# VistaVisio

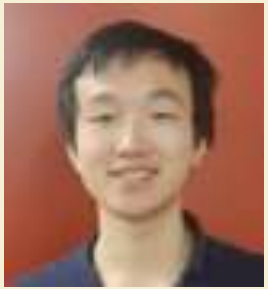
# CE Team

Ari Jacobs - Team Lead

Saul Diaz

Guntash Gill

Kevin Hernandez



Jeffery Wang

# EE Team

Preston Tran - Team Lead

Jason Li

Nathan Nhieu

Joshua Liu

# Table of contents

# Overview

# Problem

- No good interactive viewer for campus!

# Similar Products


Google Maps

- No street-view!

GoGaucho

- Has classroom search feature
- Lacks visual context!

# Our Project

- Create a virtual and interactive model of the UCSB campus
- Locations in the model correspond with accurate GPS world coordinates

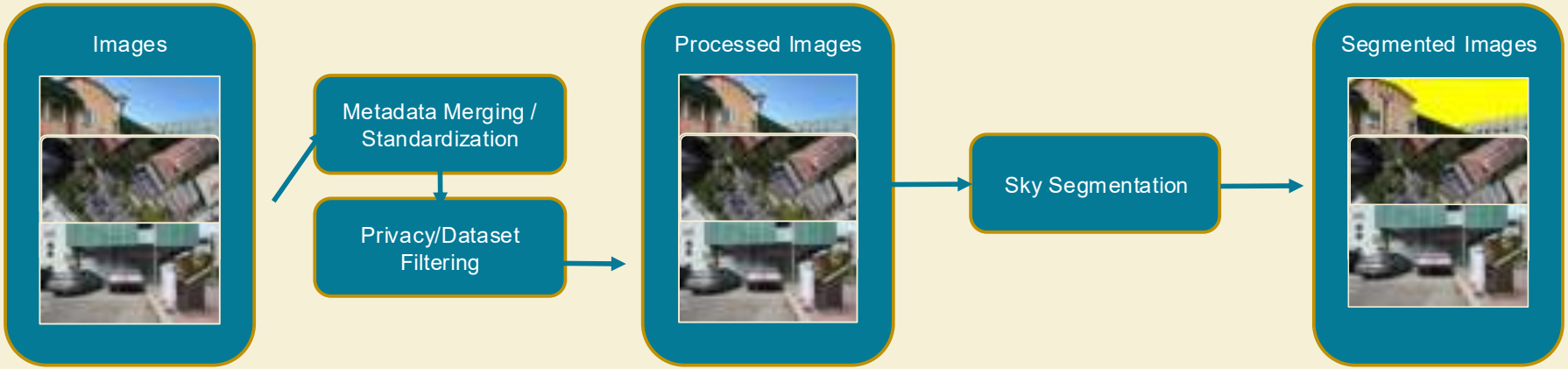
# Web Demo - Dense Point Cloud

# VR Demo

**VR Demo time!!!**

# Imaging Pipeline

# Pipeline Overview



# What makes a Good Dataset

- Need high enough sampling rate s.t. multiple photos capture same features
- Encompasses whole building
    - Ground + Aerial shots
- Accurate + Well-suited camera parameters:
    - Focal length, resolution, 3D GPS coordinates

# What makes a Good Dataset

- **Need high enough sampling rate s.t. multiple photos capture same features**
- Encompasses whole building
  - Ground + Aerial shots
- Accurate + Well-suited camera parameters:
  - Focal length, resolution, 3D GPS coordinates

# What makes a Good Dataset

- Need high enough sampling rate s.t. multiple photos capture same features
- **Encompasses whole building**
  - Ground + Aerial shots
- Accurate + Well-suited camera parameters:
  - Focal length, resolution, 3D GPS coordinates





# Image Segmentation

- Sky regions add noise, causing floating points and weaker Gaussian Splats.
- SegFormer classifies sky pixels per image, keeping only those connected to the top edge to filter false positives.
- Masking sky pixels ensures features are extracted only from structures, improving point cloud and splat quality.

# Challenges of Dataset Collection

- Privacy/Ethical usage – No identifying features on people, cars, etc...
- Non-ideal Photos – Worsen reconstruction & cause visual artifacts
- Image Density – Not enough photos = worse reconstruction
- Storage – Takes up terabytes of storage

# Scope of Campus

# Scope of Campus

# Scope of Campus

- Good coverage ✓
- Less transient objects ✓
- Keeps important buildings ✓

# Progress of Imaging

# Region of Campus Completed

Emlid Device

Wingtra Drone

DJI Drone

# Reconstruction Pipeline

# Naive Approach:

- Why not toss all of our images at once?

    1) Computationally inefficient

    1) Purely ground-level and purely drone images will not match even if GPS information says so

    1) Bundle adjustment parameters will be different for size of each building

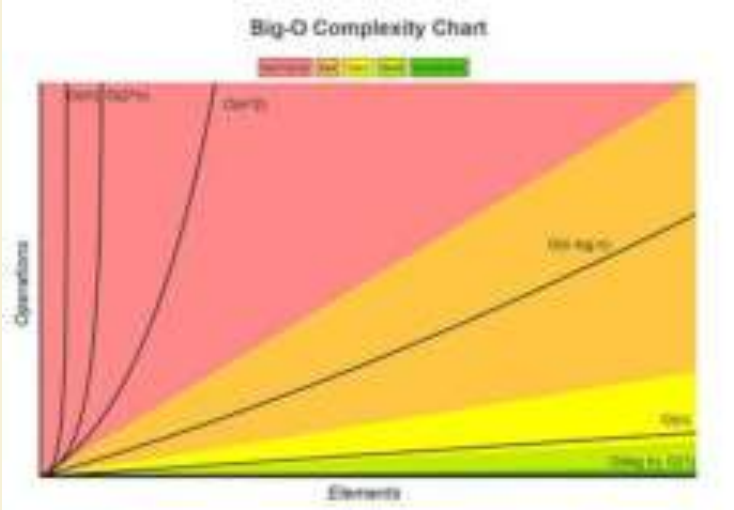
# Naive Approach:

- Why not toss all of our images at once?

