Freehand acquisition of unstructured scenes

Presented by Mihai Mudure

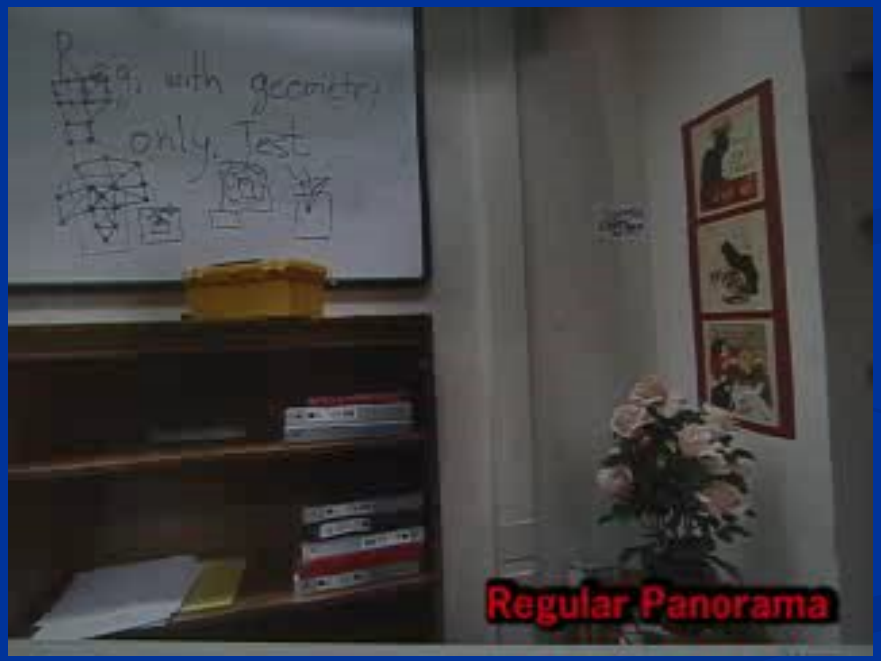
September 2006

Goals

- Acquire interactively approximate models of unstructured scenes
- Inside looking out case
- Freehand

Unstructured scenes

- Scenes that contain many small surfaces
  - Leafy plants, messy desks, coats on a rack



Unstructured scenes

- Detailed modeling requires
  - Huge time investment
  - Expensive acquisition hardware

Challenges

- Data acquisition
  - Acquire depth information from many viewpoints
- Interactivity
  - The operator must be able to get feedback during data acquisition and guide the scanning

Challenges

- Tracking the acquisition device
- Modeling

Our solution

- Use the ModelCamera for acquisition
  - Acquires color frames enhanced with 45 depth samples
  - Evolving model is a colored point cloud
  - Point cloud displayed as we scan

Our solution

- Tracking
  - Previous approach: we used calibrated features (checkers)
    - Not very robust for long sequences
    - Operator had to concentrate on maintaining registration
  - ModelCamera mounted on a mechanical tracking arm

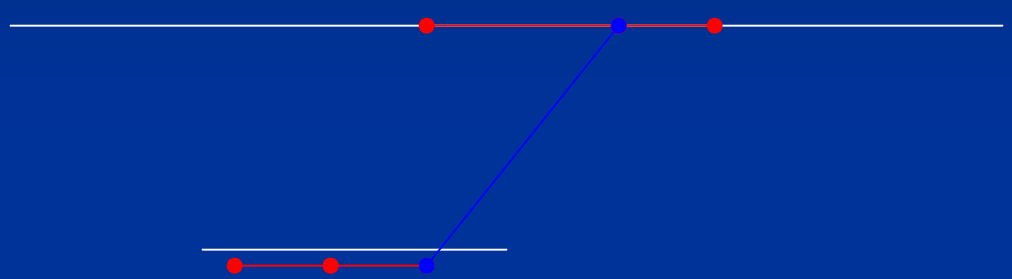
Our Solution

- Modeling
  - Disconnected representation
    - Splatting
  - Connected representation (triangle mesh)
    - Create an approximate mesh for each desired view
    - Color the mesh by projective texture mapping

Our solution

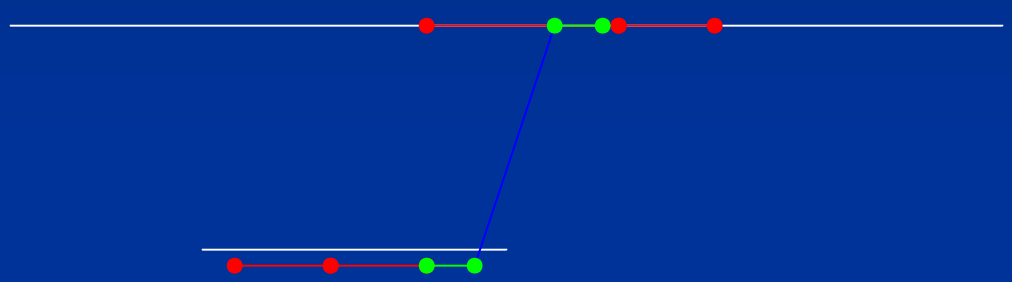
- Mesh generation
  - project points onto the desired view
  - Splat to determine visibility
  - Triangulate in 2D
  - Unproject each pixel covered by a splat into 3D, each such point will be a vertex of the 3D mesh
- Advantages
  - Reduces the size of the skins in the desired view

