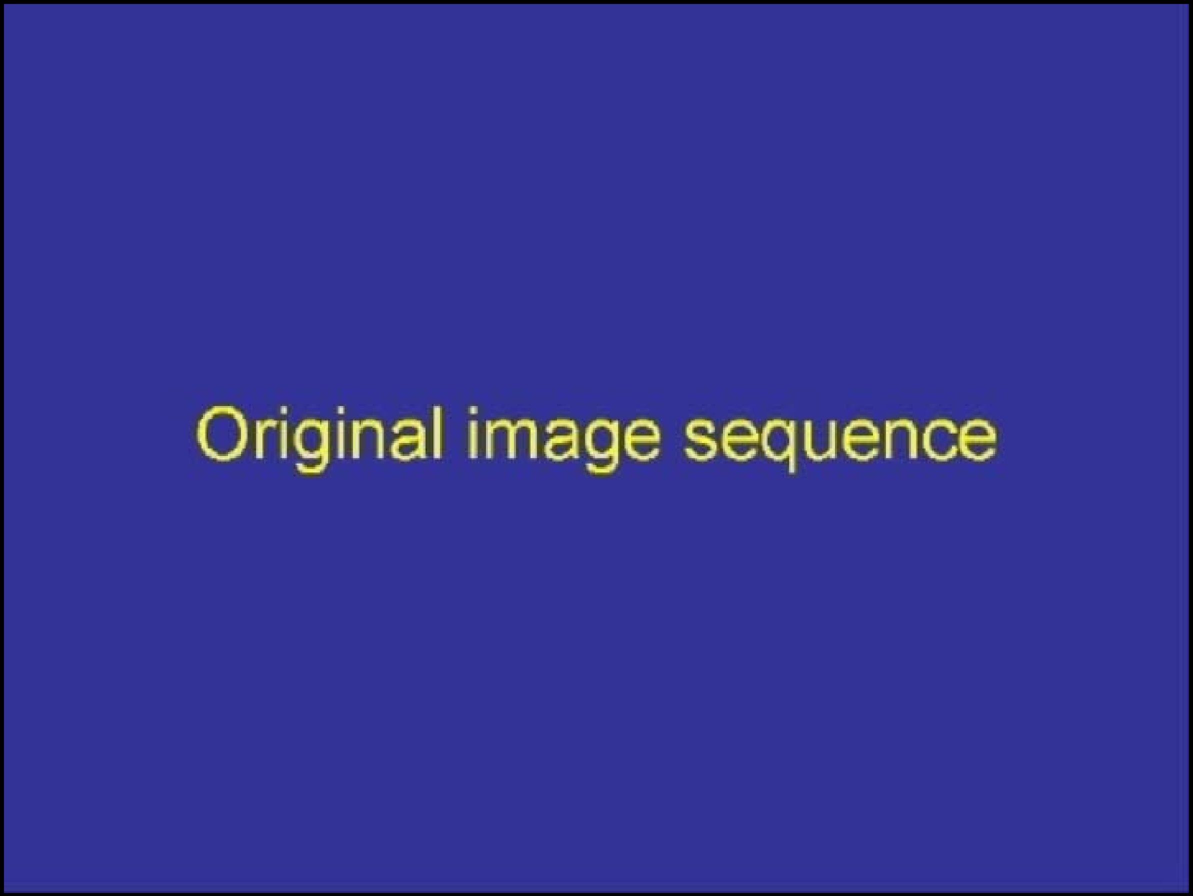
- We can obtain images of the background scene despite moving occluders

Source images were occluders attempts to cover scene continuously

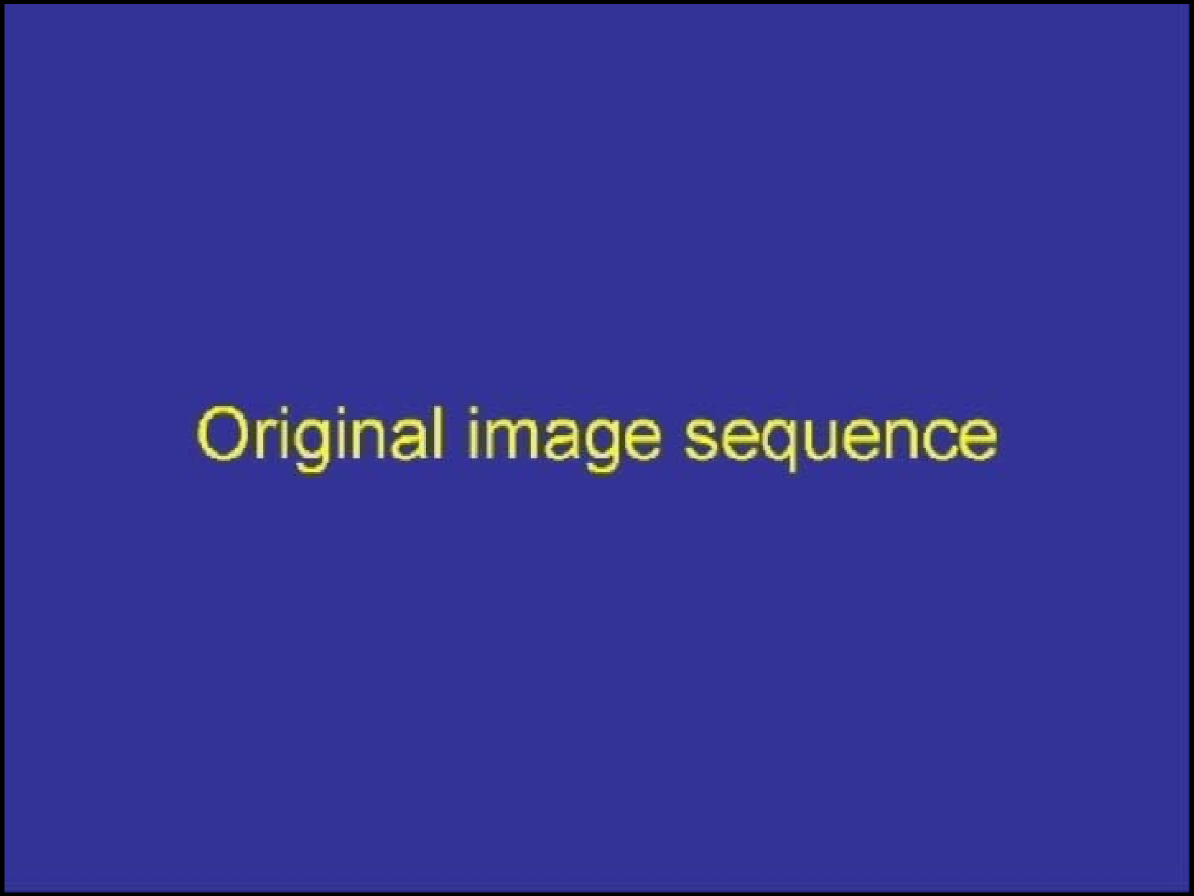
Our rendering

Naïve rendering

# Examples I



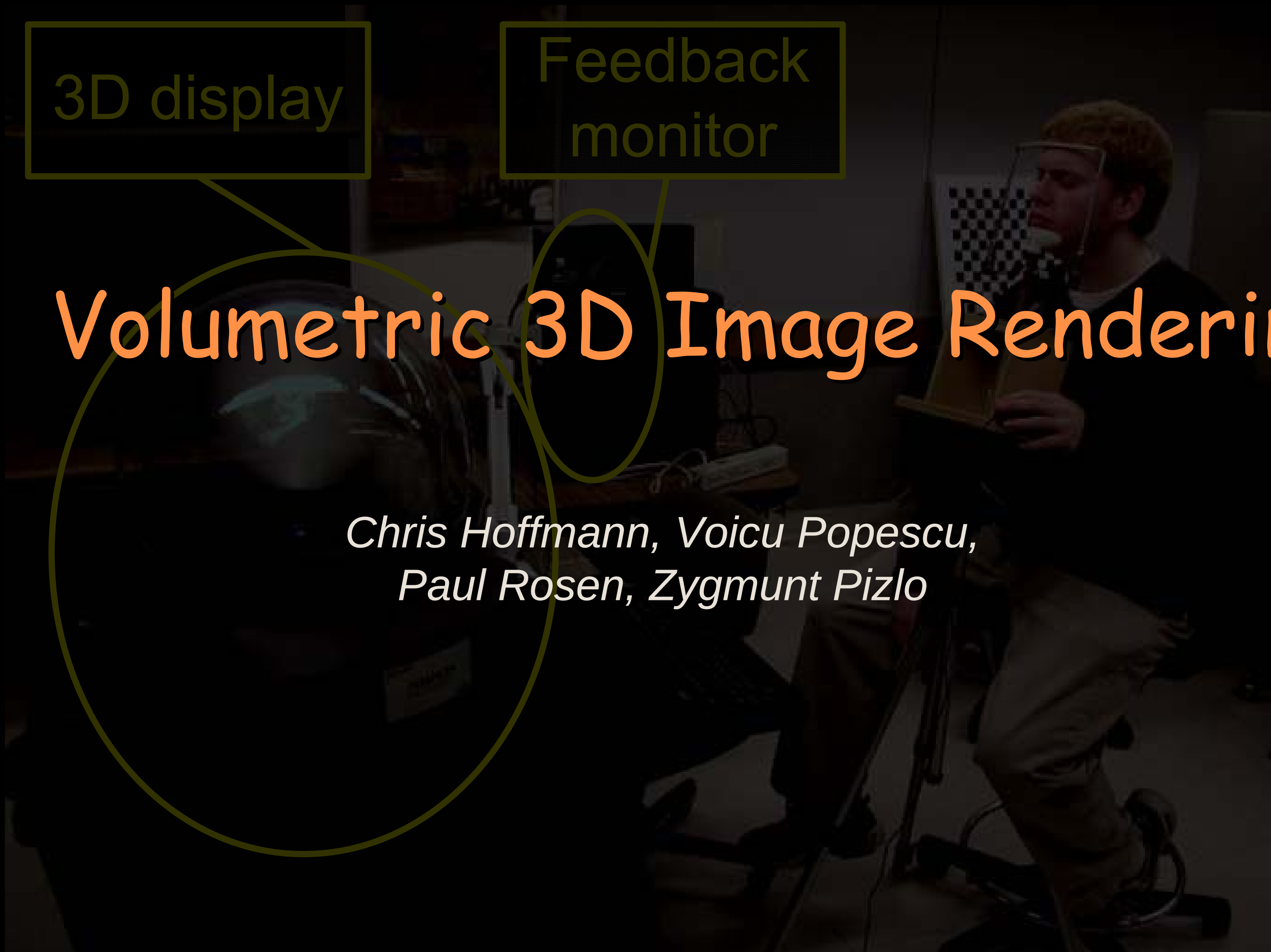
# Examples II



# Conclusions and Future Work

- First successful steps towards robust acquisition in active environments
- Clear line-of-sight is achieved at a low-level abstracted away from the operator
- Our results both analyze and confirm the viability of our designs
- Future work:
  - Address effects due to strong illumination changes
  - Incorporate with a large-scale acquisition effort
  - Add real-time visualization to LCD screen on a portable camera

# Volumetric 3D Image Rendering



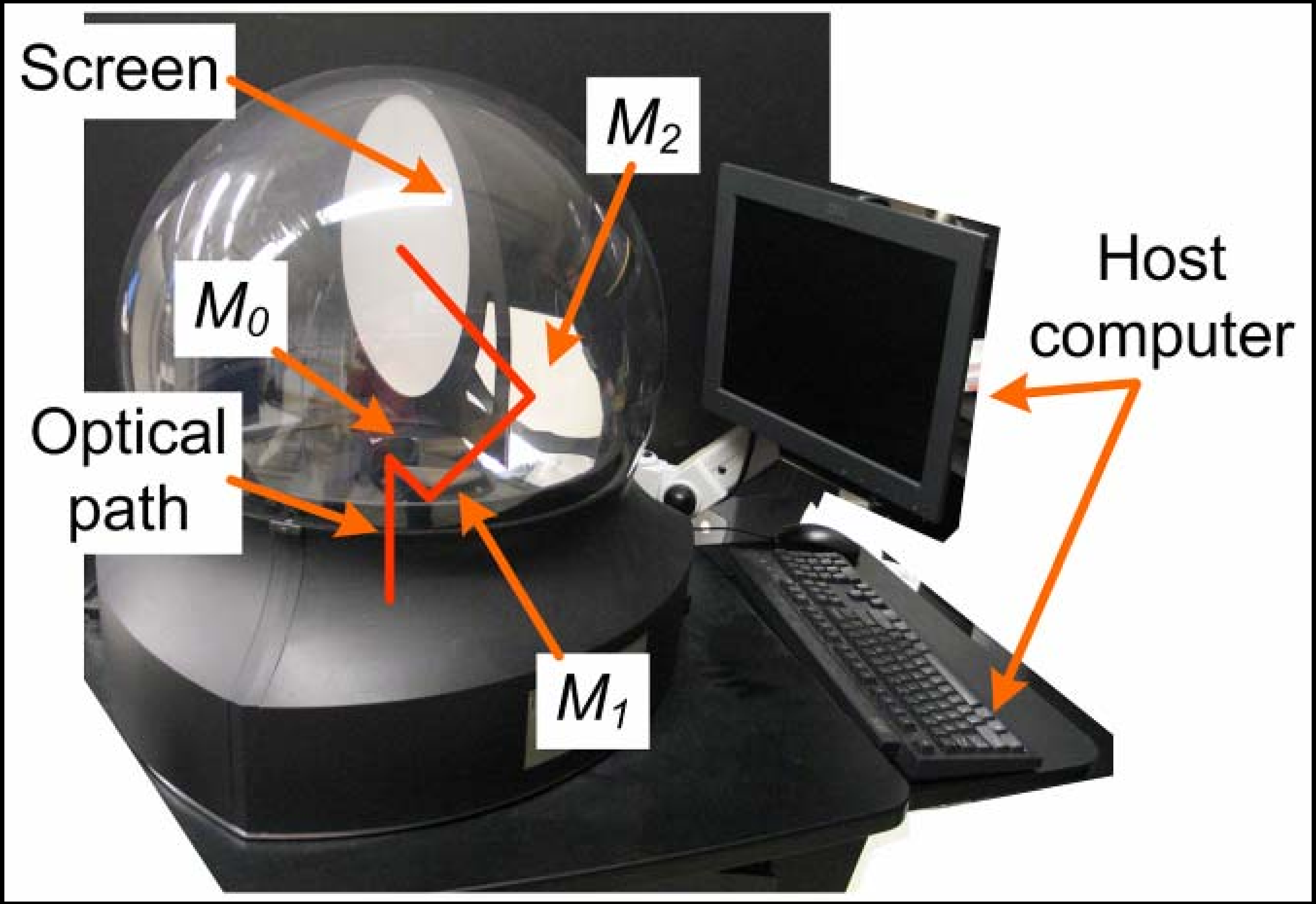

*Chris Hoffmann, Voicu Popescu,*
*Paul Rosen, Zygmunt Pizlo*

# Motivation

- Conventional 3D computer graphics employs 2D displays
  - Graphics system needs to know the view desired by user
  - Image needs to be recomputed for every new view desired by user
  - Image is flat, no stereo cues
- 3D display technology produces a 3D sculpture of light
  - Correct image simultaneously to many users, w/o the need of bulky head gear or imprecise trackers
  - Technology far from mature (low spatial and color resolution, low brightness, low update rates, no opacity)

# Volumetric 3D display

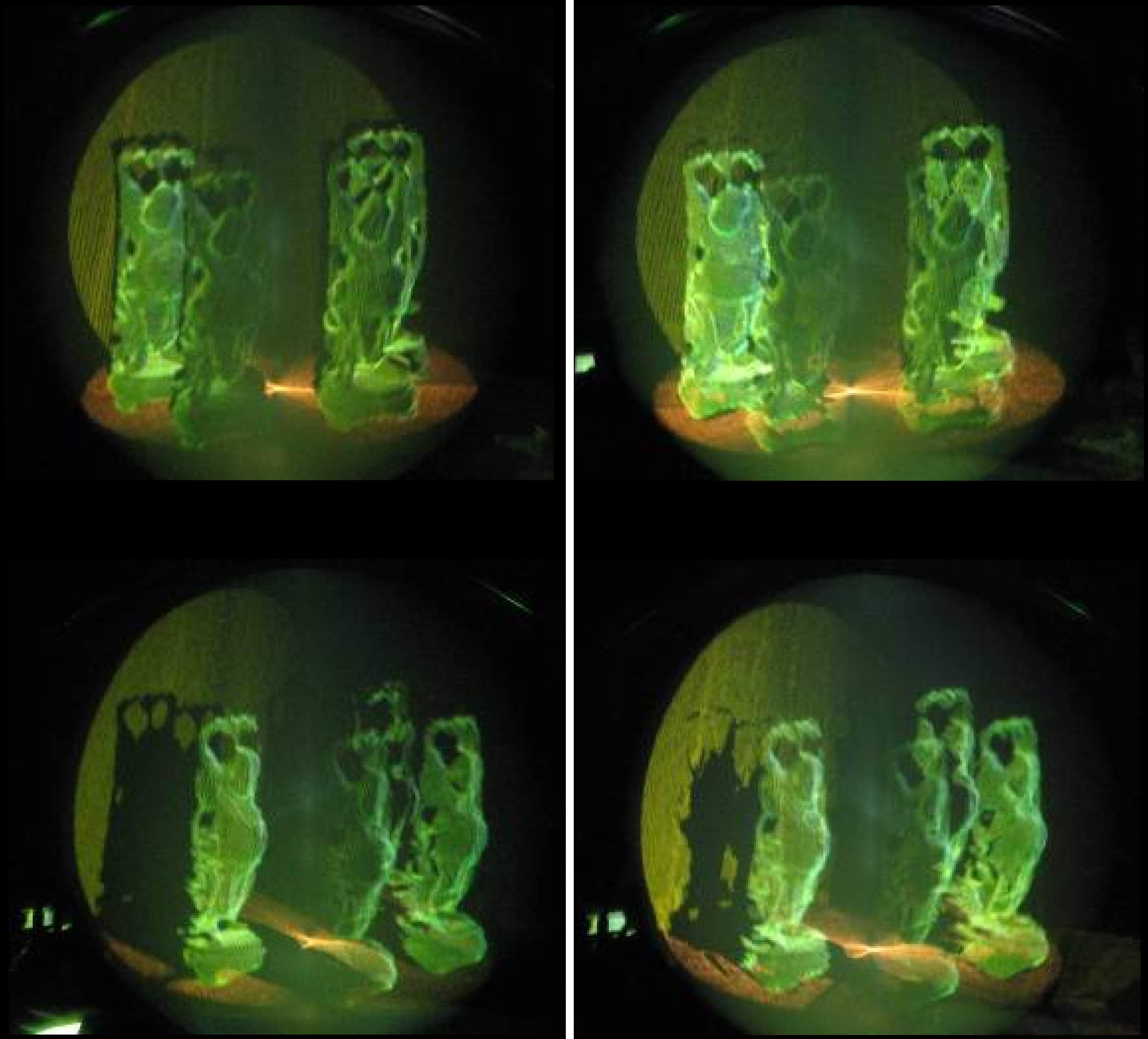
# Volumetric 3D display



# Motivation

- Improve 3D image rendering
- Understand advantages and disadvantages of 3D display
- Improve understanding of human vision system
  - 3D images are uniquely appropriate stimuli in vision psychophysical experiments

# 3D image rendering acceleration



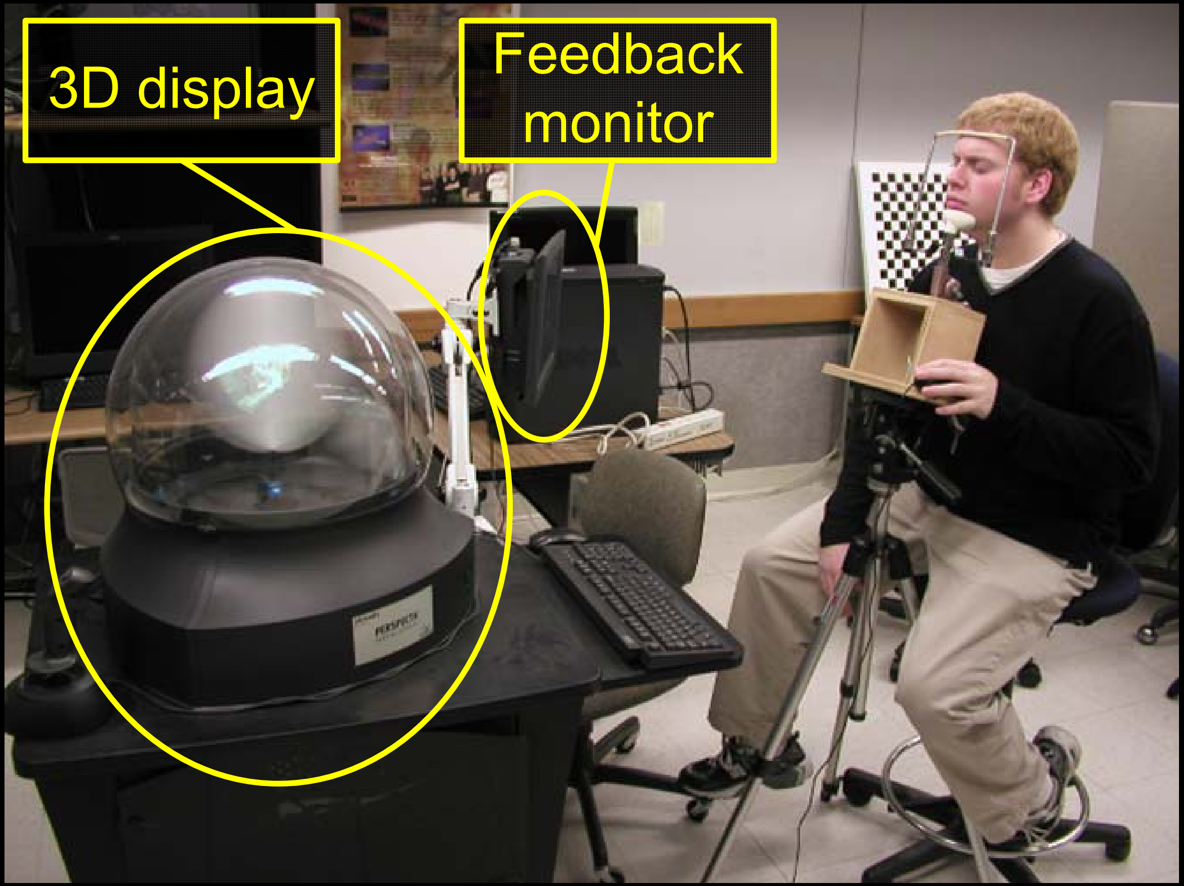
Regular reference image

Occlusion camera reference image

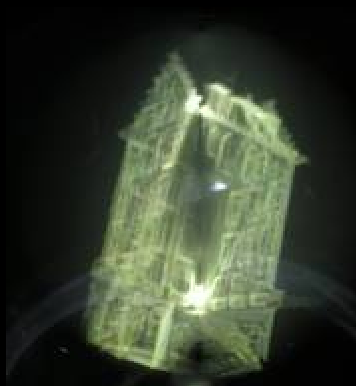
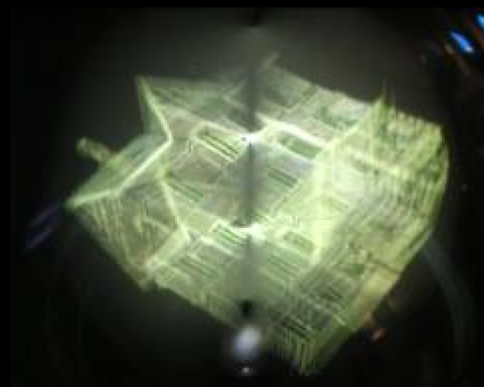
# Psychophysical experiment 1

- Compare several viewing conditions of 3D display and LCD
  - Monocular/binocular
  - With/without motion parallax
  - Near/far
  - LCD
- On each trial an object or a distorted version is shown under arbitrary orientation
- Subject is asked to decide whether the object is distorted or not

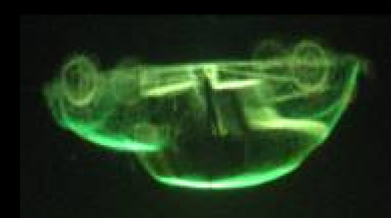
# Experimental setup



# Sample stimuli



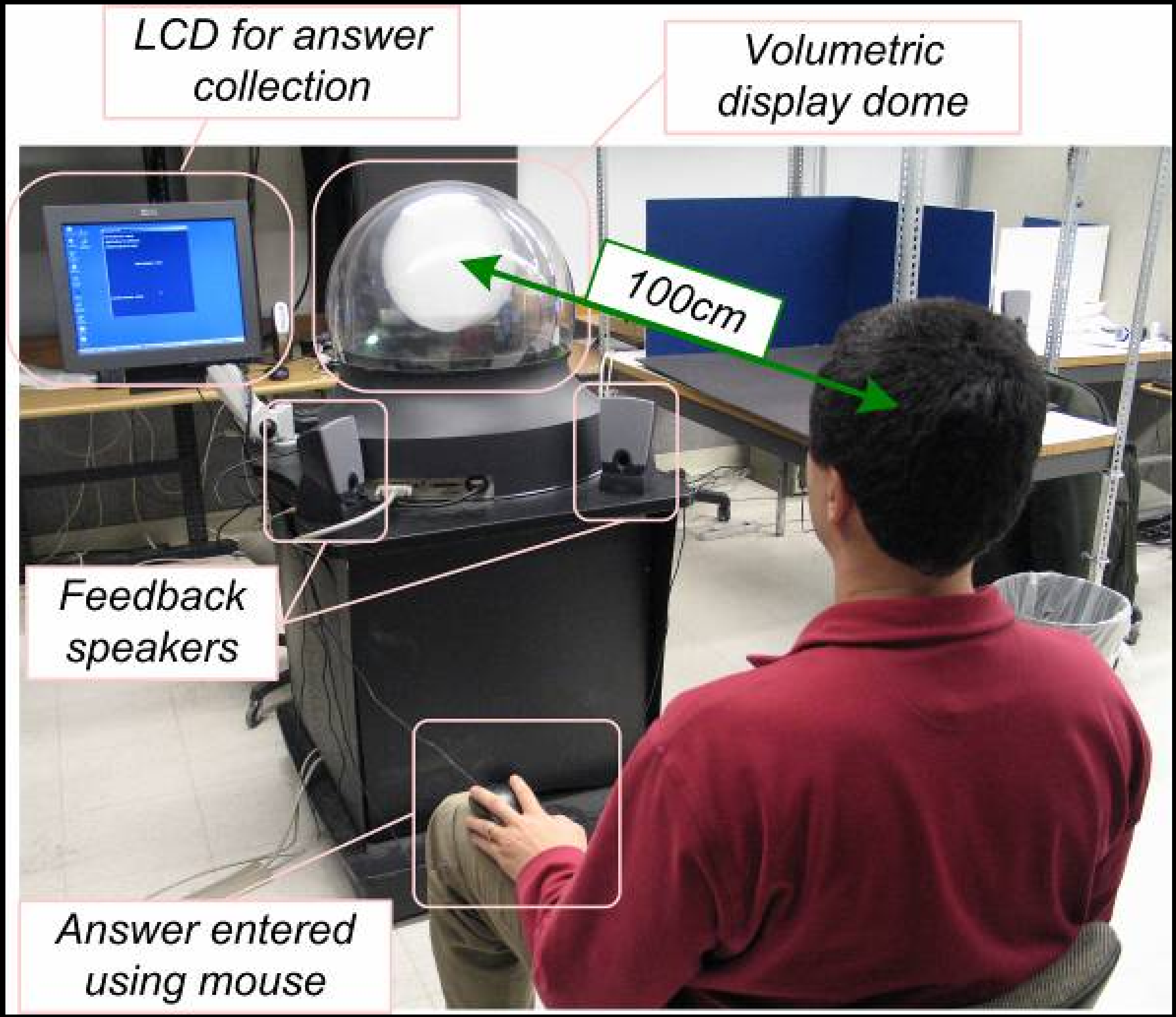
# Conclusion

- Performance in the free viewing condition is reliable and higher than monoscopic viewing of LCD
- Monoscopic viewing of LCD is better than all other 3D display viewing conditions (except free viewing)
  - Complex stimuli are rendered poorly by 3D display

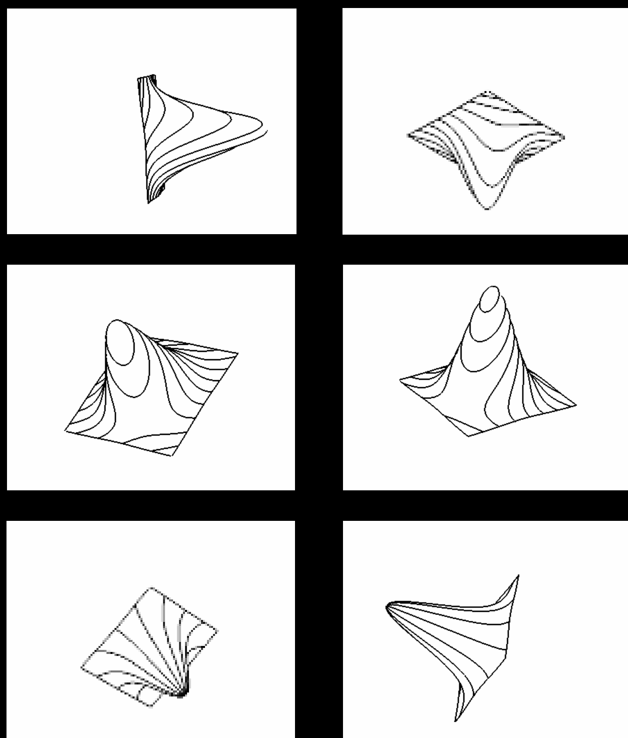
# Psychophysical experiment 2

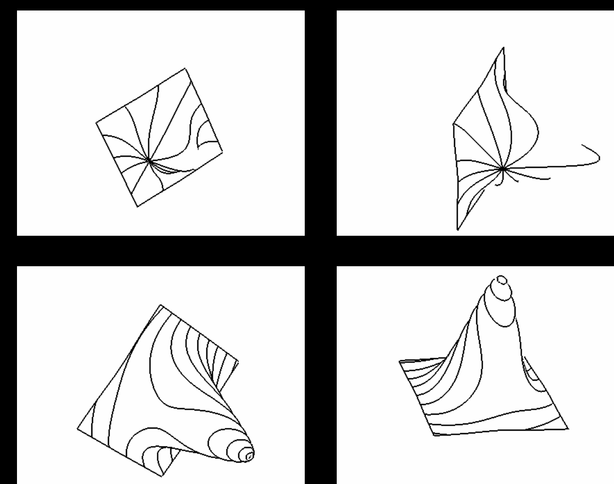
- Compare 3D display with LCD, but use simple 3D stimuli
  - 3D surfaces rendered by contours
- A surface or a distorted variant is shown on each trial
- Subject is asked to decide whether it is distorted or not