    **1) Computationally inefficient**

    1) Purely ground-level and purely drone images will not match even if GPS information says so

    1) Bundle adjustment parameters will be different for size of each building



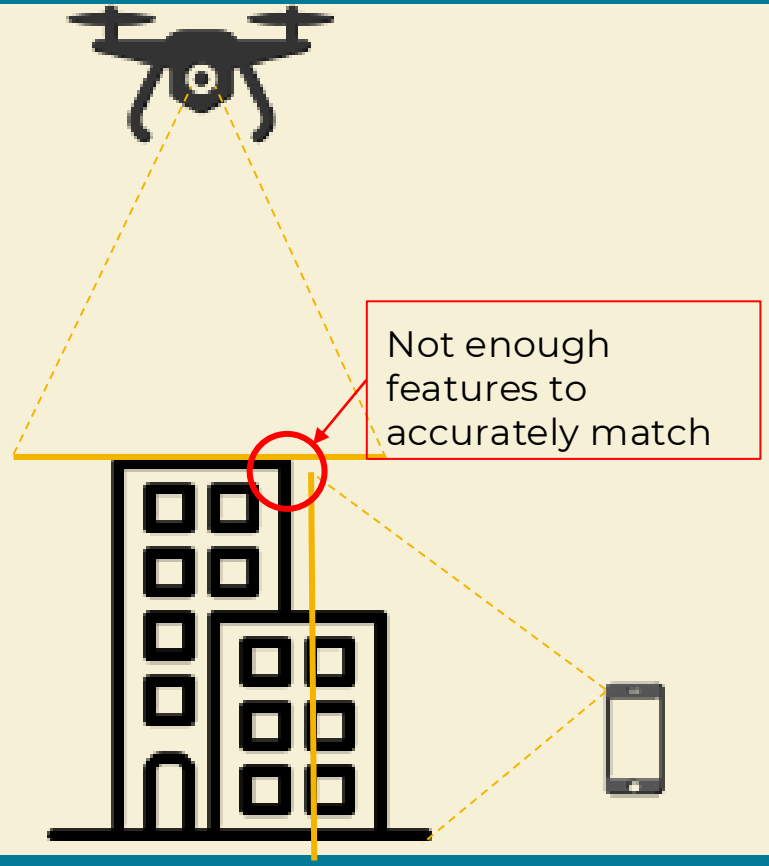
# Naive Approach:

- Why not toss all of our images at once?

  1) Computationally inefficient

  **1) Purely ground-level and purely drone images will not match even if GPS information says so**

  1) Bundle adjustment parameters will be different for size of each building



# Naive Approach:

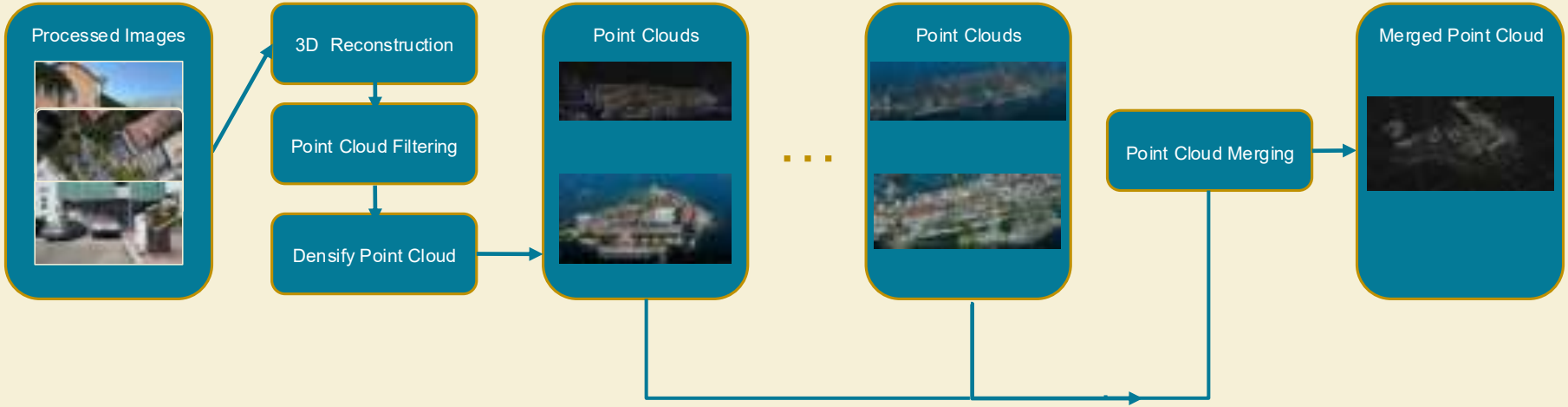
- Why not toss all of our images at once?

    1) Computationally inefficient

    1) Purely ground-level and purely drone images will not match even if GPS information says so

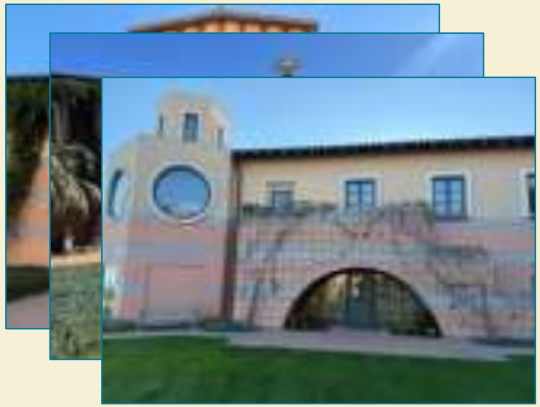
    **1) Suboptimal for hyperparameter search due to buildings of different sizes**
        - ex: Buchanan v.s. Chem/PSNB