Mesh generation



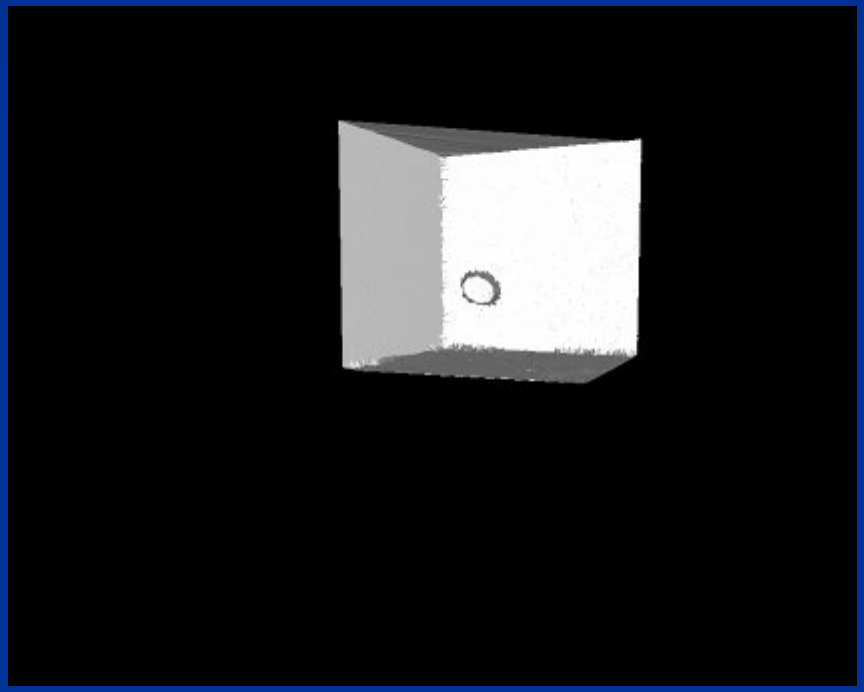
Desired View

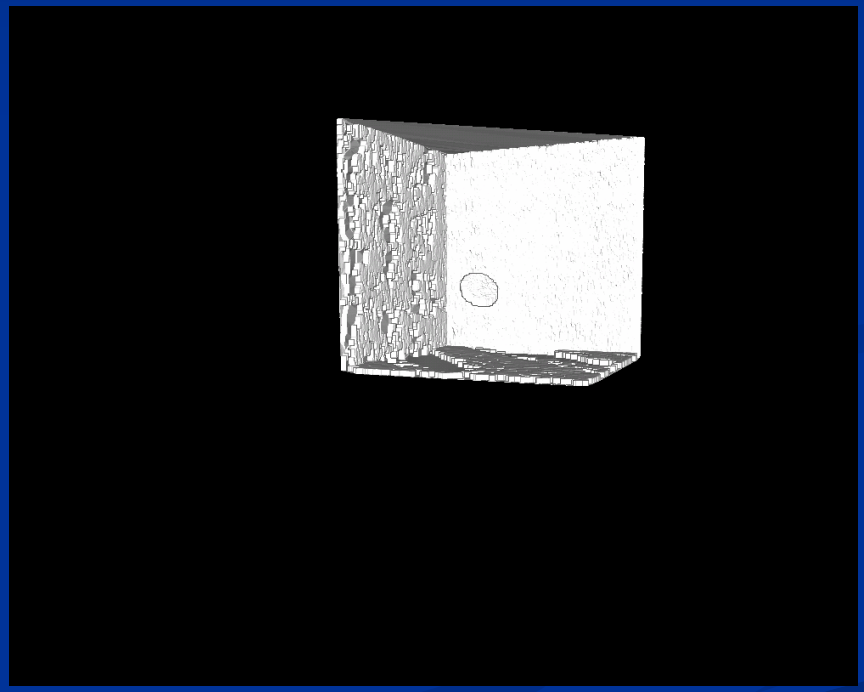
Mesh generation



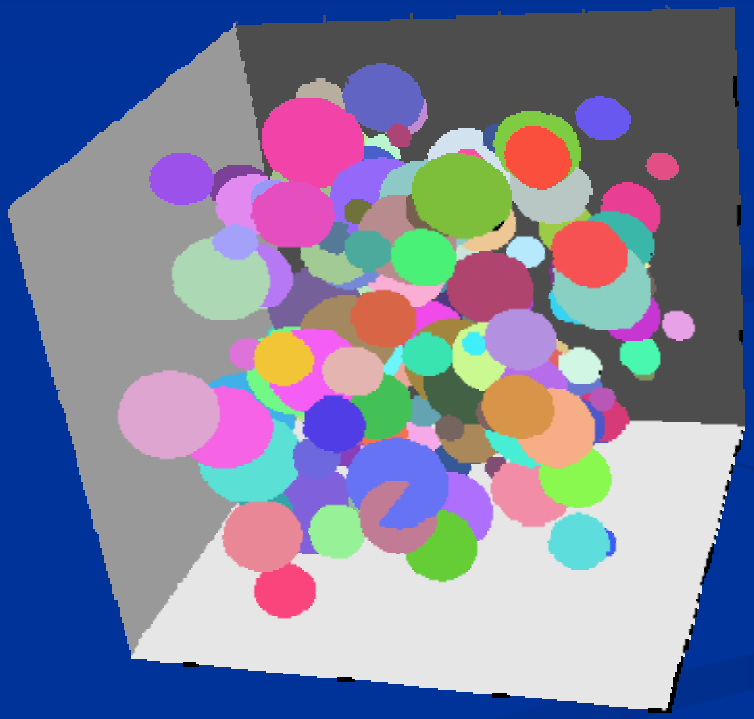
Desired View

Mesh

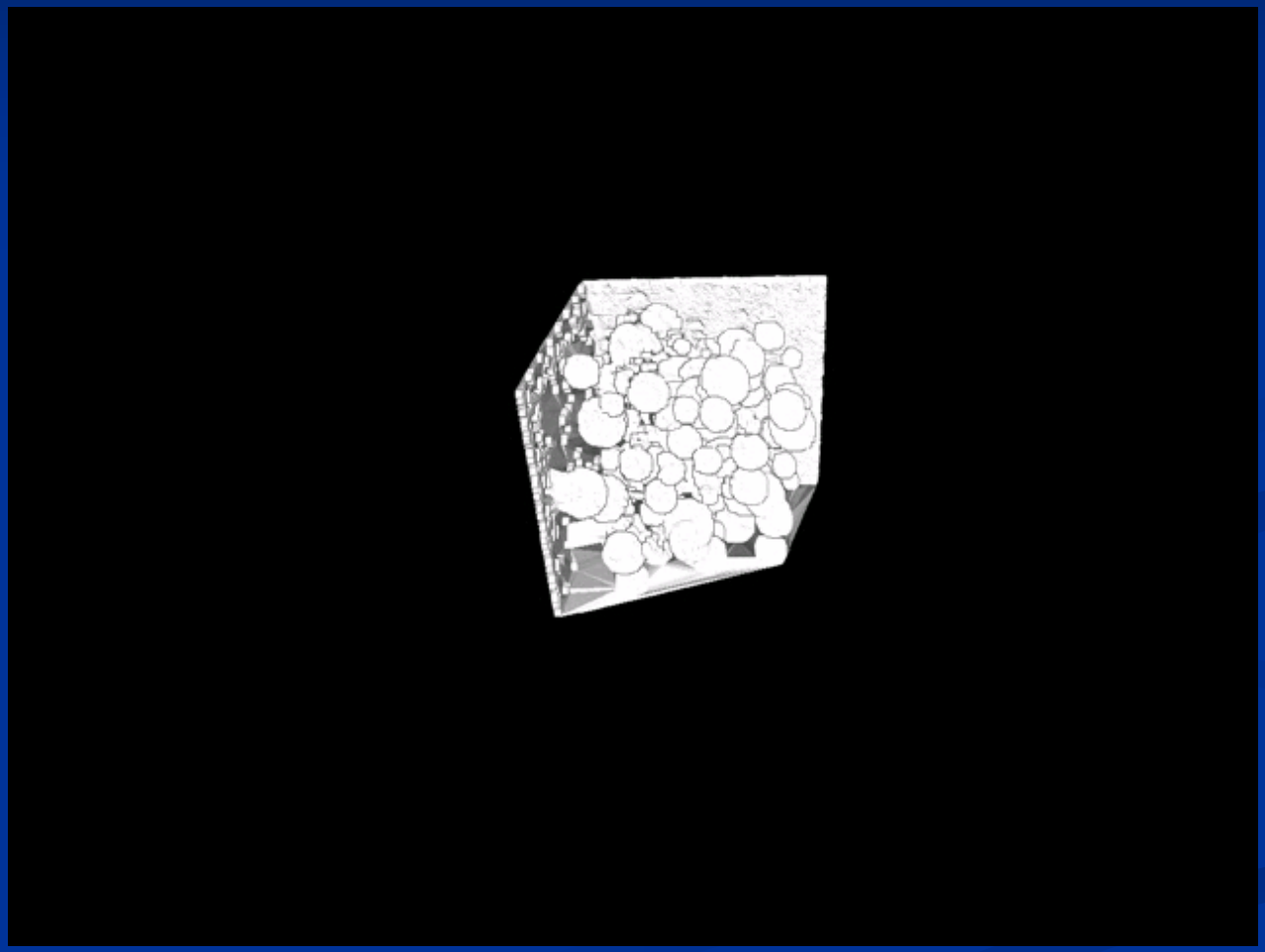




Mesh example



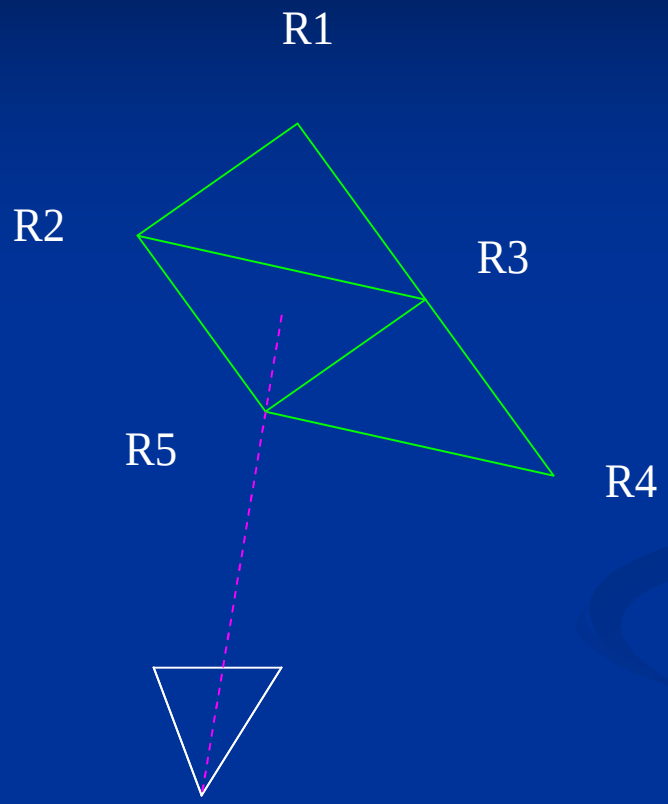
Mesh



Coloring

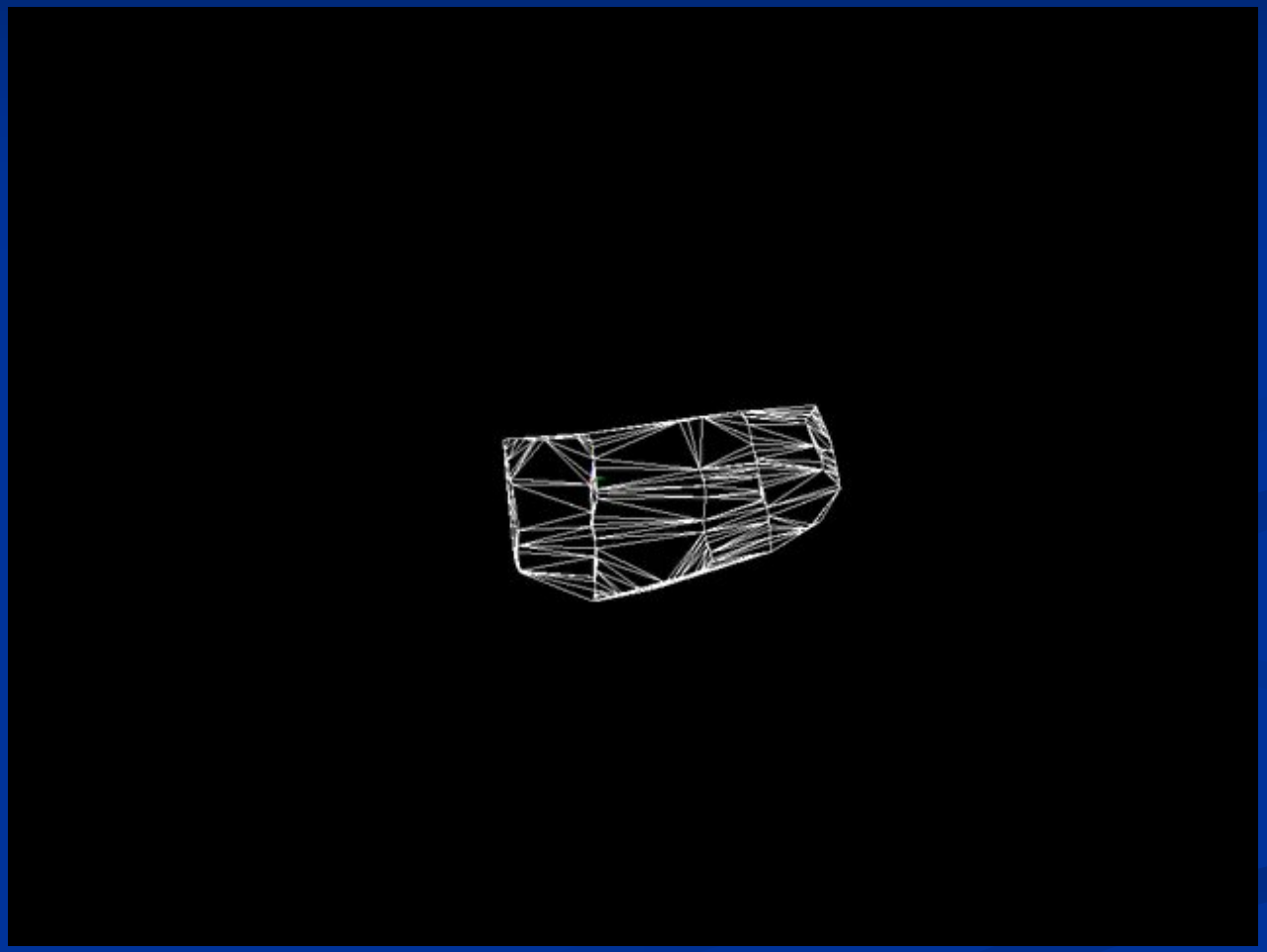
- Which reference images to use ?