# Experimental setup

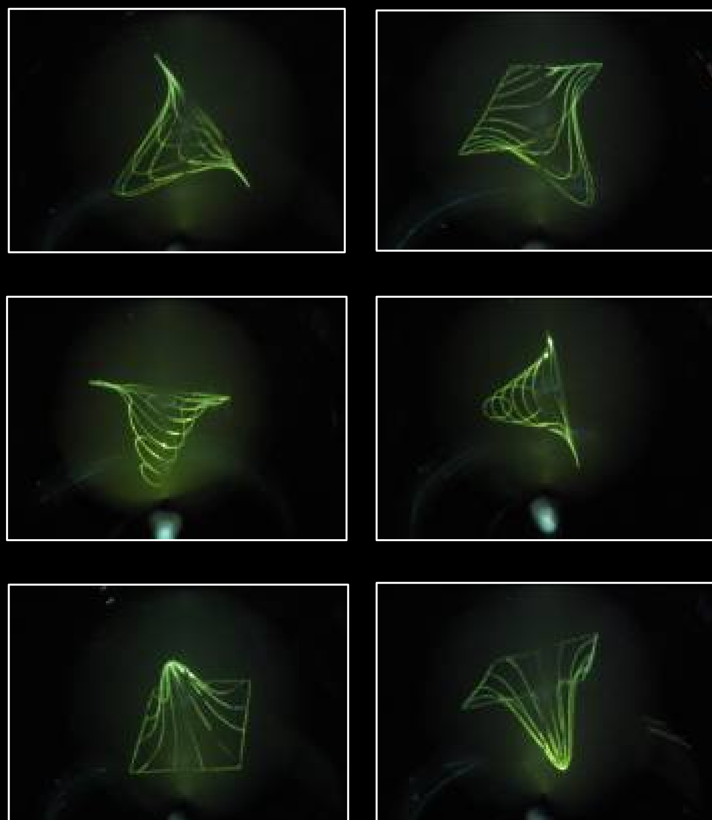


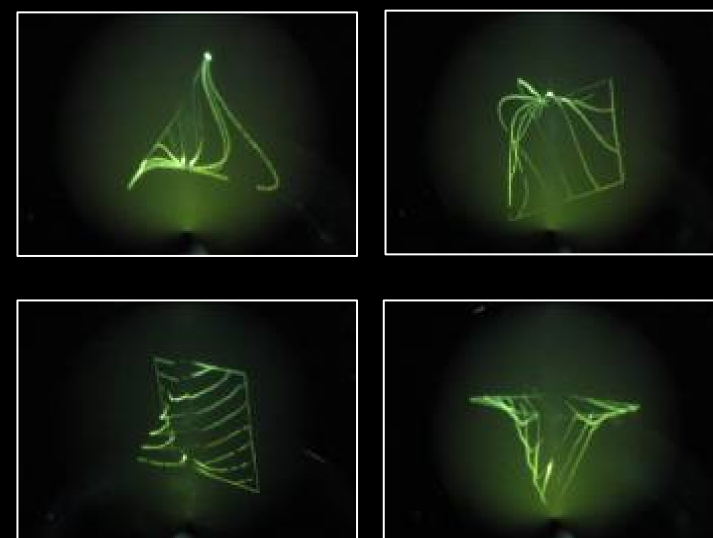
# LCD stimuli





# 3d display stimuli





# Conclusion

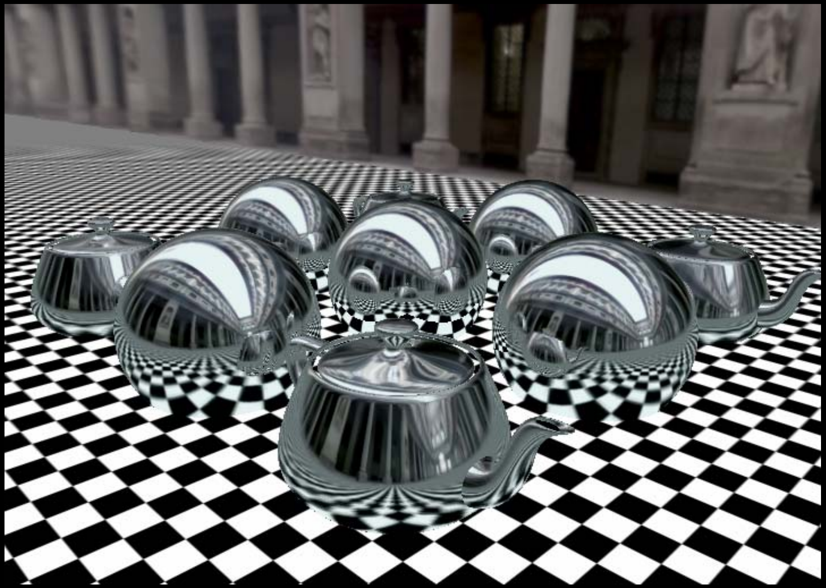
- Stereoscopic performance was substantially higher than the monoscopic performance

# Realistic Reflections at Interactive Rates

*Voicu Popescu, Chunhui Mei,*
*Jordan Dauble, Elisha Sacks*

# Reflections—a difficult problem

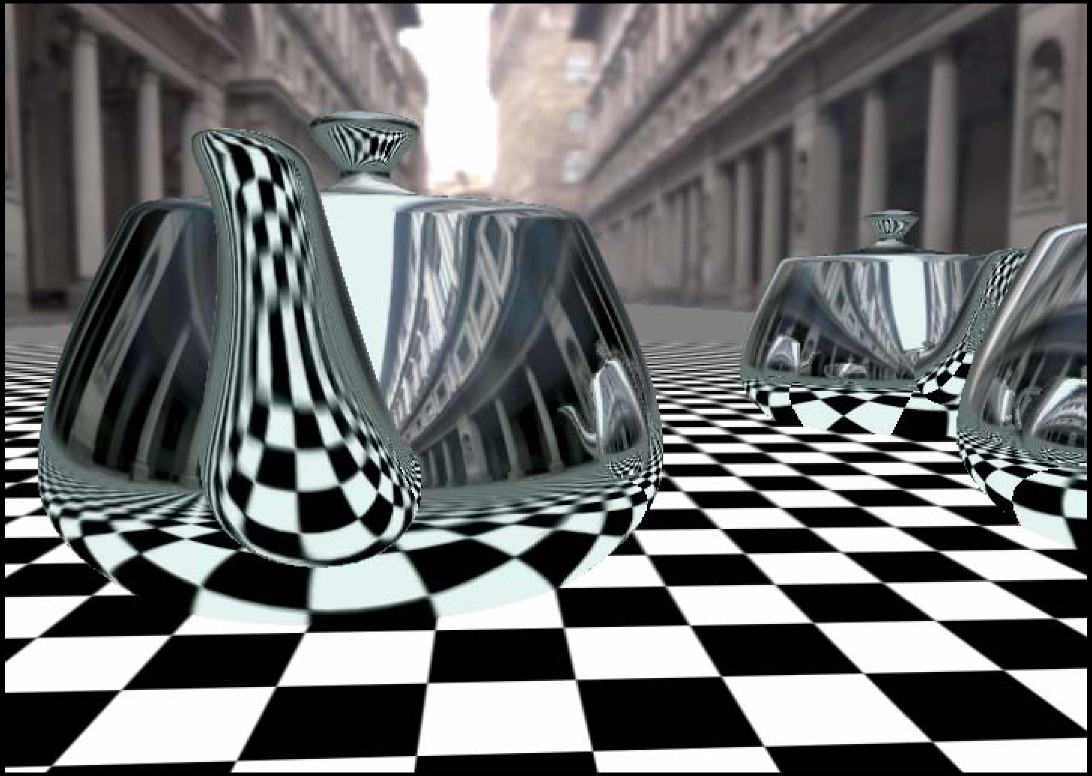
- Every reflector is a portal onto a world which is as rich as the directly observed scene and which has complex image formation laws



# Our approach

- Approximate reflected scene with impostors
  - Considerable prior work on impostors
  - Reflector surface prevents desired viewpoint from getting too close to the impostor
  - Reflection distortion hides impostor artifacts

# Example: 4 teapots



- *D* = 1, *R* = 4, *D+(R-1)+D* = 5 intersections / pix
- 12 second order reflections
- 40fps

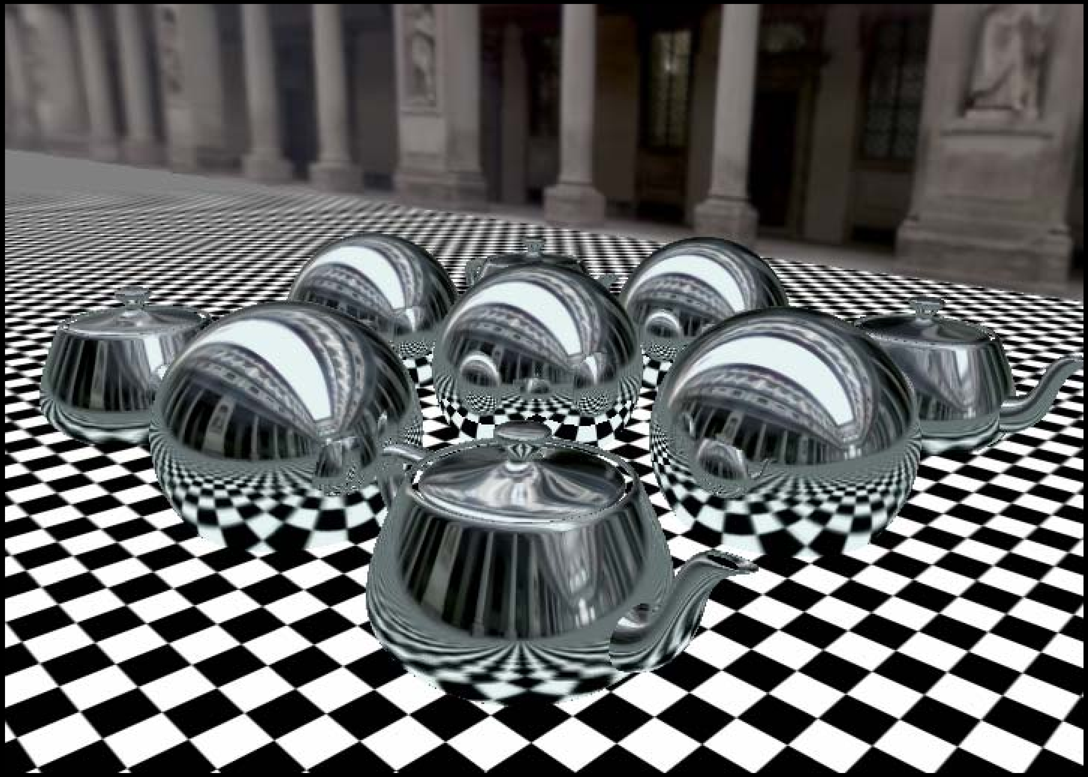
# Example: table scene

- *D* = 2, *R* = 2, *D*+(*R*-1)+*D* = 5 intersections / pix
- 2 second order reflections
- 33 fps

# Example: table scene

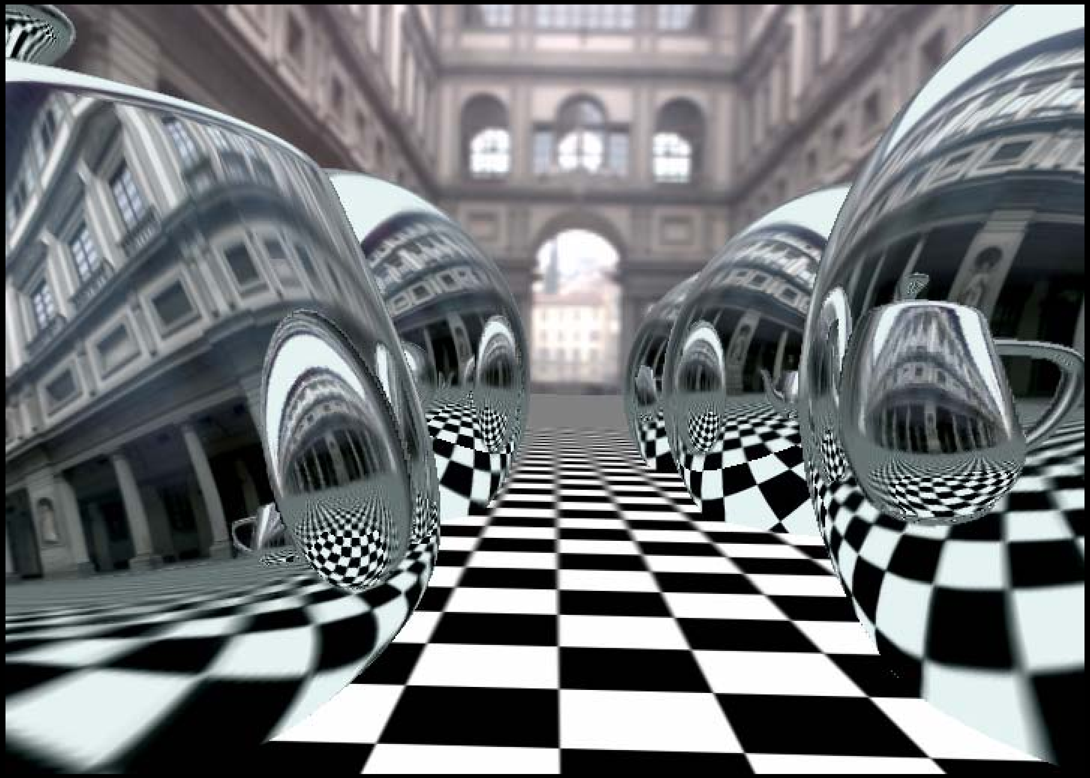
- *D* = 2, *R* = 2, *D*+(*R*-1)+*D* = 5 intersections / pix
- 2 second order reflections
- 33 fps

# Example: pushing-it scene



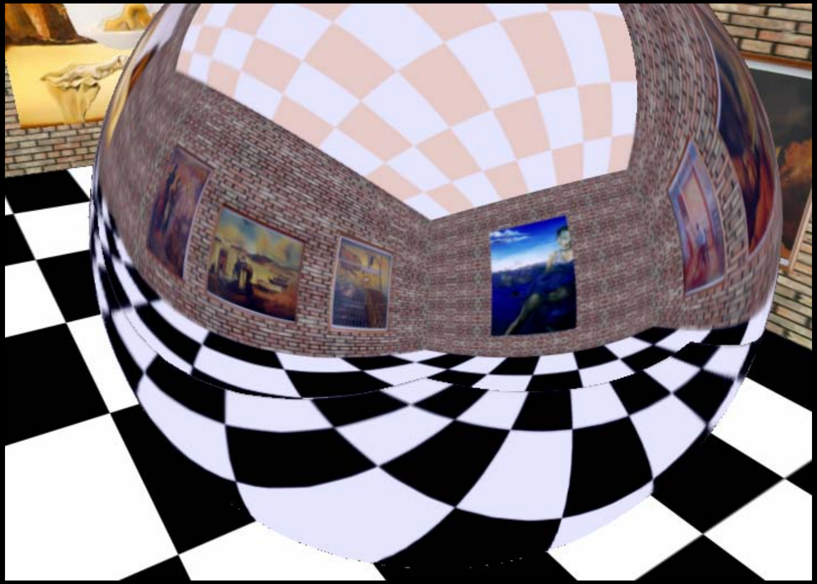
- $D = 2$, $R = 9$, $D+(R-1)+D = 11$ intersections / pix
- 72 second order reflections
- 11 fps

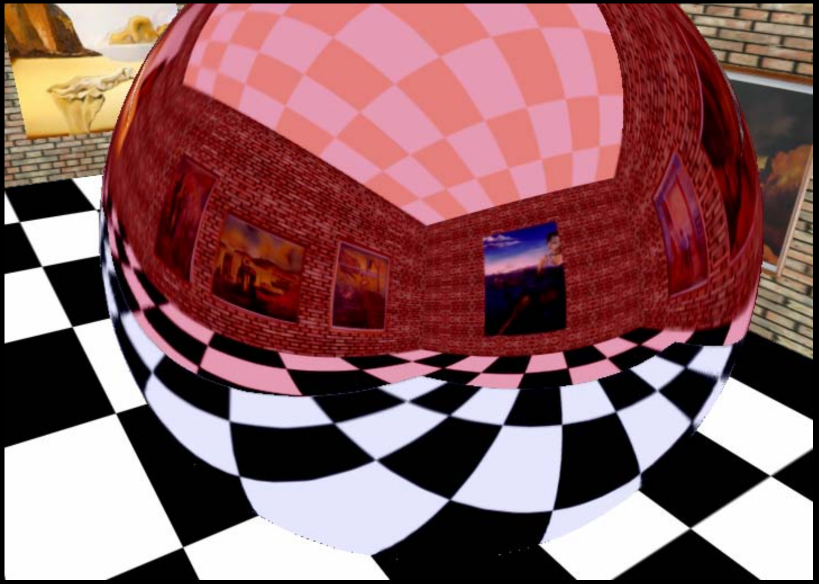
# Example: pushing-it scene



- *D* = 2, *R* = 9, *D*+(*R*-1)+*D* = 11 intersections / pix
- 72 second order reflections
- 6 fps

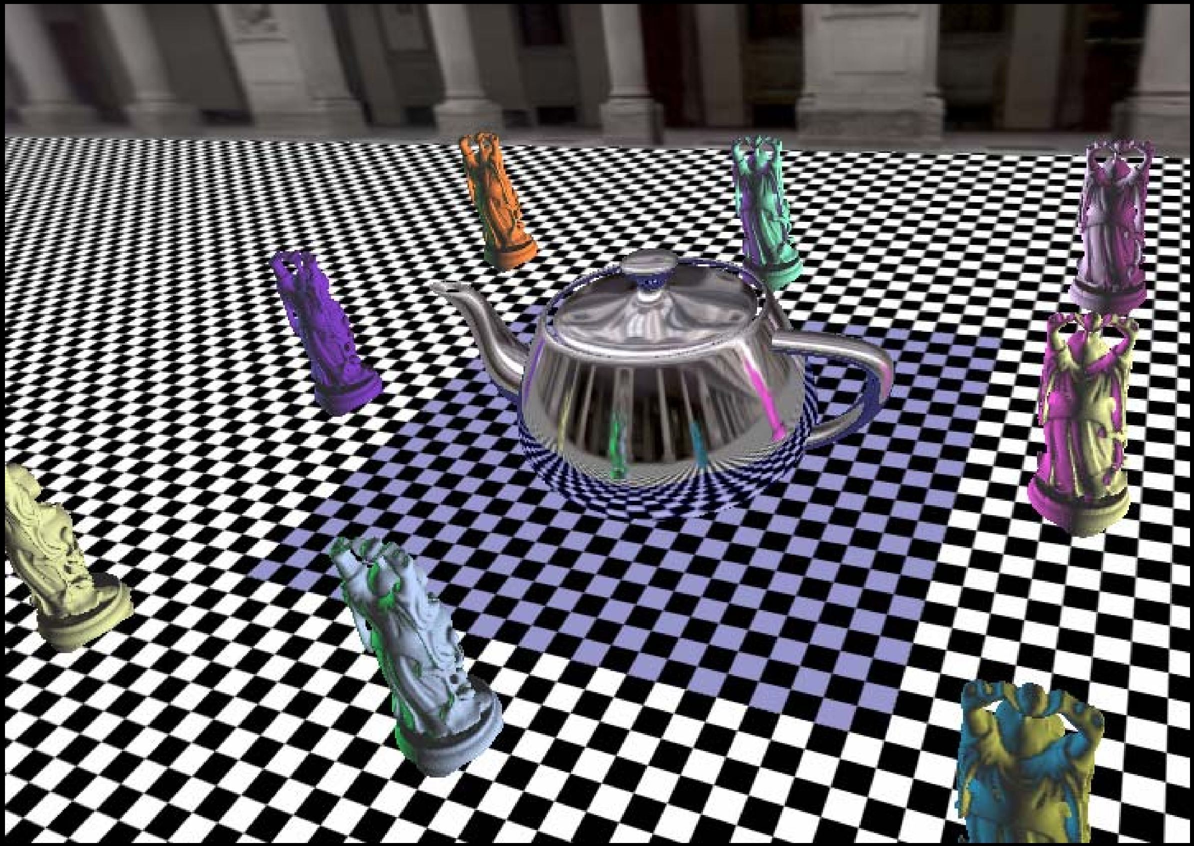
# Problem



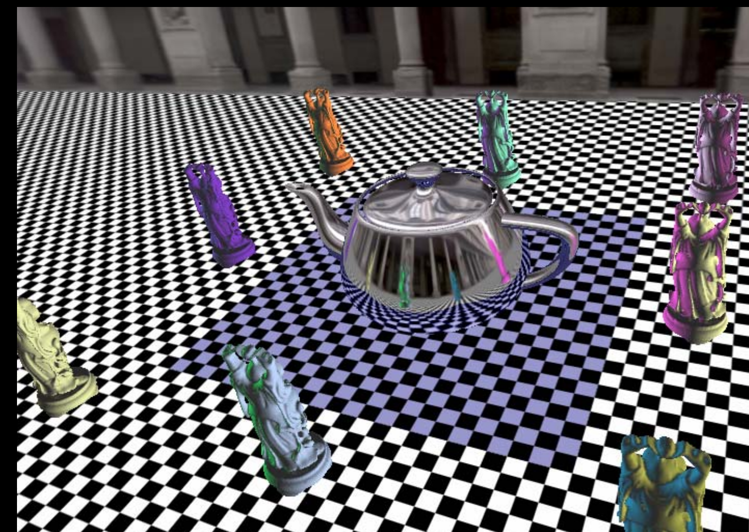


Transition from impostor to environment map (*red* in left image) is discontinuous.

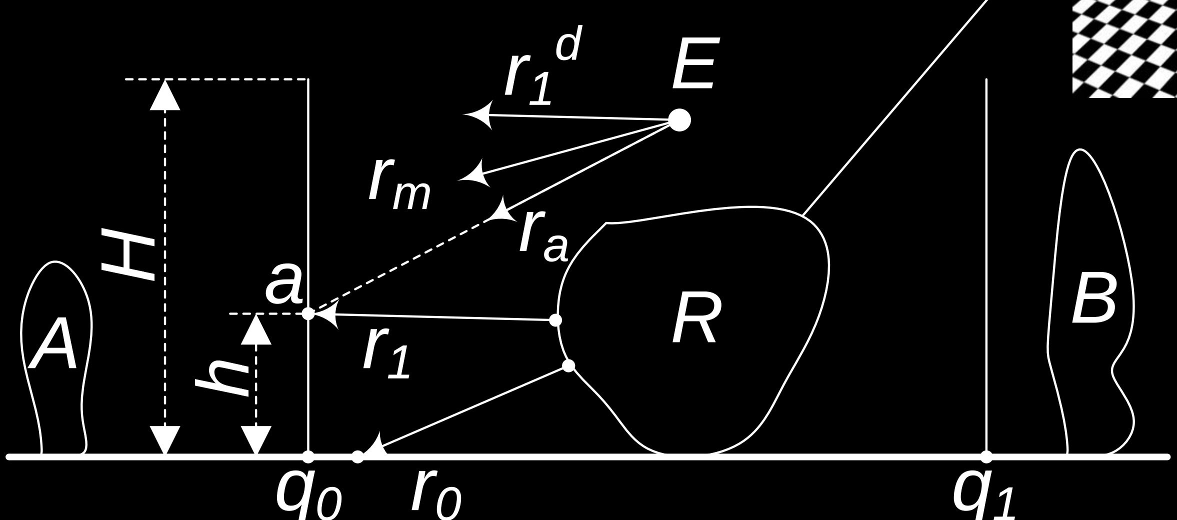
# Solution: ray morphing



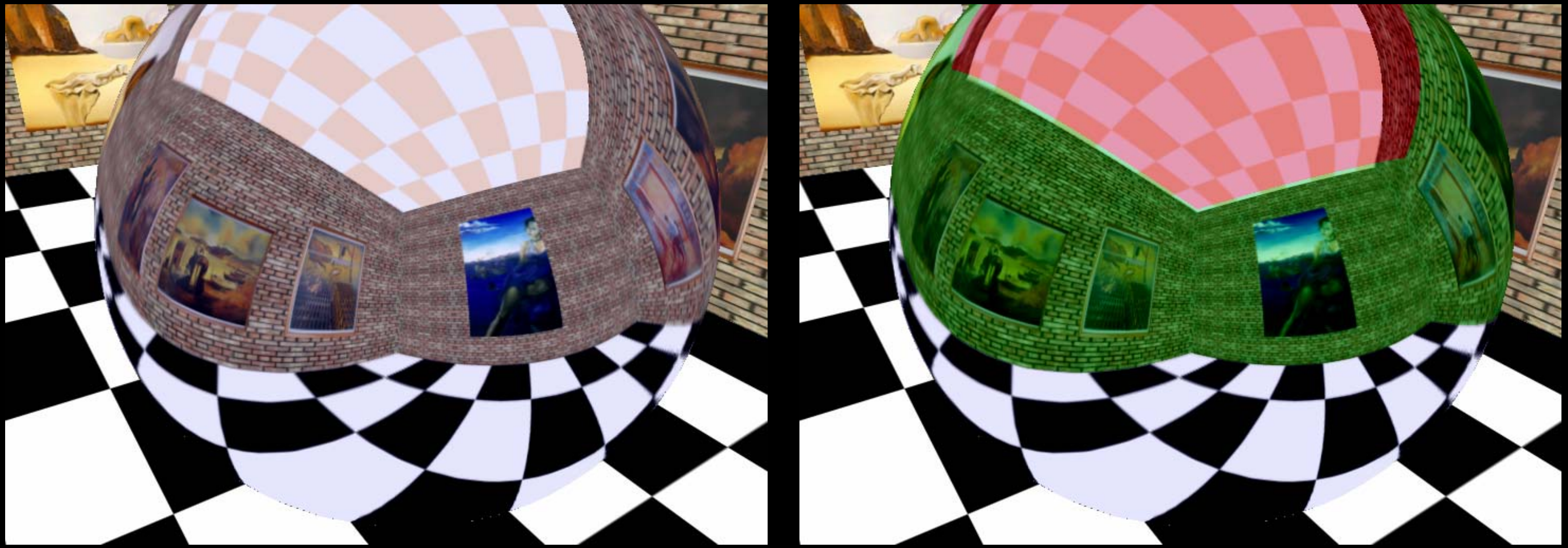
# Solution: ray morphing



$$r_m = r_a + (r_1^d - r_a)h/H$$



# Solution



*Left*—continuous transition. *Right*—morph region (*green*), environment map (*red*).