# Pipeline Overview

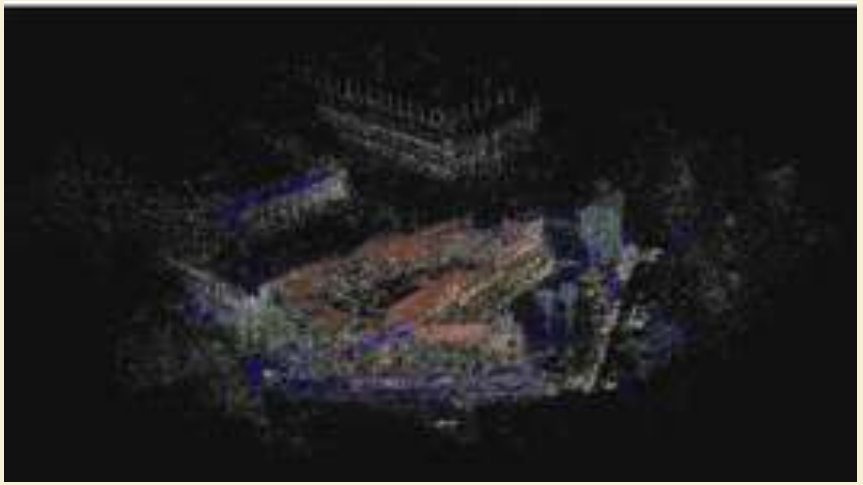


# Captured UCSB Ground Imagery

Ground Images

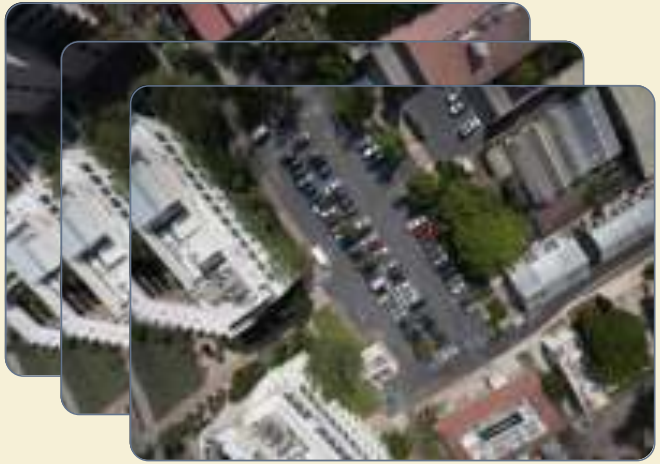


Sparse Point Cloud



# Captured UCSB Aerial Imagery

Drone Images



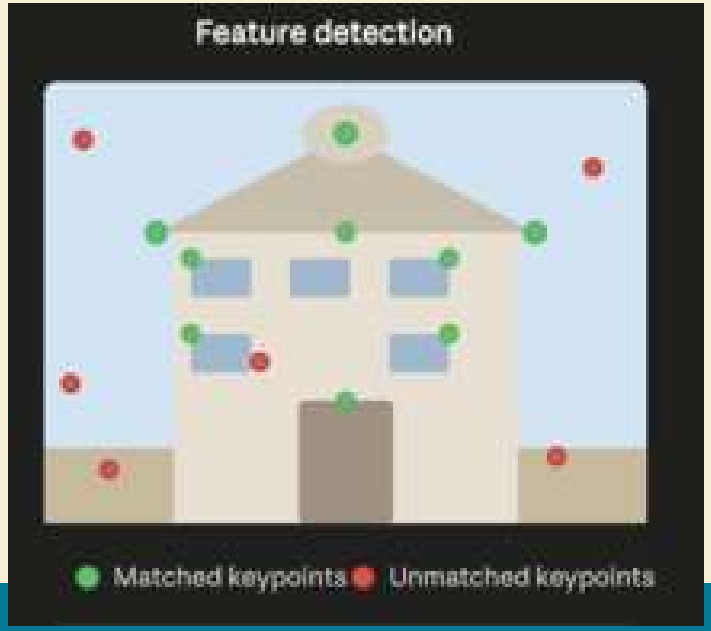
Dense Point Cloud

# How To Get Point Clouds

1. Image Retrieval
    a. Import a large dataset of images
    b. Summarizes the entire image into one representation
    c. Nearest neighbors

