

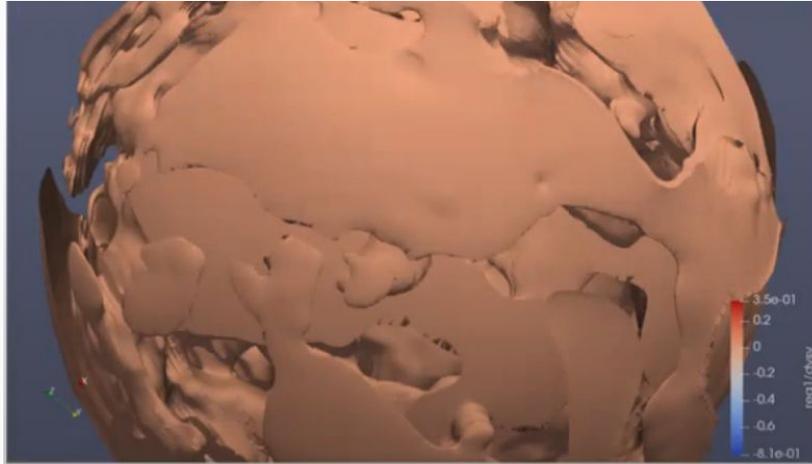
A 3D topographic map of a mountain range, rendered in shades of orange and brown. The map shows a central mountain peak with a valley running through it, surrounded by smaller hills and ridges. The terrain is detailed with contour lines and shading to indicate elevation.

Real-time in situ visualization & automated digital twin construction

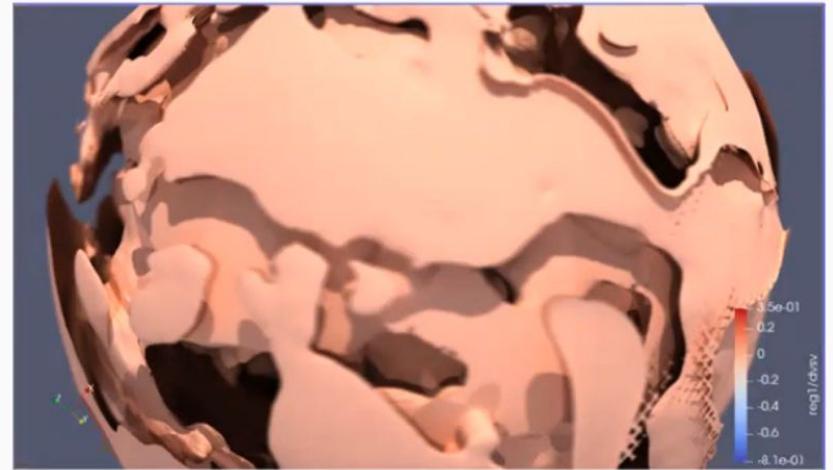
Krishna Kumar, The University of Texas at Austin
Paul Navratil, Andrew Solis, Greg Abrams, Sean McLean

IXPUG: Intel eXtreme Performance Users Group, April 2024

Why do we need ray tracing in scientific visualization?



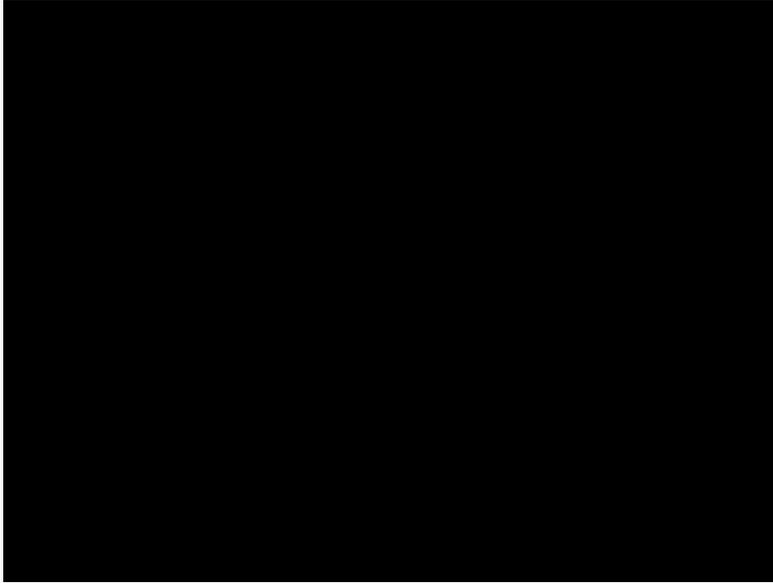
Without Ray Tracing



With Ray Tracing

Earth Interior (Kronos Group 2020)

Ray-tracing for engineering problems

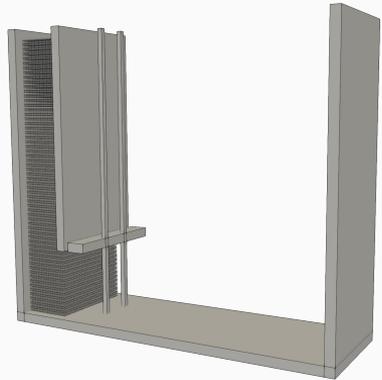


Courtesy of Satori Tsuzuki

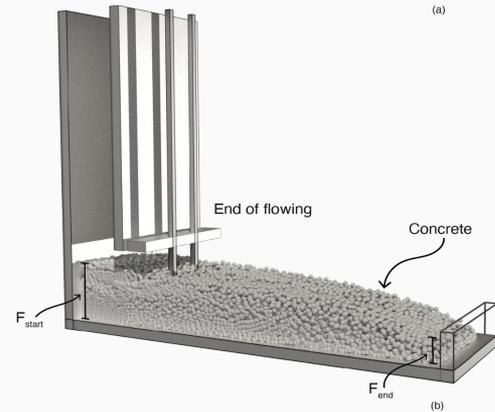
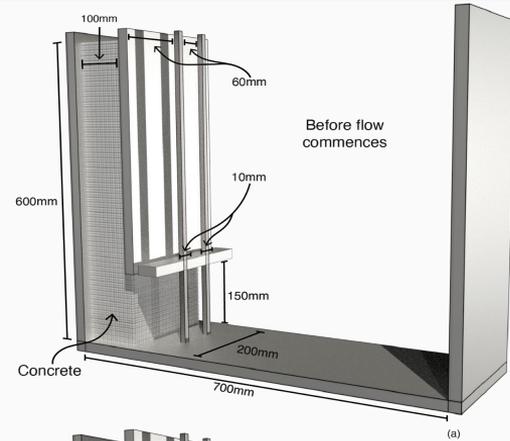


Dreamworks

Ray Tracing in ParaView (Intel OSPray-enabled): Concrete Flows

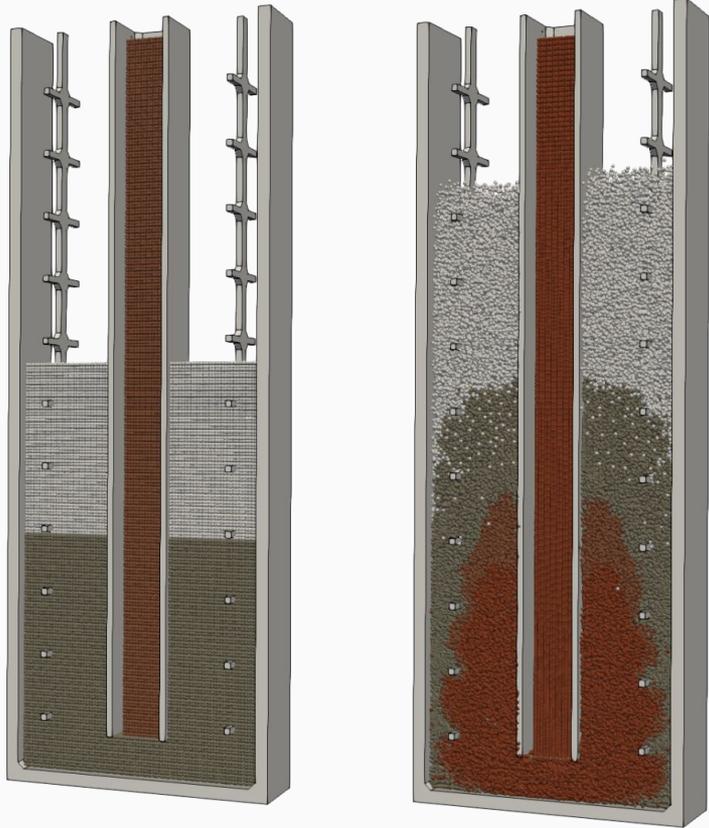


Without Ray Tracing

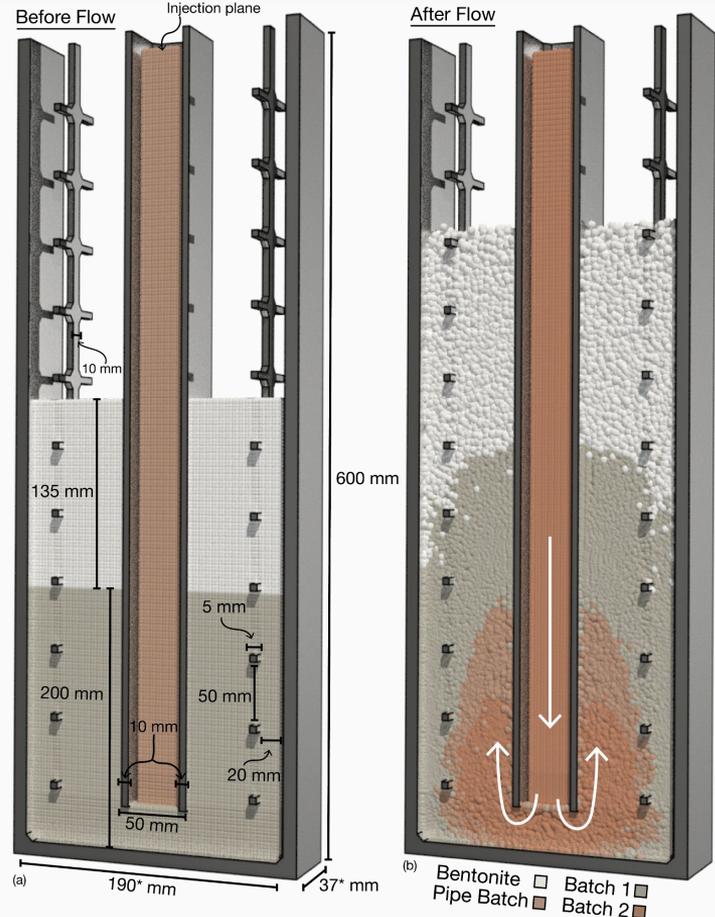


With Ray Tracing

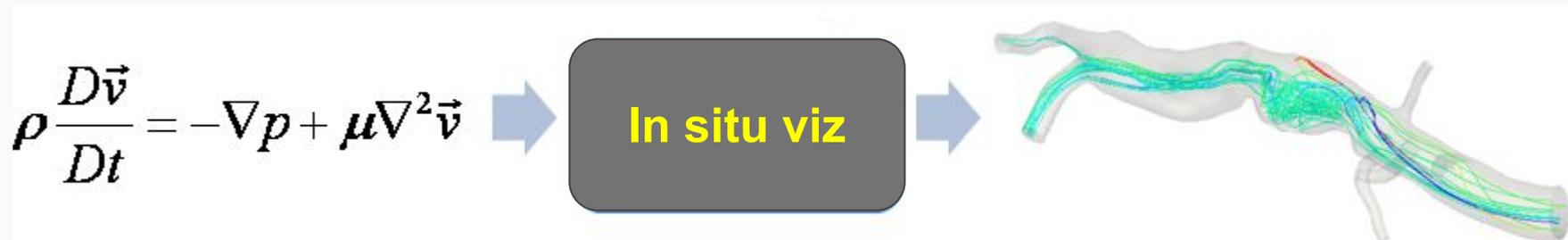
Ray Tracing in ParaView (Intel OSPray-enabled): Concrete Flow