# Attenuation w/ distance

# Fresnel

# Combined effects

# Billboard limitations

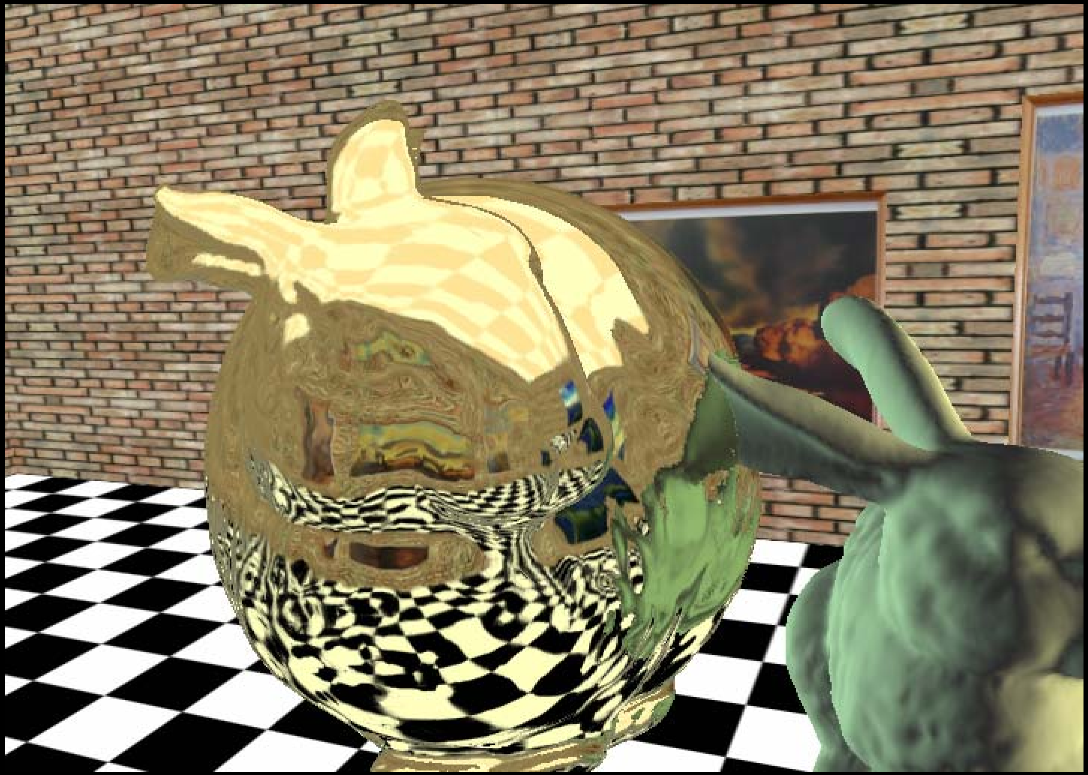
- No support for objects very close to the reflector
- Limited accuracy
  - Flat reflection
  - Lack of motion parallax

# Depth image impostor results

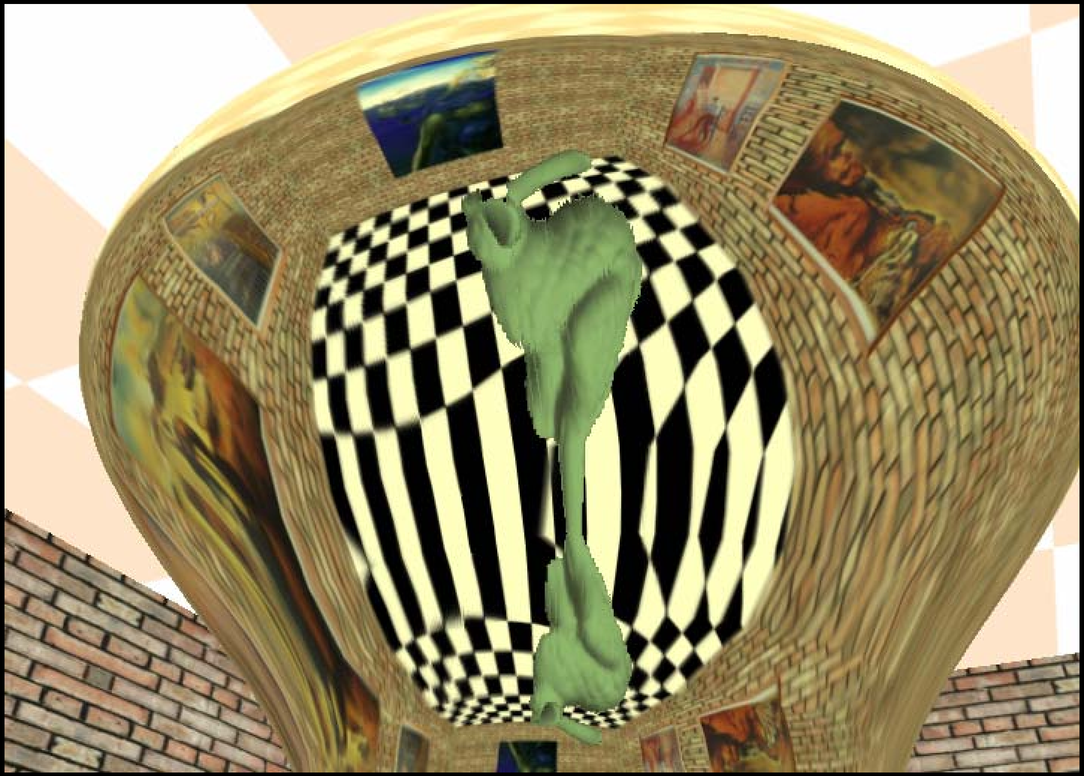
# Depth image impostor results

# Depth image impostor results

# Depth image impostor results



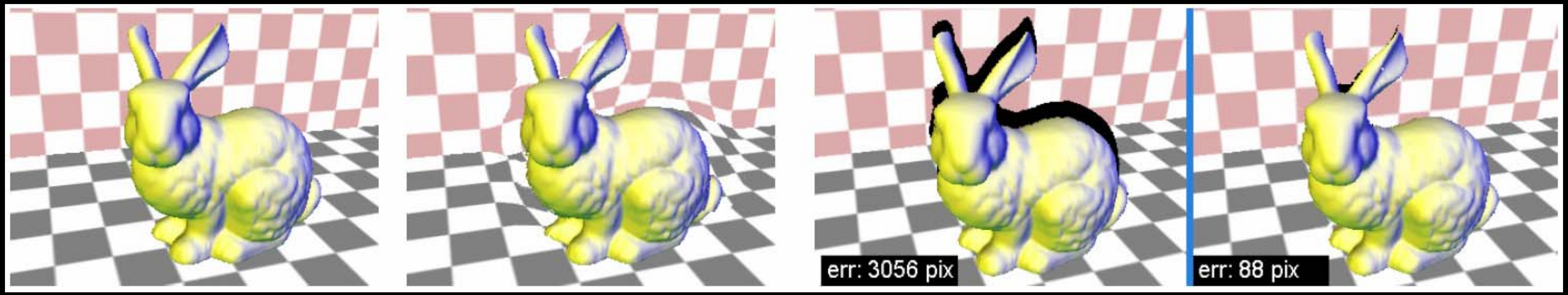
# Depth image impostor results



# Depth image impostor results

# Future work

- Other types of impostors
  - occlusion-resistant



# The end

- The end

# Conclusions

- The reflected-impostor approach works
  - Fast, realistic
  - Increased modeling effort
- Rendering reflections reduced to the lesser problem of rendering w/ impostors

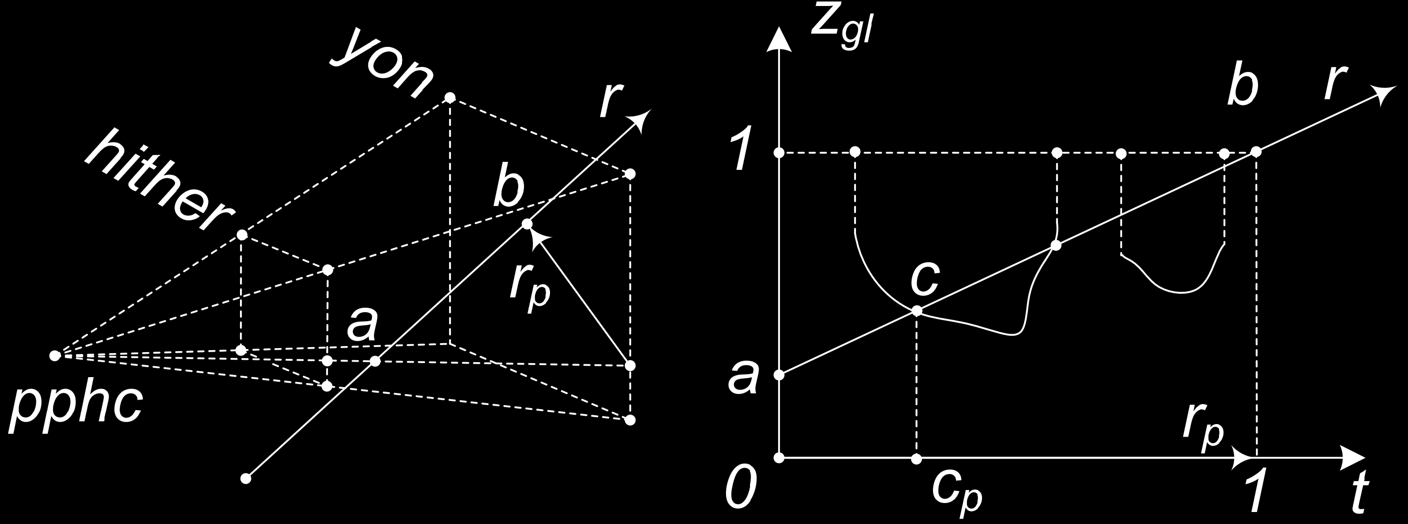
# Future work

- Other types of impostors
- Other BRDFs
- Self-reflections
- Constructing the SRDMs on the GPU

# Depth image impostors

- Impostor has to provide
  - Fast construction YES
  - Fast intersection with ray ???
  - Antialiasing YES

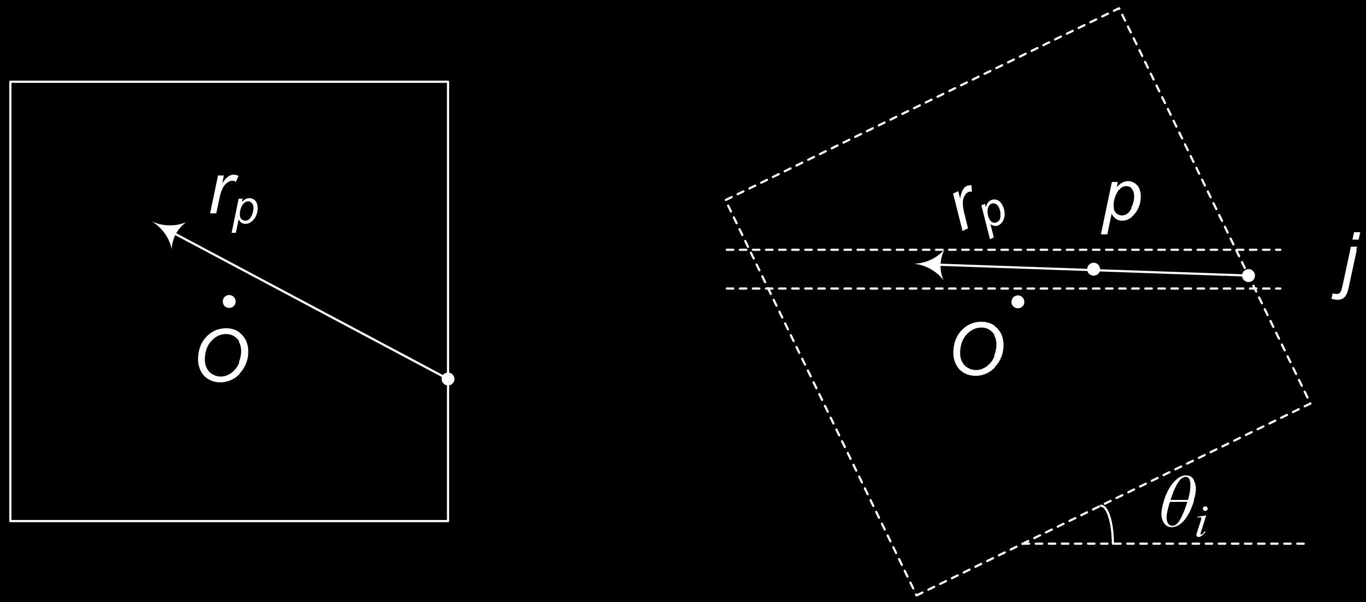
# Depth image—ray intersection




Epipolar-like constraints: intersection computed as 1D search

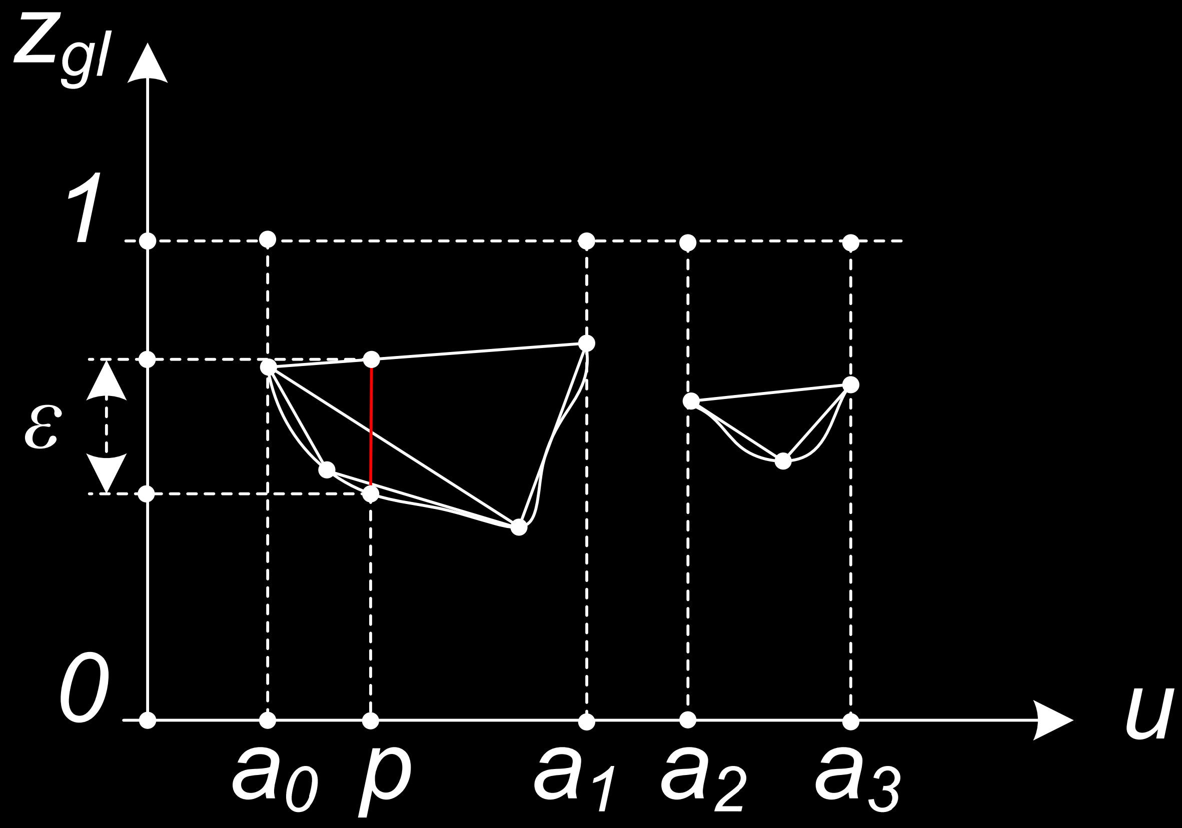
Still too many steps along epipolar segment

# Simplified Rotated Depth Maps




Pre-rotate depth map.
All rays ever needed project to rows.
Pre-simplify rows.

# Simplified Rotated Depth Maps



# Visualization of 9/11 Attack on the Pentagon

*Voicu Popescu, Chris Hoffmann,*
*Sami Kilic, Mete Sozen, et al.*

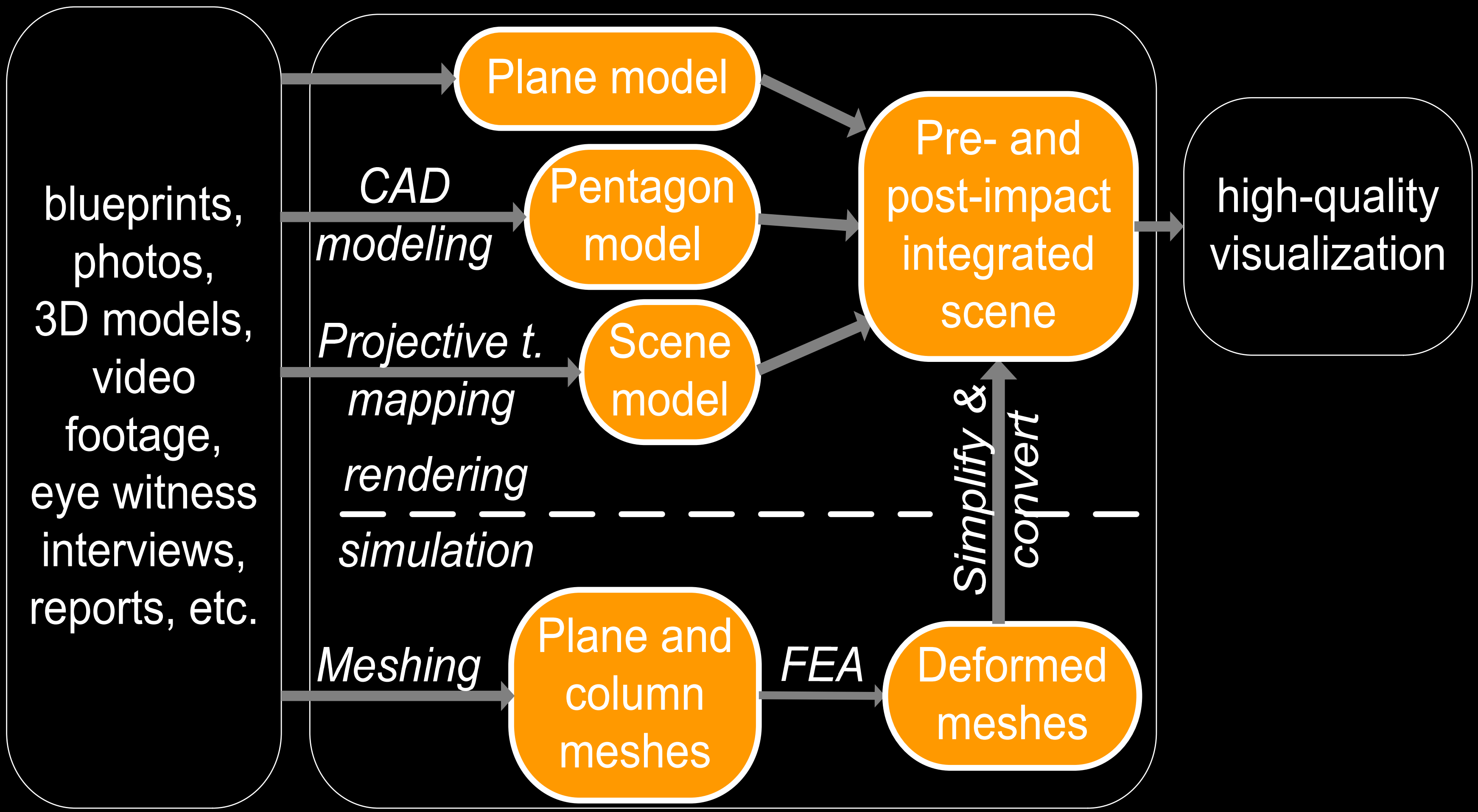
# Motivation

- Understand the behavior of the Pentagon building during the September 11 attack
- Improve visualization of simulation results
  - Use state-of-the-art graphics and visualization techniques
  - Enable realistic visualization, for dissemination of simulation results to non-specialists

# Approach

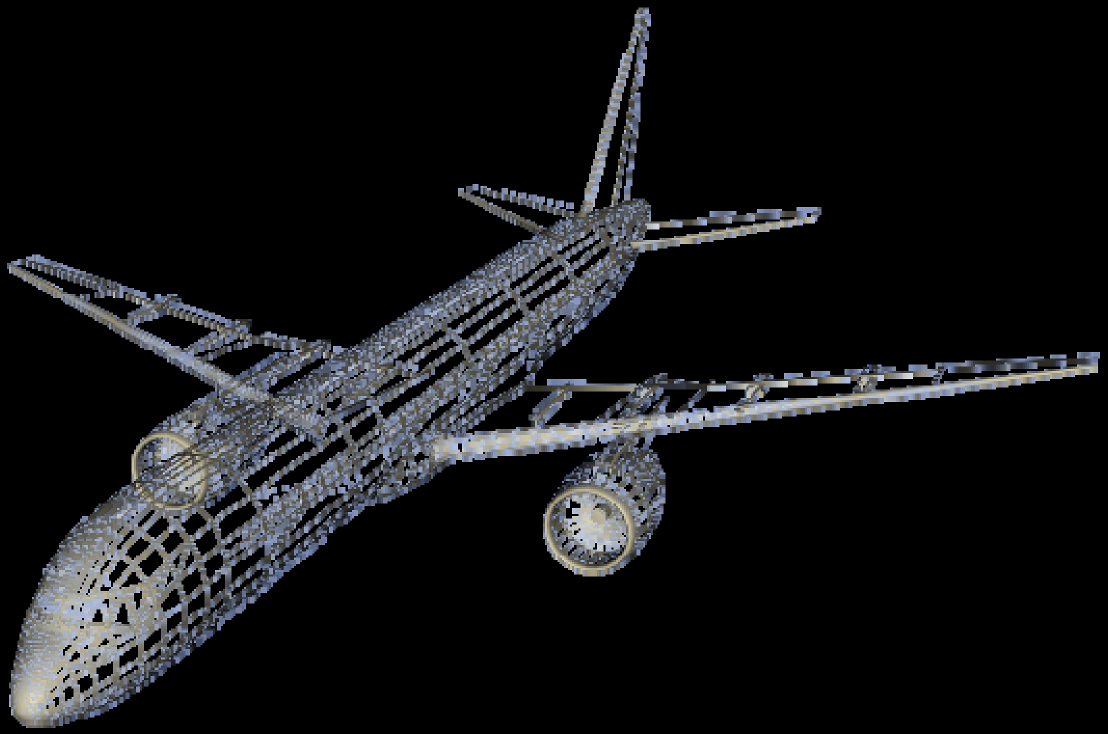
- Create a general, scalable, and reusable link between the worlds of simulation (*accurate physics*) and animation (*accurate visuals*)
  - Import simulation results into animation SW system

# Overview

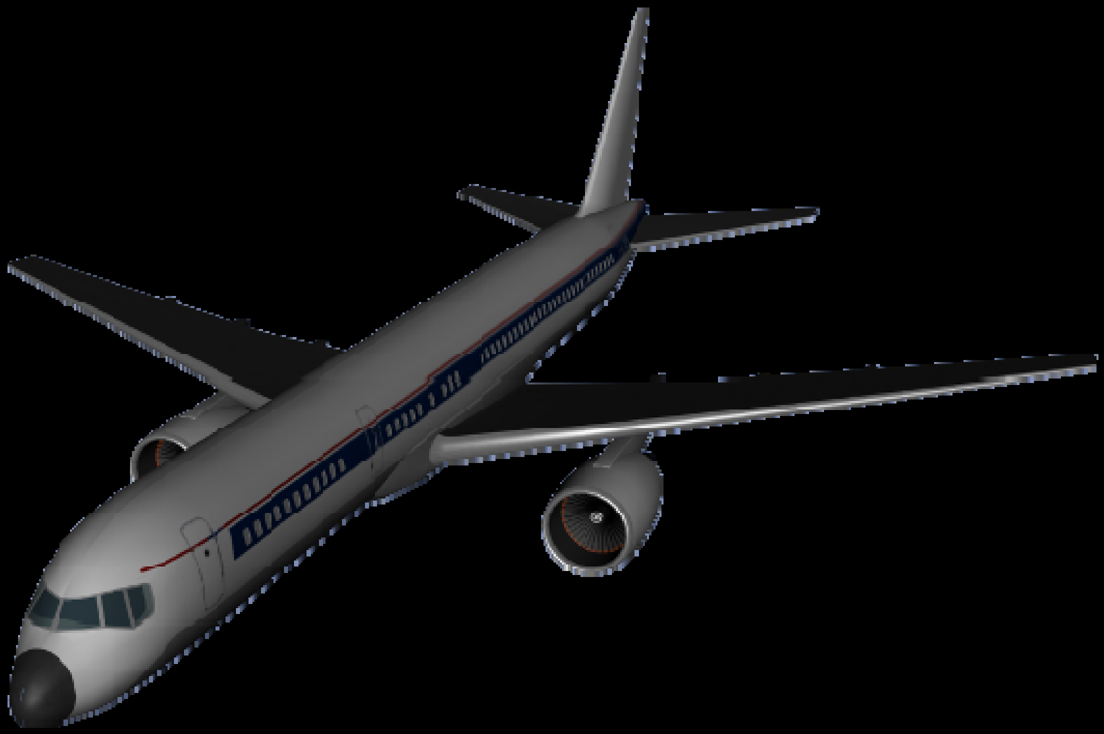


blueprints, photos, 3D models, video footage, eye witness interviews, reports, etc.
Plane model
CAD modeling
Pentagon model
Projective t. mapping
Scene model
rendering
simulation
Meshing
Plane and column meshes
FEA
Deformed meshes
Simplify & convert
Pre- and post-impact integrated scene
high-quality visualization

# (rendering) Plane model

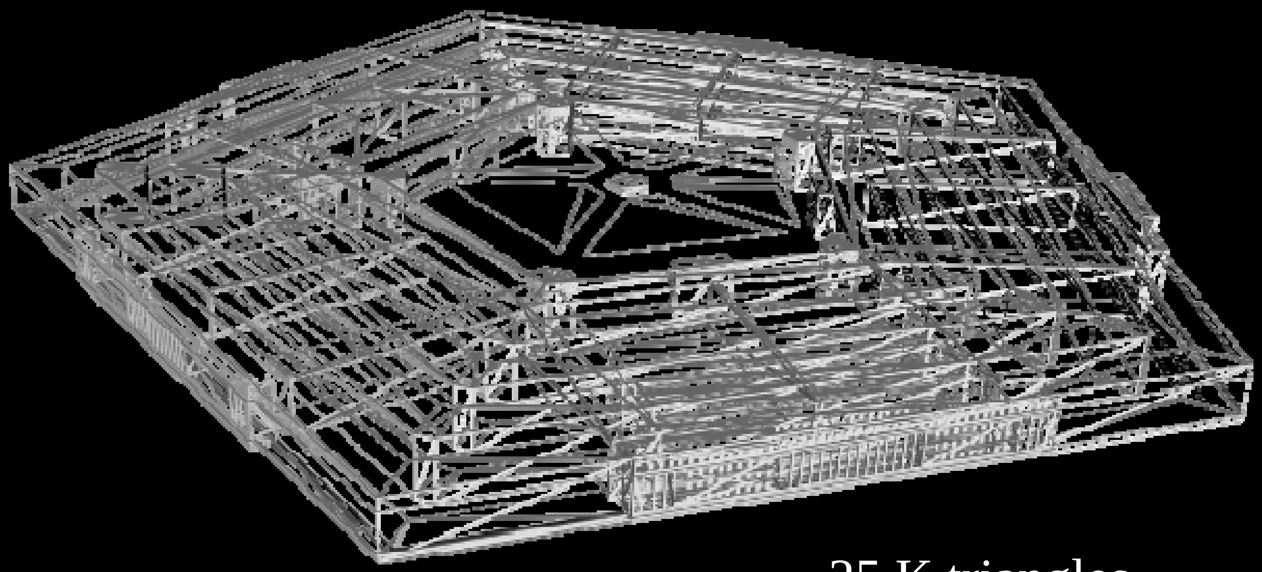


# *(rendering)* Plane model



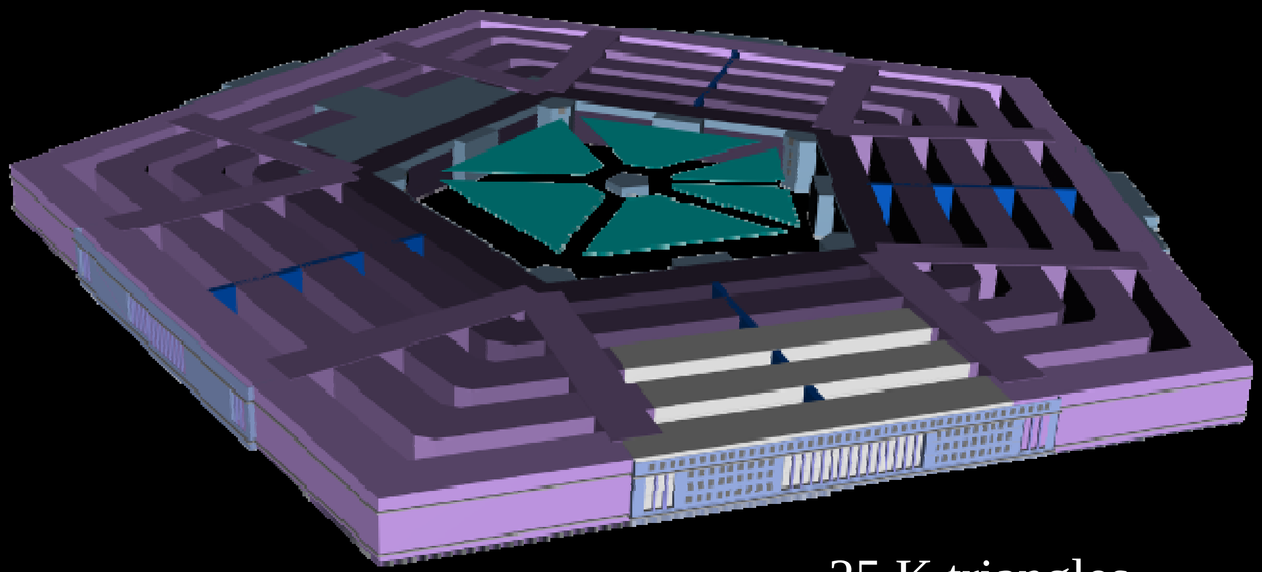
Courtesy Amazing 3D graphics Inc.