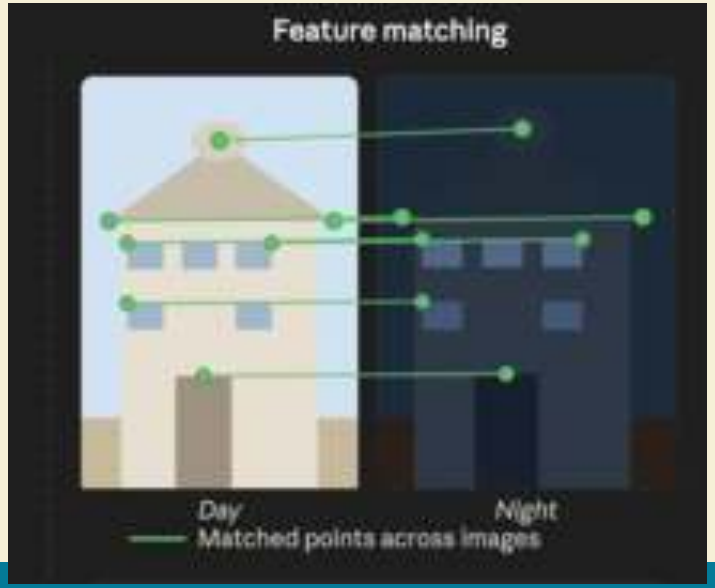
# How To Get Point Clouds

2. Feature Detection and Extraction
    a. Identify key points and features
    b. Local feature descriptors



# How To Get Point Clouds

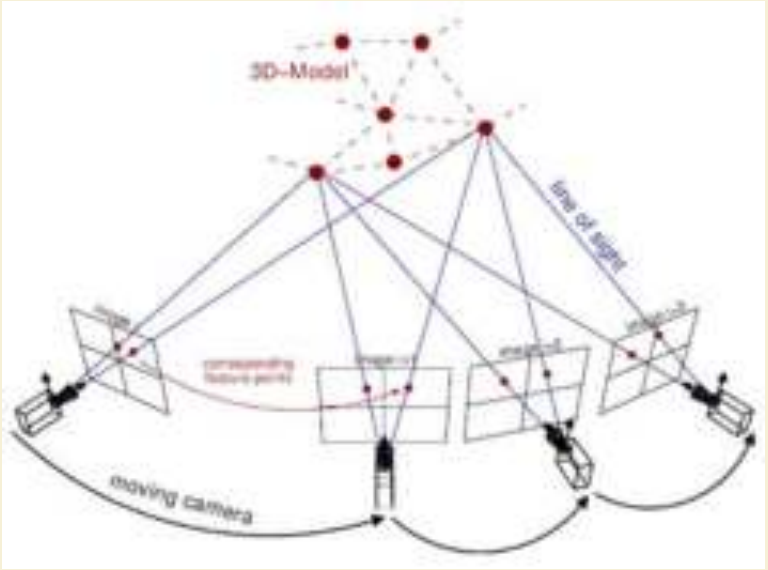
3. Feature Matching
    a. Match/pair key points between images to find correspondences
    b. Geometric verification



# Reconstruction Tools

4. Structure-from-Motion (SfM)
    a. Recovers the camera viewpoints from multiple images
    b. Creates an initial **sparse** point cloud of the scene

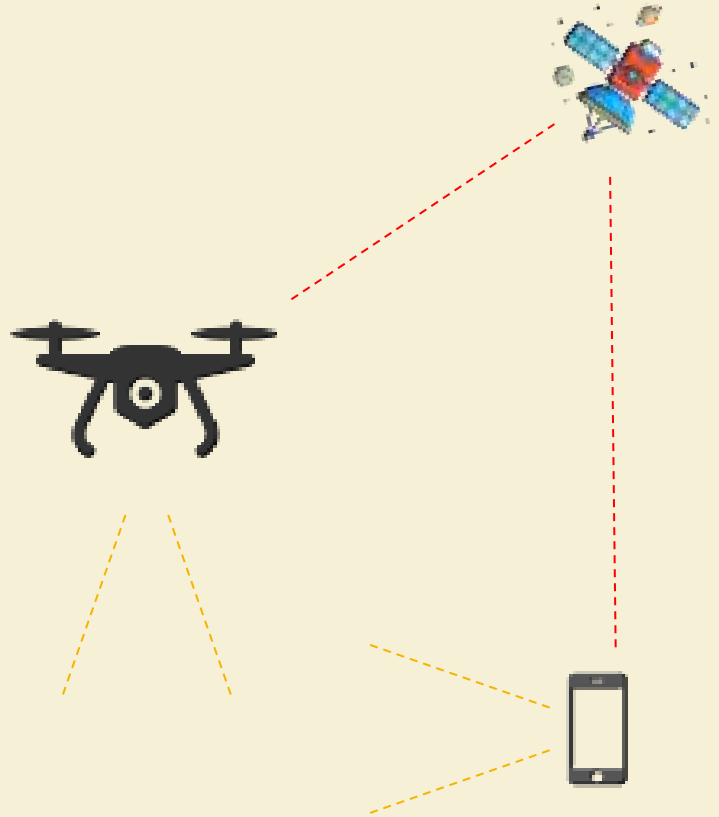
5. Multi-view Stereo (MVS)
    a. Creates **dense** point cloud from sparse reconstruction
    b. Millions of points instead of thousands Important for improving render.




3D reconstruction processed via COLMAP (Schönberger & Frahm, 2016).

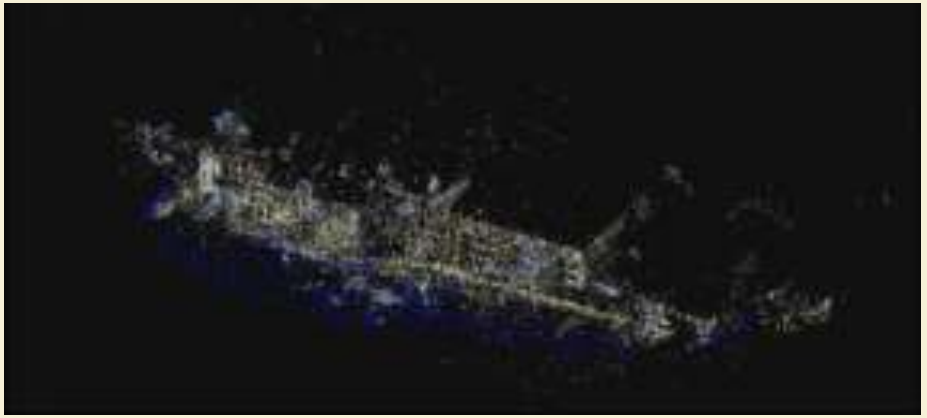
# How To Make Accurate Point Clouds

- One issue of implementing both drone and ground imagery is that some locations will have weak overlap in the 2D images gathered by both
- Solution: Give the computer a “starting guess” prior to mapping based on GPS data



# Point Clouds

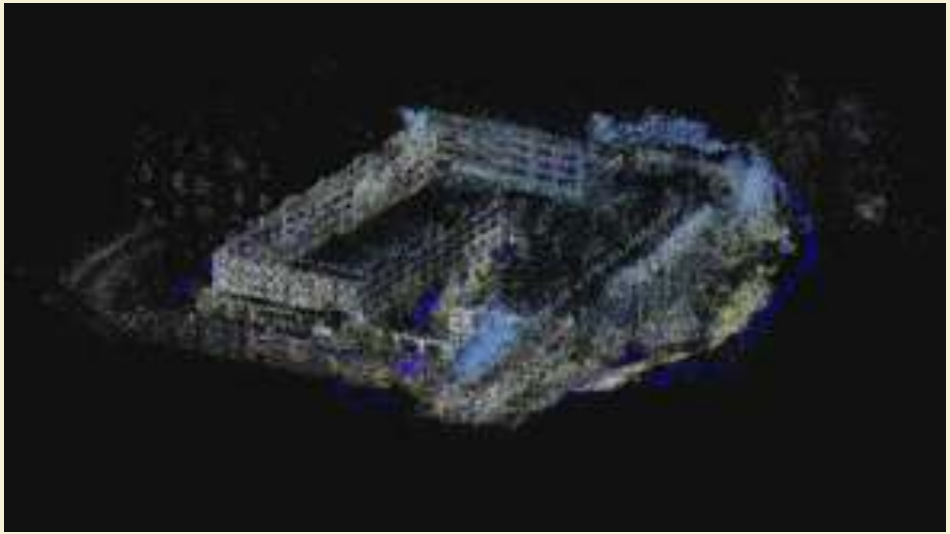
- Collection of 3D points combine to create a “scaffold” of a real world scene
- Made from images with overlapping details
- 3D ‘skeleton’ of scenes where each point has its own position.
- Blue triangles represent where the camera location is