Without ray tracing



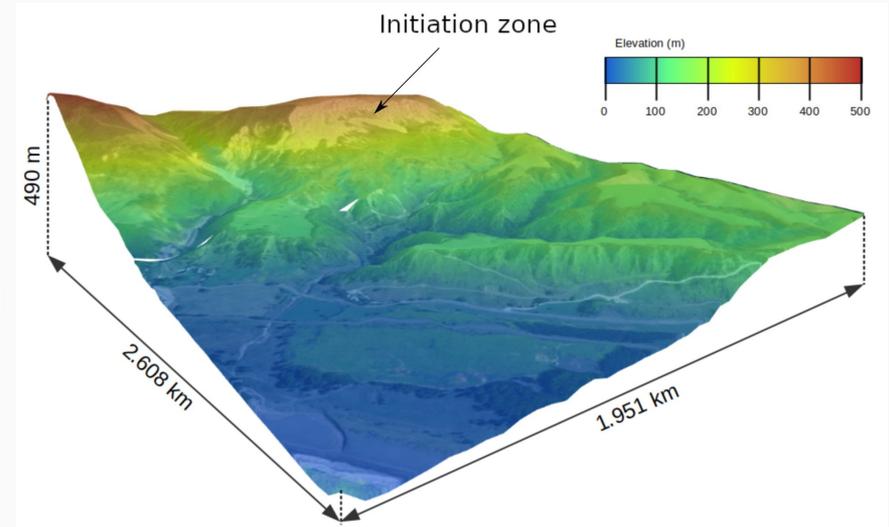
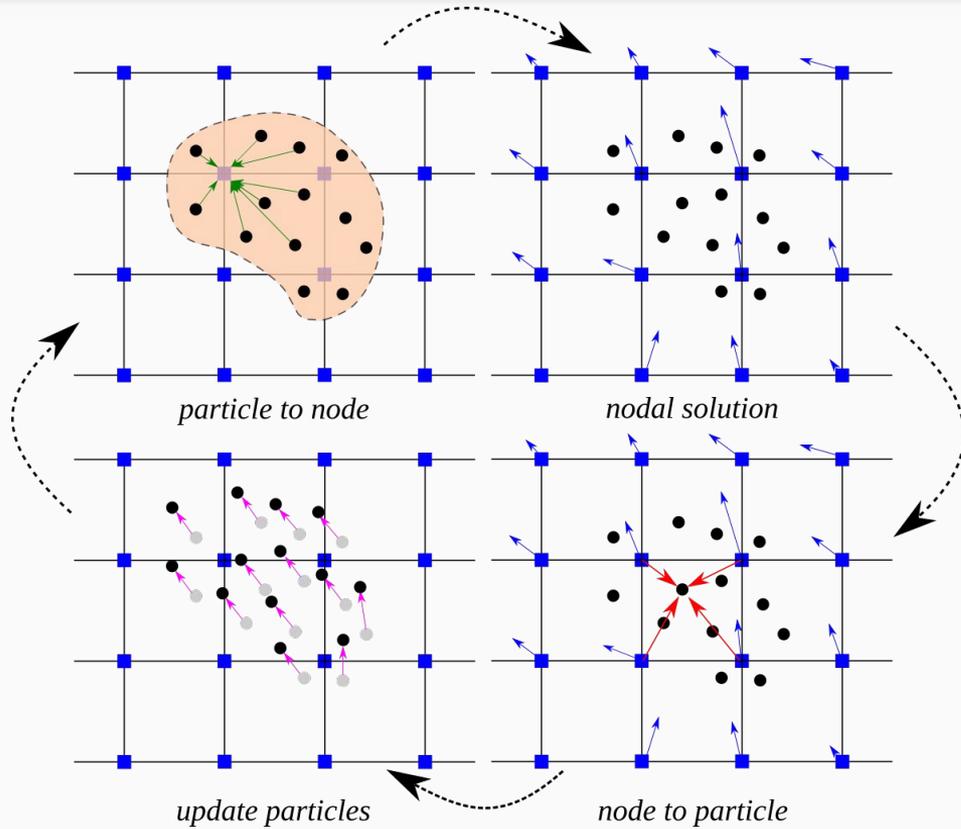
In situ visualization



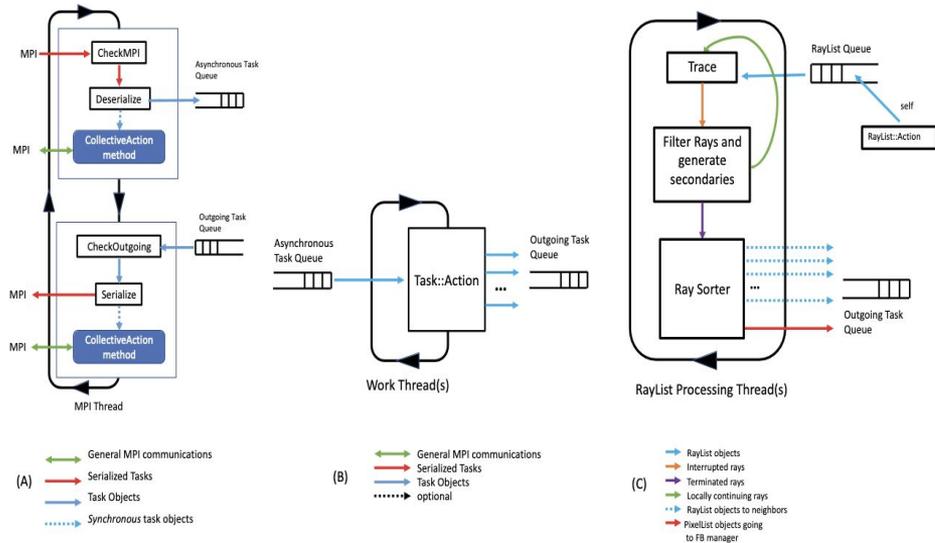
Challenges in visualization

- Viz is sequential: Input/BCs -> Simulation -> Post-hoc viz
- I/O is the bottleneck - Terabytes of data
- Visualization at arbitrary discrete time steps (loss of useful data)
- No live visualization / feedback
- Shared visualization is difficult
- Distributed Viz is challenging and often chunked

Material Point Method for large-deformation modeling



TACC Galaxy: Asynchronous ray tracing powered by OSPRay and Embree



Abrams et al (2018)

Intel OSPRay

Application (e.g., ParaView, VisIt)

Intel® OpenImageDenoise

Intel® OSPRay

ISPC Device

MPI Distributed Device

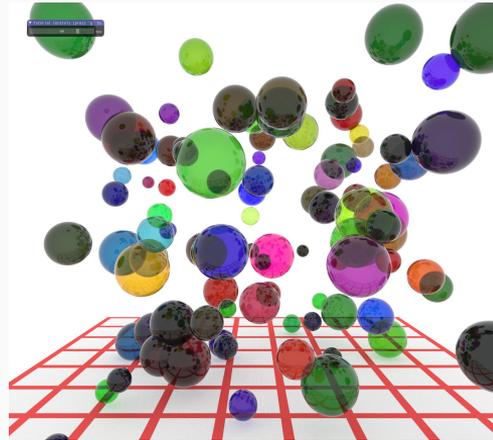
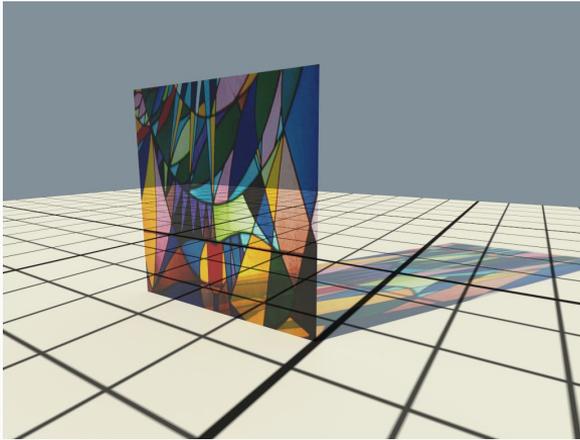
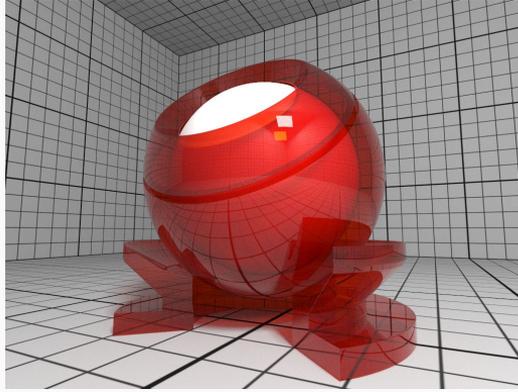
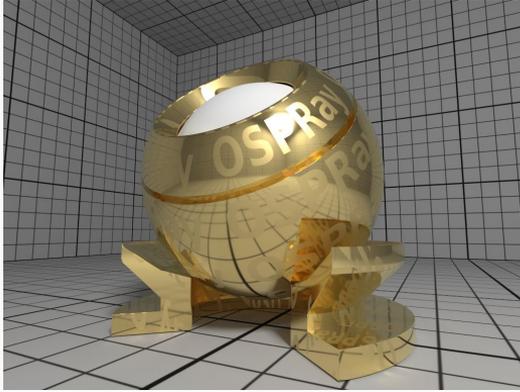
Intel® Embree

Intel® OpenVKL

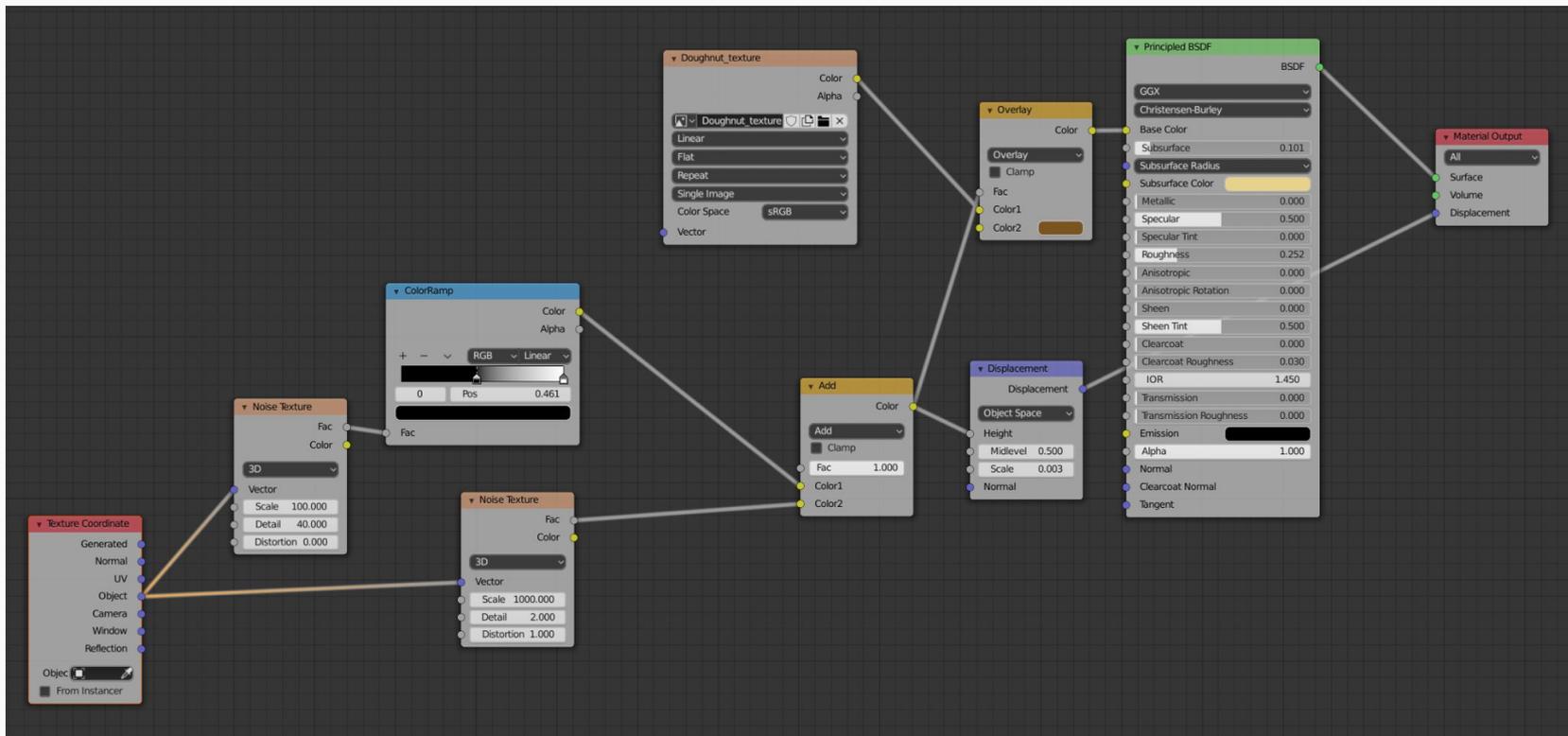
OpenGL, Mesa, etc.