# (rendering) Pentagon model



25 K triangles

# (rendering) Pentagon model



25 K triangles

# Satellite images (Ikonos, 1 m, true color)

September 07, 2001

Courtesy Space Imaging

# Satellite images (Ikonos, 1 m, true color)

September 12, 2001

Courtesy Space Imaging

# Satellite images (Ikonos, 1 m, true color)

September 12, 2001

Courtesy Space Imaging

# (rendering) Scene model

# (rendering) Scene model

# (rendering) Visualization of approach

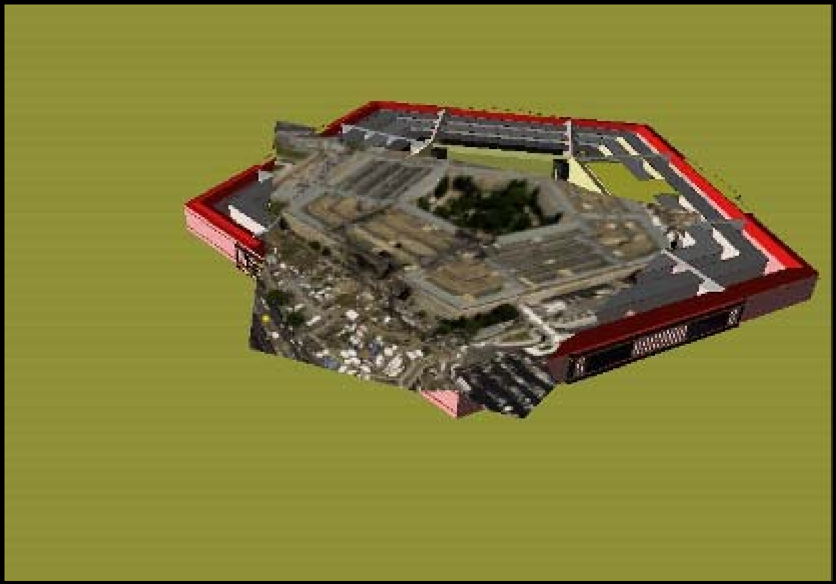
# Helicopter mounted camera

- Texturing with high resolution aerial and satellite images

Courtesy ASCE

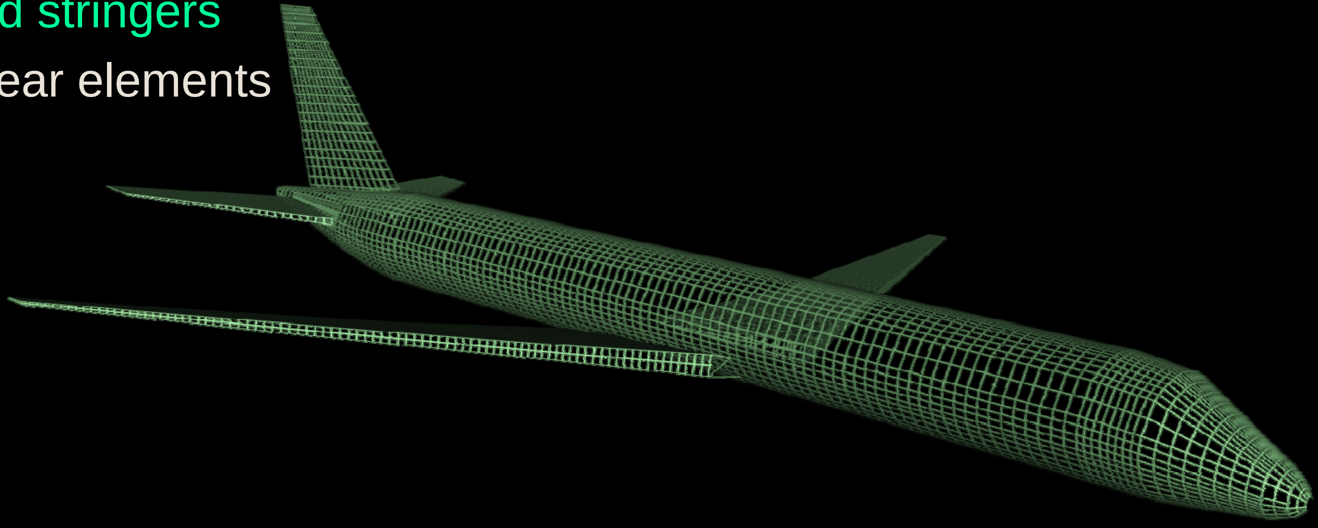
# (rendering) Scene after impact

- Difficult to model conventionally
- Modeled using projective texture mapping



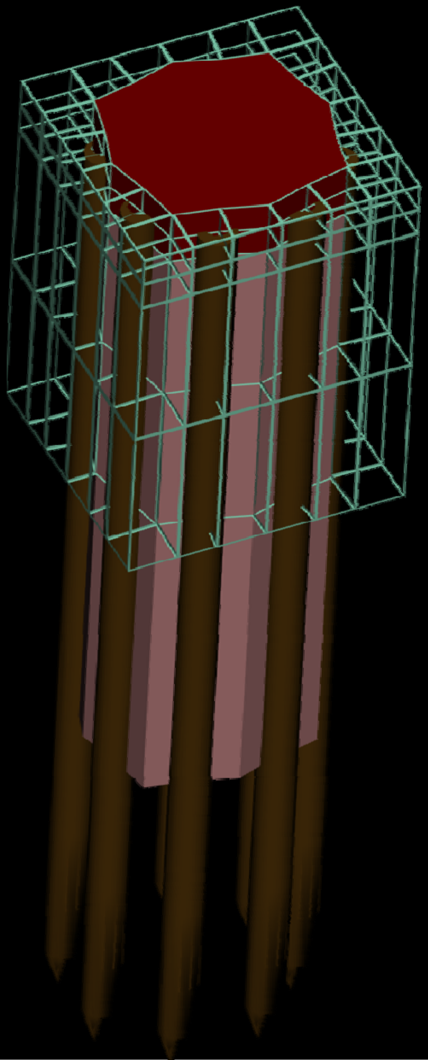
# (simulation) Finite element meshes

- Custom mesher
- Scene simplified to most relevant components
  - Aircraft
    - Tanks filled with fuel
      - Eulerian mesh, 15 $cm^3$ hexahedral elements, fractional occupancy
    - Fuselage & floor
      - Quadrilateral thin shell elements
    - Ribs and stringers
      - Linear elements

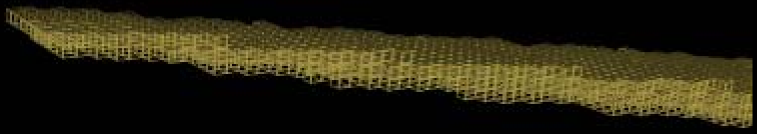


# (simulation) Finite element meshes

- Custom mesher
- Scene simplified to most relevant components
  - Aircraft
  - Building columns
    - Concrete core
      - 7.5cm x 7.5cm x 15cm hexahedral elements
    - Rebars
      - 15 cm linear elements
    - Fluff
      - 7.5cm x 7.5cm x 15cm hexahedral elements
- Total 954 K nodes

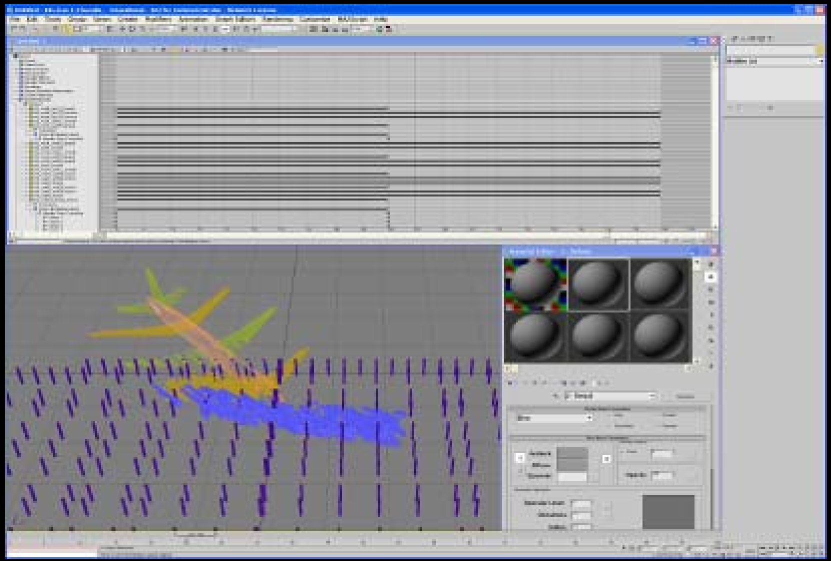


# (simulation) Finite element meshes

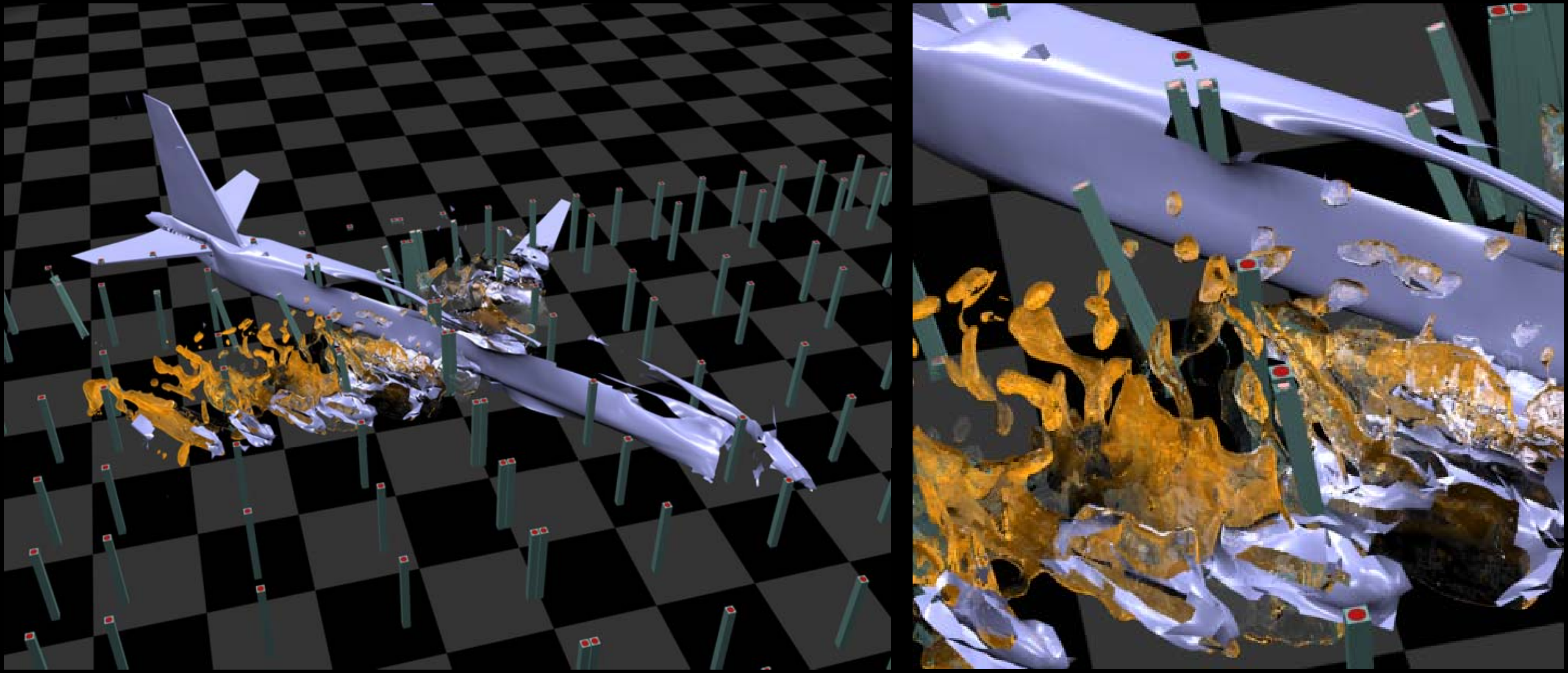


# Visualization of simulation results

- Custom plugin to import FEA results into animation software
  - Complex light and material editors
  - Complex rendering algorithms
  - Good camera control
  - Integration with surrounding scene



# Visualization of simulation results



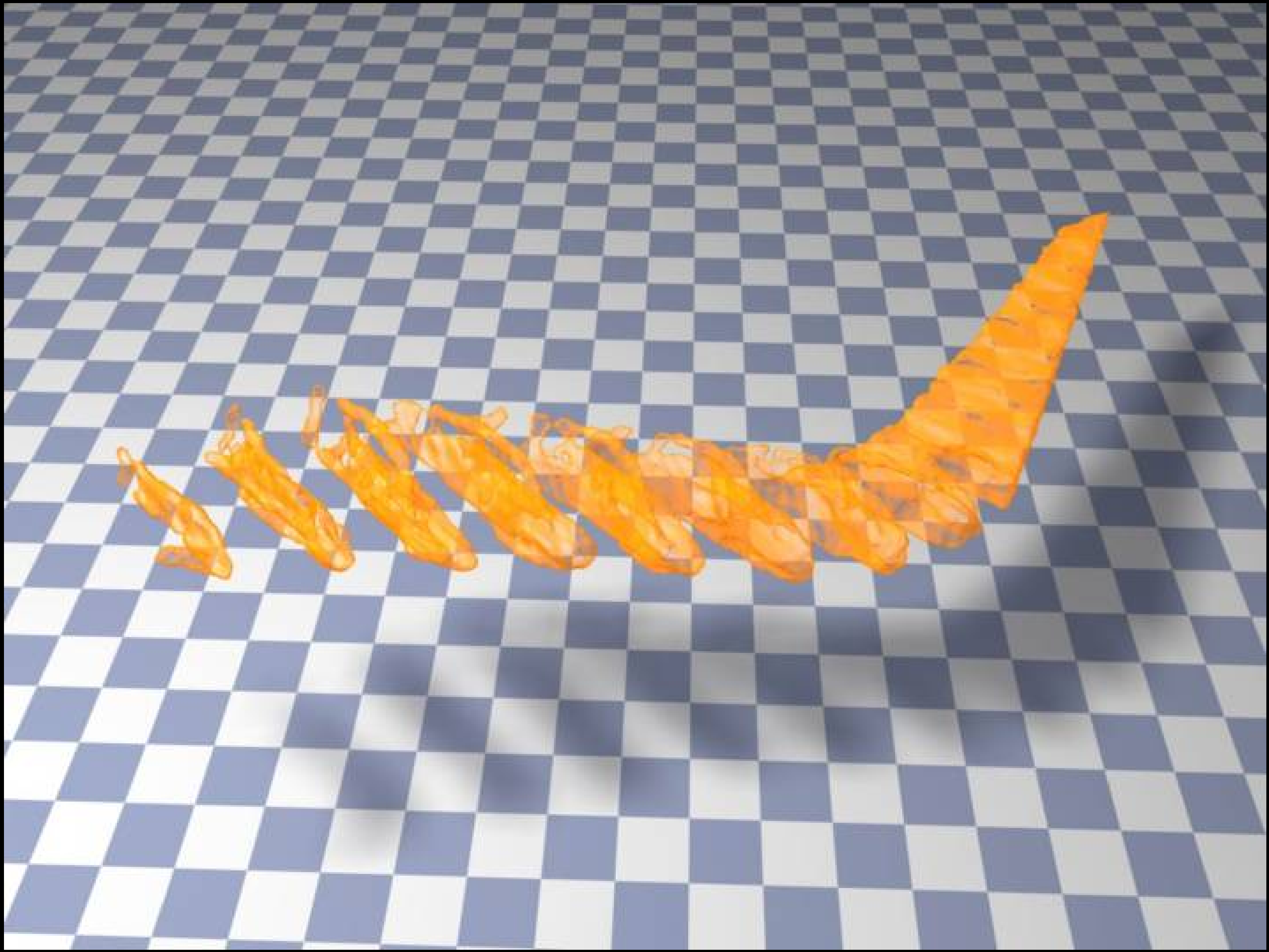
# Visualization of simulation results

# Visualization of simulation results

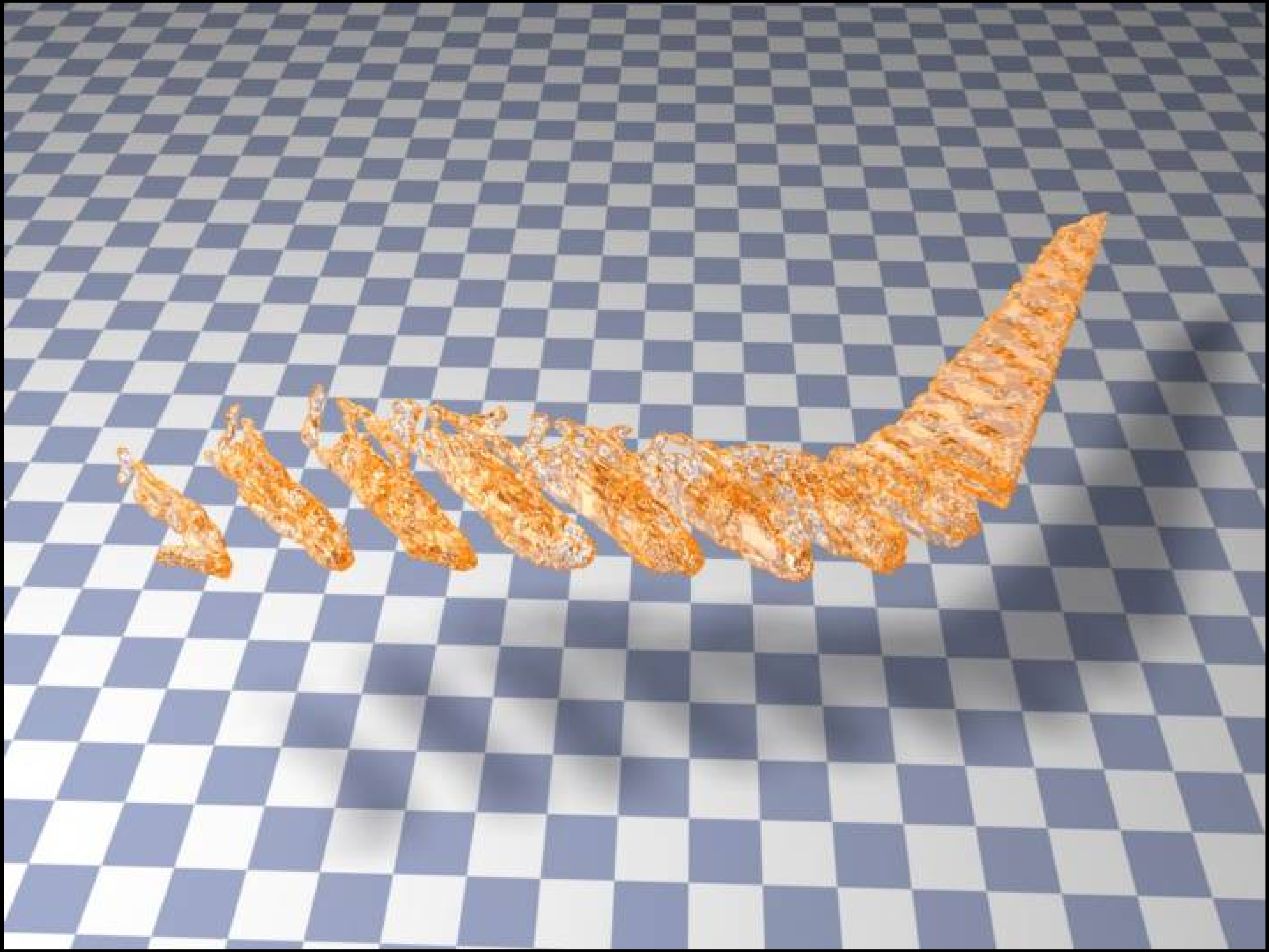
# Liquid objects (jet fuel)

- Liquid modeling and animation: challenging
  - Solution 1: per state isosurfaces
    - (+) Good shape approximation,
    - (-) Discontinuous animation
      - 30 s: 50 states over 900 frames
  - Solution 2:
    - Threshold fractional occupancy data
    - Remove internal faces
    - Relax mesh
    - Animate nodes linearly between FEA states
    - Phase state liquid in and out

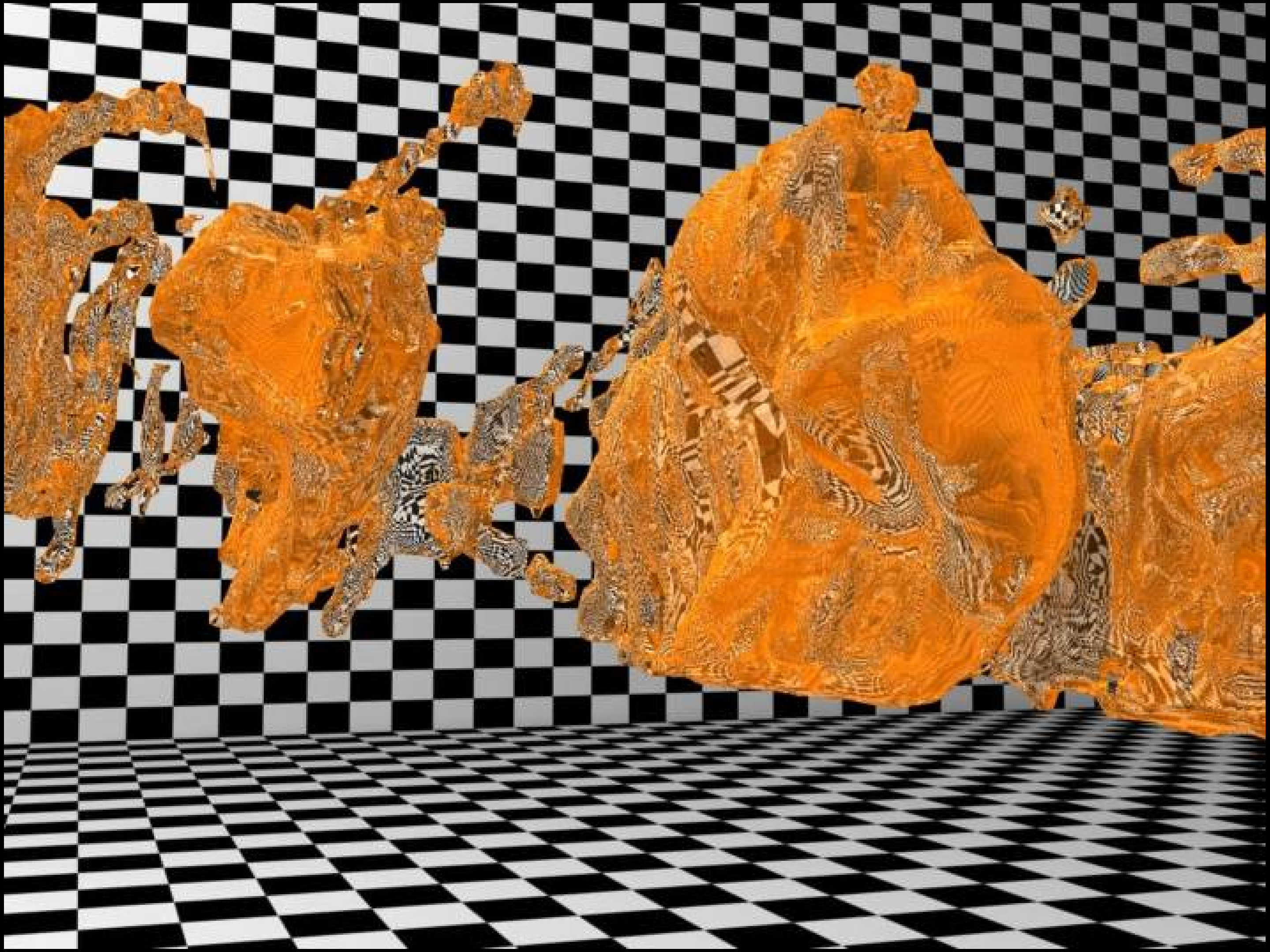
# Liquid visualization (alpha blending)



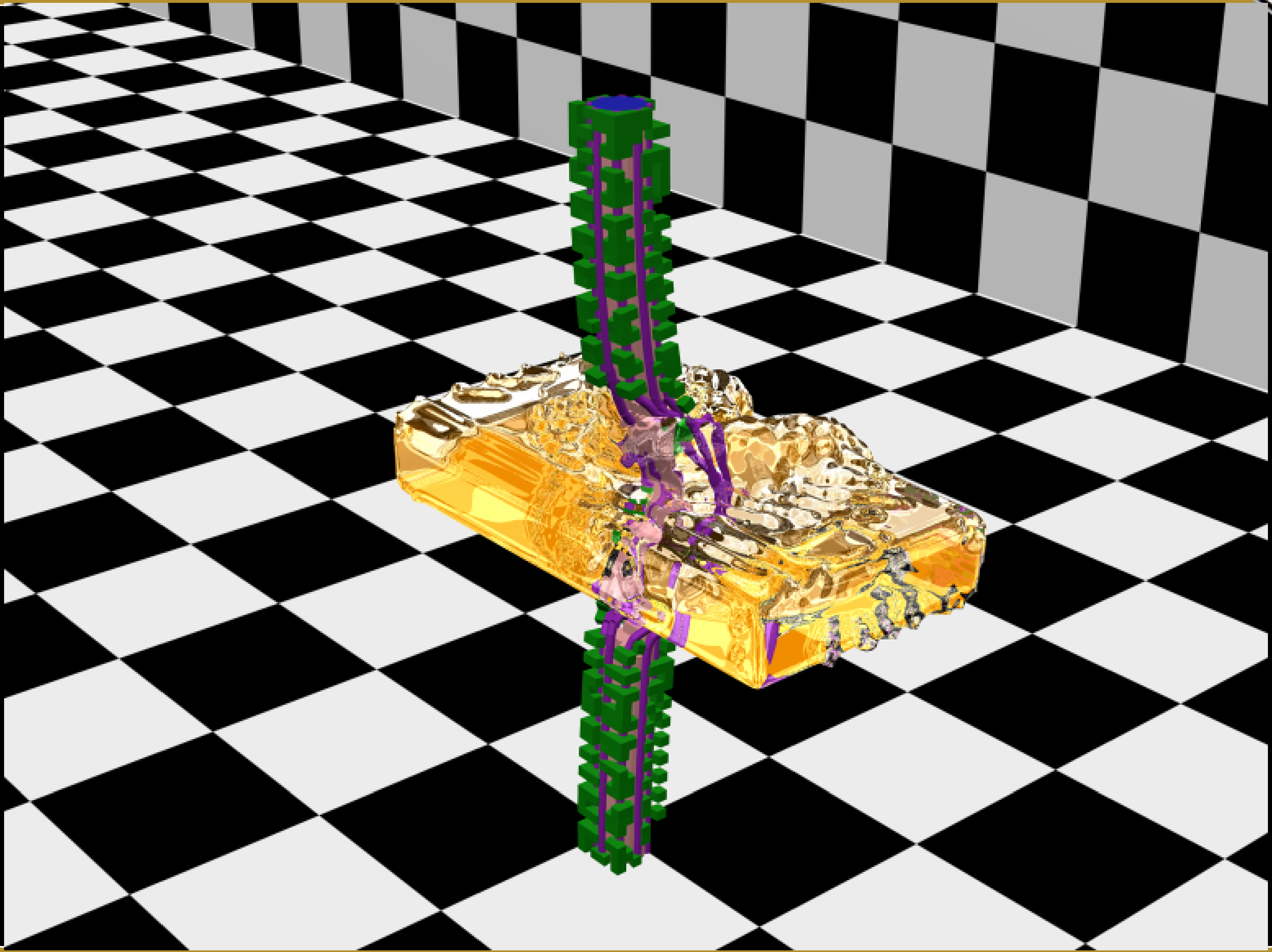
# Liquid visualization (raytracing)



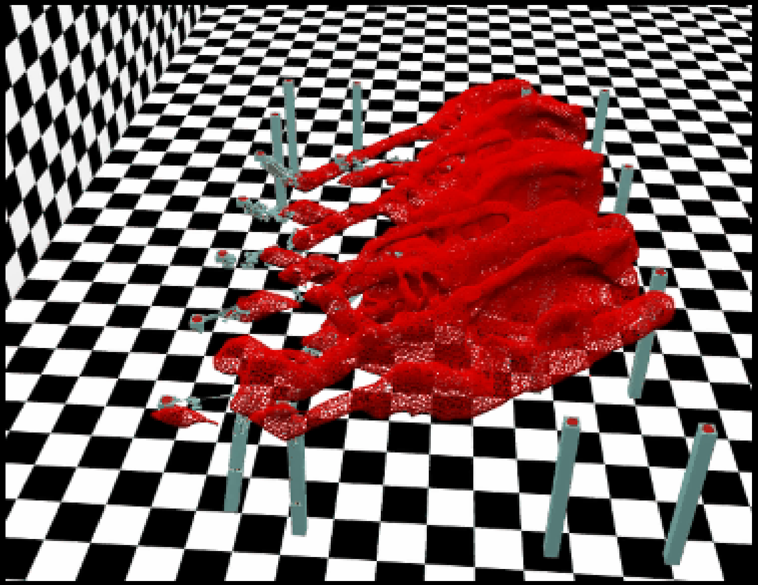
# Liquid visualization (raytracing)



# Liquid visualization (raytracing)

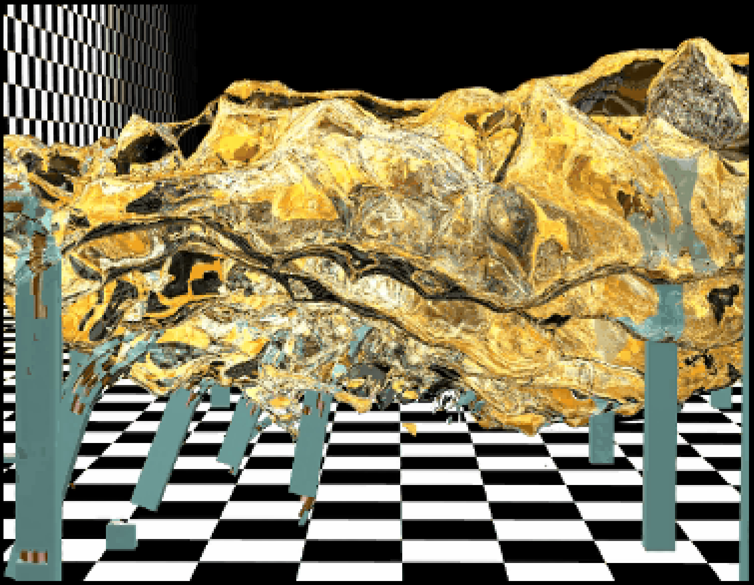


# Liquid visualization (wireframe)



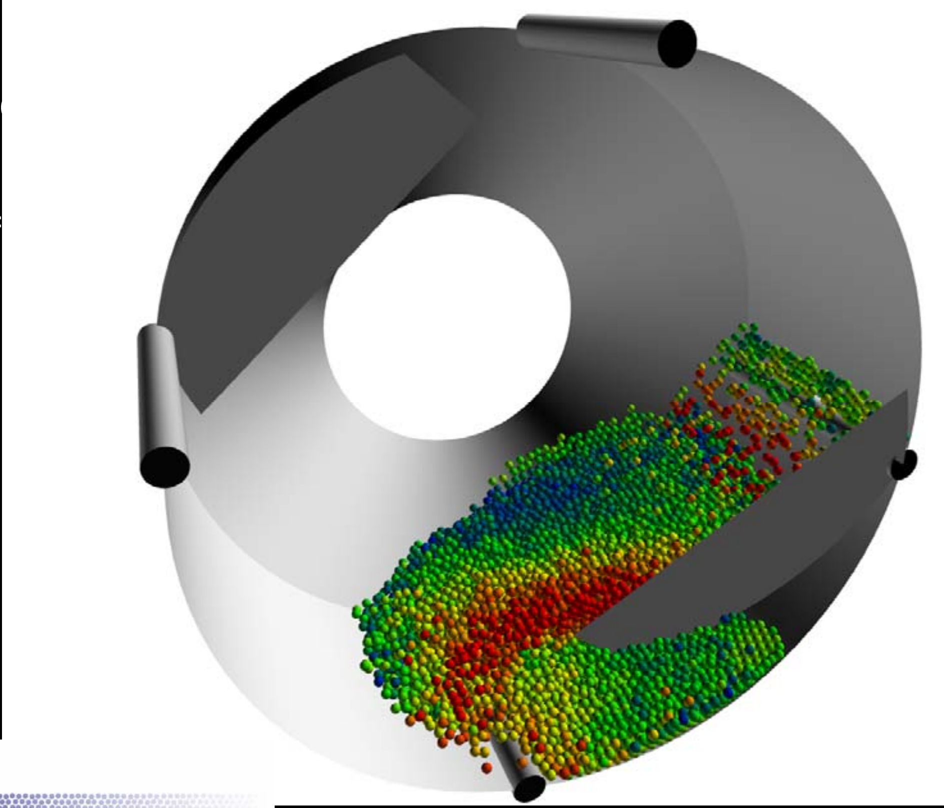
# Liquid visualization (raytracing)

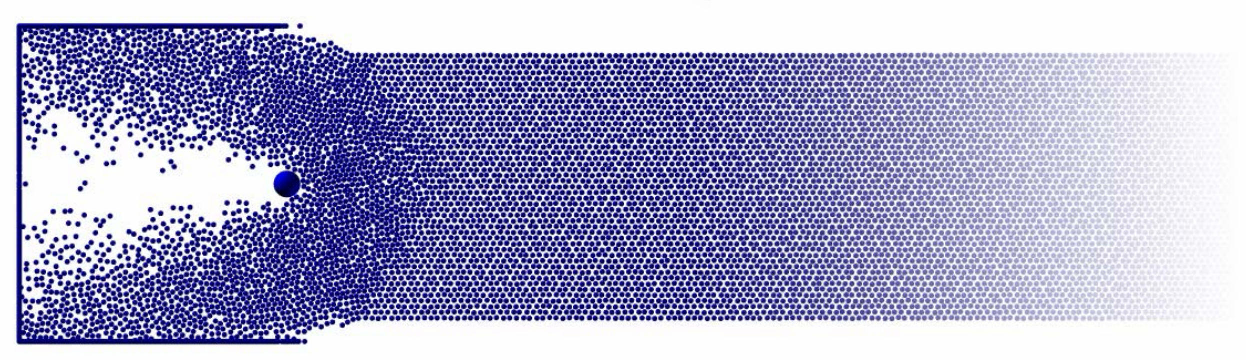
20 hours per frame



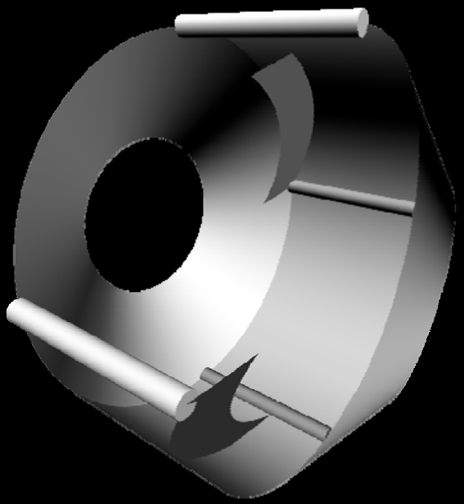
# Visualization of DEM Simulations

- The "Discrete Element Method," or DEM, is a numerical modeling technique for simulating collections of discrete objects.
- Computationally intensive: number of particles in a typical serial DEM simulation is on the order $10^4$,sometimes $10^5$.
- DEM is well suited for studying the behavior of *granular materials:* collections of discrete macroscopic particles. Examples:
    - Sand
    - Powders
    - Avalanching rocks
    - Ice flows
    - Tablets!