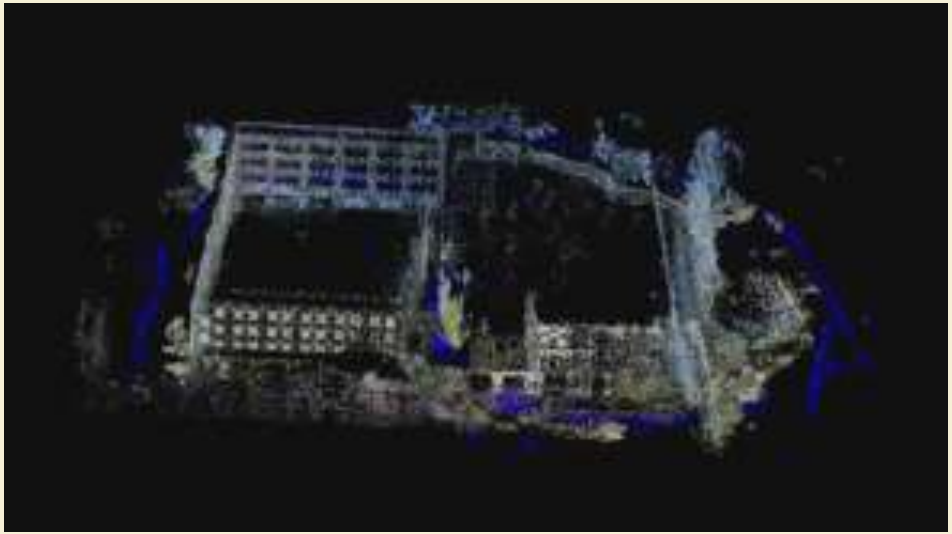
**Chemistry Building**

# Point Cloud Filtering

Before



After



# Point Cloud Merging

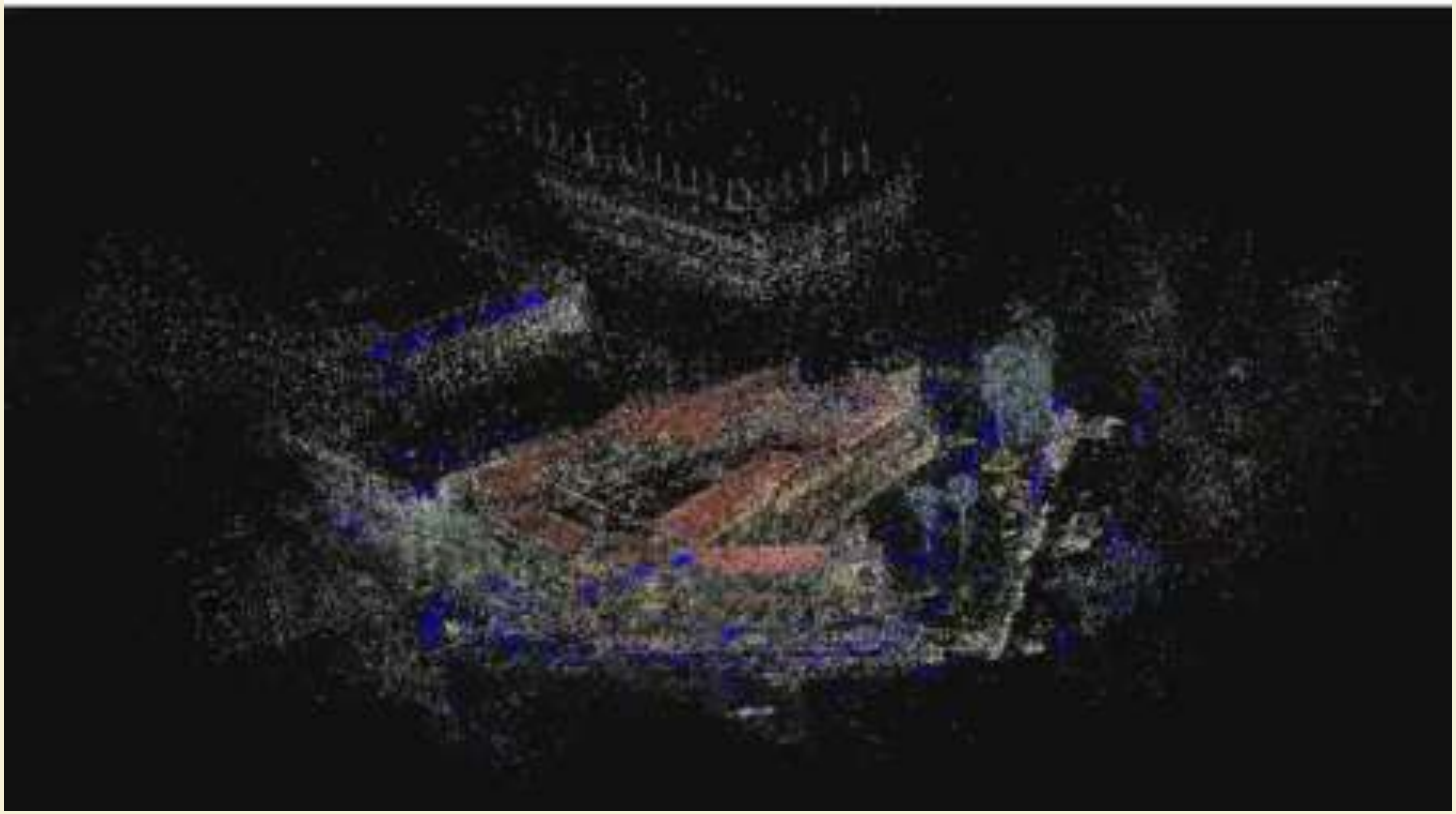
- Are we just going to make a single point cloud all at once? **NO**