  - Project reference COPs onto a sphere centered around the object
  - Triangulate projections
  - When rendering, project the desired view COP onto the sphere, find the triangle and color using the corresponding reference cameras
- Assumption : the entire object is visible in the reference images
  - Enforced during preprocessing

Coloring



Desired View

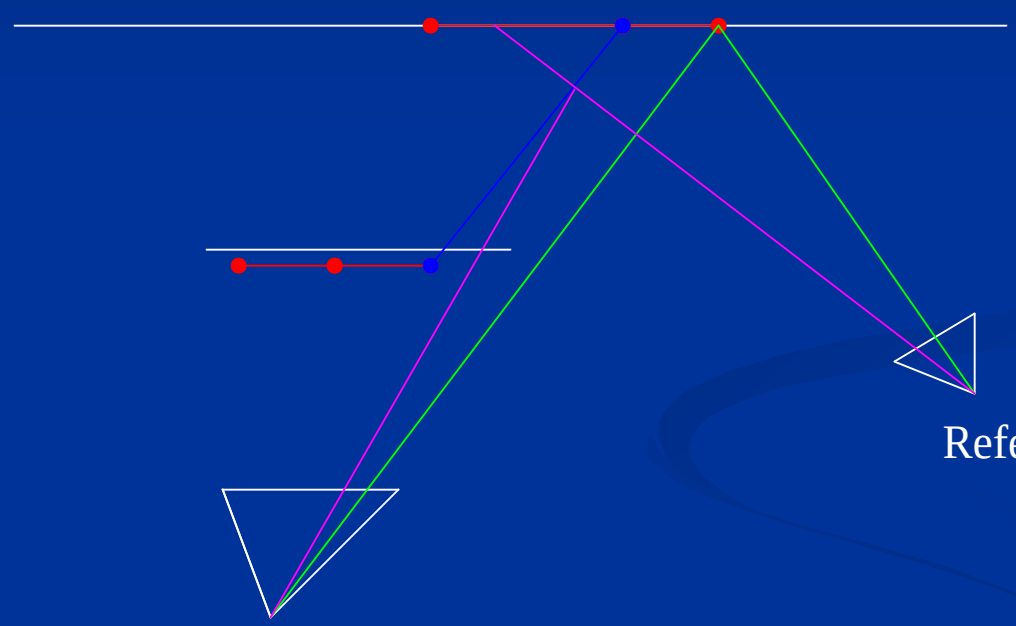
Coloring



Coloring

- Order reference cameras by the distance between the desired view COP projection and the reference camera COP projections onto the sphere
- For each desired view pixel find the pixel in the reference image where the corresponding 3D point projects
- Compare the depth of the point with the depth in the reference image (zbuffers for reference images are pre-computed)
- If the point is visible in the reference image, assign color