Intel® Xeon®, Xeon Phi™, future GPUs

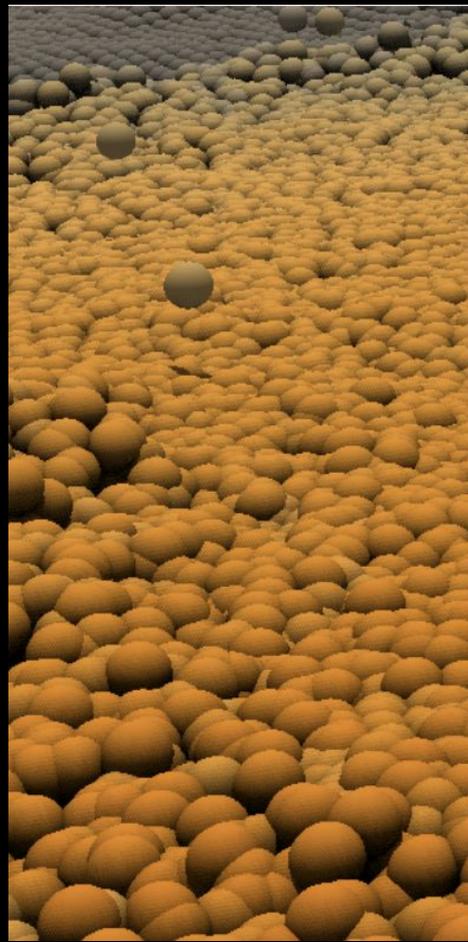
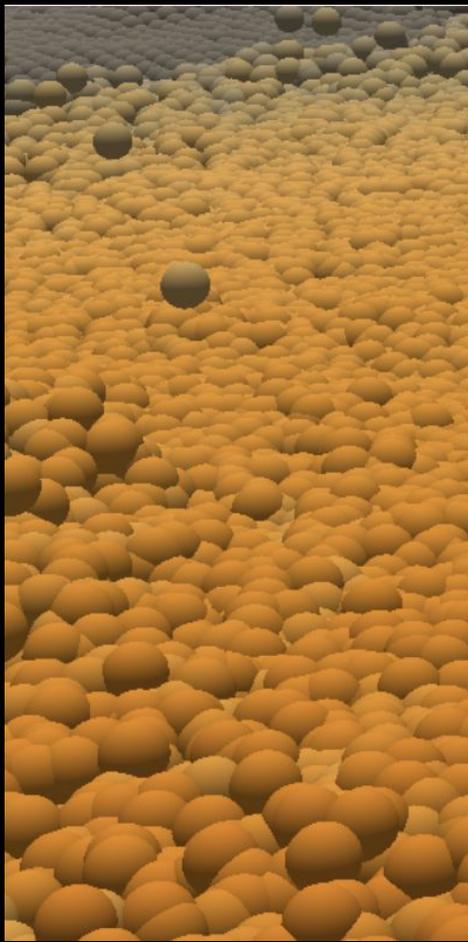
Intel OSPRay and Materials



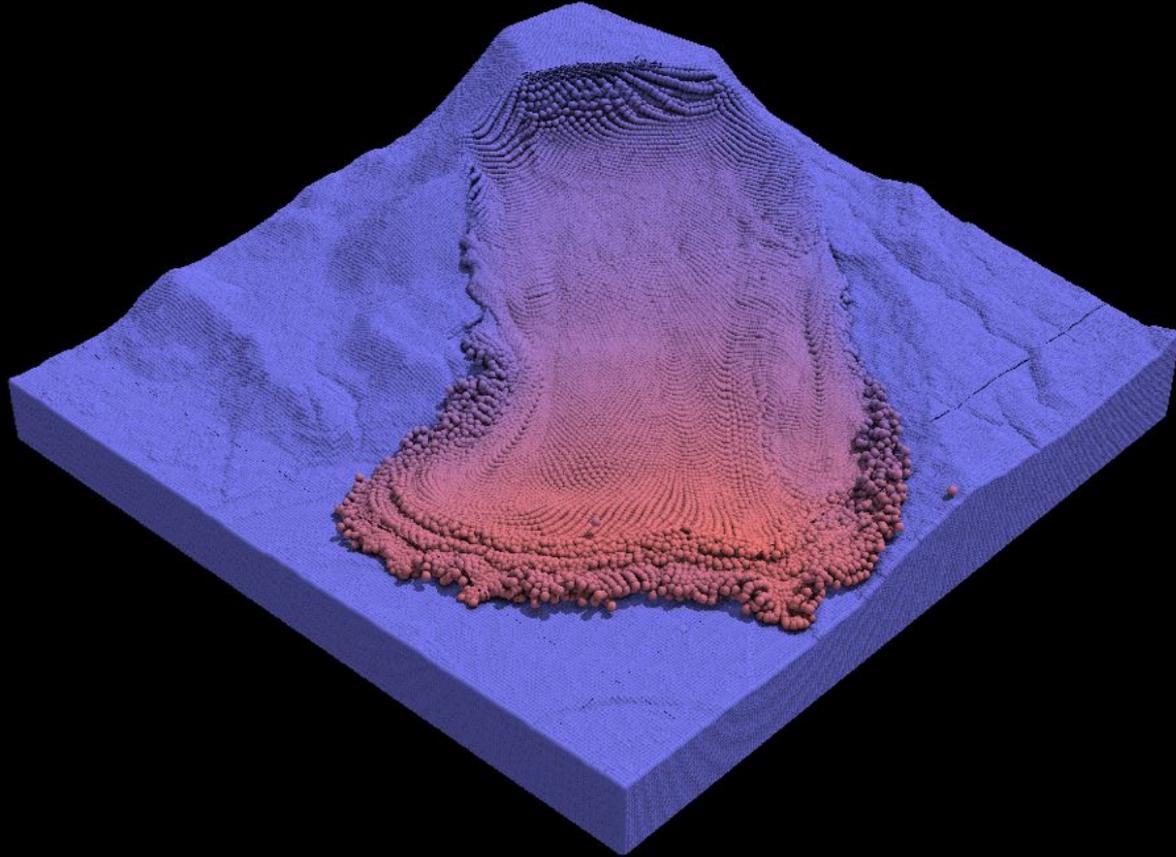
In-situ viz: Workflow integration



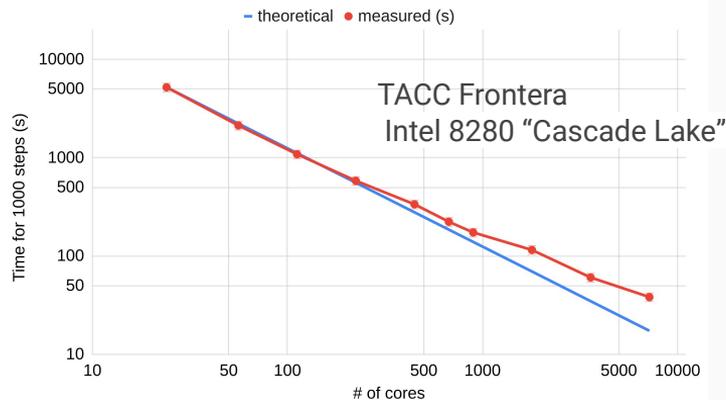
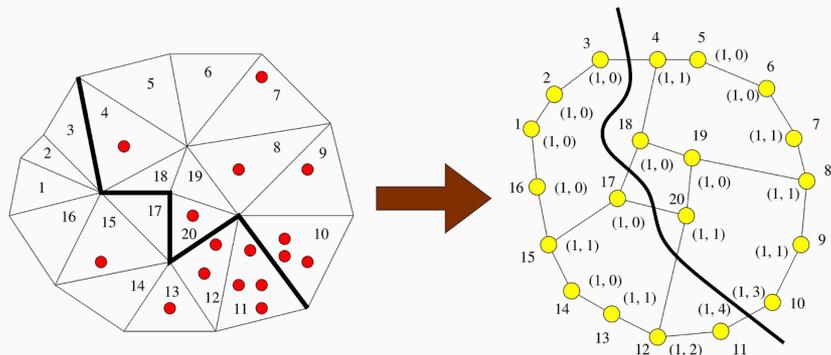
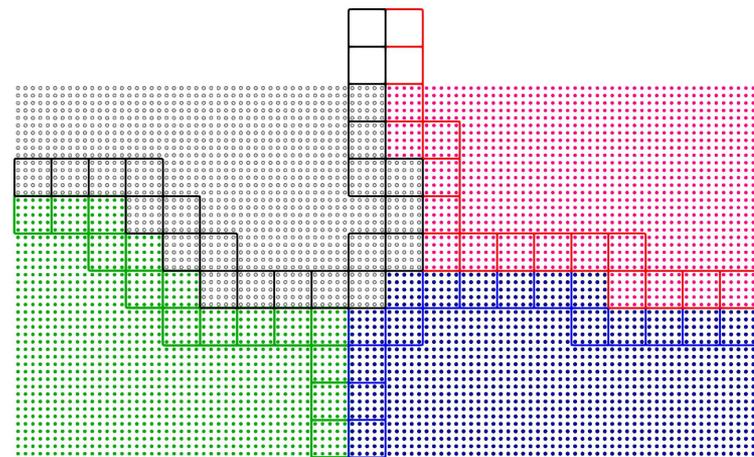
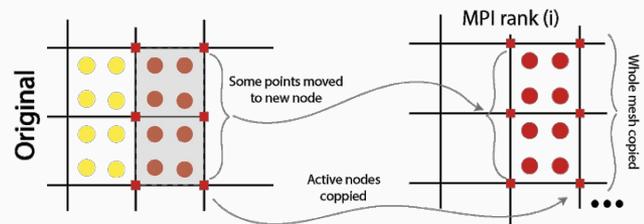
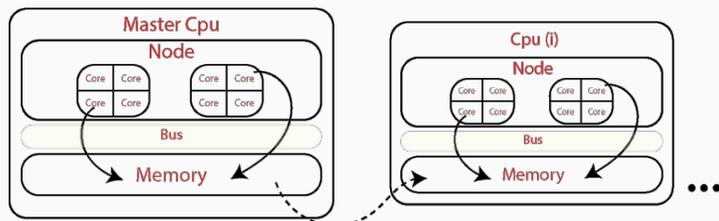
Rendering with Global Illumination