- We merge point clouds based on their gps data
  - Accurately place them onto their global coordinate system along with other reconstructions

- This allows us to incrementally build the campus
- We find the optimal scale, rotation and translation and apply it to all our 3D points

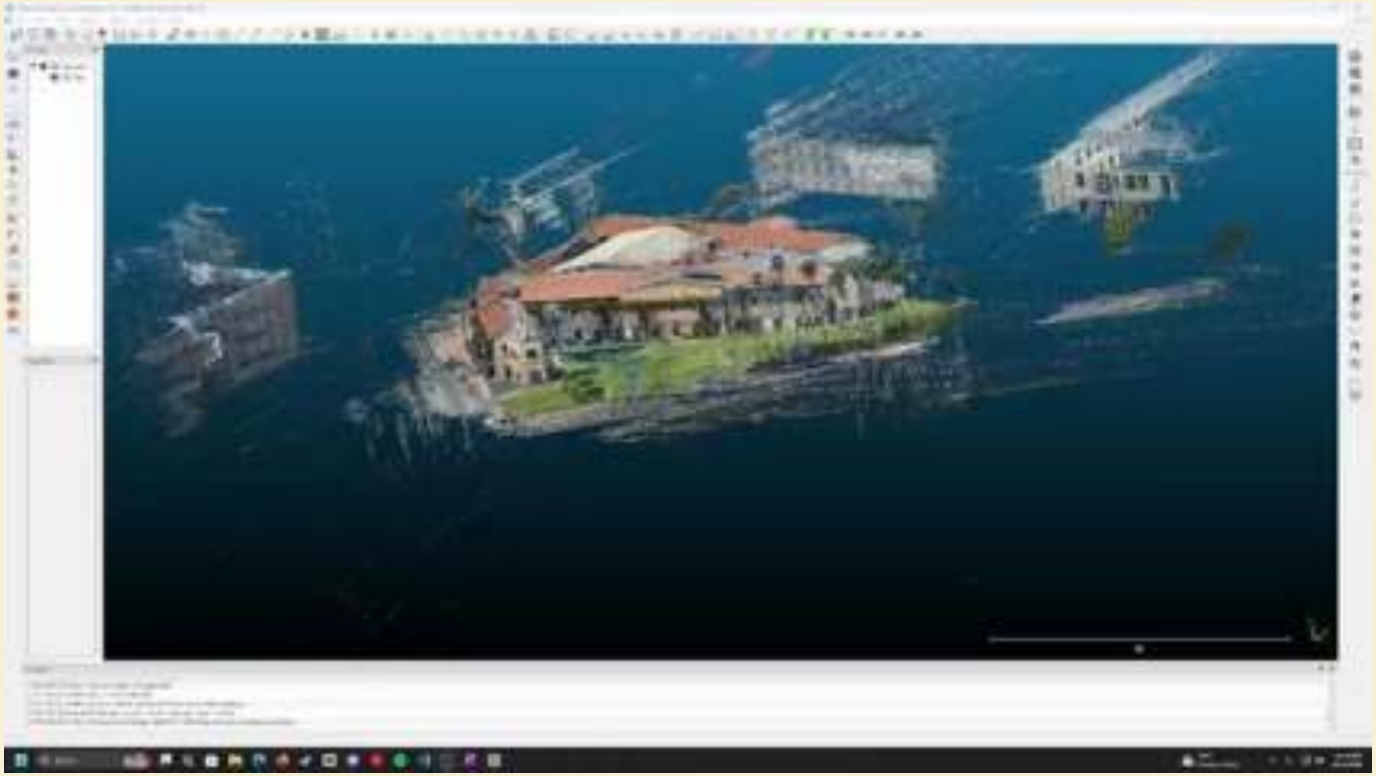
# Sparse Reconstructions

KITP



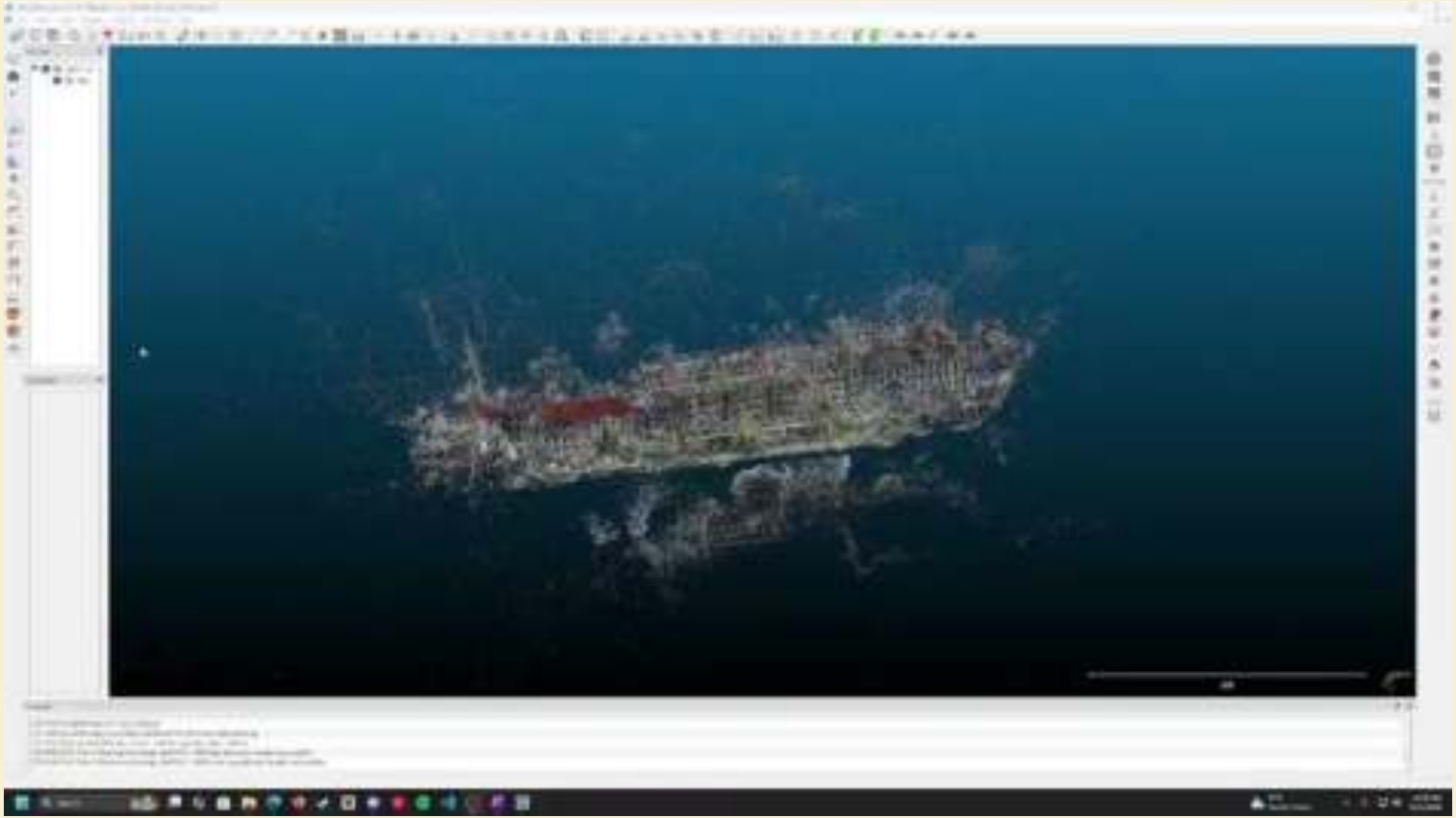