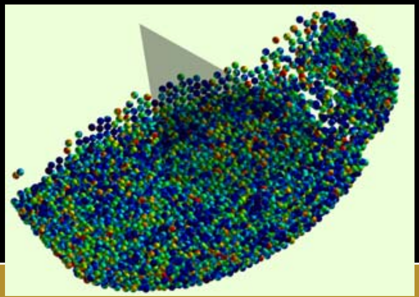


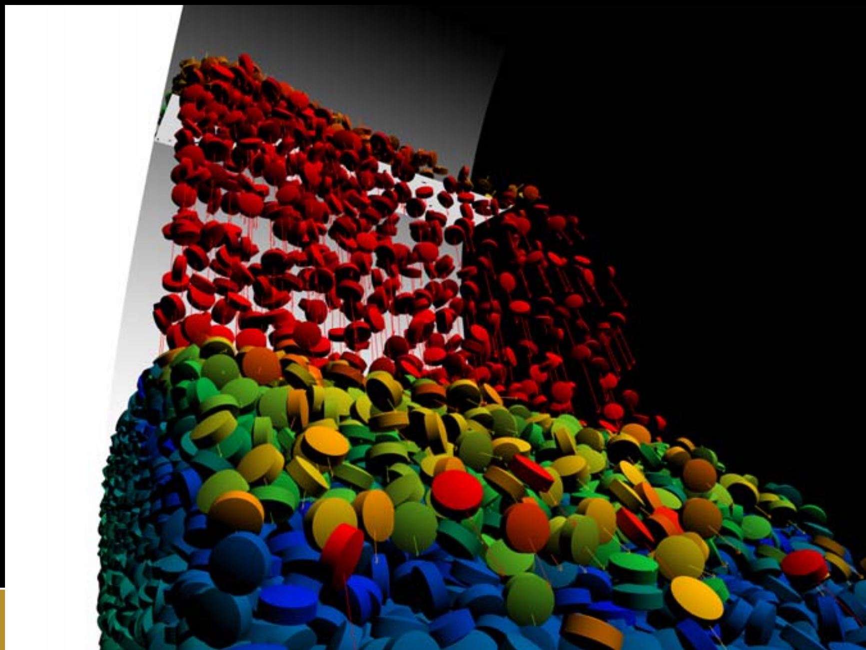


# Pharmaceutical Application: Tablet Coating Project



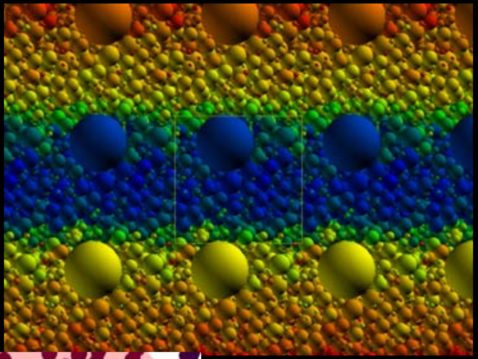
- Pan coating: a pharmaceutical manufacturing operation
- Apply a spray coat to tablets
  - For aesthetic purposes
  - To control release
  - To add a second active ingredient within the coating
- Process parameters are poorly understood
- Critical phase in drug manufacturing
- By modeling the pan coater and spray we hope to better understand and optimize the process

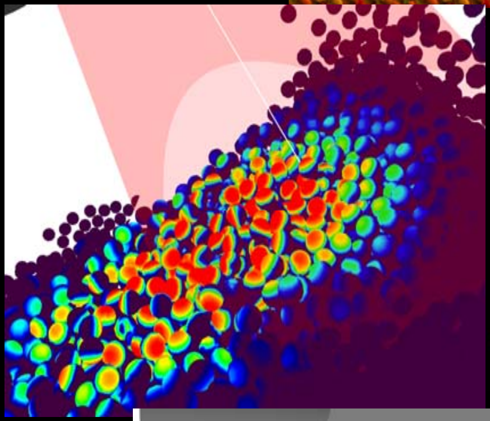


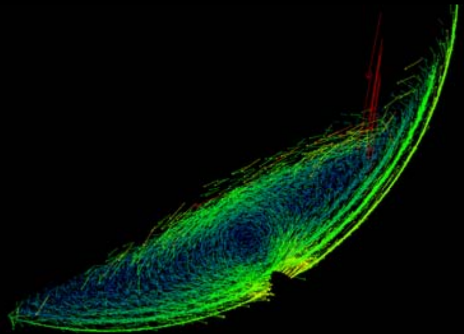


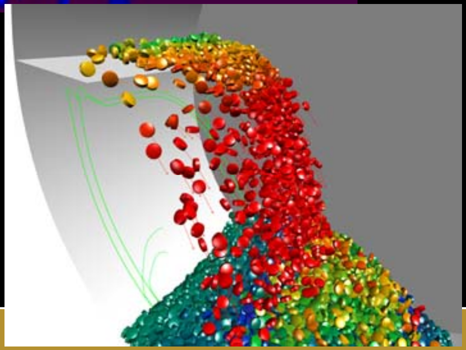
# Visualization Package

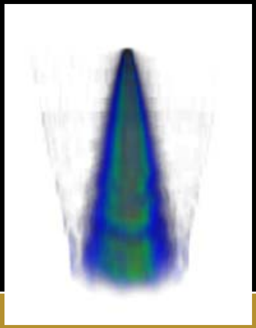
- “ParticleVis” application
- Visualization of particle simulations (such as DEM)
- C++, uses OpenGL for rendering
- Real-time exploration and analysis of large sets of DEM data
- Several techniques used to render all associated data: particle trajectories, velocity fields, even spray density











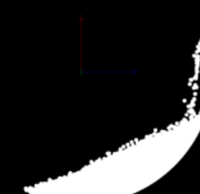
# Visual Analysis Examples

Generating images from the data to compare against real-world results:

Filtered Video Footage

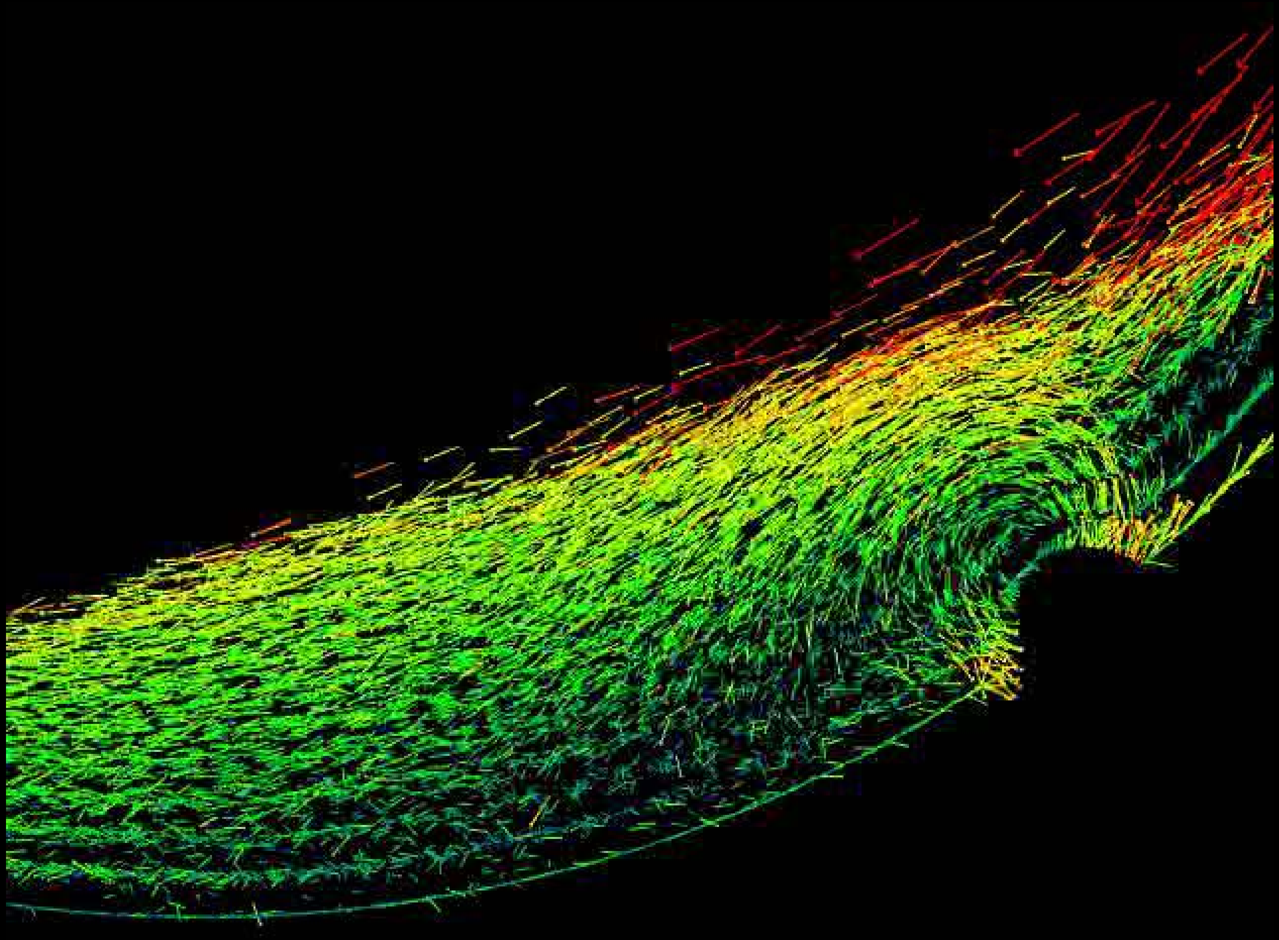
Visualized Simulation Data

Goal: Fast and Intuitive Validation



# Visual Analysis Examples

Visualization of intrapan flow patterns:



# WTC North Tower on 9/11


Faculty:

*Christoph M. Hoffmann, CS*
*Ayhan Irfanoglu, CE*
*Voicu Popescu, CS*
*Mete Sozen, CE*

Students:

*Oscar Ardila-Giraldo, CE*
*Ingo Brachmann, CE*
*Paul Rosen, CS*

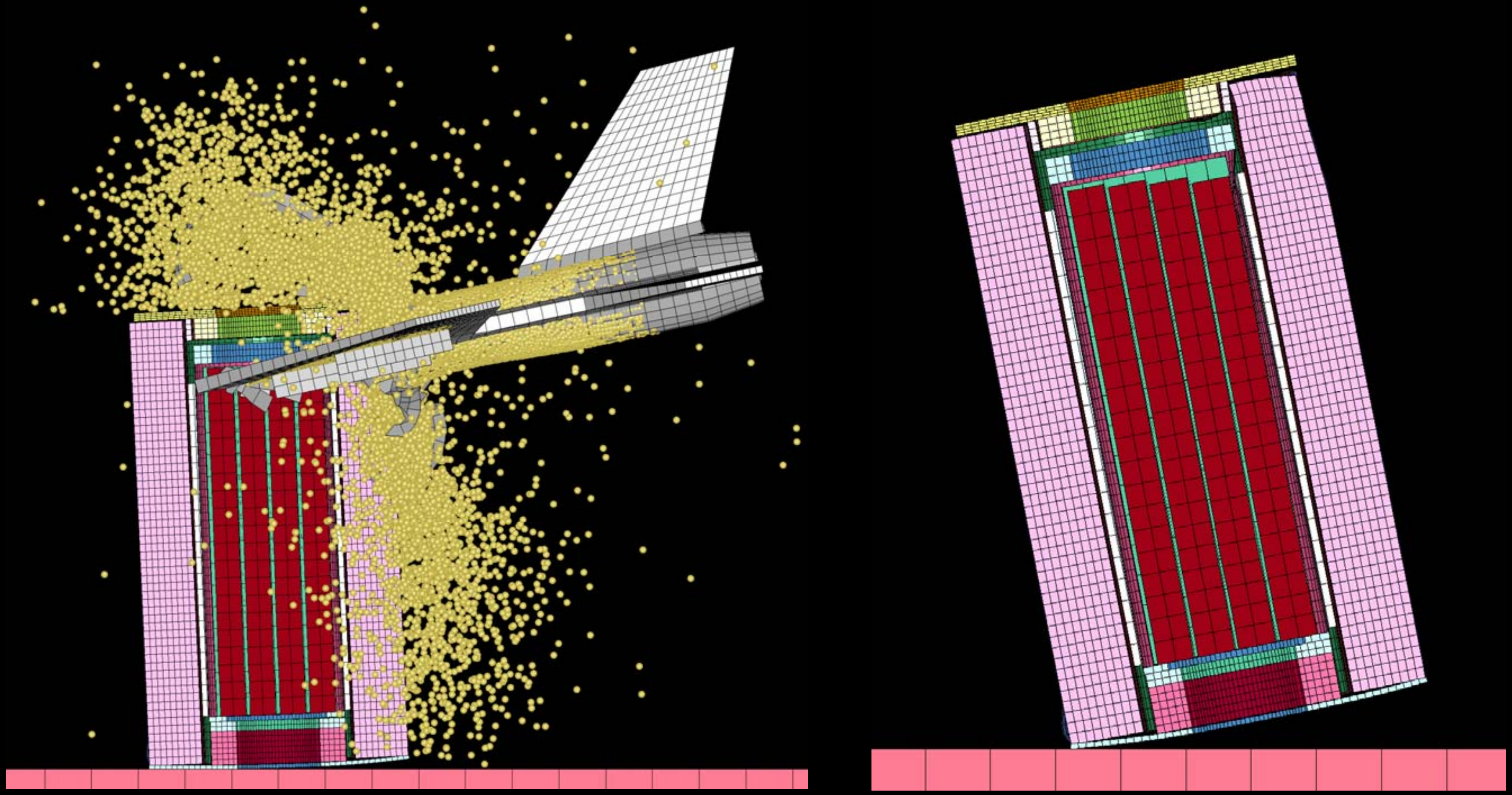
# Objective

- Simulate the crash of AA-11, a Boeing 727-200ER aircraft, into WTC-1
- Learn what plausible damage there could have been to the building, and whether the crash alone could have been survived

# Approach and Time Line

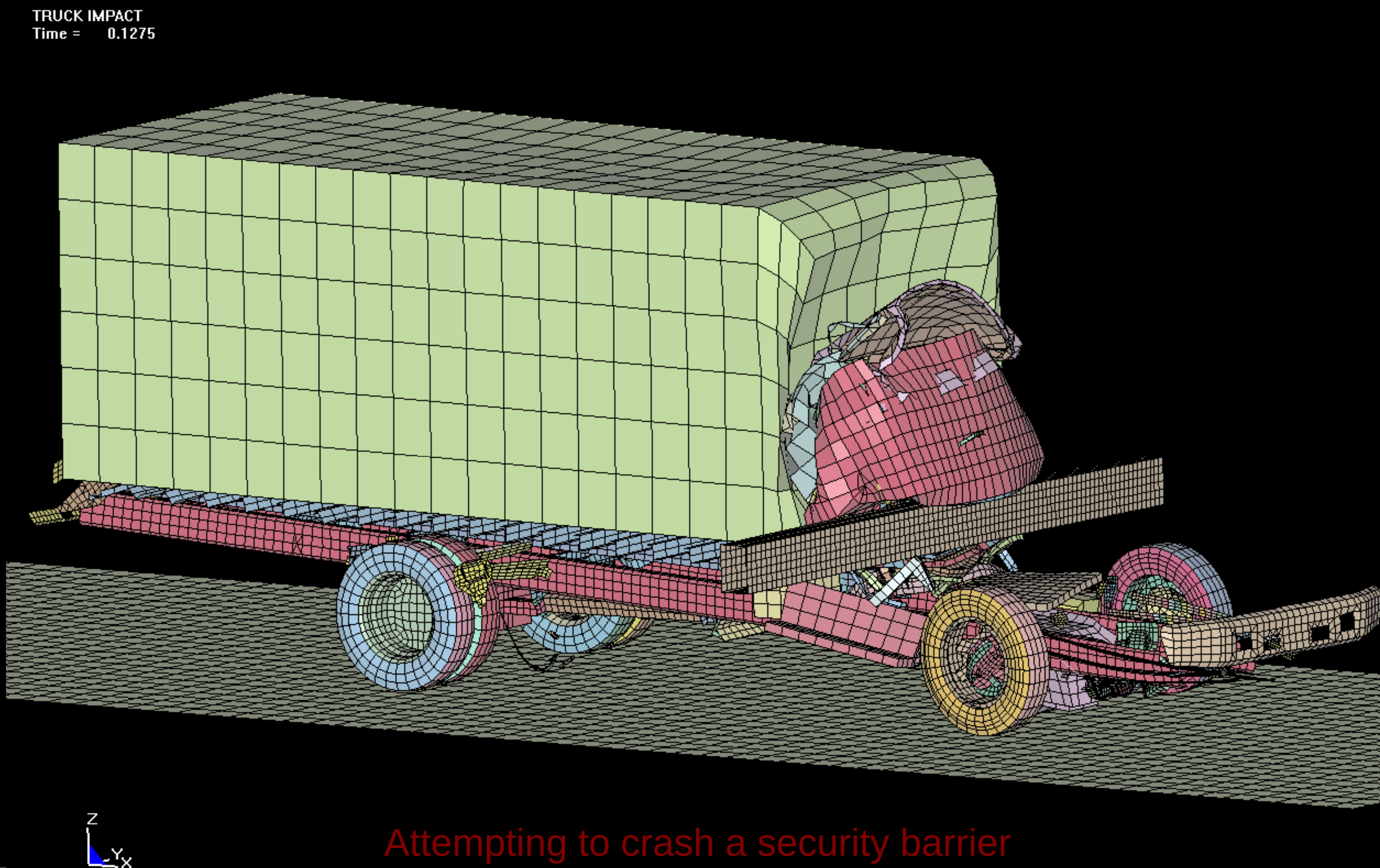
- Using LS-Dyna for simulations:
  - Construct a finite element model of the building support structure (20 months)
  - Construct a finite element model of the aircraft (8 months)
  - Calibrate the models using accepted engineering techniques and physical experiments (6 months)
  - Simulate the impact (1 month and ongoing)

# LS-Dyna Capabilities, 1



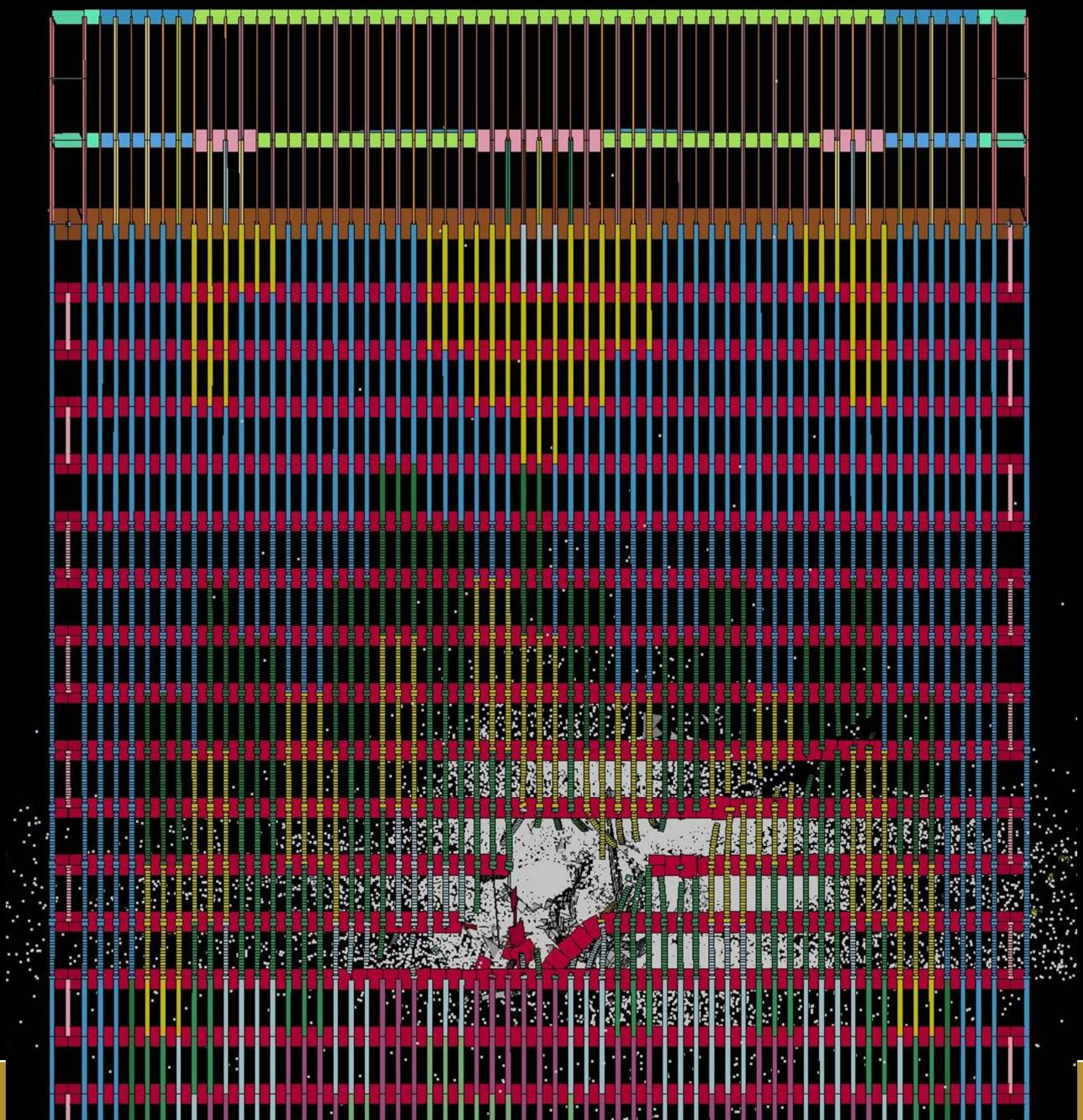
Spent nuclear fuel storage accident

# LS-Dyna Capabilities, 2

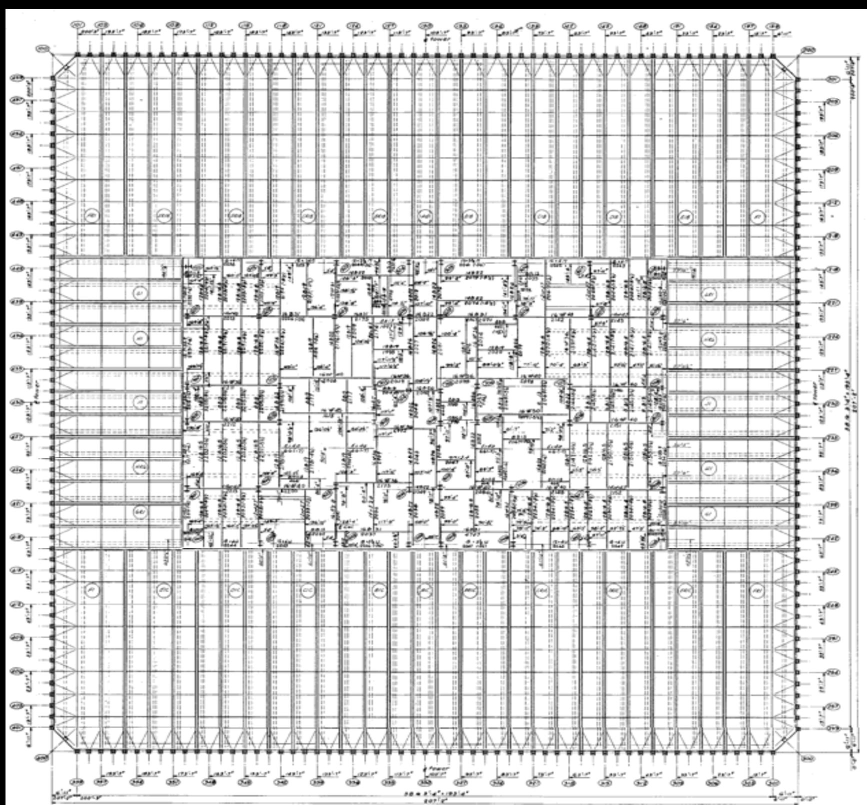


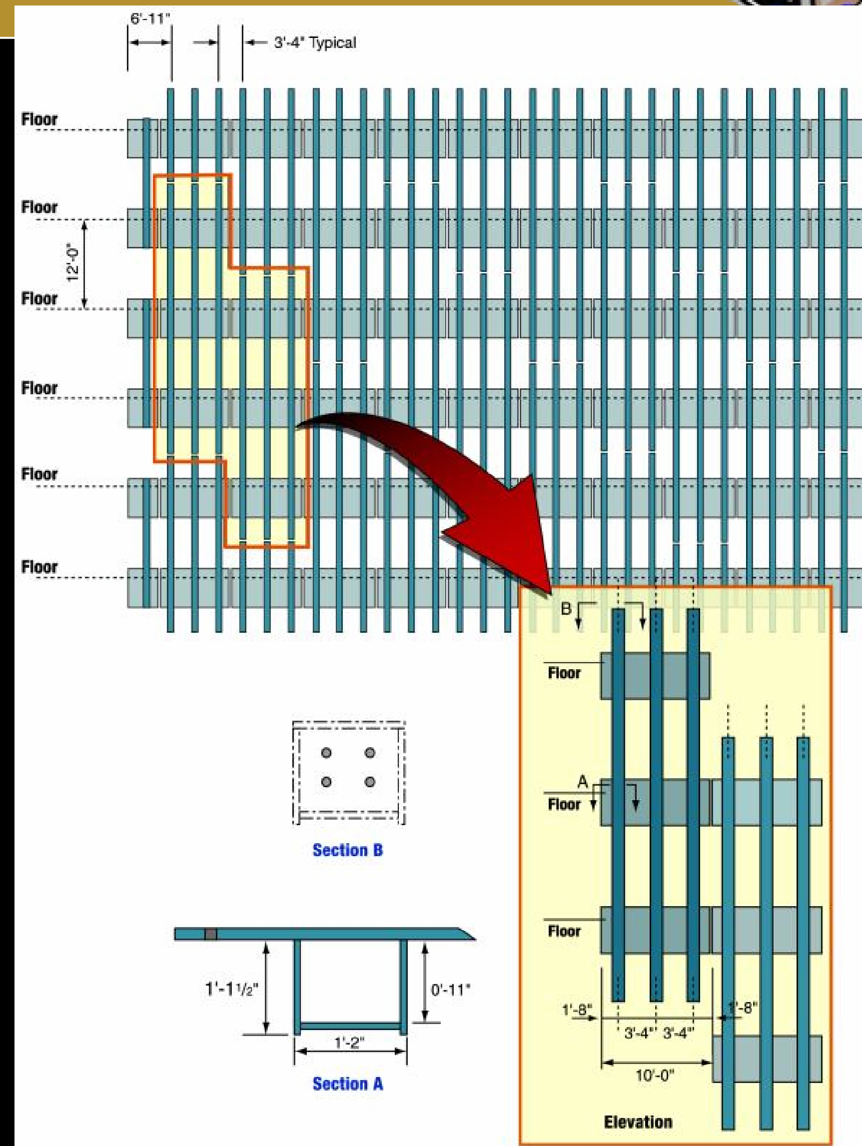

Attempting to crash a security barrier

# LS-Dyna Capabilities, 3



# WTC-1 Structure





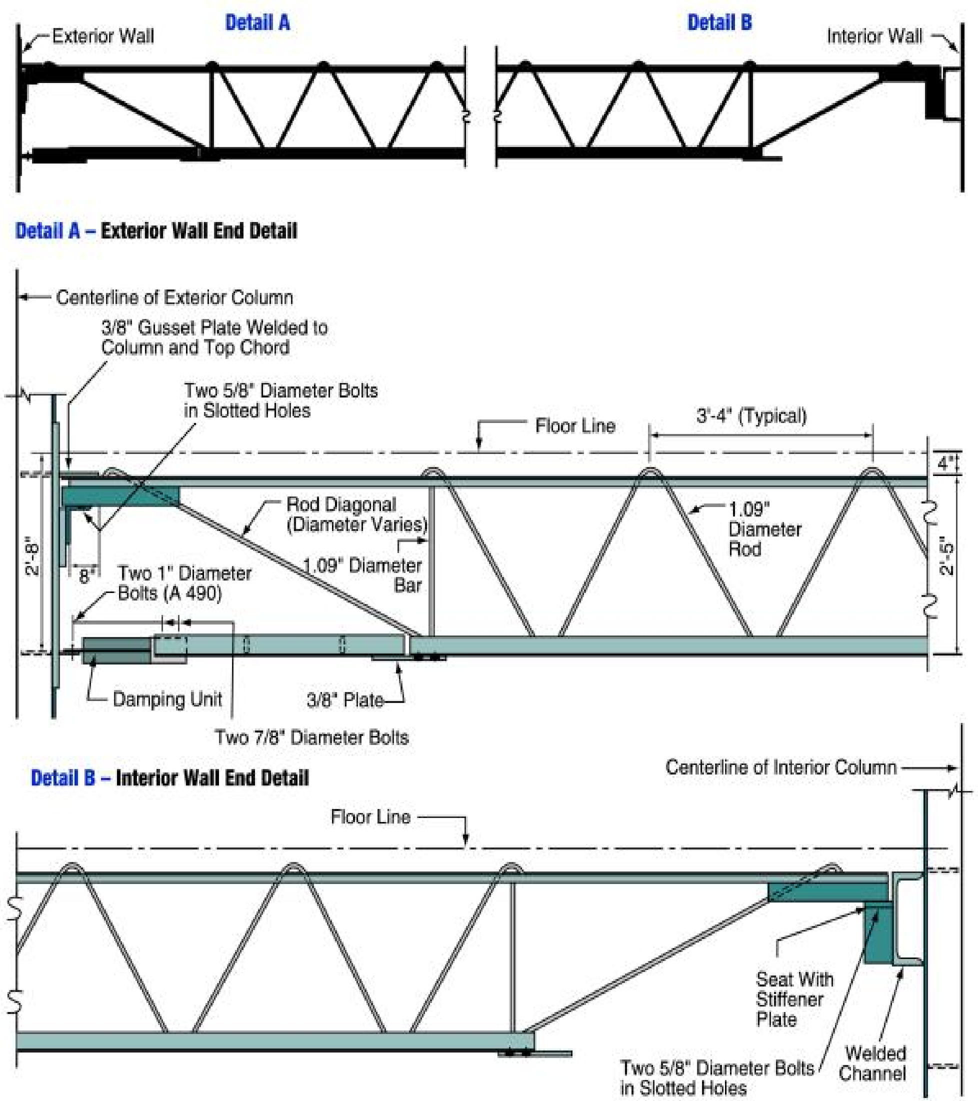
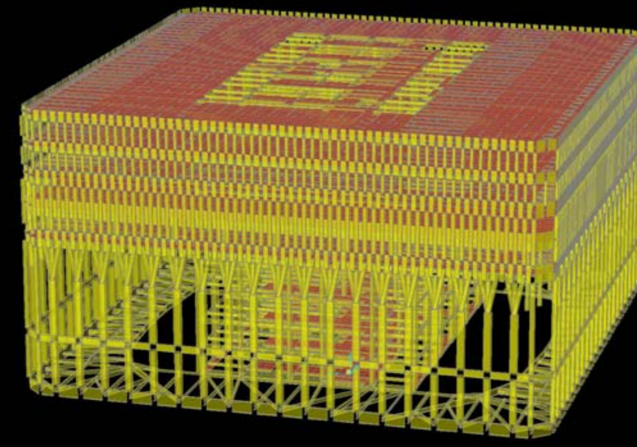
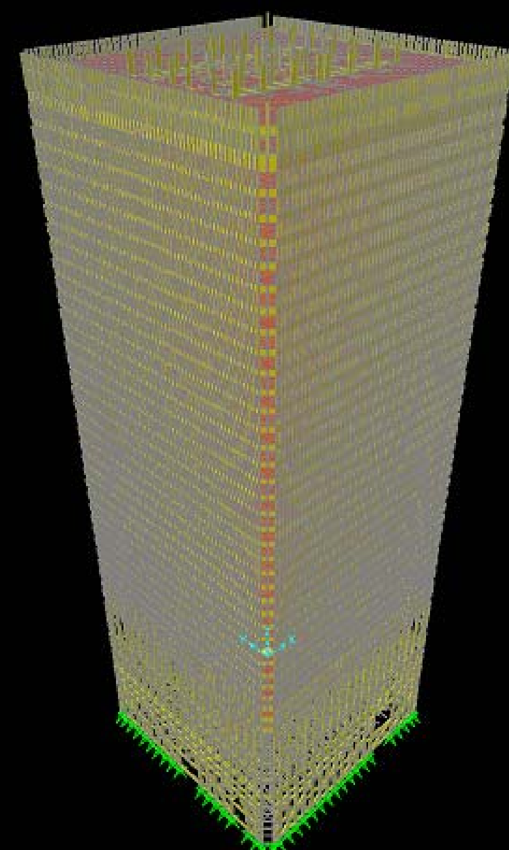