Coloring



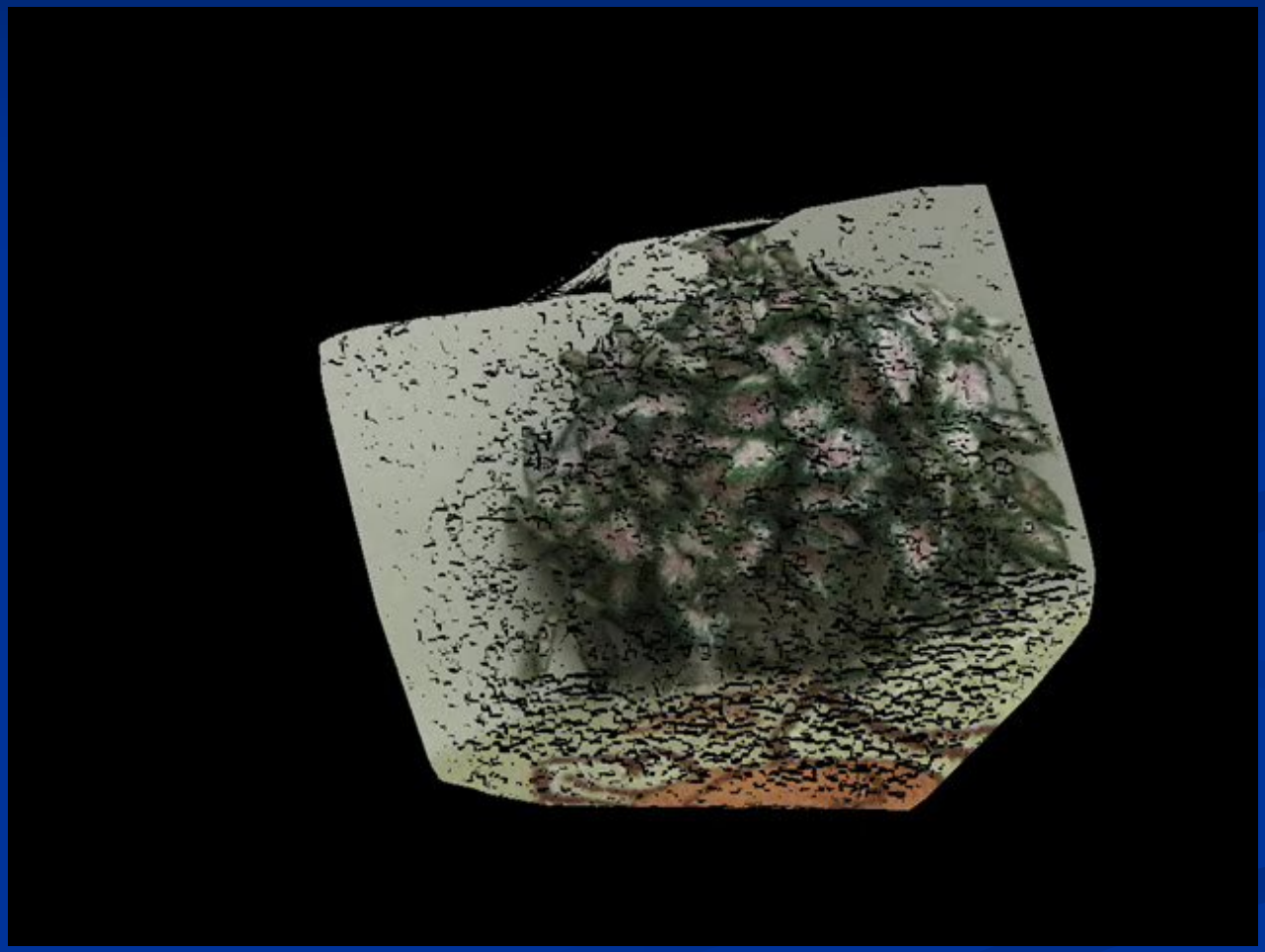
Coloring skins

- No good solution
- Skins are approximations of the surface
- They will get incorrect color from the reference cameras

Coloring skins

- Current solution
  - Simply fill in the missing color by averaging the neighbors
  - Works well as long as skin size remains relatively small (a few pixels wide)