# Dense Reconstructions

**KITP**



# Sparse Reconstructions

**CHEM/PSNB**



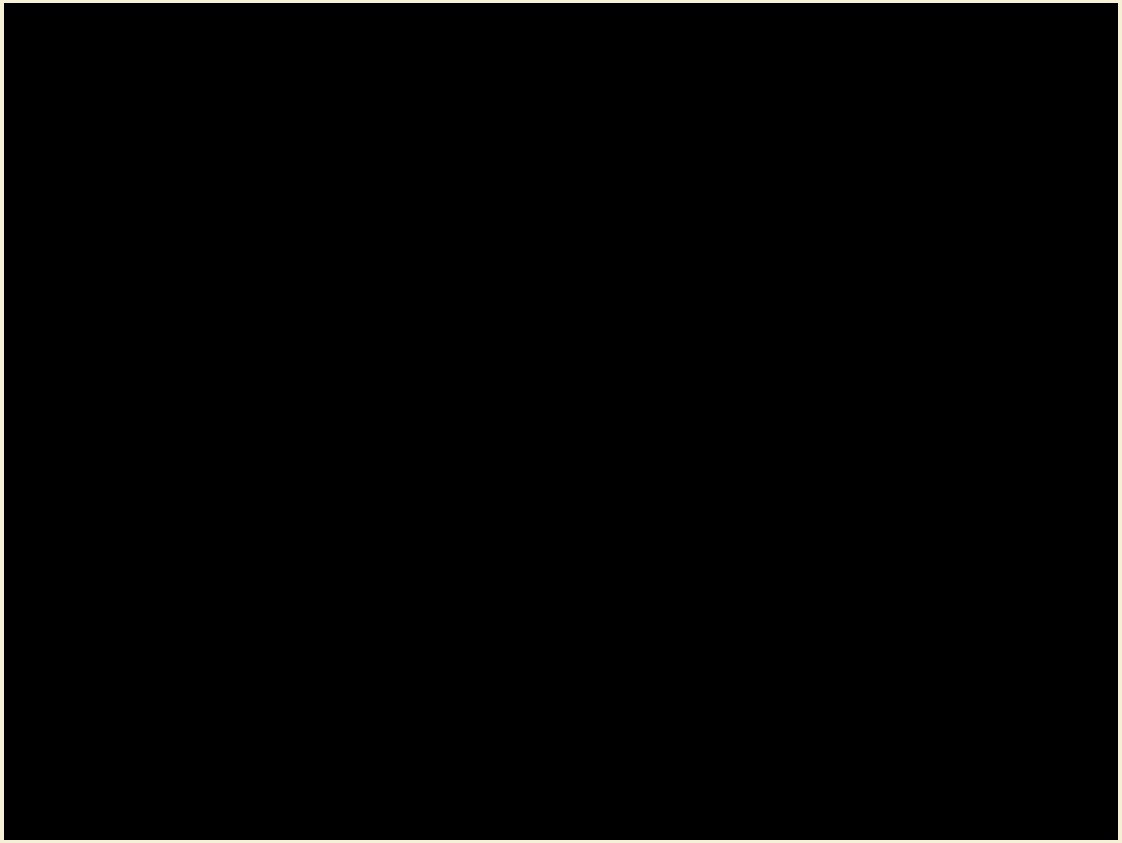
# Dense Reconstructions

**CHEM/PSNB**

# Merged MRL + DJI drone + HFH Dense

# Dense Reconstructions

**Drone - Lagoon Flight 1**



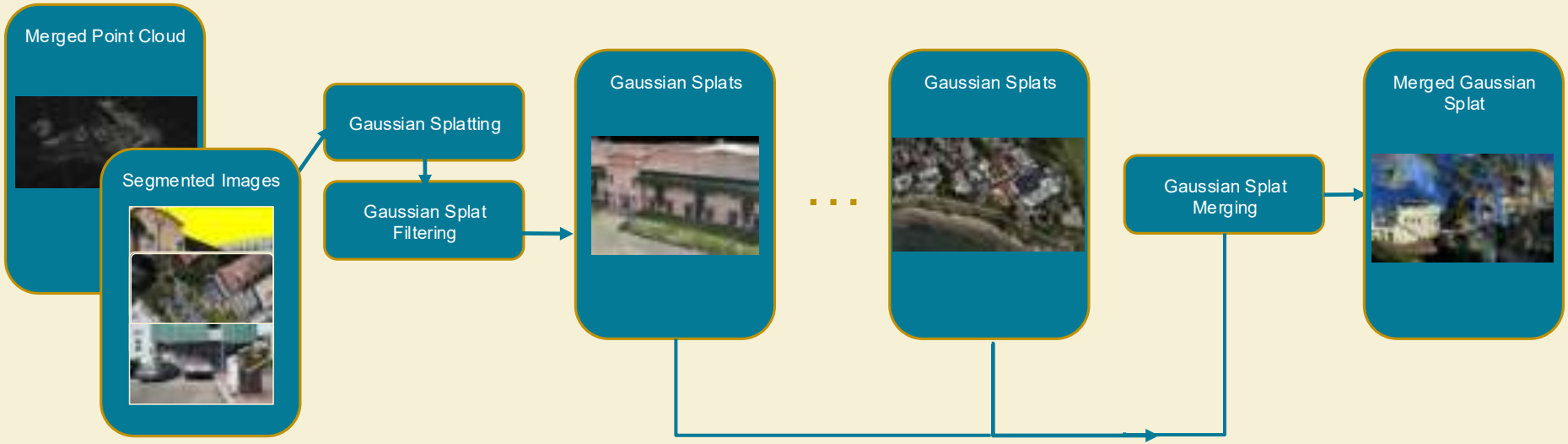
# Dense Reconstructions

**Drone - Lagoon Flight 2**

# Gaussian Splats

What are they, why are they useful, demo

# Pipeline Overview



# Gaussian Splatting Pipeline

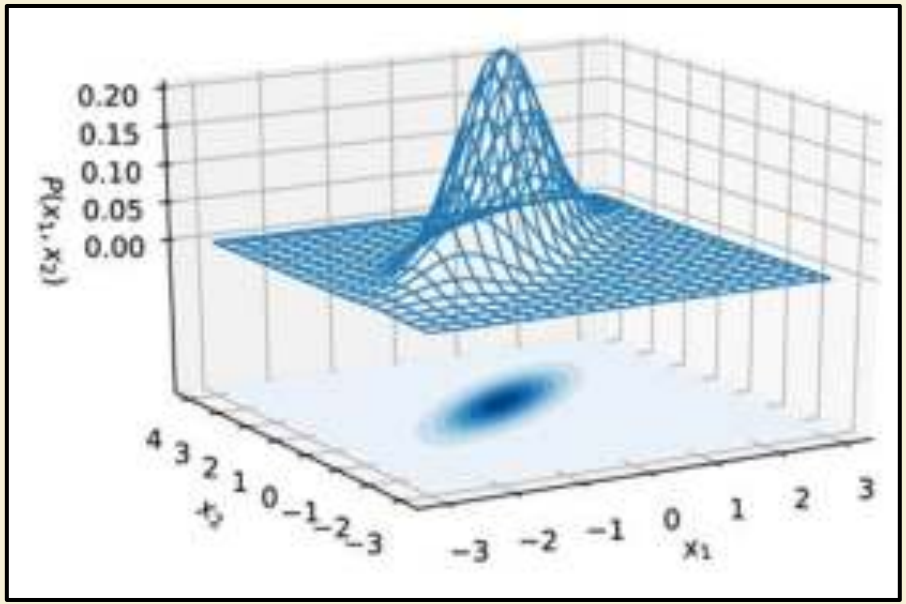