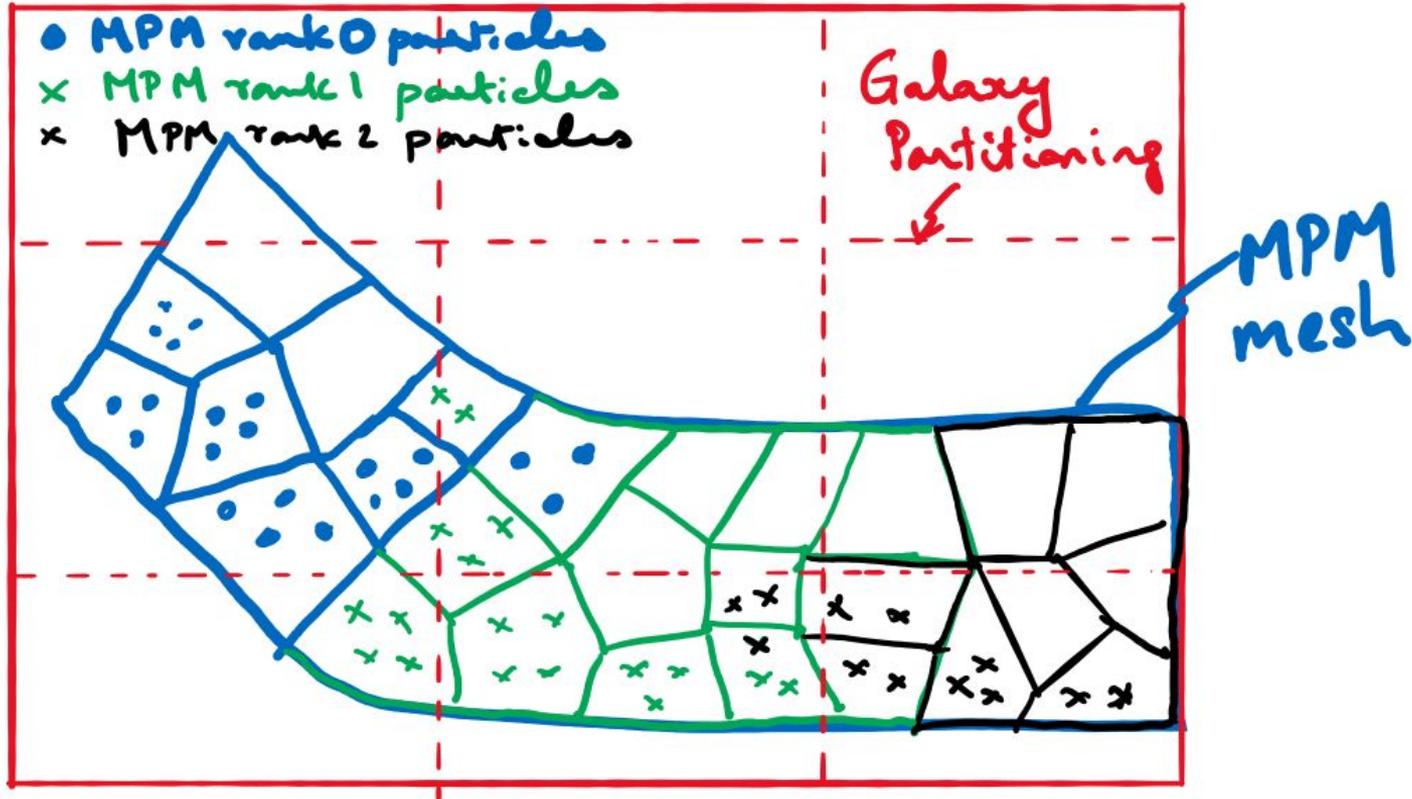
TACC Galaxy and MPM Rendering of Oso Landslide (8 MPI Viz and 16 MPI Compute)



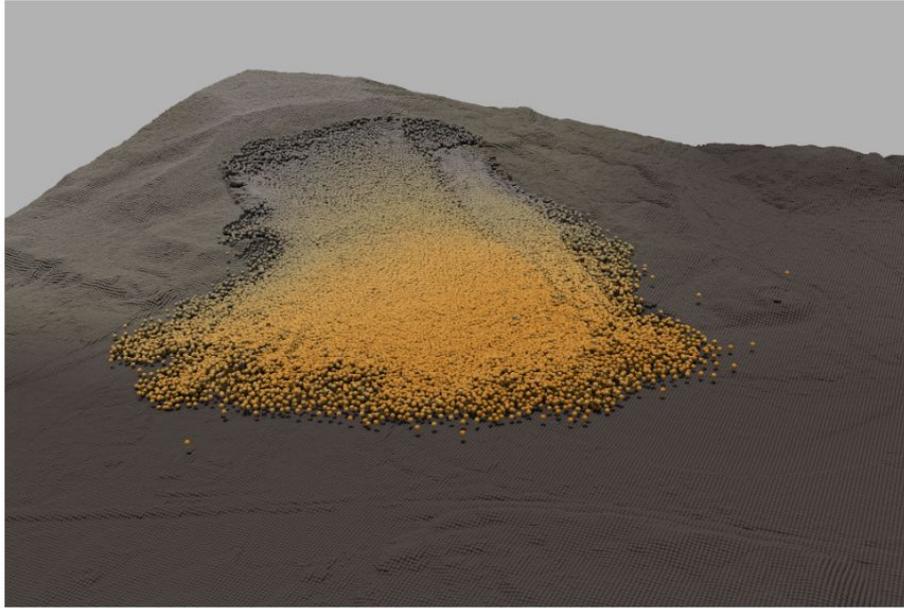
HPC Distributed Simulations: Domain Decomposition



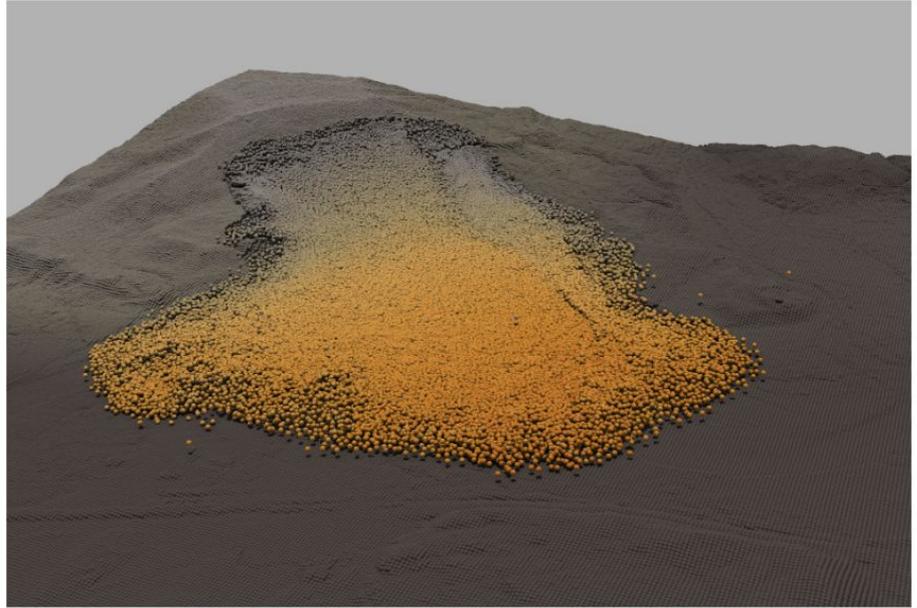
TACC Galaxy: Asynchronous Ray Tracing



Galaxy in situ rendering of the Oso Landslide



(a) runout after 50 s

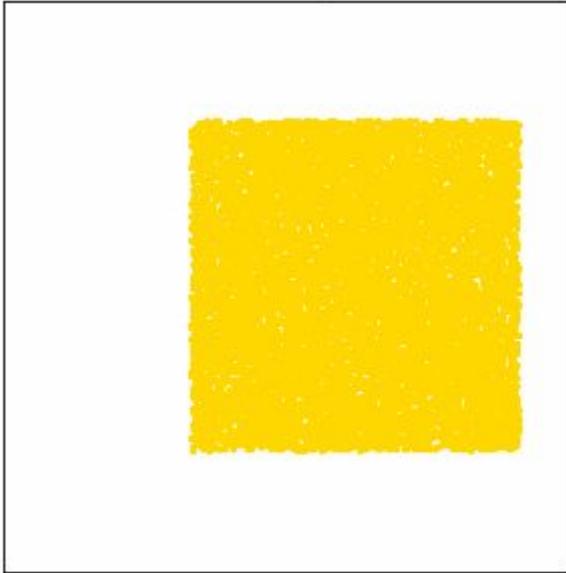


(b) runout after 74 s

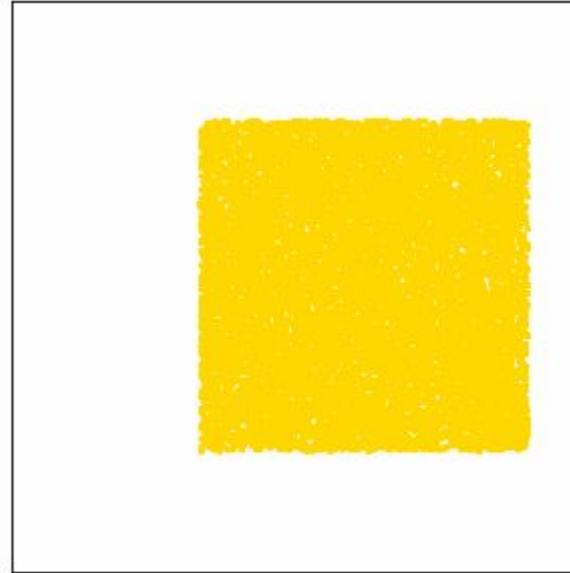
Fig. 5: In situ rendering of MPM simulation with Galaxy. The color gradient shows the amount of particle displacements from their original position.

ML prediction of granular flows

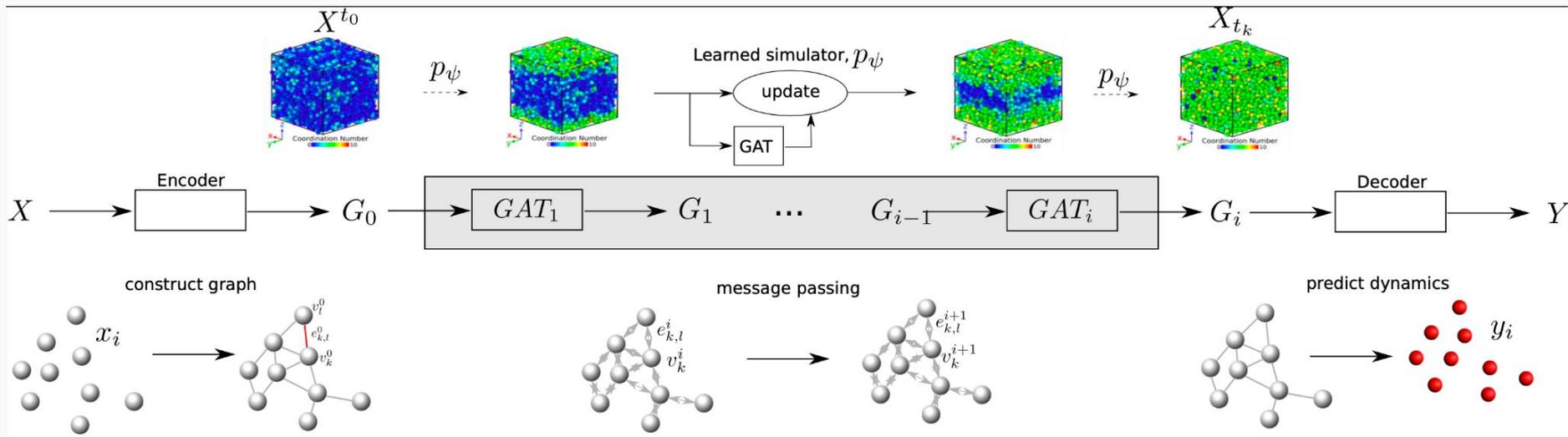
Reality



GNS

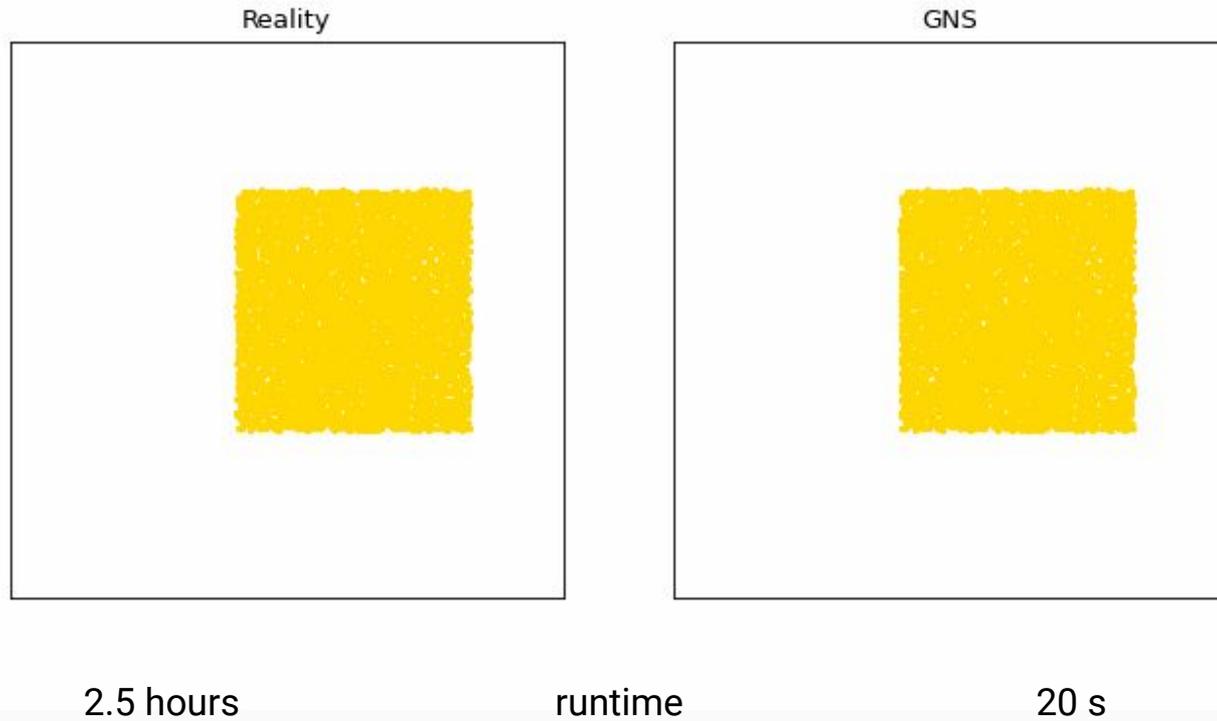


How does it work?