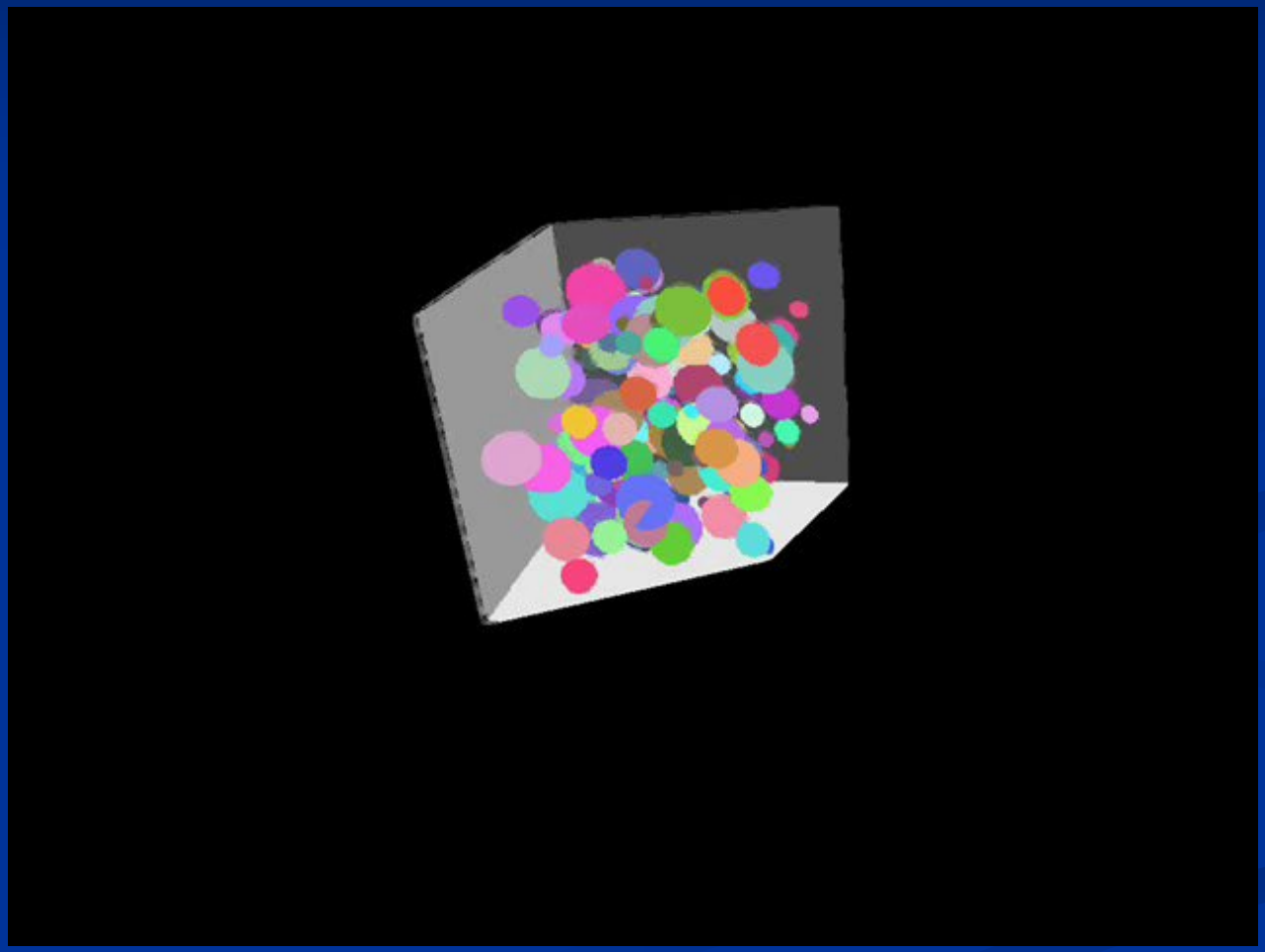
Coloring skins



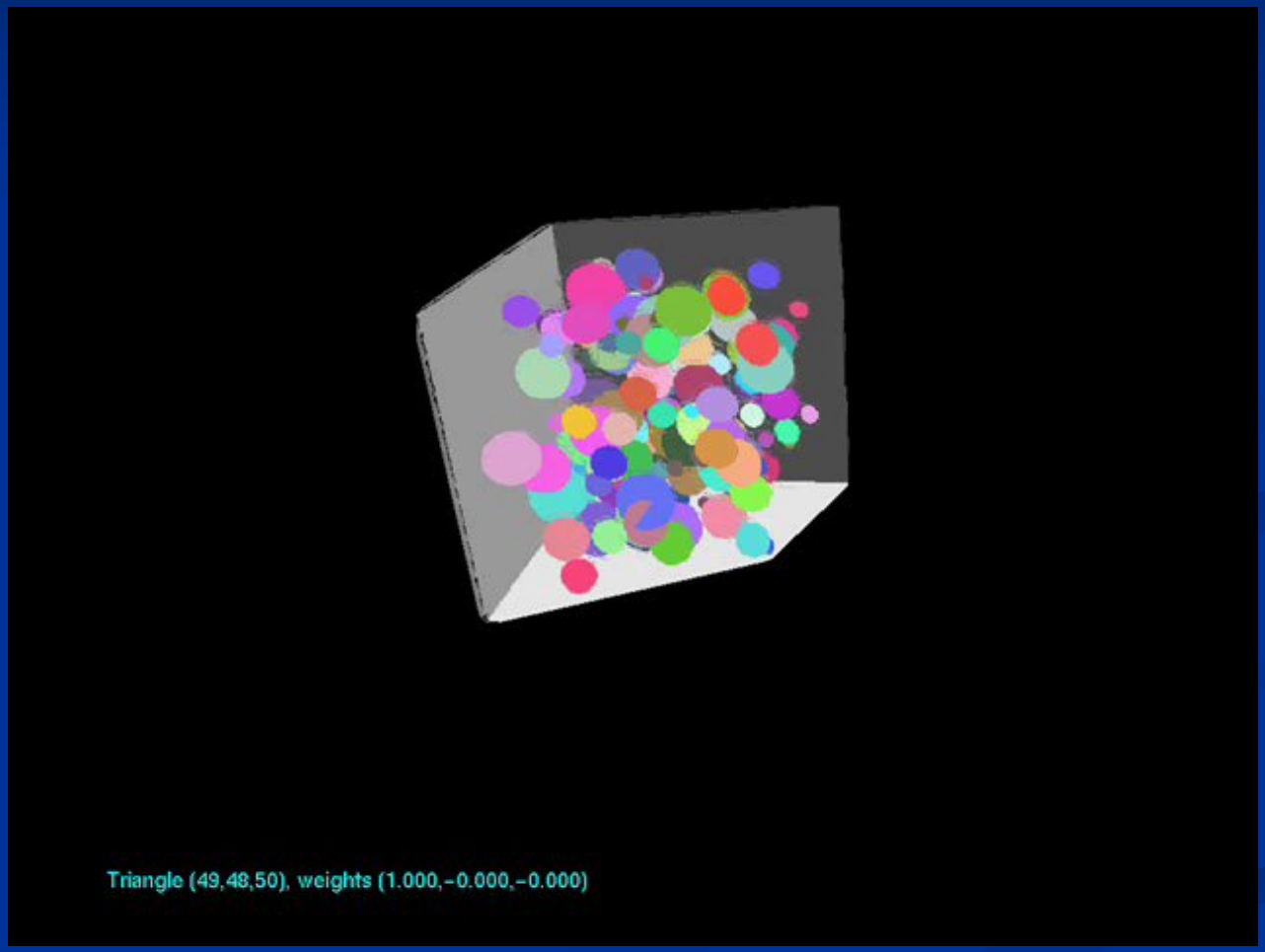
Coloring

- Another problem : popping
  - When the desired view changes from one set of 3 reference cameras to another we get very annoying popping
  - This is due to the approximate geometry + skins
  - Solution : render image 3 times using each of the 3 reference cameras as the first in the list, then blend