*Figure 2-6 Floor truss member with detail of end connections.*





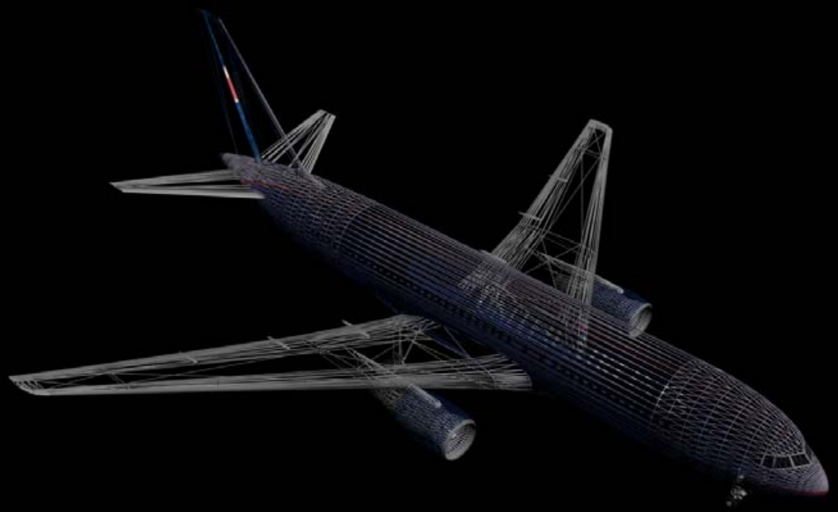
Number of Nodes: 640,000
Number of Beam Elements: 530,000
Number of Shell Elements: 360,000

# Finite Element Aircraft Model

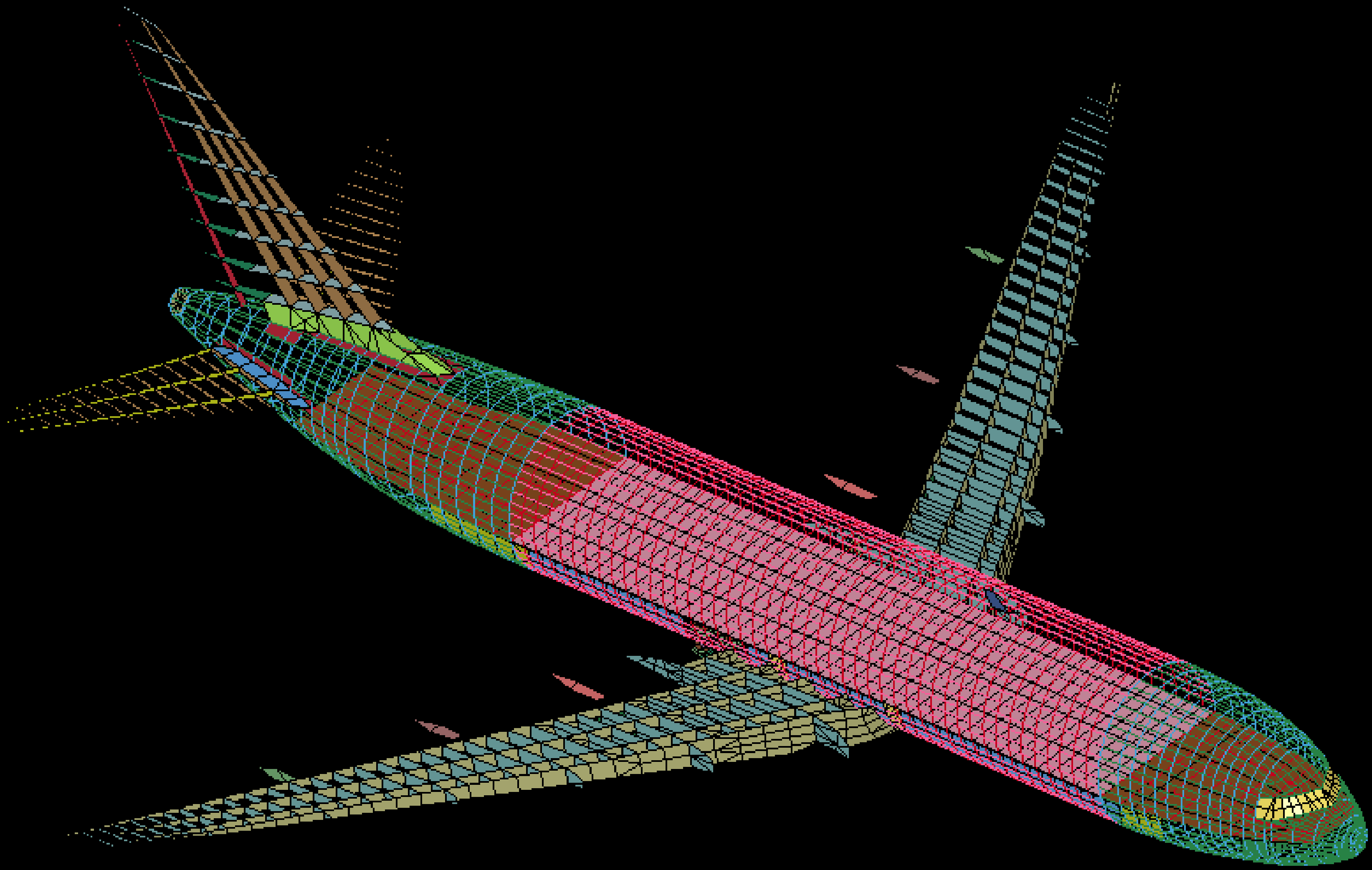
- Problem:
  - Much of the structural and dimensional data is proprietary
- Our Solution:
  - Get the overall dimensions from public sources
  - Use a graphics model as starting point
  - Add structural detail according to public drawings
  - Consult libraries, experts, internet sources

# Aircraft Model, 1

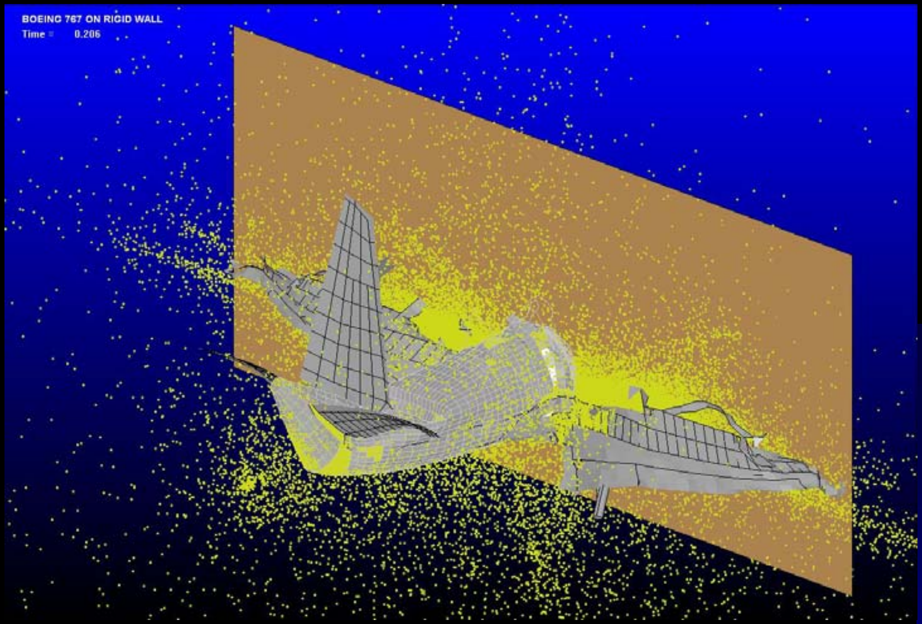

*Exterior:*
graphically pretty
but unsuitable for FEA

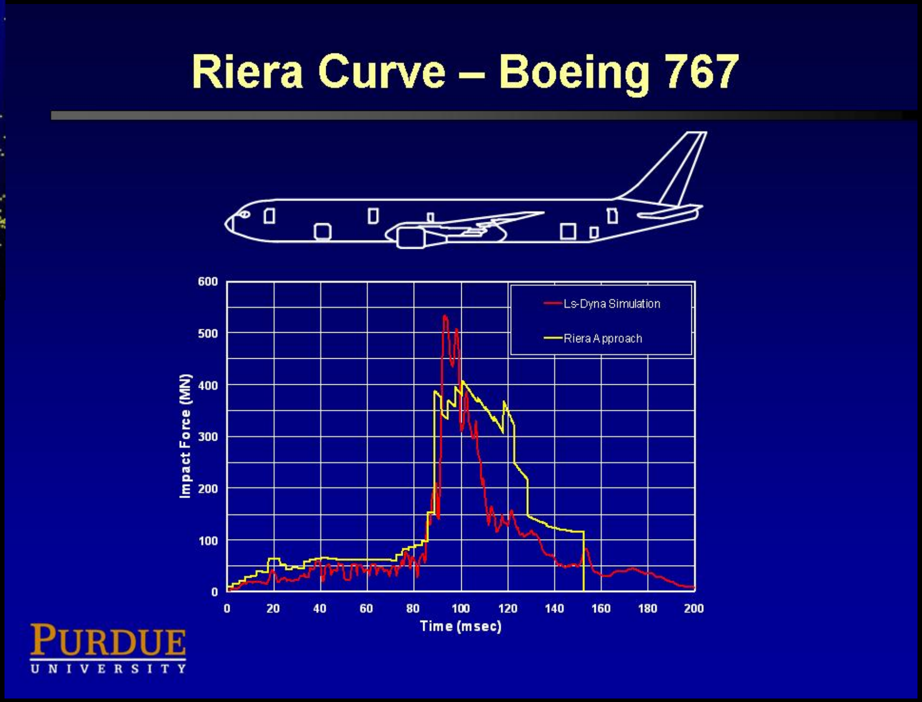


# Aircraft Model, 2

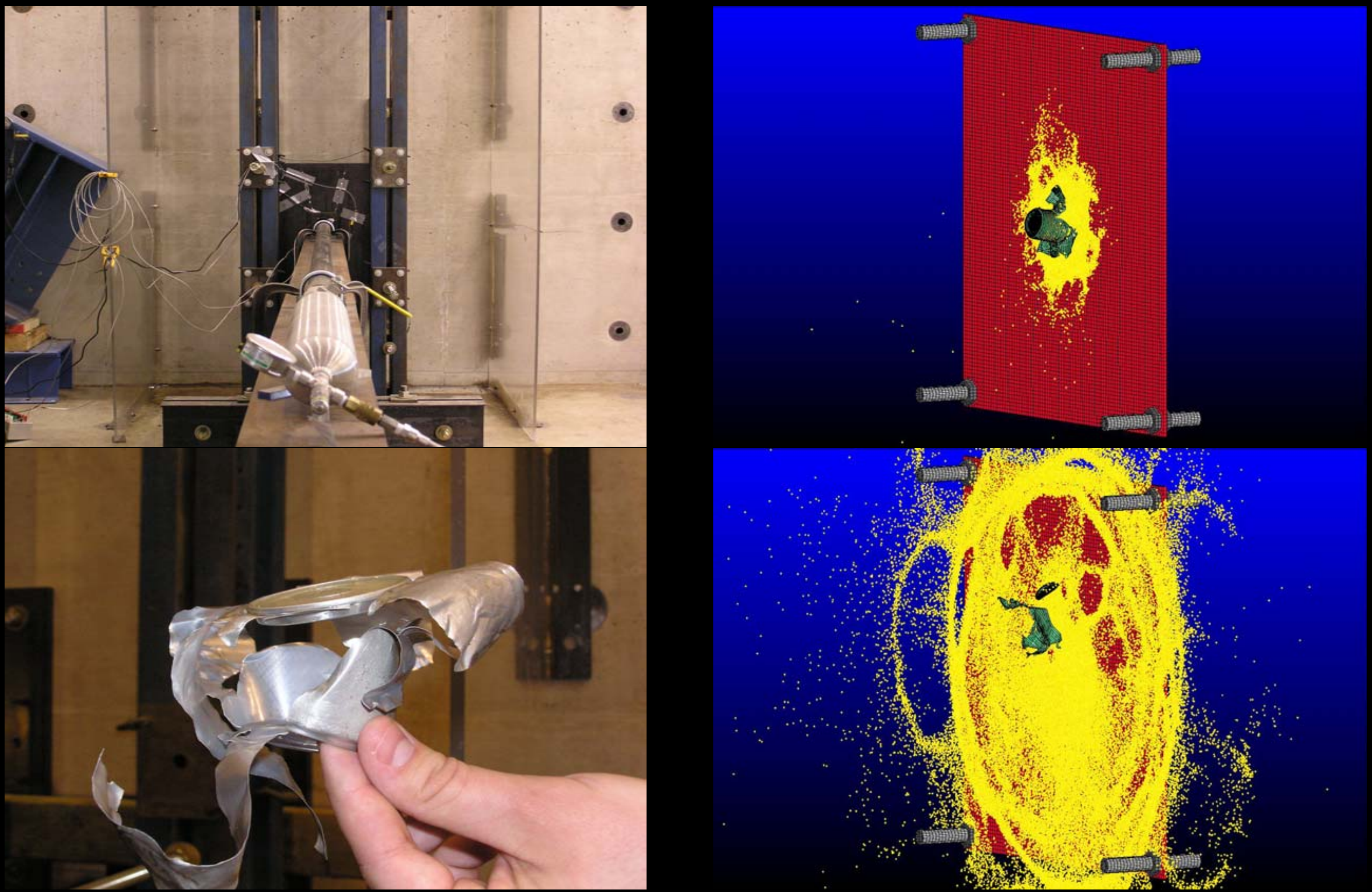


# Riera Calibration

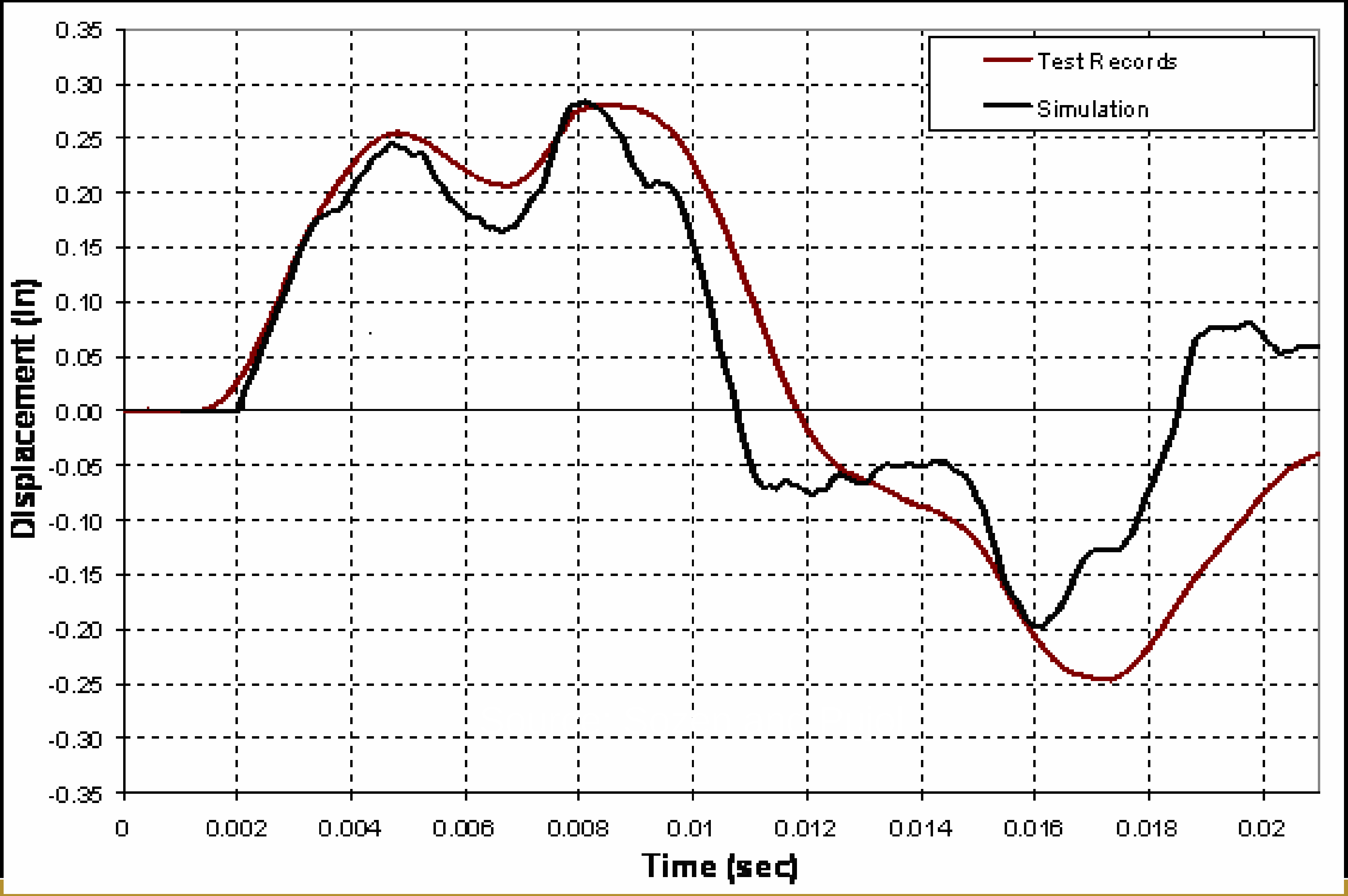


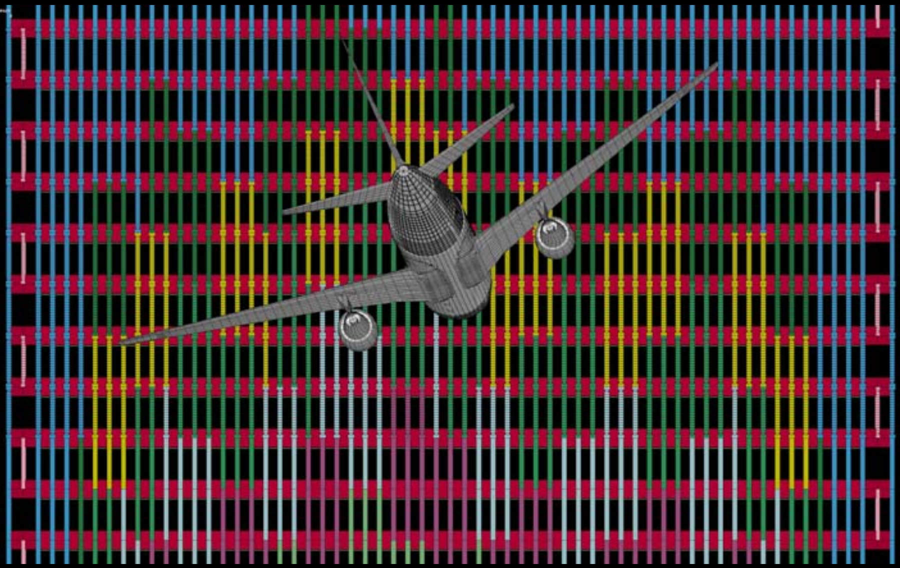
# Fluid/Structure Interaction

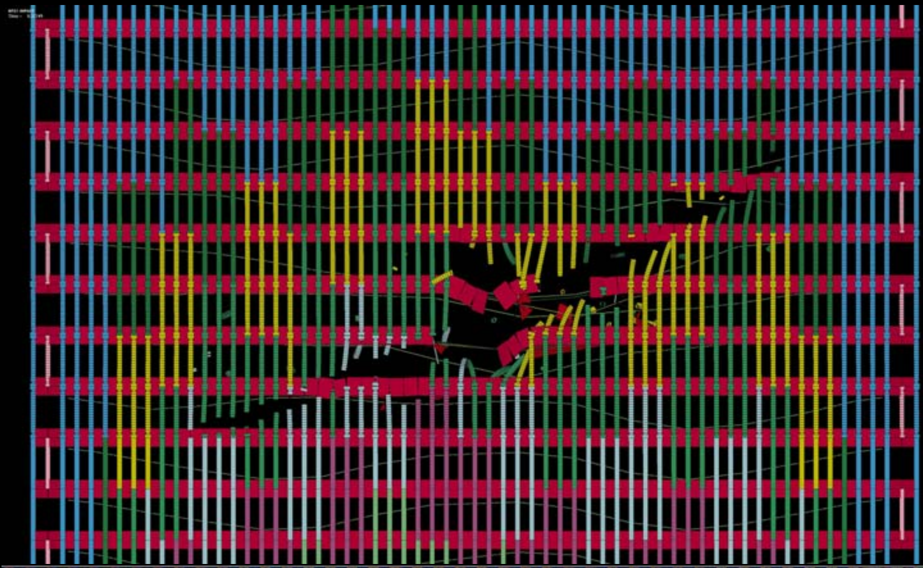


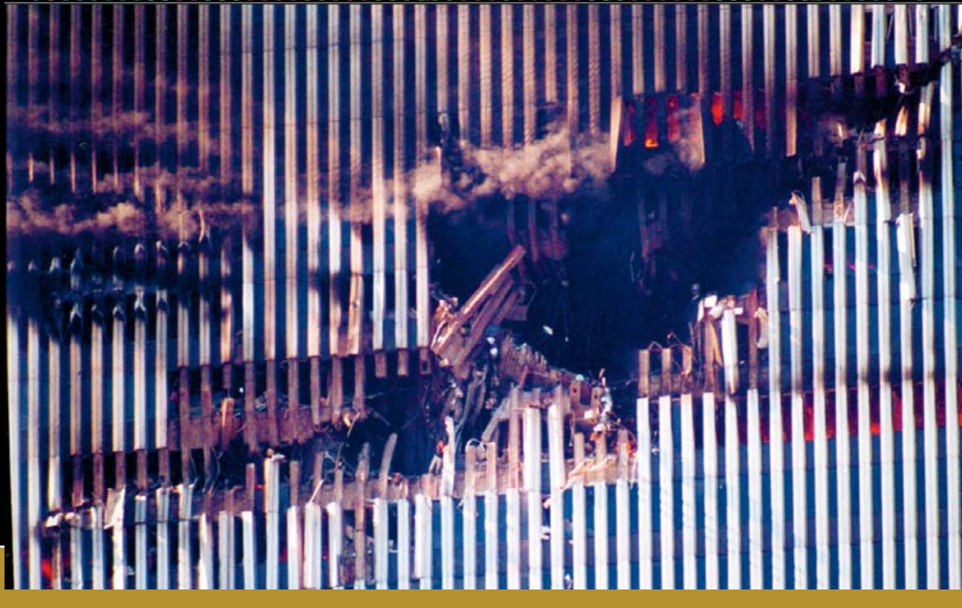
# Deflection Results



# Simulation Images

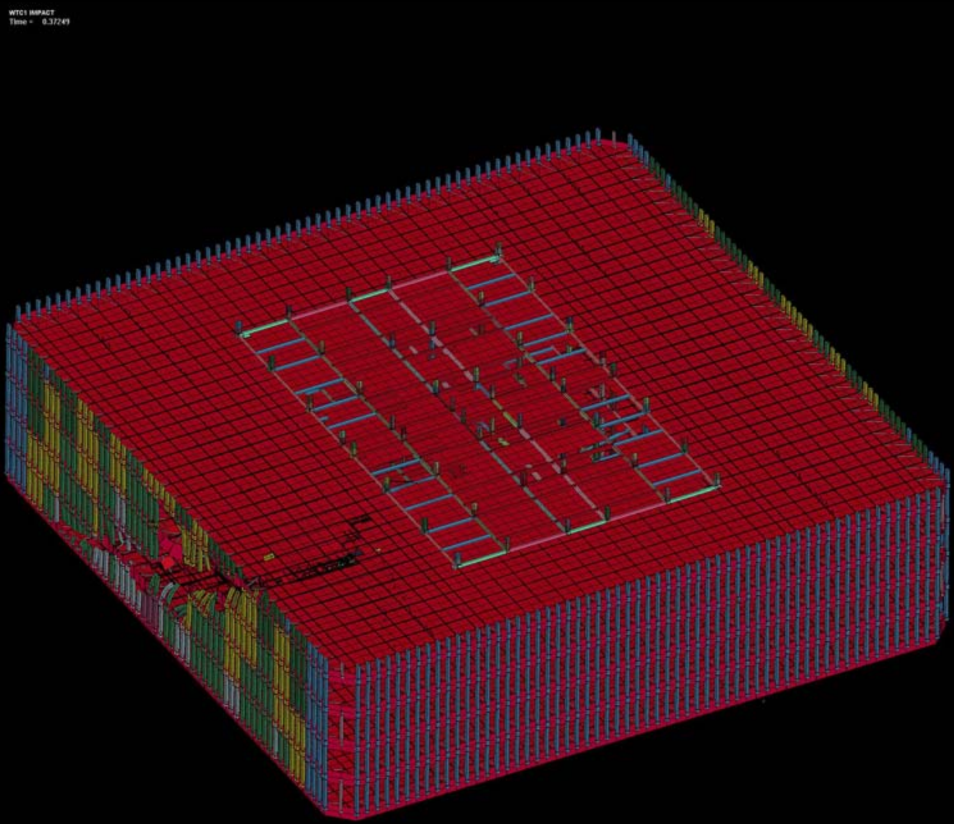


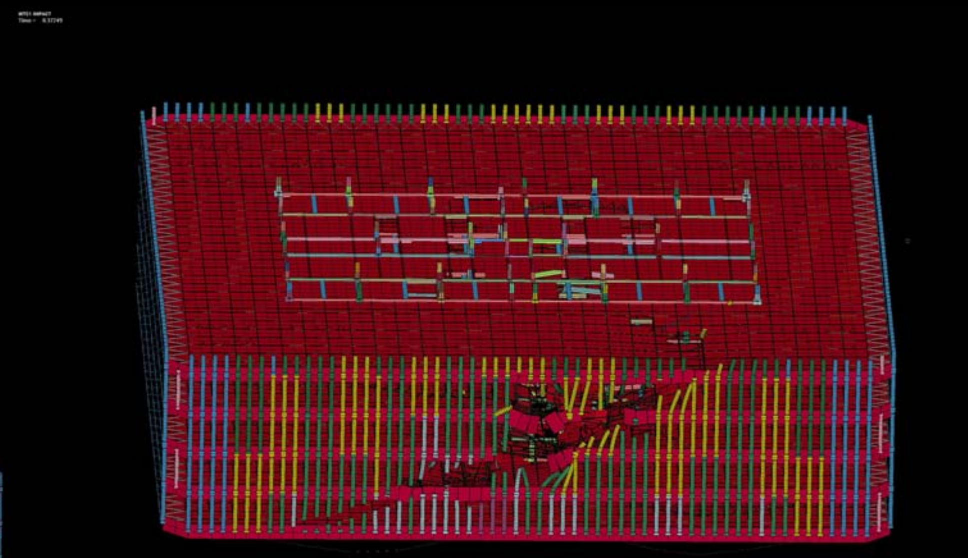




Before impact
After Impact; simulation
After impact; actual event

# Affected Floors



View of floors 94 – 97
0.37 sec after impact

# Effective Distance Learning

## through Sustained Interactivity and Visual Realism

*Voicu Popescu, Cristina-Nita Rotaru, Chun Jia, Radu Dondera, Melissa Dark, Carlos Morales, Gary Bertoline, Laura Arns*