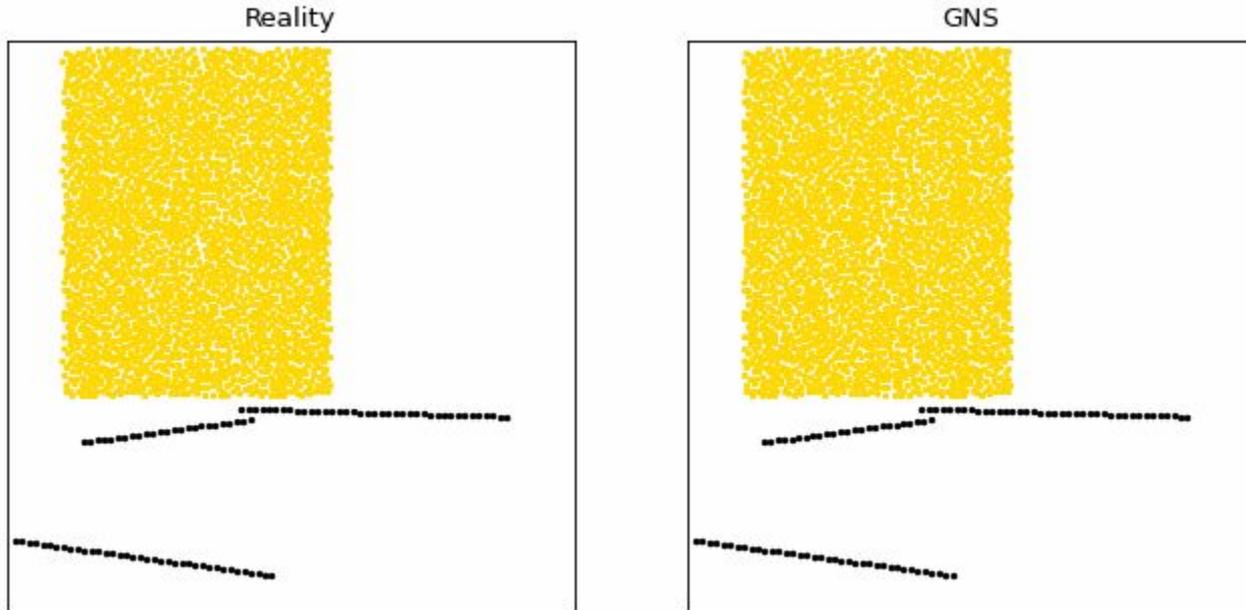


<https://github.com/geoelements/gns>

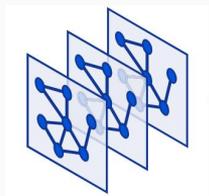
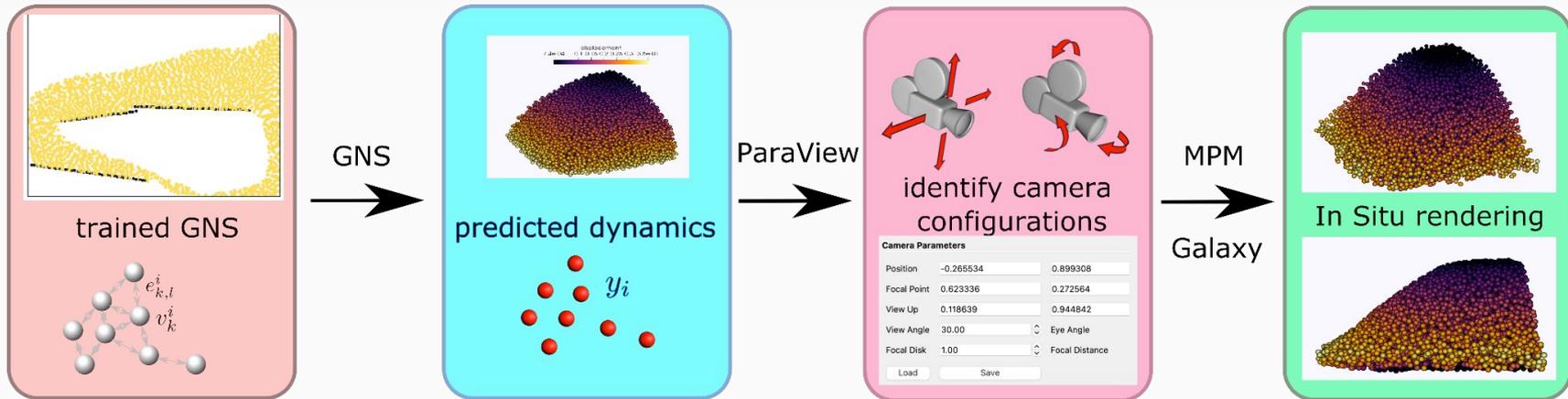
GNS predictions in 3D



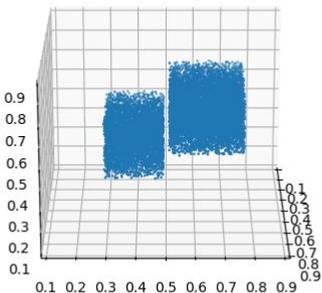
GNS prediction of granular flows with obstacles



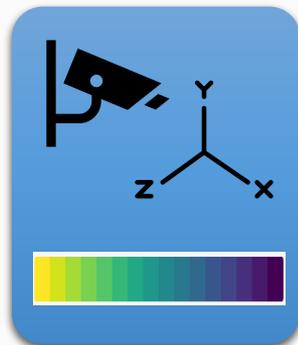
A machine learning oracle for in situ viz



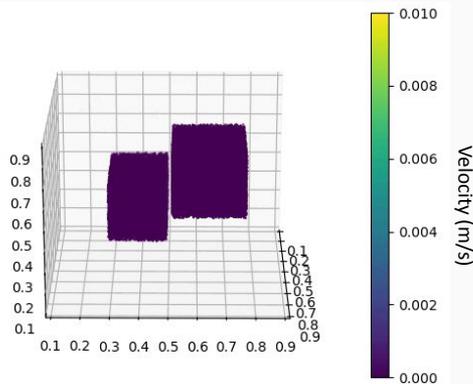
GNS



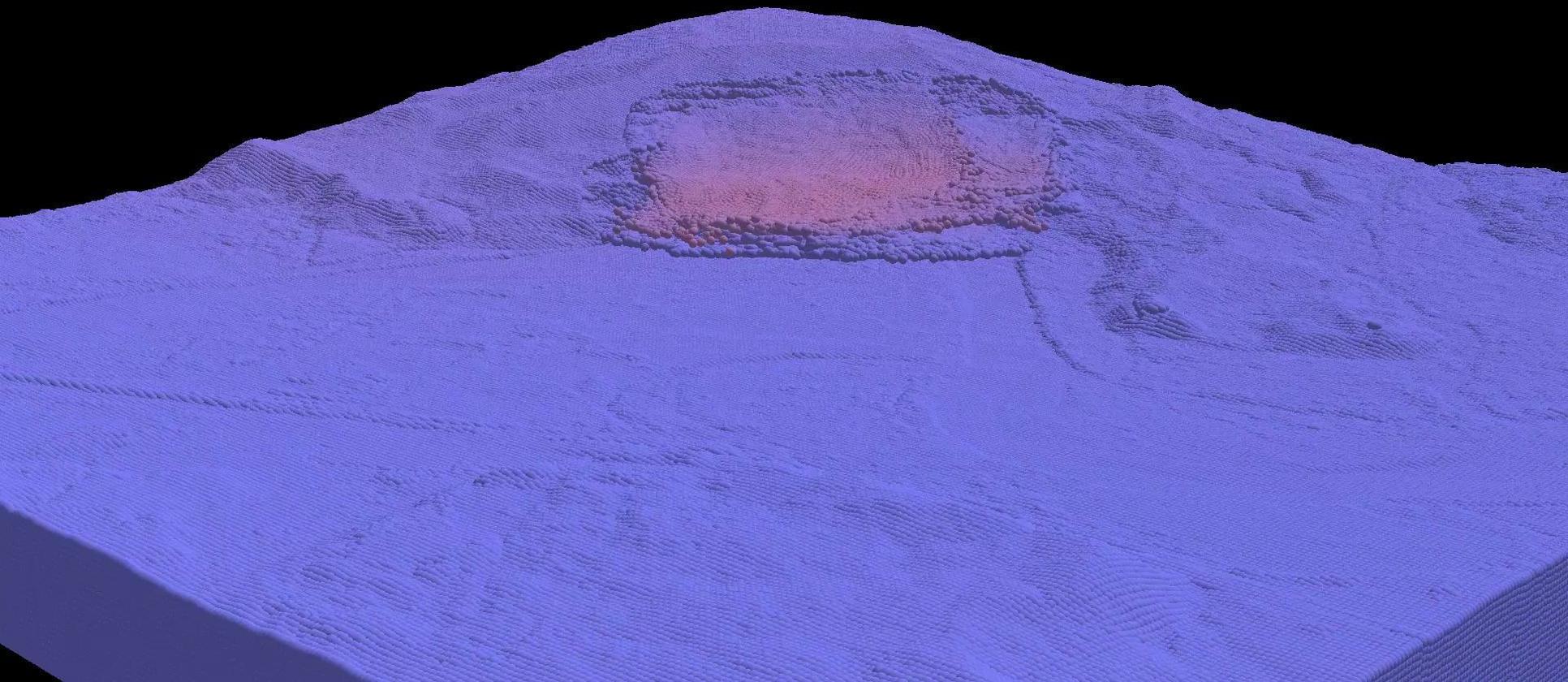
Predict dynamics



Predetermine
visualization parameters



In-situ visualization



Galaxy + MPM: In situ workload

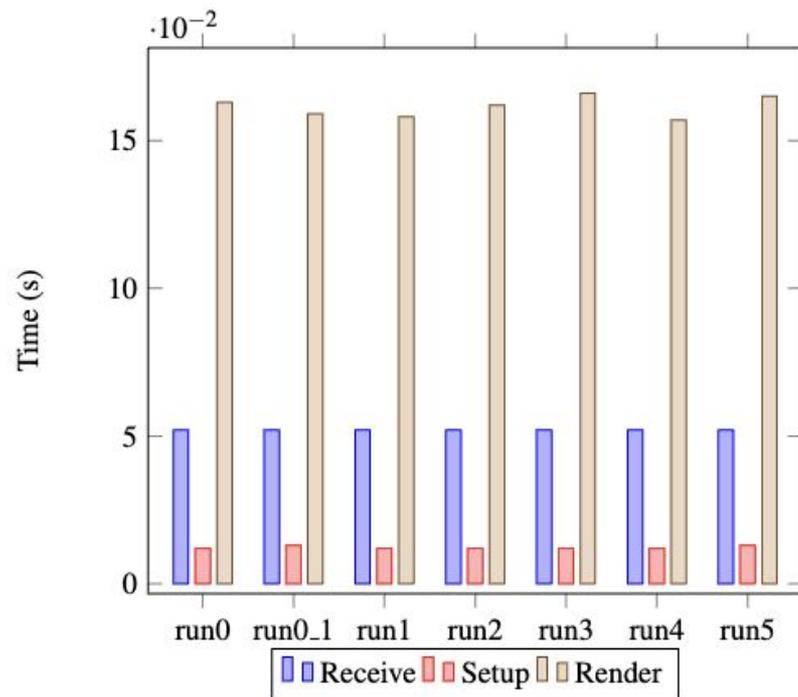


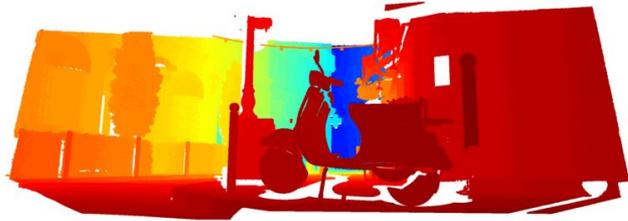
Figure 5: Graph of time for each in situ step. The breakdown shows the time spent for: Galaxy receiving data, setting up for rendering, and the rendering time for 15k particles simulation.