No Blending



Blending



Results

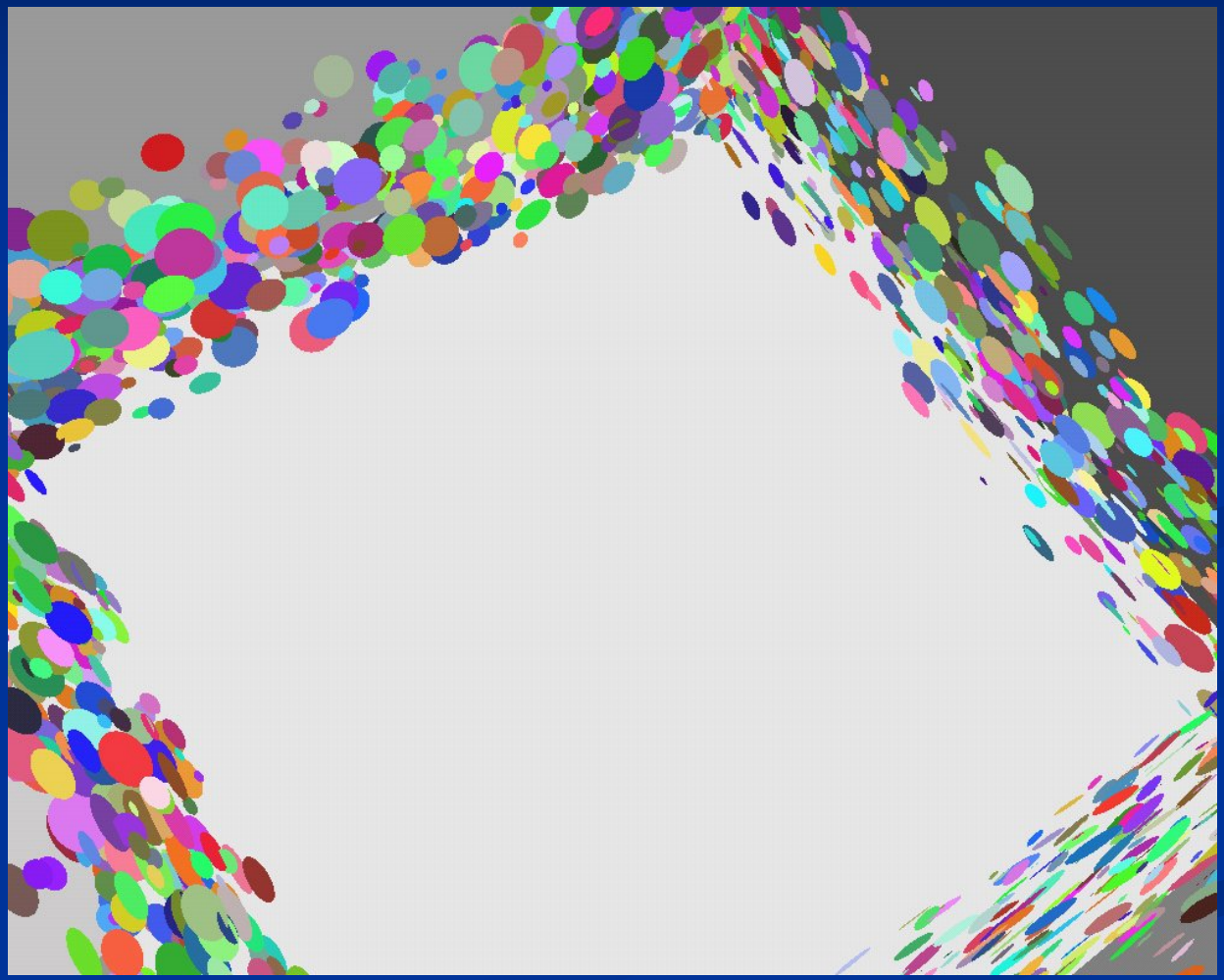
Inside looking out

- Mesh generation works for this case as presented
  - Splat size must be changed according to the desired view
- Coloring :
  - Which set of images to use to guarantee coverage of the entire scene
  - Current solution :
    - keep a list k of cameras that see a particular point (preprocessing step)
    - If a point is visible in the 3D mesh, use one of the cameras in its list to color
    - We are looking at determining a set of cameras, as small as possible, that cover the scene
    - Coherence from using one reference image to color a large part of the scene
    - Blending
  - How to blend to avoid the popping artifacts