# Motivation

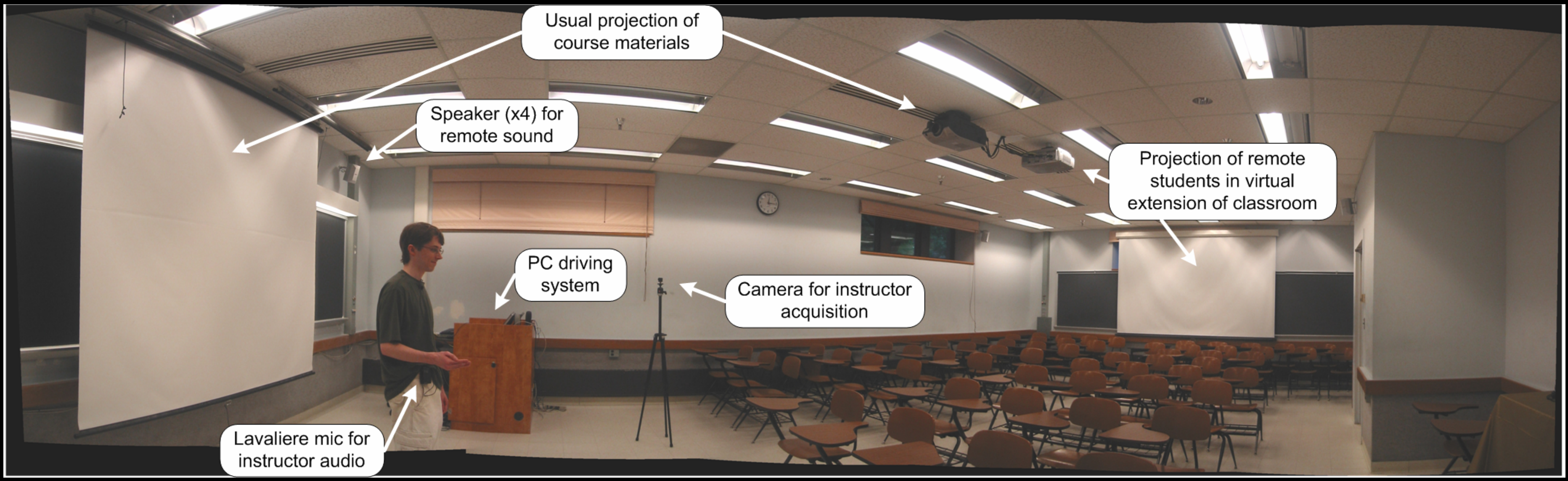
- Over 50% of US higher education institutions offer distance learning services
- Present distance learning systems are ineffective
  - Remote students are isolated
  - Remote students do not have access to many proven on-campus learning activities such as study groups and office hours
- Present distance learning systems are expensive
  - Parallel activity on campuses: special hardware, special courses, special training for instructors

# Approach

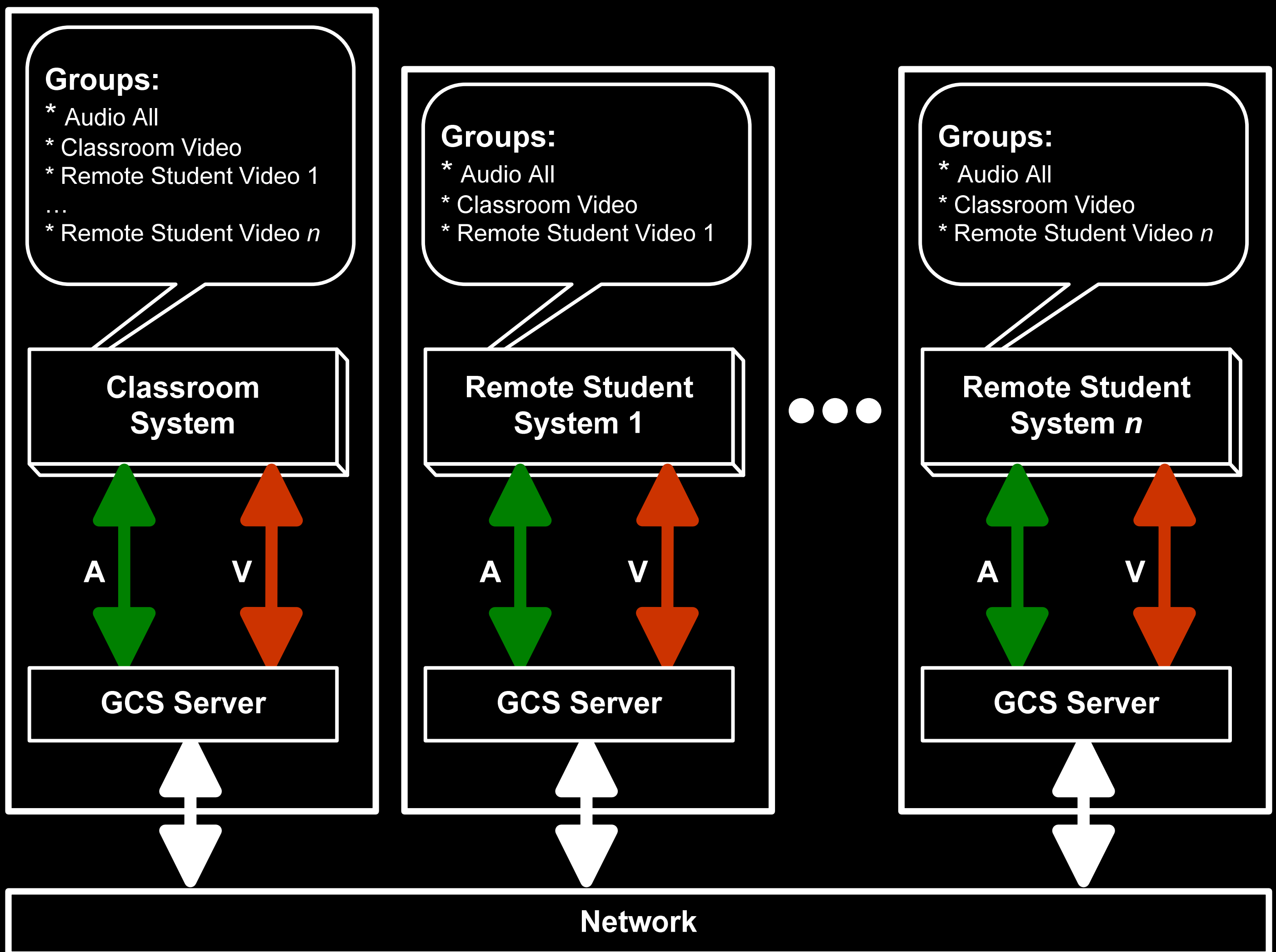
- Integrate distance learning with conventional on-campus learning
  - Extend classroom to accommodate remote students
  - Rely on commodity networking, multimedia, graphics, and computing components
  - Provide support for all learning activities (e.g. office hours, study groups)

# Distance learning system prototype

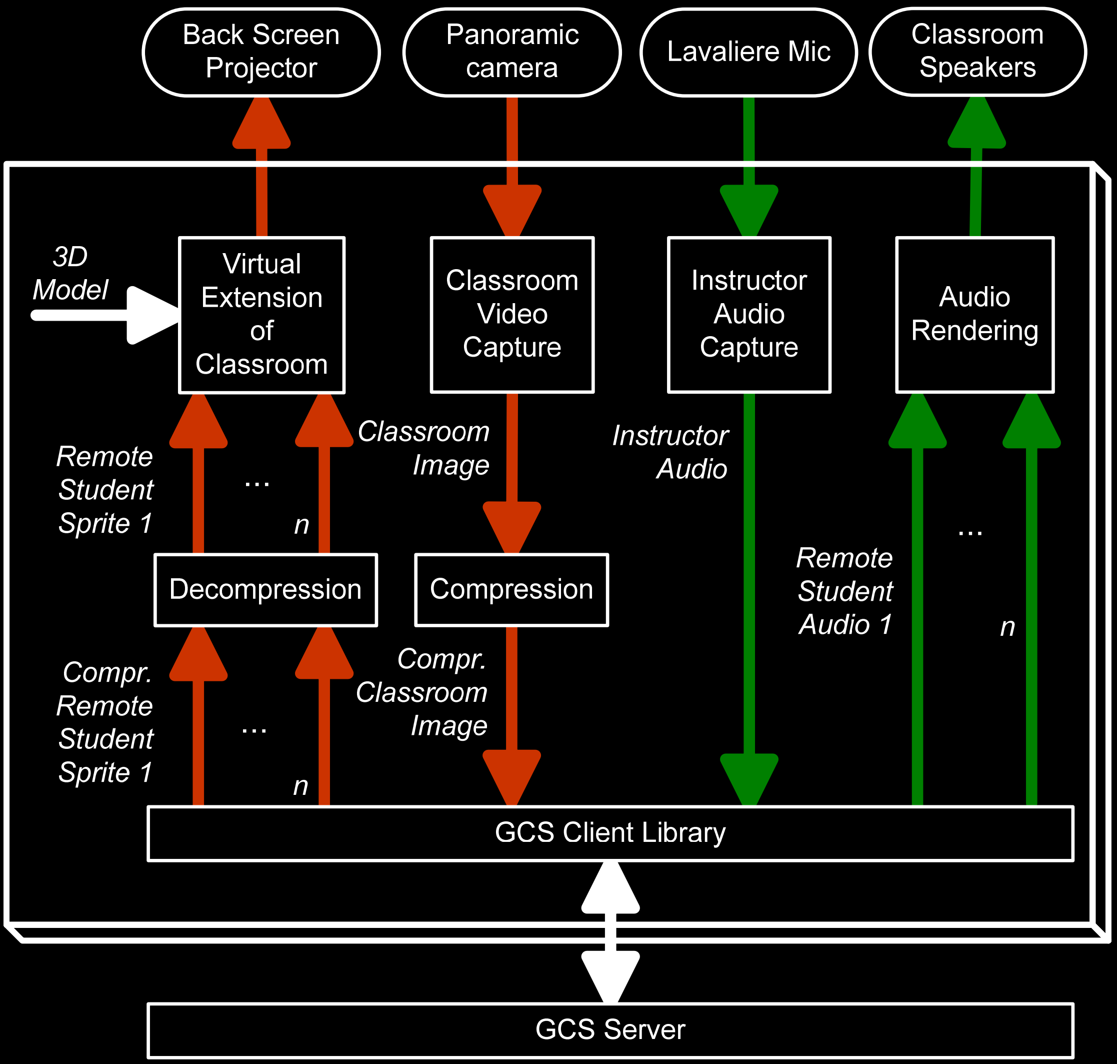
# Minimal hardware requirements



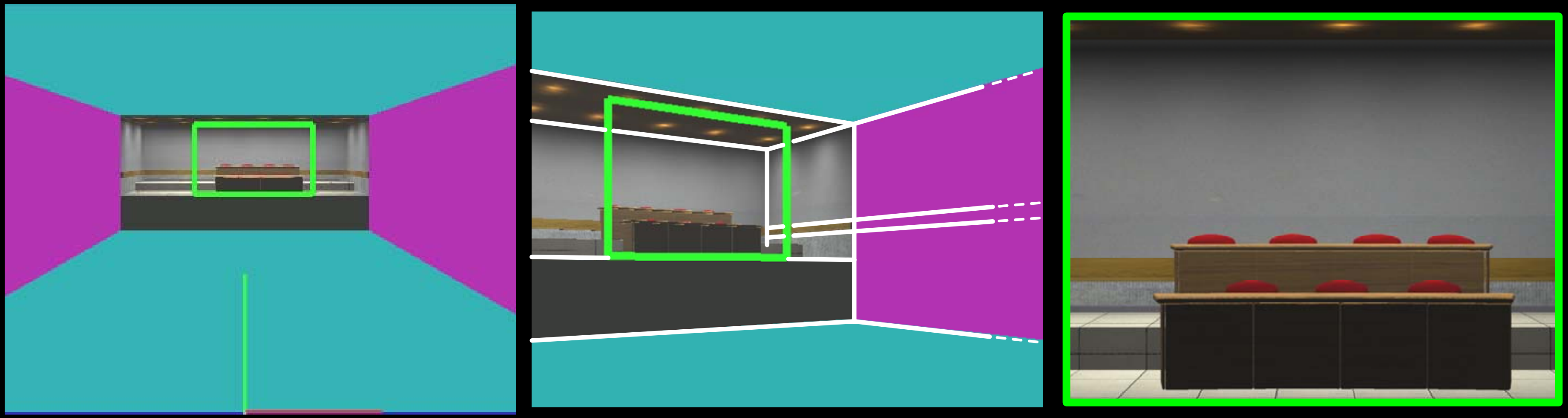
# System architecture



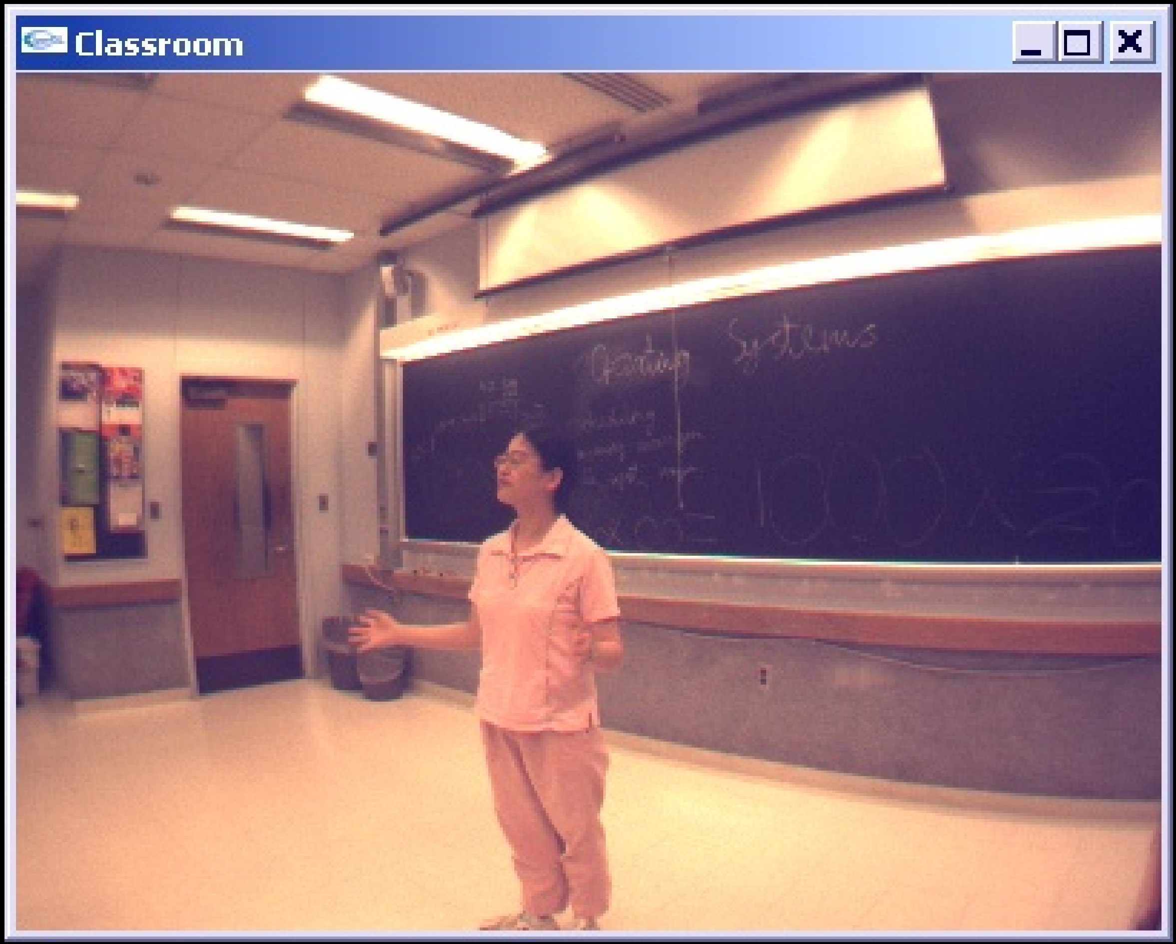
# Classroom system



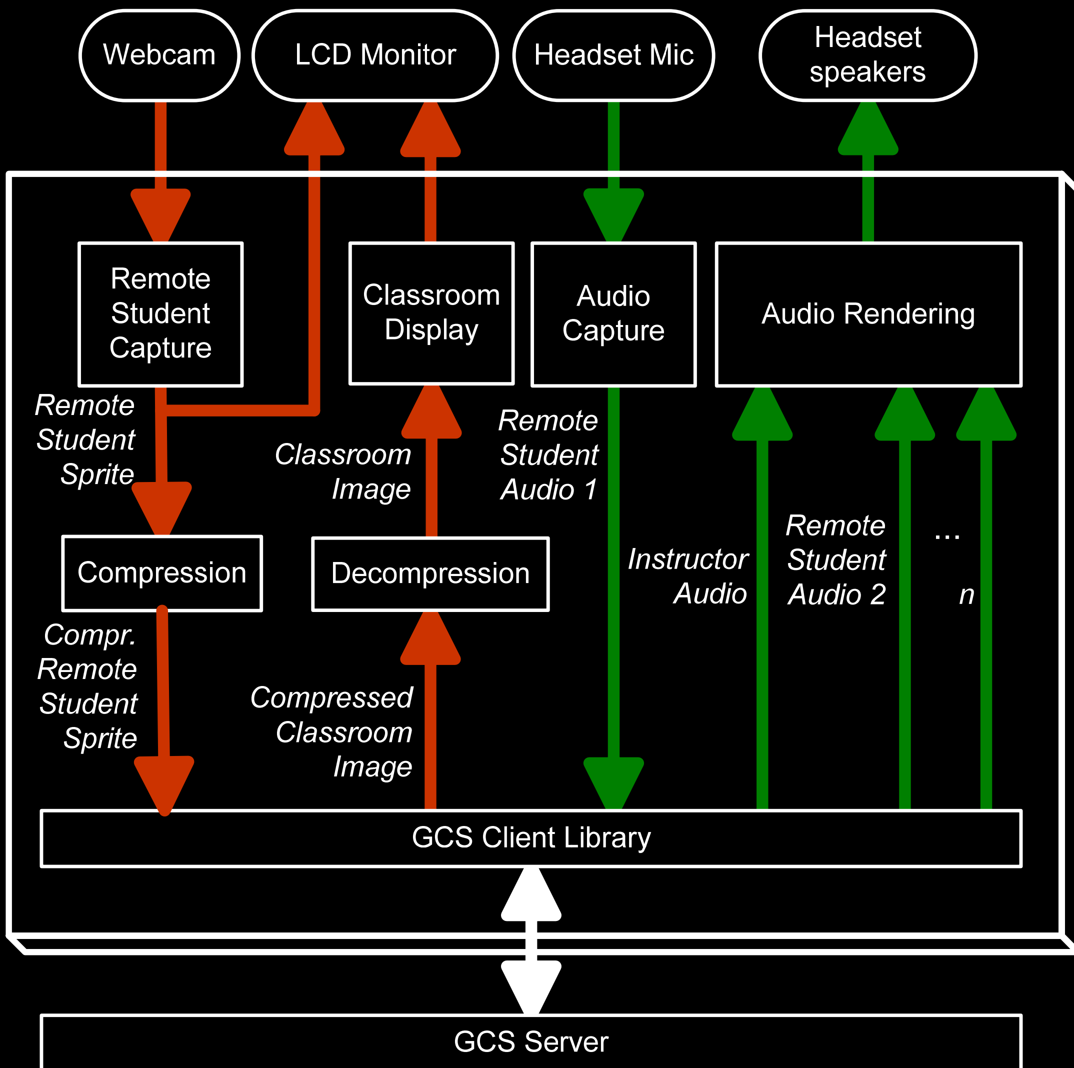
# Virtual extension of classroom



# Remote student view of classroom



# Remote student system



# Real-time background subtraction

# Future

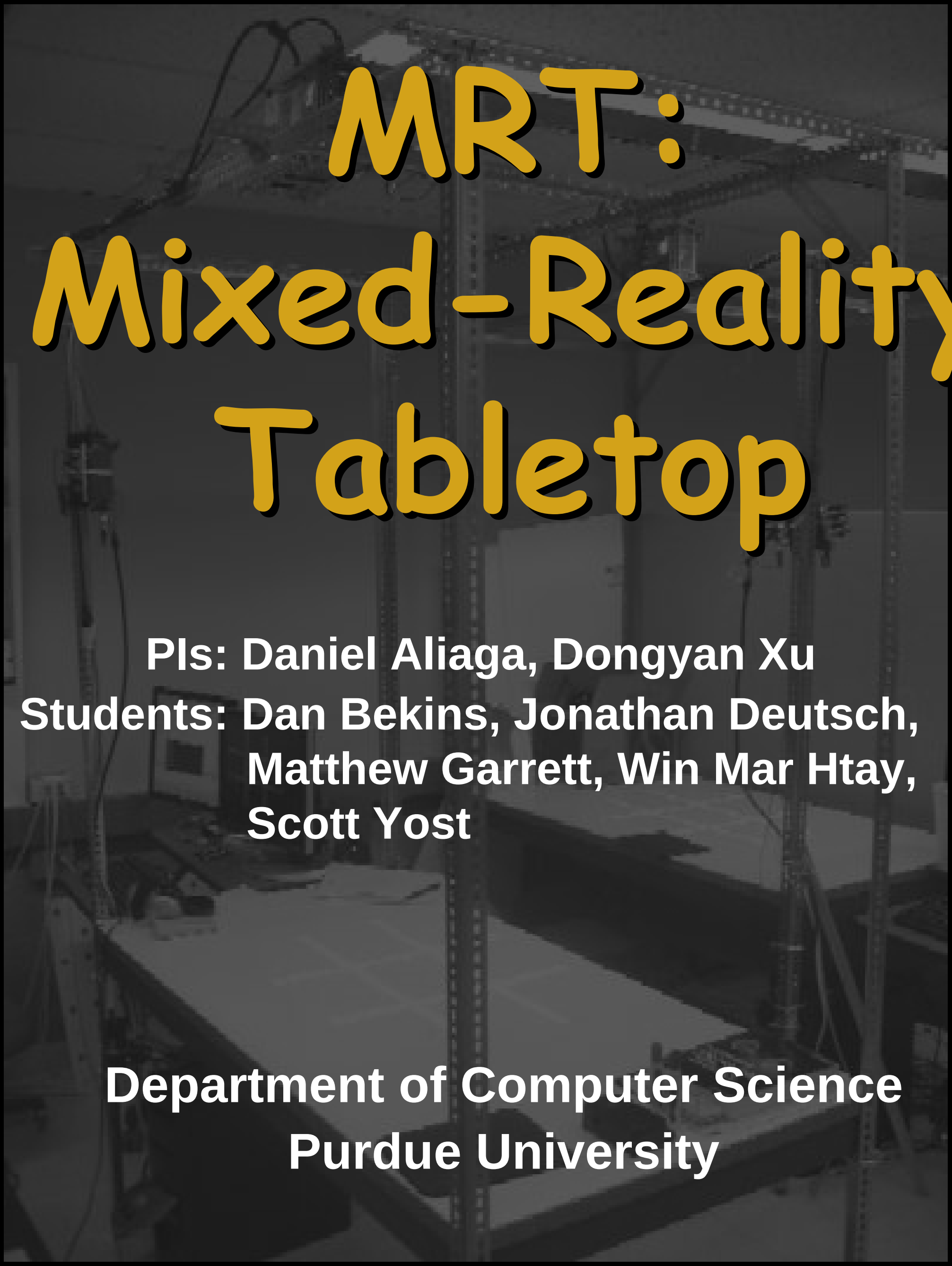
- Add instructor tracking
- Deploy system on wide geographic area
  - Image-based compression of remote student sprite
- Support study groups and office hours
- Formally asses educational value
  - Pilot studies
  - Actual classes

# MRT: Mixed-Reality Tabletop


PIs: Daniel Aliaga, Dongyan Xu

Students: Dan Bekins, Jonathan Deutsch, Matthew Garrett, Win Mar Htay, Scott Yost

Department of Computer Science
Purdue University

# Motivation

- Immersive learning in Year 2020
  - “There is a power in virtual interaction” – Rita R. Colwell
- Going beyond current-generation whiteboard
  - Provide a natural focus of attention: lab table, desk, counter…
  - Support rich and intuitive interactions among distributed users
- Adding virtual and real objects to the equation
  - Mix real and virtual objects in the same focus of attention
  - Create virtual venue and context for interactions
- Wider deployment than full-fledged VR systems
  - Lower cost
  - Less infrastructural requirement
  - Easier to develop, install, and operate

# Related Work

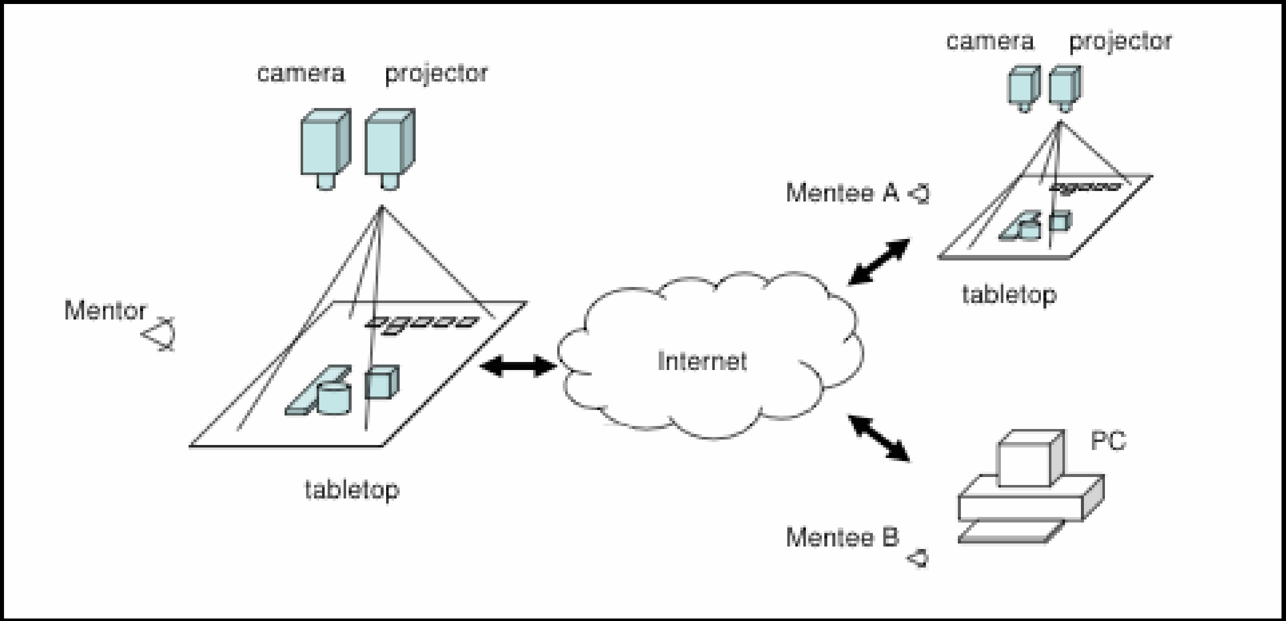
- Whiteboards
- HMD-based VR systems (UNC-CH, Feiner at Columbia)
- The Workbench (Barco, 3rd Tech)
- Tangible user interfaces (MIT, UVA)
- Emancipated Pixels (SIGGRAPH ’99)
- Shader Lamps (Raskar at MERL)
- Everywhere Displays (IBM)

# Goals

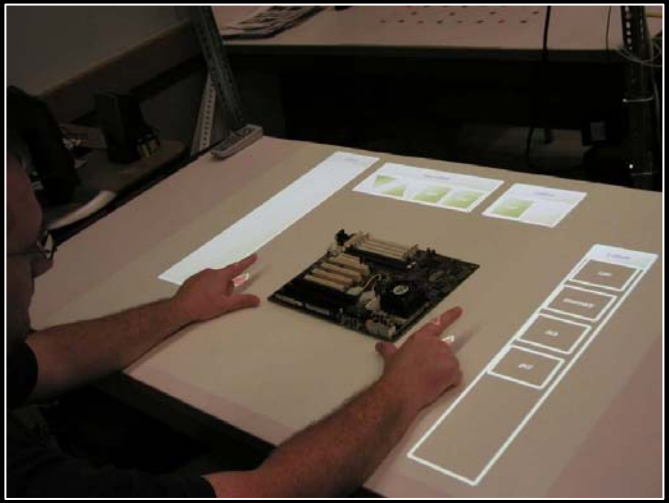
- Create a *common locus* for virtual interaction without having to shift attention between input and display devices
- Compose and synchronize *mixed-reality* video and audio for local and distant participants
- Create a *low-cost* scalable system that integrates multiple data streams over a uniform distributed platform

# Mixed-Reality Tabletop (MRT)

- Create *stations* containing a tabletop, camera, and projector to provide intuitive, device-free interaction
- Support both virtual and real objects on same tabletop
- Connect stations by transporting multimedia data over the network for composition and display on remote stations
- Provide a software toolkit for fast application development