LLMs and
Differentiable
Rendering

Digital Twins

Simulating sensors

- LIDAR data (point clouds are easier to aggregate together to construct a single scene from multiple sensors).
- We can simulate different camera angles and camera settings and environmental conditions



Raytracing simulation of sensors

Obstacle Example



Lower light →



Lower focus →



Ray tracing different lighting conditions and object detection

CAR 0.87



CAR 0.83



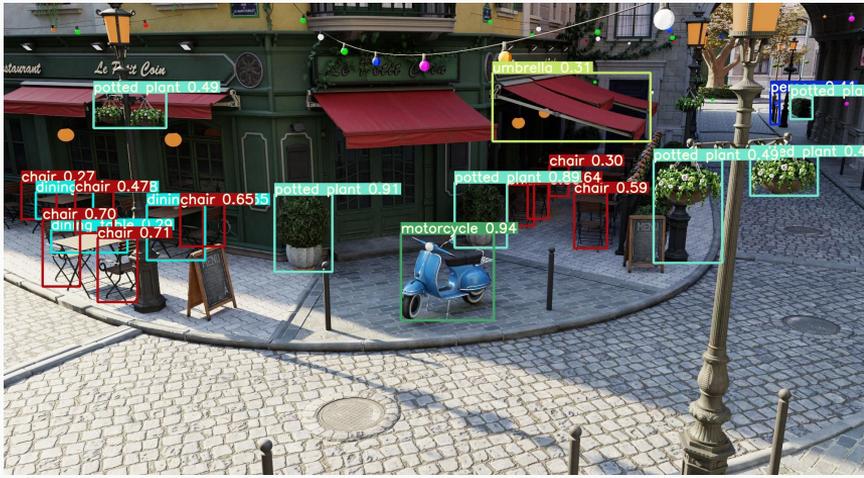
CAR 0.90



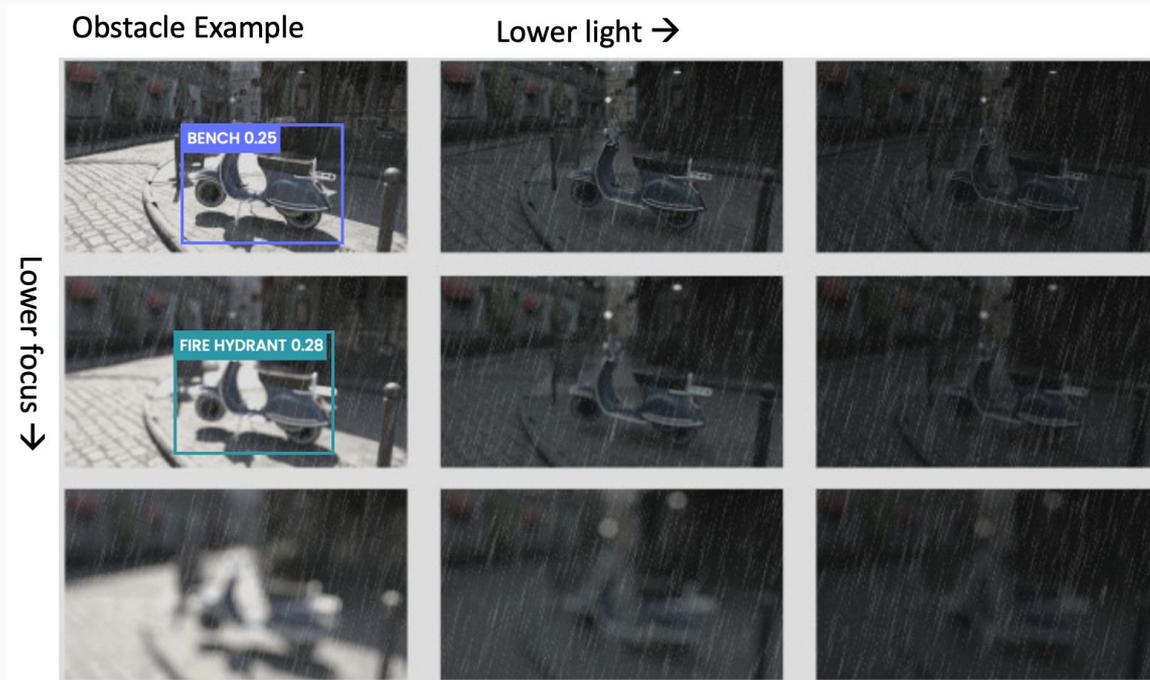
CAR 0.80



Ray tracing for multi-modal sensing - Effect of lighting conditions



YOLO v7 fails to detect objects in adverse weather conds



SoTA CNN object detection fails in adverse weather conditions

Object detection in different socio-economic backgrounds



Upscale neighborhood



Low-income neighborhood

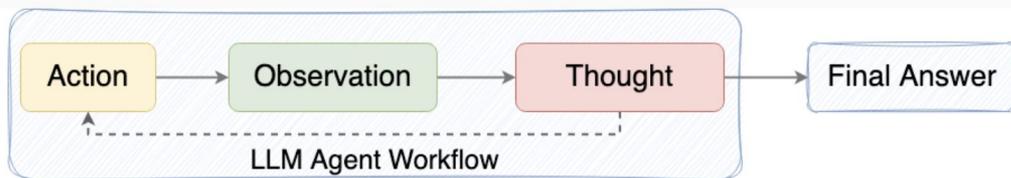
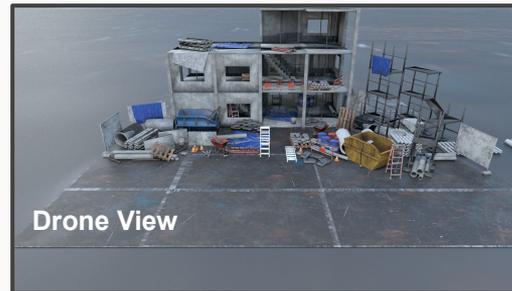
Constructing a Digital Twin

What if I want to tell the drone to go explore the construction site?

"Find the red ladder in the construction site"

"Explore the inside of the structure"

"Complete a loop around the perimeter of the construction site and build a 3D digital model"



How does GPT work?

The geotechnical engineer discovered that the site with dry collapsible soil in arid Arizona _____.

was not suitable for construction

was = 76.71%
is = 6.40%
has = 5.28%
had = 5.00%
contained = 2.67%

not = 24.05%
unstable = 10.56%
highly = 8.36%
composed = 7.48%
too = 6.89%

suitable = 84.58%
stable = 7.63%
well = 1.05%
very = 0.98%
able = 0.79%

for = 99.59%
to = 0.4%
as = 0.00%
foundation = 0.00%
enough = 0.00%

construction = 71.2%
project = 9.70%
building = 6.23%
structure = 5.3%
develop = 3.94%

(a) temperature = 0

How does GPT become creative?

The geotechnical engineer discovered that the site with dry collapsible soil in arid Arizona _____.

was highly saturated with water

was = 77.67%

is = 6.40%

has = 5.10%

had = 4.88%

contained = 2.63%

not = 23.92%

unstable = 10.58%

highly = 8.54%

compose = 7.48%

too = 6.93%

compress = 31.96 %

saturated = 22.10 %

unstable = 12.32 %

perme = 9.56 %

compact = 6.22 %

with = 53.4 %

and = 31.39 %

<end> = 14.46%

, = 0.43%

due = 0.14%

water = 77.03%

groundwater = 17%

moisture = 4.36%

ground = 0.63 %

clay = 0.19 %

(b) temperature = 1

LLM Agents

- LLM is framed as the decision maker (agent) given some parameters
 - **Planning:** agent breaks down large tasks into smaller subtasks, can also refine tasks as it observes issues
 - **Memory:** agent recalls previous information to make better decisions
 - **Tool use:** agent can be given access and information about external tools

LLM Agent Example - Calculator (No Agent)

✦ GPT-4

S what is $(4.5 \times 2.1)^{2.2}$?



Let's break down the calculation step by step:

First, multiply 4.5 and 2.1:

$$4.5 \times 2.1 = 9.45$$

Next, raise the result to the power of 2.2:

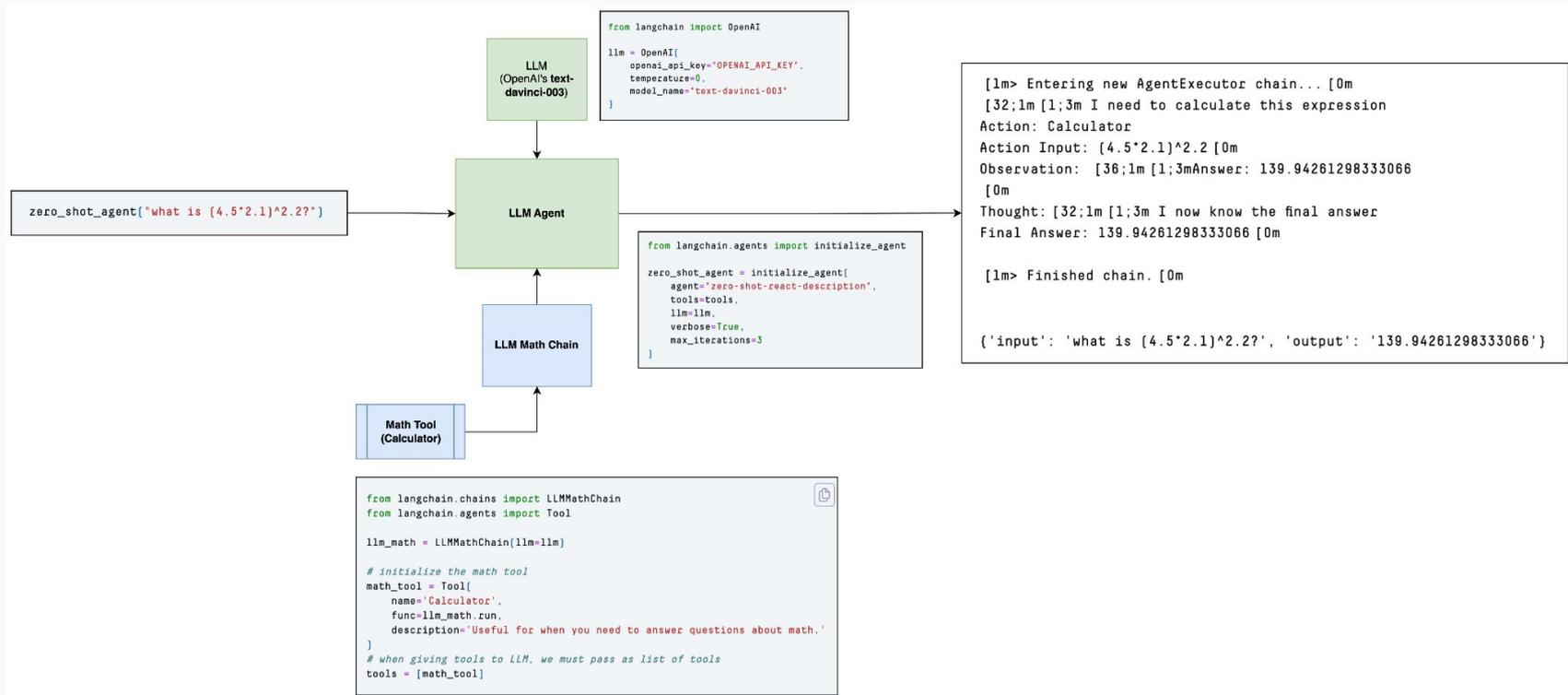
$$9.45^{2.2}$$

To compute this, it's a bit more complex and requires more precision than manual calculations would typically allow. Most would use a calculator or software to determine this value. Let's estimate the value:

Using a calculator, $9.45^{2.2} \approx 104.3123$ (rounded to four decimal places).