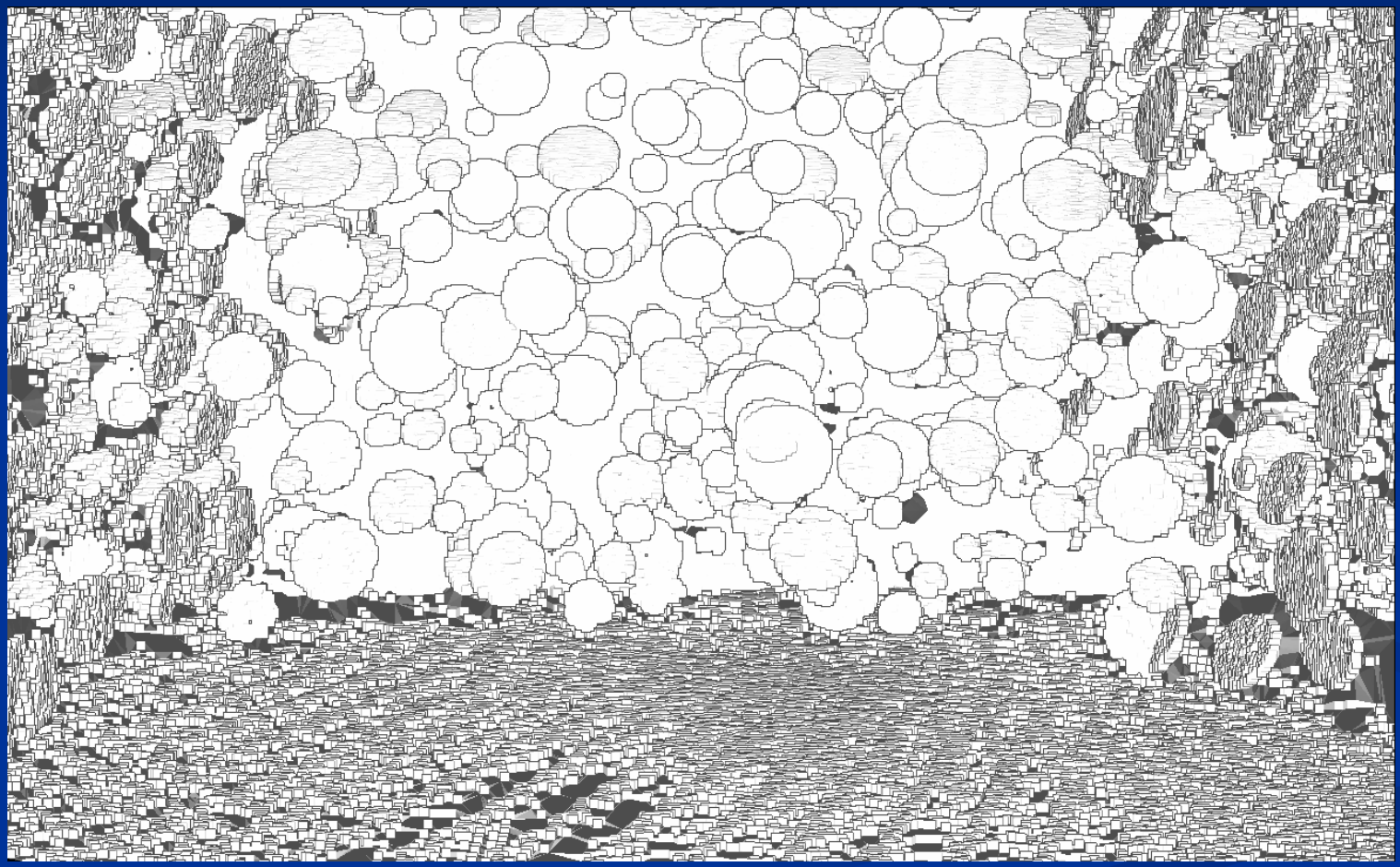
Inside looking out

- Speed
  - An entire room can have a lot of 3D points => slow mesh generation
  - If many images are needed to cover the whole desired view => slow coloring of the scene

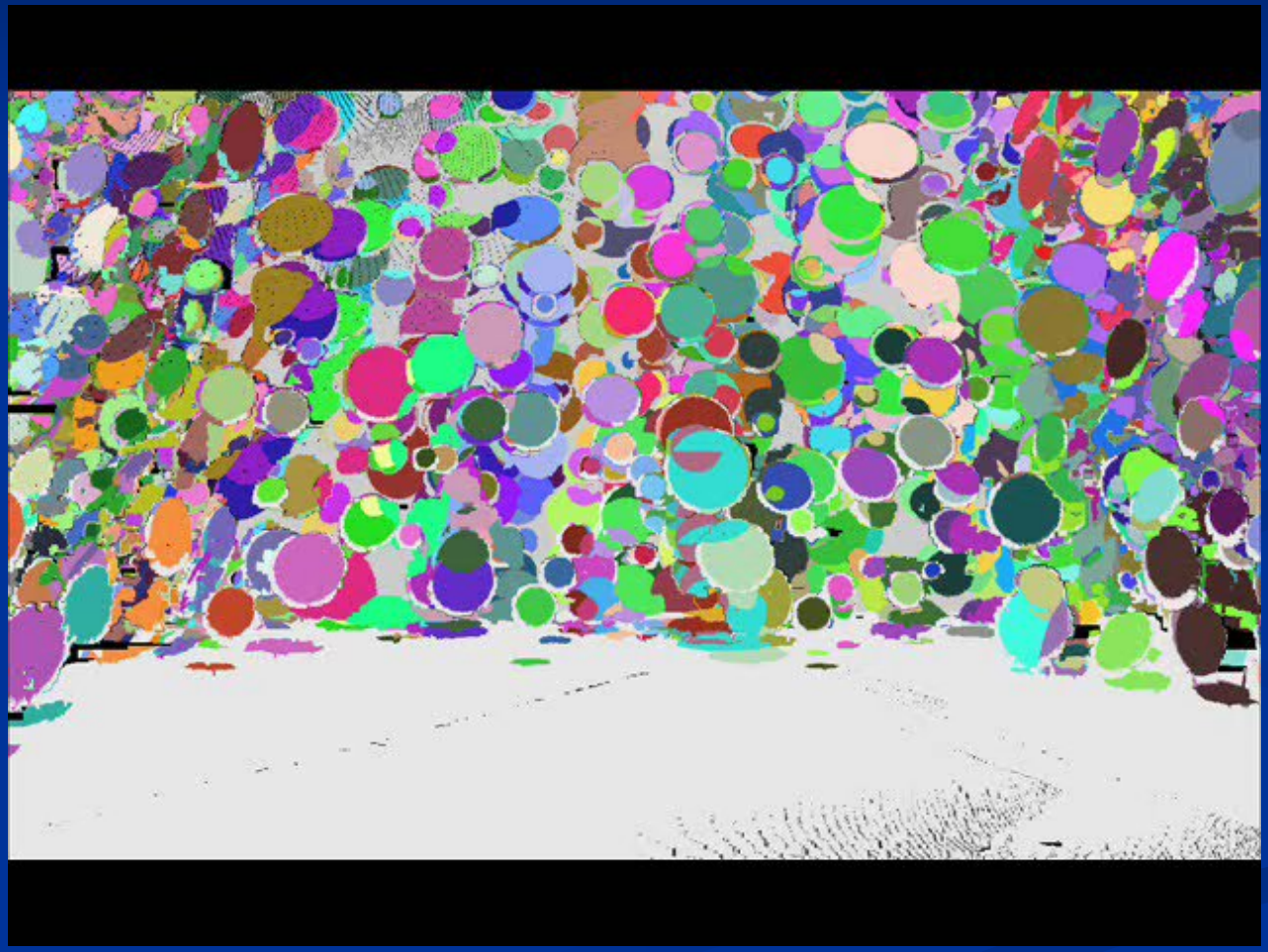
Simulator scene



Simulator scene



Demo



Thank you