# Example MRT Applications

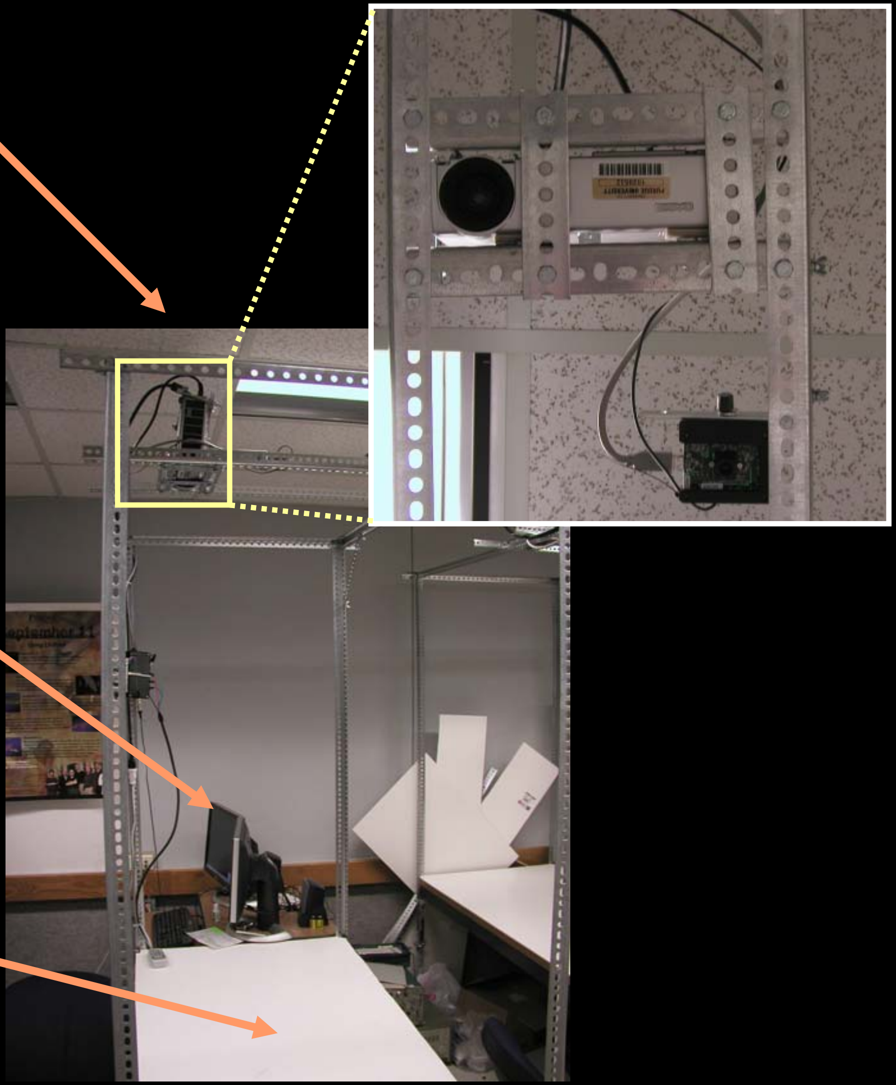


# Presentation

- Introduction
- ➔ System Overview
  - MRT Station
  - Station Pipeline
- Key Components
  - Synchronization
  - Calibration
  - User Interface
- Applications
  - API Framework
  - Interactive Classroom
  - Interactive Physics
  - Interactive Origami
- Conclusions

# MRT Station

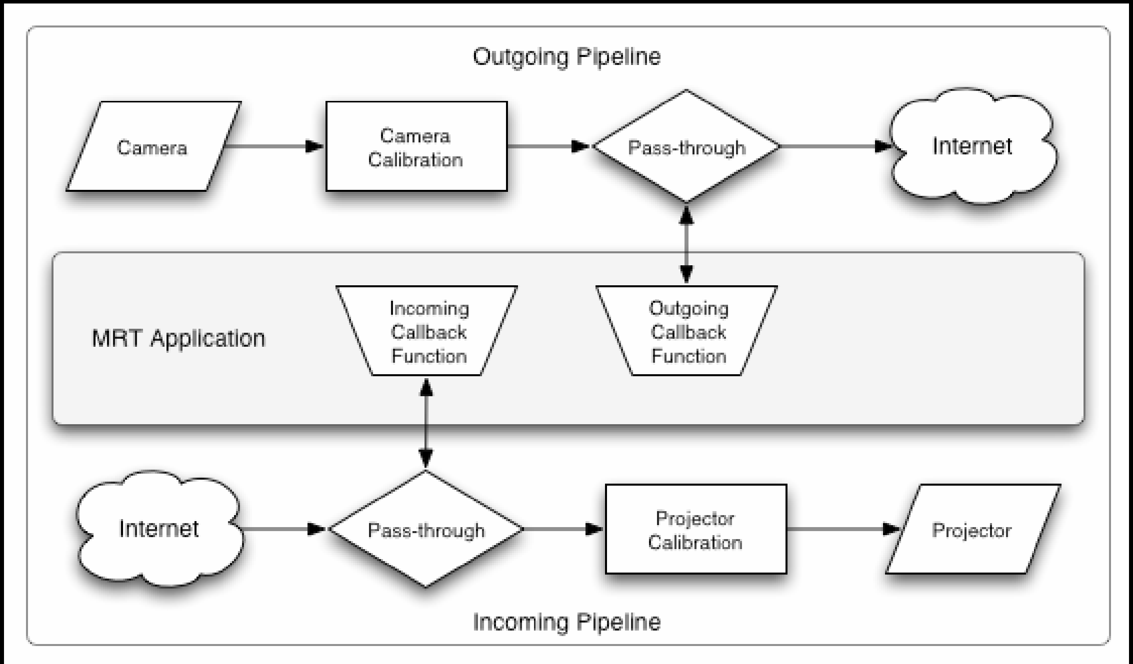
- Projector and camera
- PC workstation
- Tabletop



# MRT Software-only Station

- PC only
  - Mouse movements are mapped into MRT environment

# MRT Station Pipeline



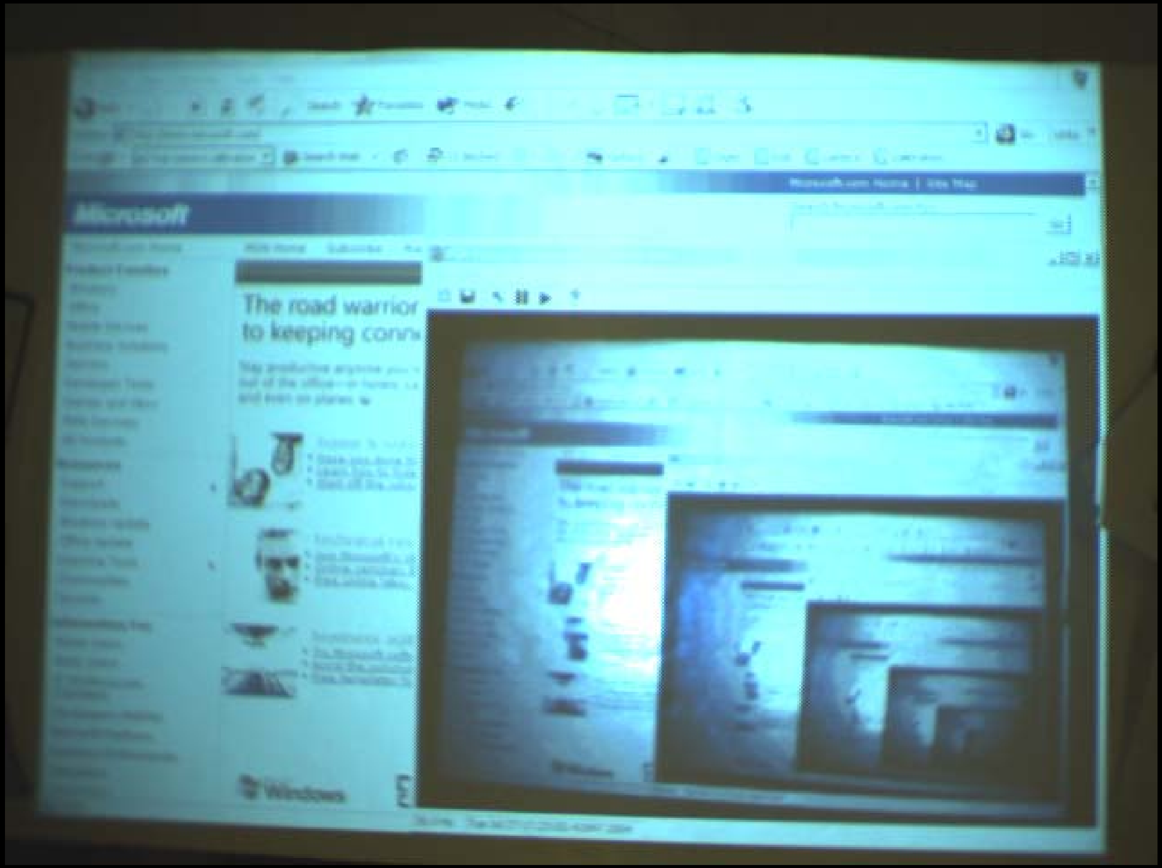

- The stations are interconnected by a programmable pipeline for “composing” real and virtual imagery over a network

# Presentation

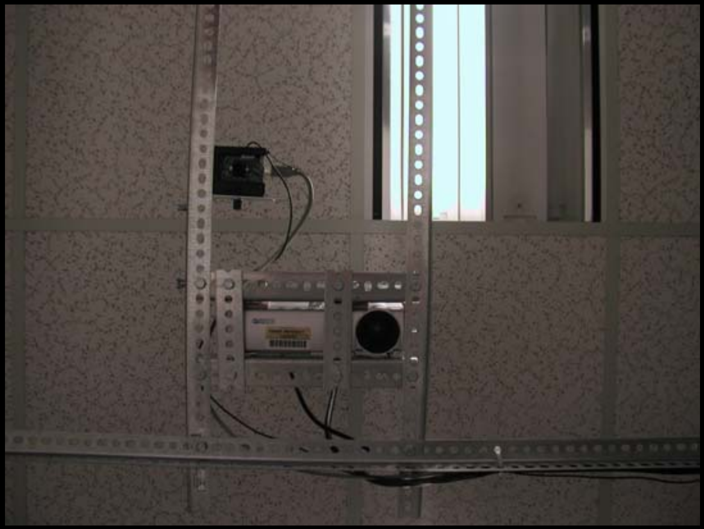
- Introduction
- System Overview
  - MRT Station
  - Station Pipeline
- ➔ Key Components
  - Synchronization
  - Calibration
  - User Interface
- Applications
  - API Framework
  - Interactive Classroom
  - Interactive Physics
  - Interactive Origami
- Conclusions

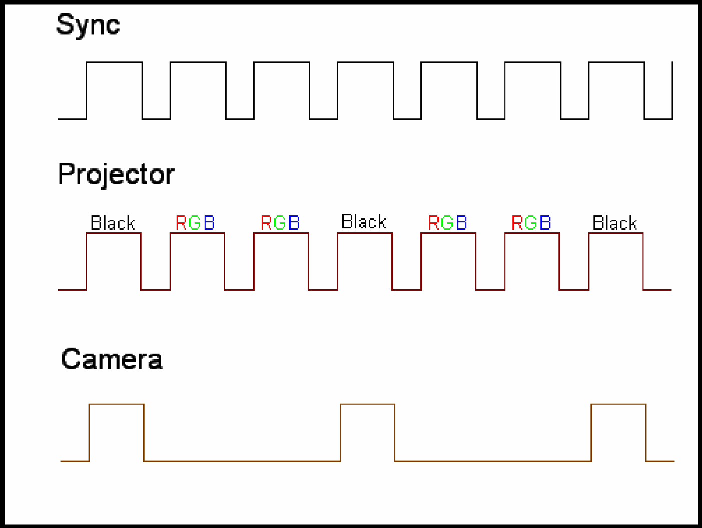
# Camera-Projector Synchronization

- Synchronize the camera and projector to prevent an "infinite mirror" effect



# Camera-Projector Synchronization



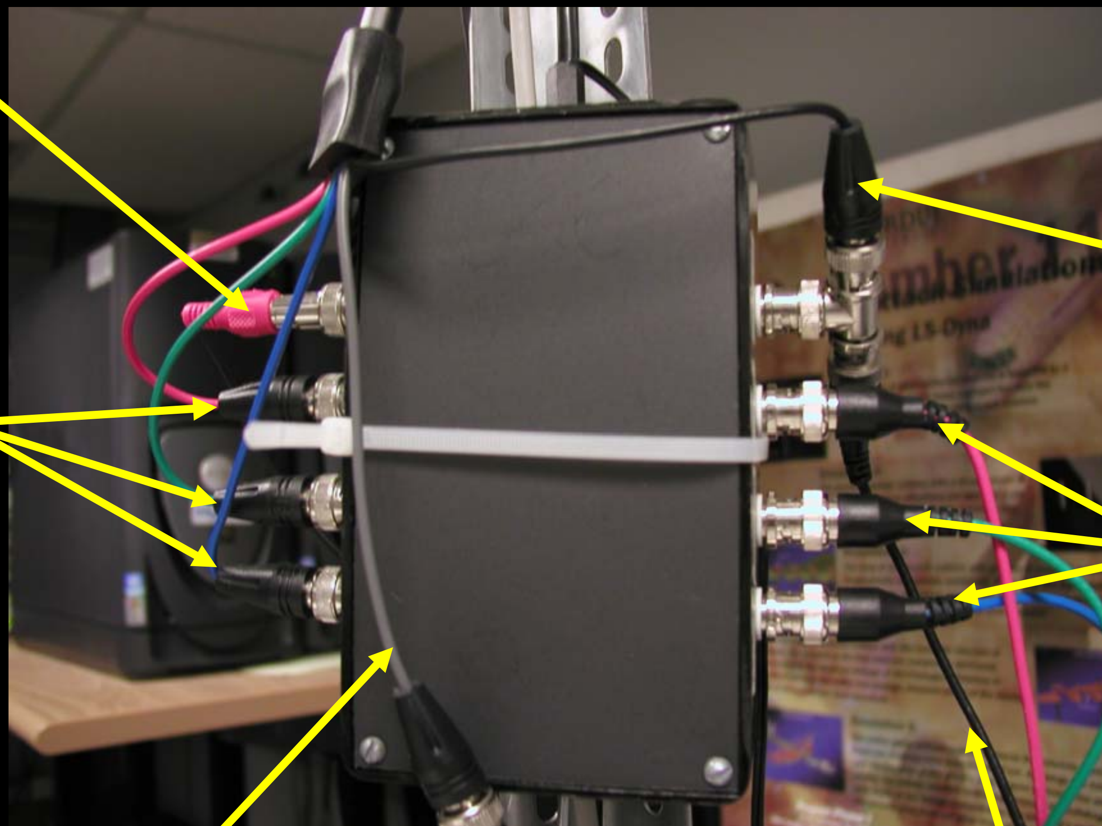



- Frame 1
  - Camera triggered
  - Black image projected
- Frame 2
  - RGB image projected
- Frame 3
  - RGB image projected
- and so on…

# Camera-Projector Synchronization

V-sync to camera

RGB to Projector



V-sync to projector

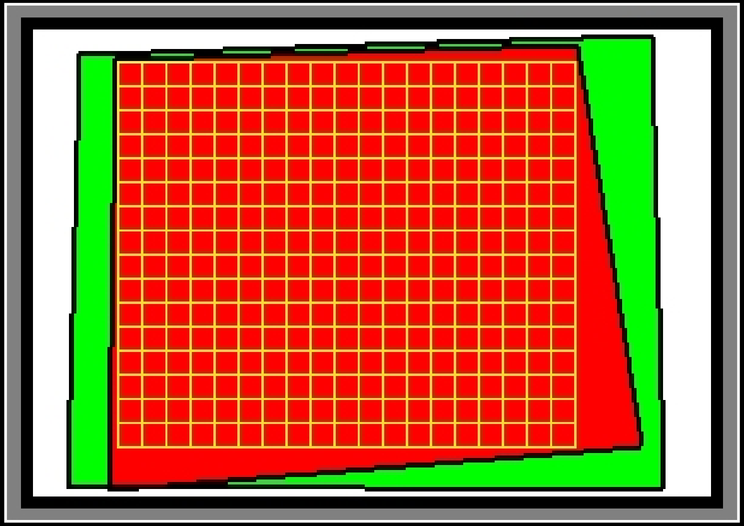
RGB from Video Card

HV-sync (bypasses black box)

V-sync split from VGA signal

# Calibration

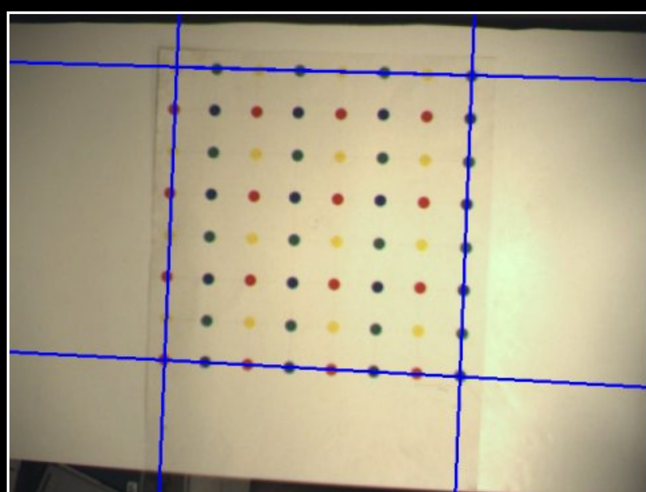
- Perspective and lens distortion cause the raw camera and projector images to be misaligned with the tabletop and each other
- Determine a mapping to warp from the camera's coordinate system to the tabletop's coordinate system

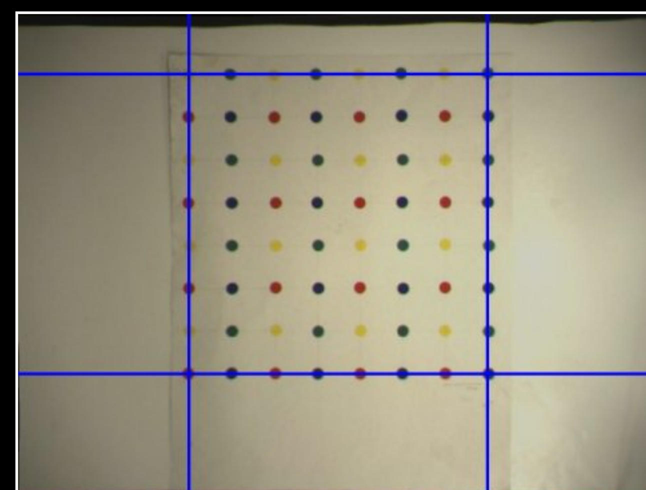


**Tabletop overhead**: the visible camera area (green) and projector area (red) are aligned with the tabletop (white) to form a rectilinear grid (yellow)

# Calibration: Camera

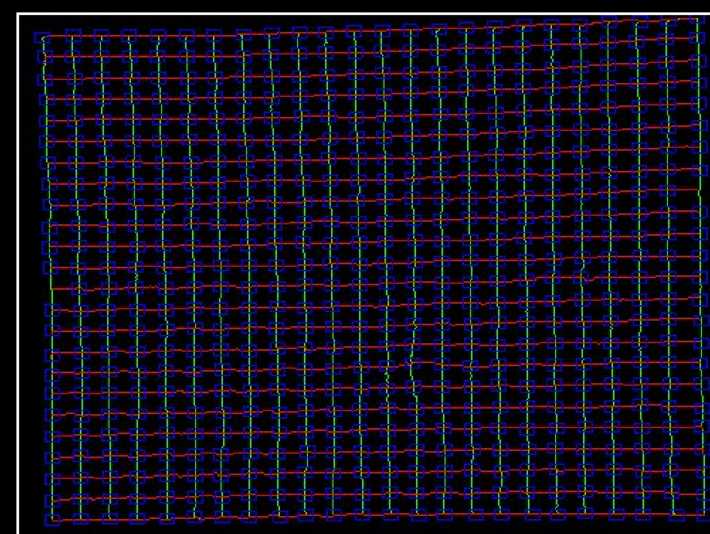
- A snapshot is taken of a rectilinear grid on the tabletop
- Known points on the grid are corresponded to their pixel locations in the snapshot
- The point correspondences are used to approximate the camera warp [Tsai87]





# Calibration: Projector

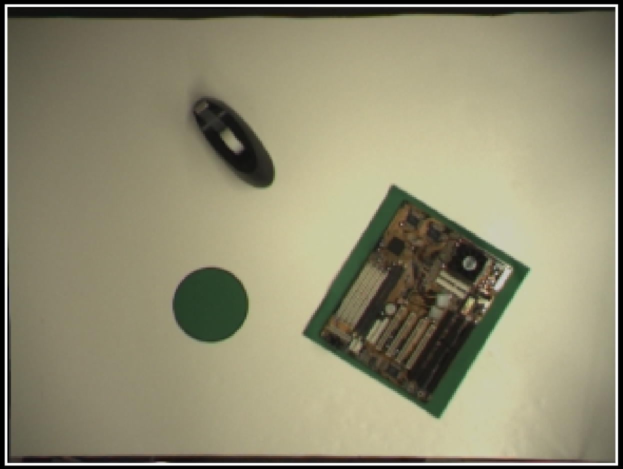
- A rectilinear grid is projected onto the tabletop and recorded by the camera
- The recording is transformed by the camera warp
- Points on the grid are corresponded to their pixel locations in the warped camera image

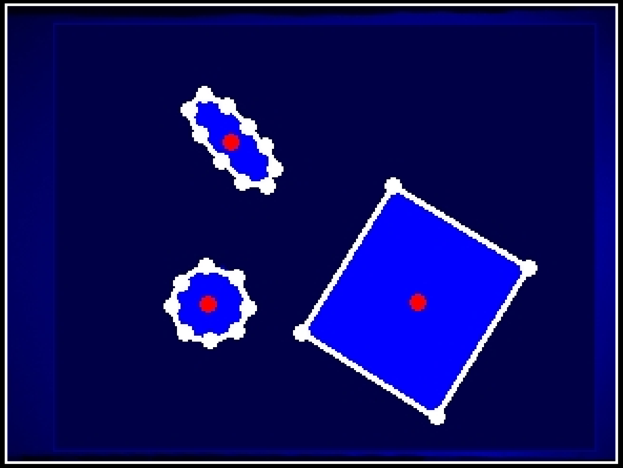


# User Interface

- Provide an intuitive graphical user interface with no interaction with keyboard or mouse
- Support object tracking and recognition
- Adopt same interface for PC-only mode

# User Interface: Tracking Objects

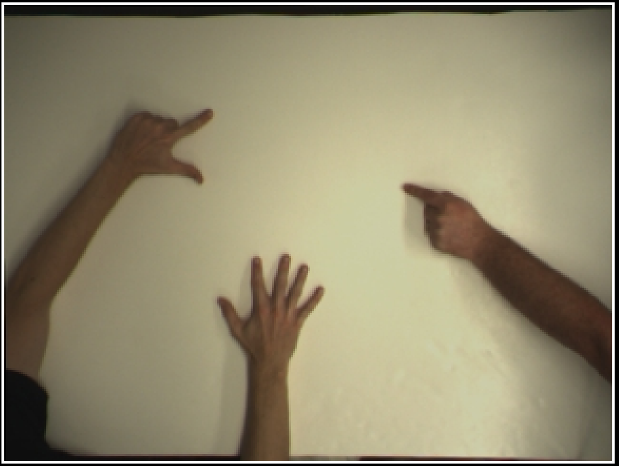


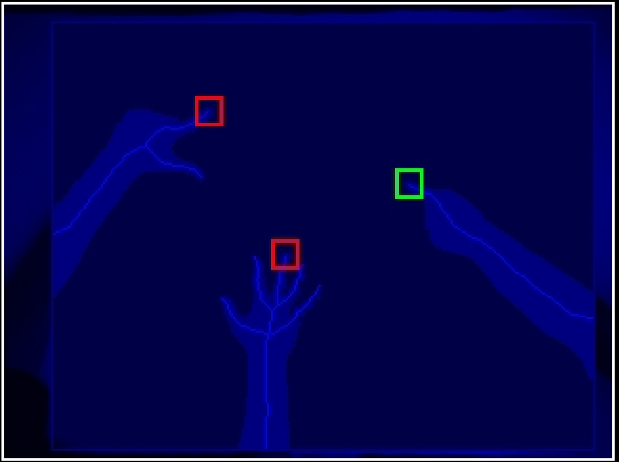


- Objects, in the interior of the table, are distinguished from the white table background using an intensity threshold
- Objects are tracked from frame to frame by considering attributes like pixel area and average position

# User Interface: Tracking Hands

- Foreground regions touching the edge of the table are considered hands or pointers
- Mouse press events are simulated by the opening and closing of the hand
- Hand regions are thinned to produce a single-pixel thick skeleton and a graph is created to describe the skeleton’s connectivity





# MRT Configuration and Performance

- Station specs:
  - Pentium 4 @ 3.2 Ghz, 512 Mb RAM
  - 100 Mbit Ethernet
  - 640x480 resolution camera triggered at 20 FPS
  - 1024x768 DLP projector at 60 FPS
  - (total cost ~$4000)
- Per frame processing:
  - video capture and warp: ~15 msec
  - object tracking: 1 to 10 msec (depending on object count)
  - network streamed video: ~7 msec
- Overall performance:
  - 20 FPS, limited by projector synchronization

# Presentation

- Introduction
- System Overview
  - MRT Station
  - Station Pipeline
- Key Components
  - Synchronization
  - Calibration
  - User Interface
- ➔ Applications
  - API Framework
  - Interactive Classroom
  - Interactive Physics
  - Interactive Origami
- Conclusions

# API Framework

- Provide basic controls like buttons, numeric selectors, and panels
  - Use C++ inheritance to create custom controls from a base control class
- Provide programmable event control
  - networking
  - mouse click, move, drag n' drop
  - object tracking
- Render graphics using DirectX/OpenGL

# Application #1: Interactive Classroom

- Uses an Instructor / Student model
  - One instructor and multiple students
- Designed for use with students from grade 6 and up
- Instructor can use environment for:
  - Demonstrations and labs (e.g., biology dissections)
  - “Show and Tell” (e.g., describe parts of circuit board)

# Instructor and Student Environments

- Instructor environment includes:
  - Programmable labels
  - Extendable list of students
  - Composable multiple-choice quizzes
  - Movable button panels
- Student environment includes:
  - Movable labels
  - Ask-question and submit-response buttons
  - Viewable user list
  - Movable button panels

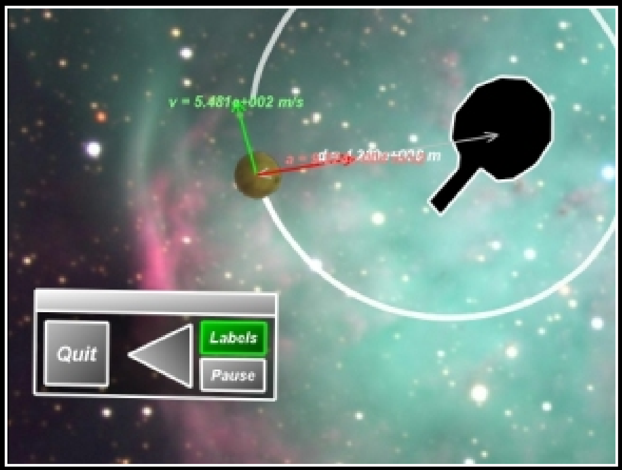
# Interactive Classroom

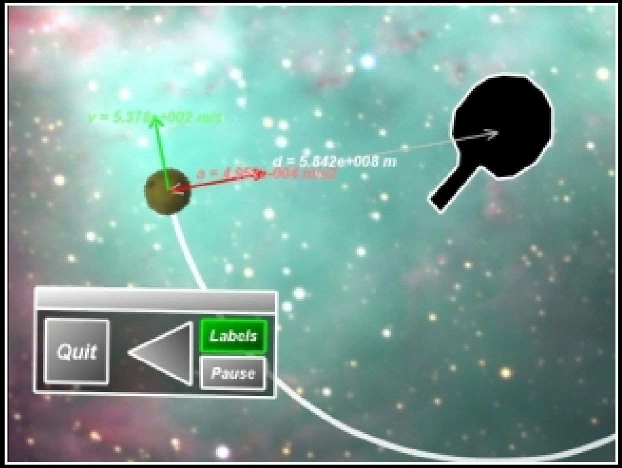
**Teacher's side…**

# Application #2: Interactive Physics

- Allow students to interactively experiment with physics concepts in mixed reality
- Allow remote tables to interact in a common physical simulation environment
- Take advantage of object tracking to model real physical characteristics
- Display interactive labels such as vector arrows

# Interactive Physics: Orbital Motion




- Students learn about 2D orbital motion and Newton's law of gravity

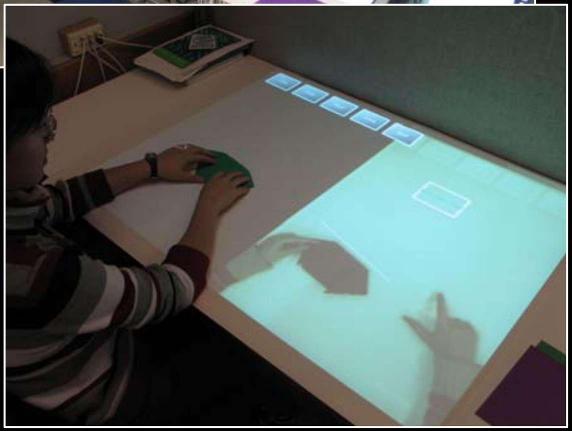
$$F = ma = G\, M_0 M_1 / d^2$$

- Students and teacher set the mass of an object placed on their respective tables
- The teacher sets the scale of the universe
- The student sets the initial velocity vector for the orbiting object

# Interactive Physics: Orbital Motion

# Application #3: Interactive Origami

- Teaching and learning origami in mixed reality
- Table divided into teacher's and student's sides
- Virtual illustration tools allow teacher to effectively show student how to fold origami
- Allow student to save and review origami folding steps



# Presentation

- Introduction
- System Overview
  - MRT Station
  - Station Pipeline
- Key Components
  - Synchronization
  - Calibration
  - User Interface
- Applications
  - API Framework
  - Interactive Classroom
  - Interactive Physics
  - Interactive Origami

➔ Conclusions

# Conclusions and Future Work

- MRT creates a common tabletop for interaction among human users and objects
- MRT composes and synchronizes virtual and real objects for shared virtual venues involving local and remote users
- MRT demonstrates a low-cost scalable system that integrates multiple data streams over a uniform distributed platform

- Future Work
  - Provide richer virtual interactions and scenario creation (e.g., urban planning, emergency response training, …)
  - Extend to more pervasive display and surfaces (“Mixed Reality Room”)
  - Enhance user’s perception by improving camera/projector synchronization (e.g., DLP synchronization, projecting non-black images, …)