So, $(4.5 \times 2.1)^{2.2}$ is approximately 104.3123.

LLM Agent Example - Calculator (Agent)



Autonomous System Experiments

- Simulated experiments using 3D computer graphics scene
 - Using Mitsuba 3: computer graphics library that simulates light realistically
- 3D construction scene for experimentation, this will simulate the *complex environment*

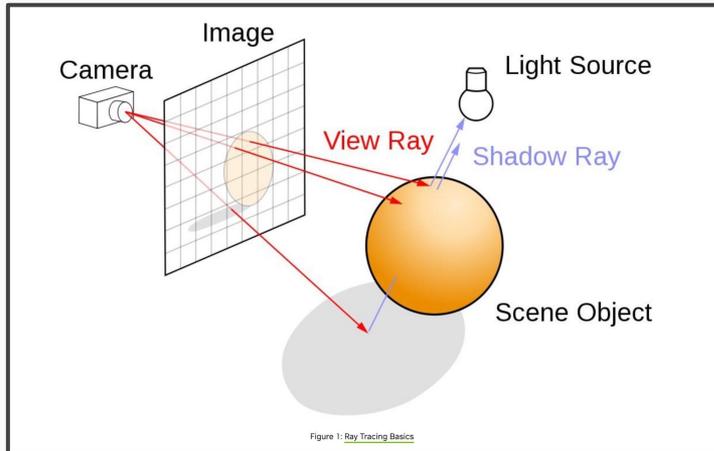


Autonomous Drone Exploration

- Features/requirements of Autonomous System:
 - **Vision System** - understand 2D
 - **LiDAR System** - understand 3D
 - **Localization & Mapping** - understand location and world
 - **Motion Planning** - decide where to move
 - **Interface** with humans to take new directions (LLM Agent)

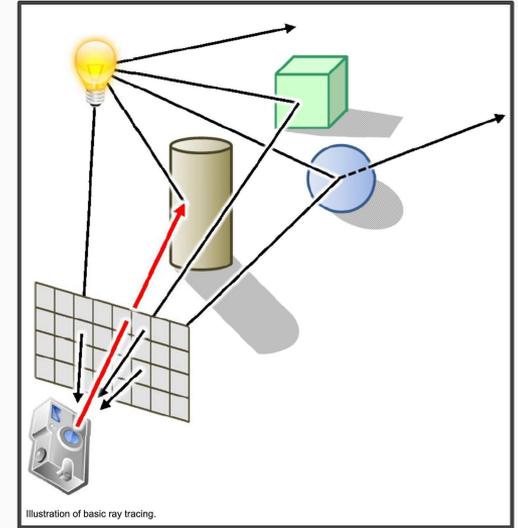
Vision System

- Render images from the drone camera view at fixed intervals
- This generates live video view when enough intervals are rendered
- Movement of camera view / drone object provides exploration capabilities



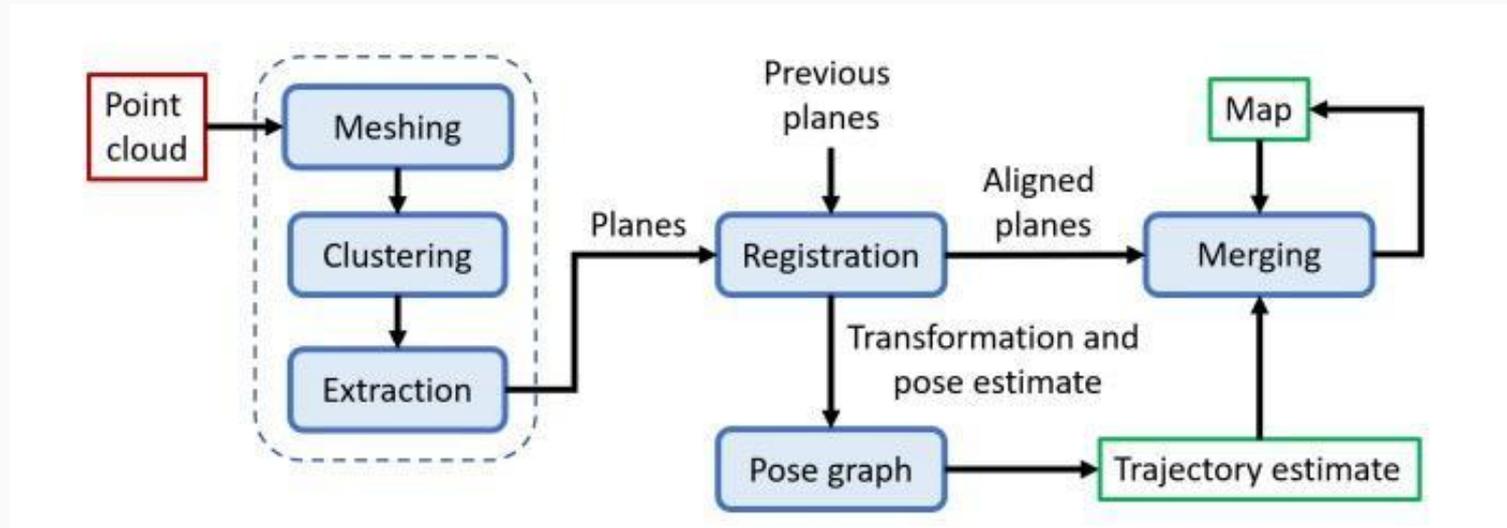
LiDAR System

- Simulate LiDAR data through rendering process
- Rendering our scene from a specific camera location allows us to record all the intersections of light vectors (ray tracing) sent out from the camera location with objects in the scene on first hit
- Remove roughly 90% of the points for computational efficiency



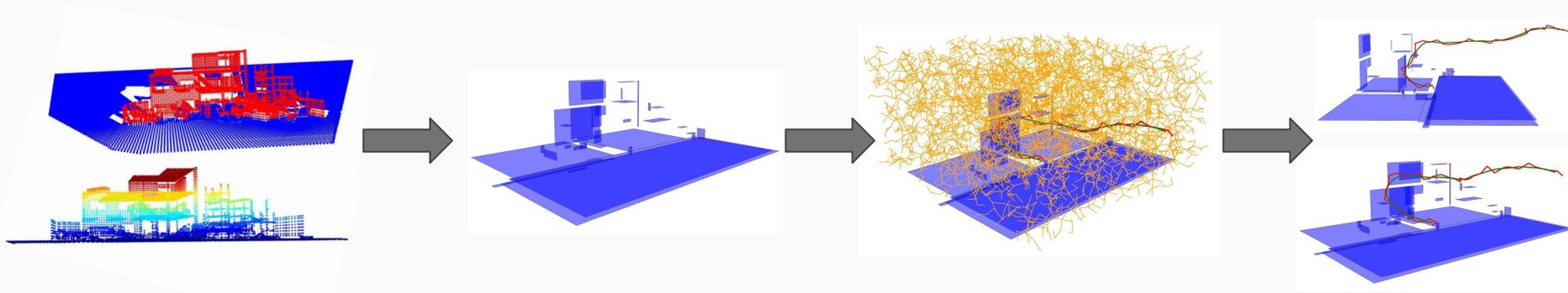
Localization & Mapping: PlaneSLAM

- 3D LiDAR SLAM algorithm which extracts planar features from point clouds to achieve lightweight, real-time localization and mapping



Motion Planning: Rapidly exploring random tree (RRT)

- RRT is an algorithm designed to efficiently search nonconvex, high-dimensional spaces by randomly building a space-filling tree



Raw Point Cloud (Generated from Drone location)

Point Cloud converted to Lightweight Plane Objects

RRT Algorithm generates random paths from starting point to destination

Final path is chosen

PlaneSLAM & Motion Planning Integration

- As we can see, the localization and mapping works with motion planning and execution, but how can we expose this system to a user in an intuitive way?
 - Both the starting location and the target location were given to RRT (x, y, z)
 - One solution is to utilize the **vision system**

Vision System & Grounding DINO

- As a user interacting with an autonomous system, I want an easy way to tell the drone where to go and what to do
- **Grounding DINO**
 - Object detection model
 - Combined model that has deep understanding of text and visual content and can relate the two
 - Grounding DINO excels at detecting objects even when they are not part of the predefined set of classes in the training data
 - Referring Expression Comprehension (REC) - Identifying and localizing a specific object or region within an image is possible by giving a textual description



Grounding DINO



"red box above the door"

Grounding DINO



"wine glass on the far left"

Grounding DINO Prompt Engineering



"man with sunglasses"



"man with sunglasses in blue shirt"

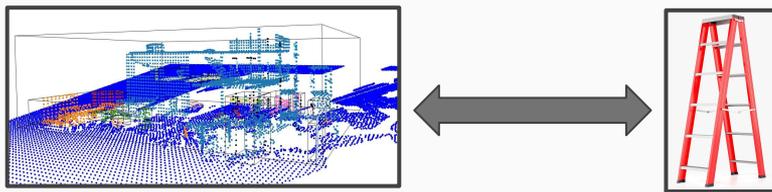
2D to 3D: Make-It-3D

- We can use Grounding DINO to establish object locations within an image in 2D, however, we need to find the 3D location with respect to the current location
- Solution:
 - Use **Make-It-3D** to convert image to a 3D mesh to do a similarity search

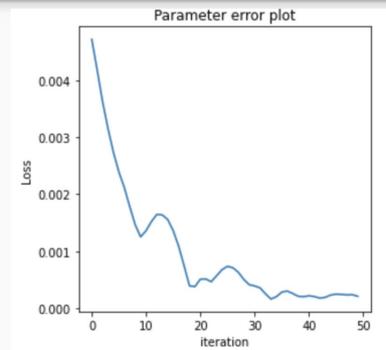
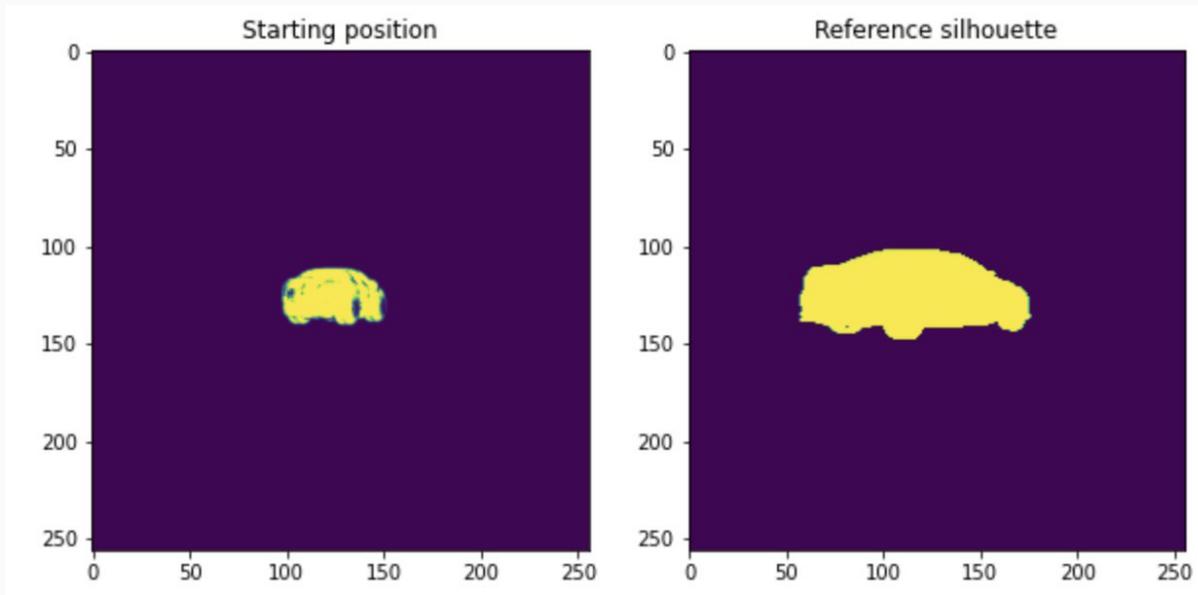


2D to 3D: Similarity Search

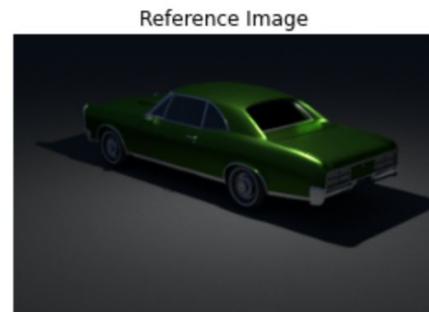
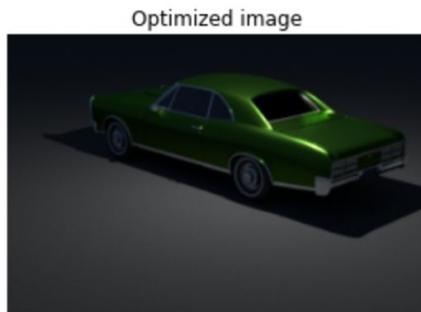
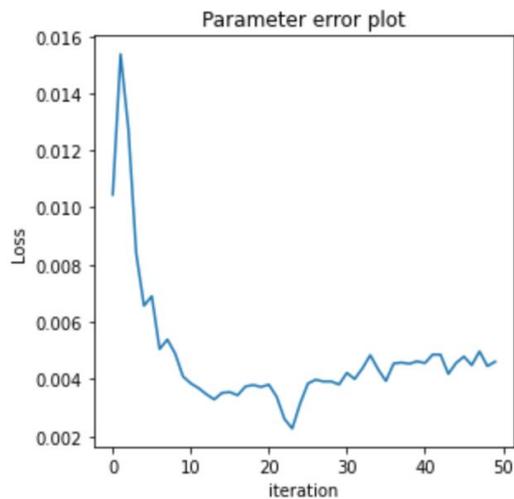
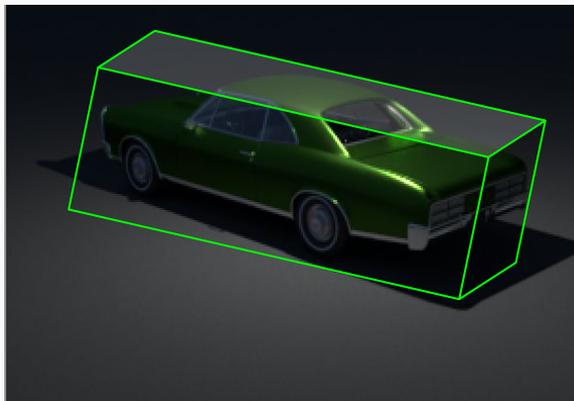
- We can use DBSCAN to perform clustering on generated LiDAR and then create bounding boxes around the unique clusters
- Then, we can do similarity comparison between our target object and the established bounding boxes



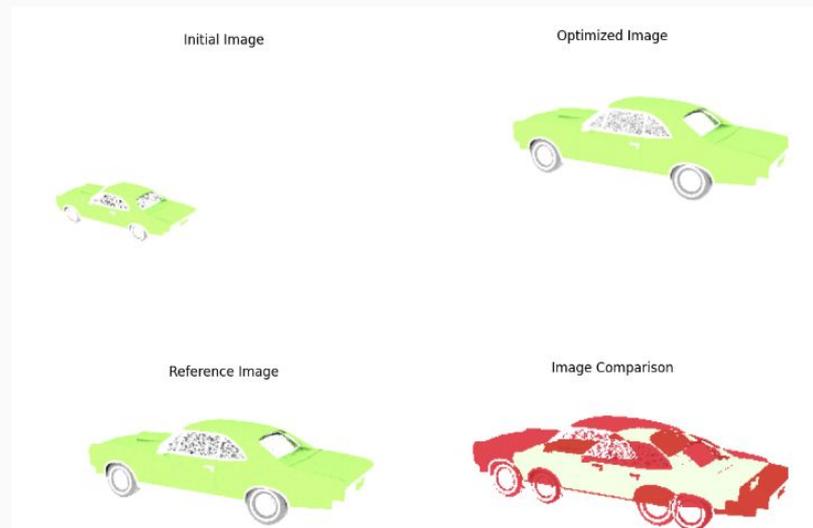
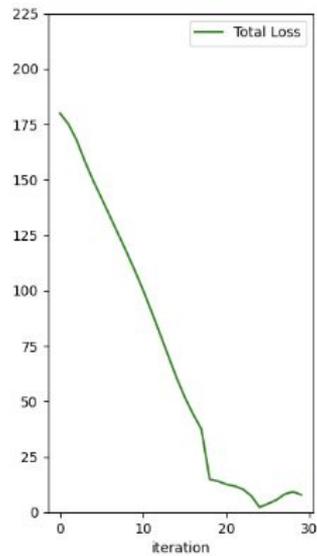
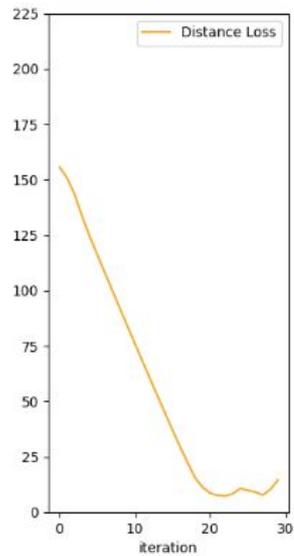
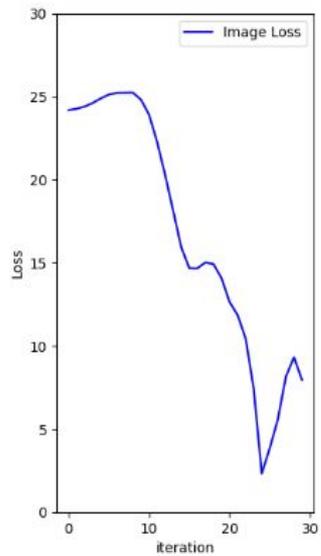
Differentiable rendering using mesh rendering



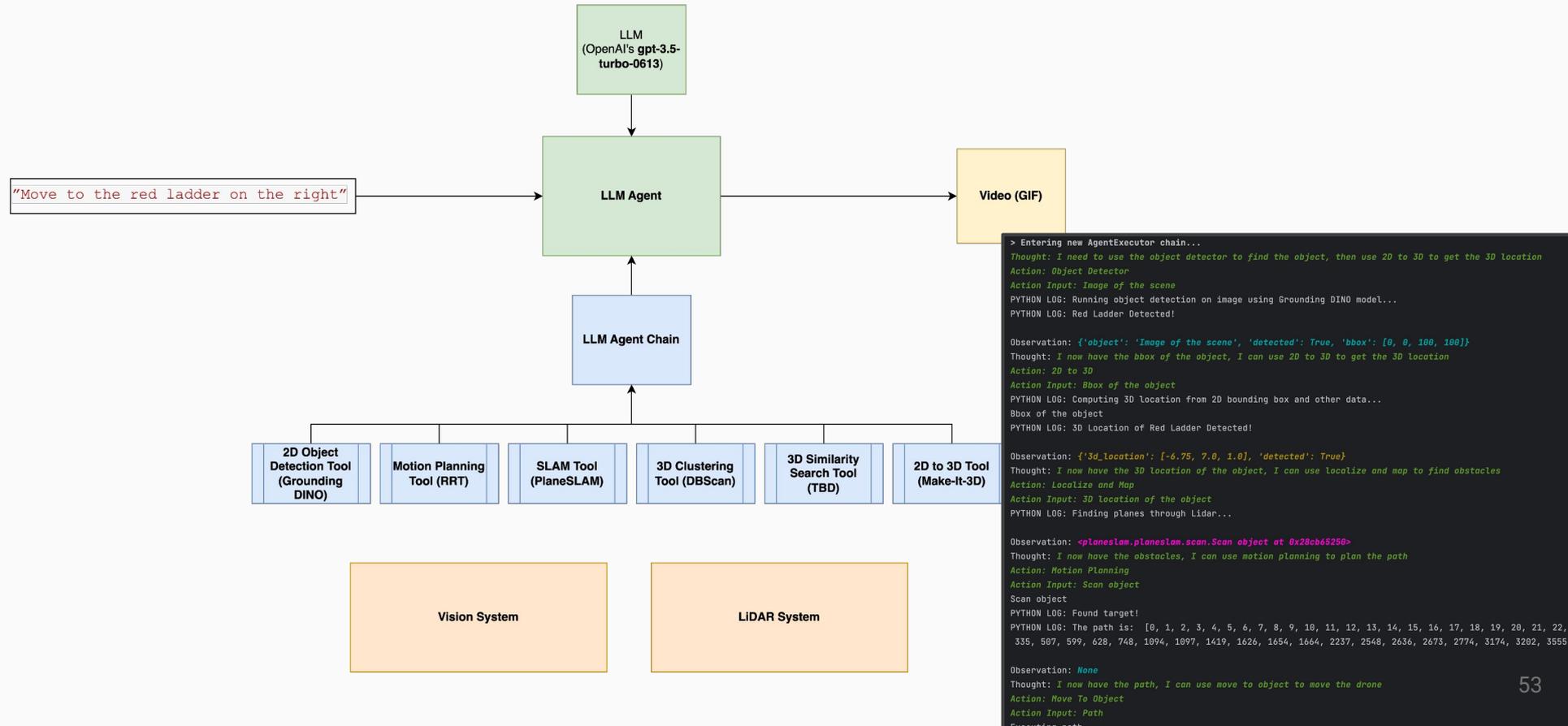
Differentiable rendering with raytracing depth and texture (3D bounding box)



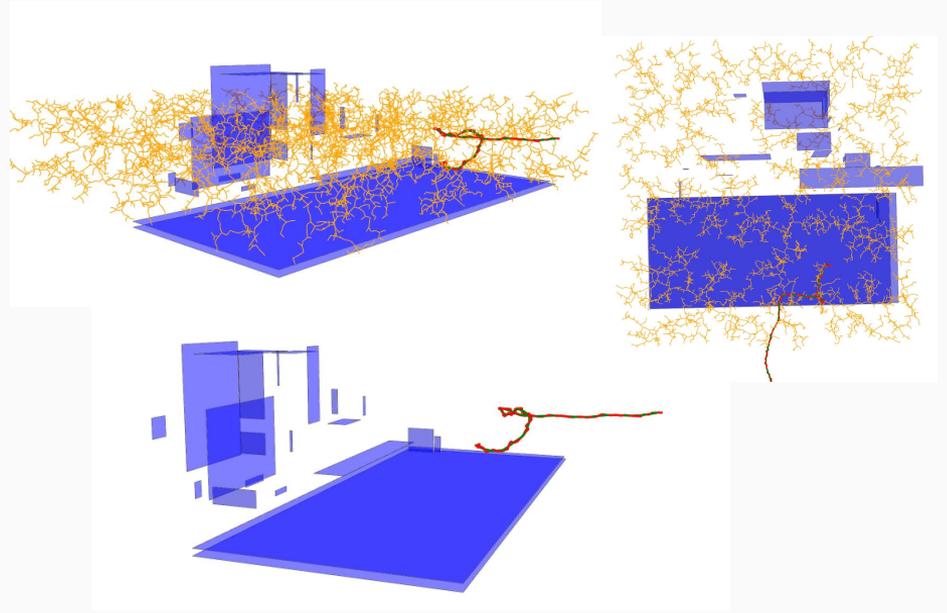
Two-stage loss: Image + Distance



LLM Agent Autonomous Drone System Architecture



LLM Agent Autonomous Drone System Demo



Controlling Robots/Drones with LLMs



Hook 'em Horns!

krishnak@utexas.edu
<https://geoelements.org>