Sparse Point Cloud

Gaussian Splat Model

# Gaussian Splats

- 3D scene representation technique that uses millions of anisotropic Gaussians

- Each Gaussian has:
  - Position
  - Color
  - Opacity
  - Covariance



# Gaussian Splats v.s. Traditional MVS

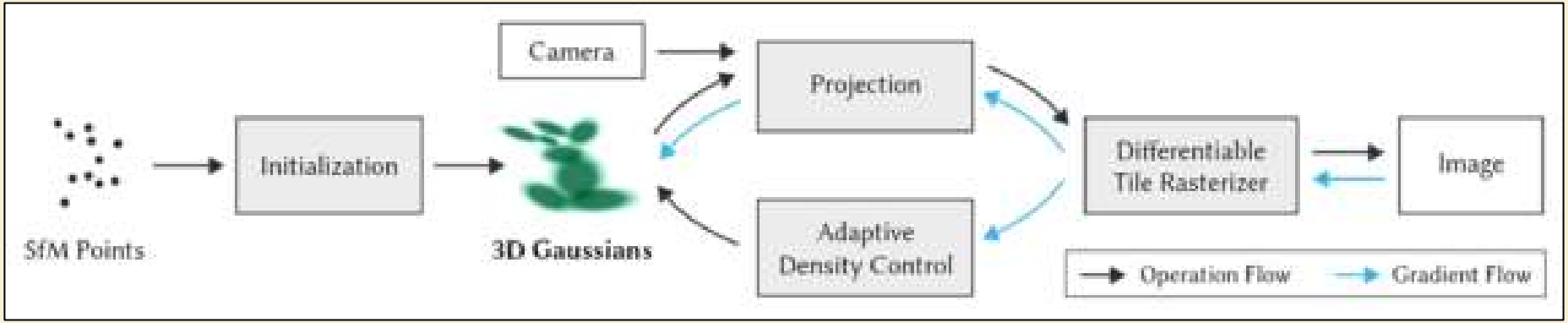
## Traditional MVS

- Rendered once
- Estimate depth & geometry, then texturize per image
- High accuracy
- Mesh output

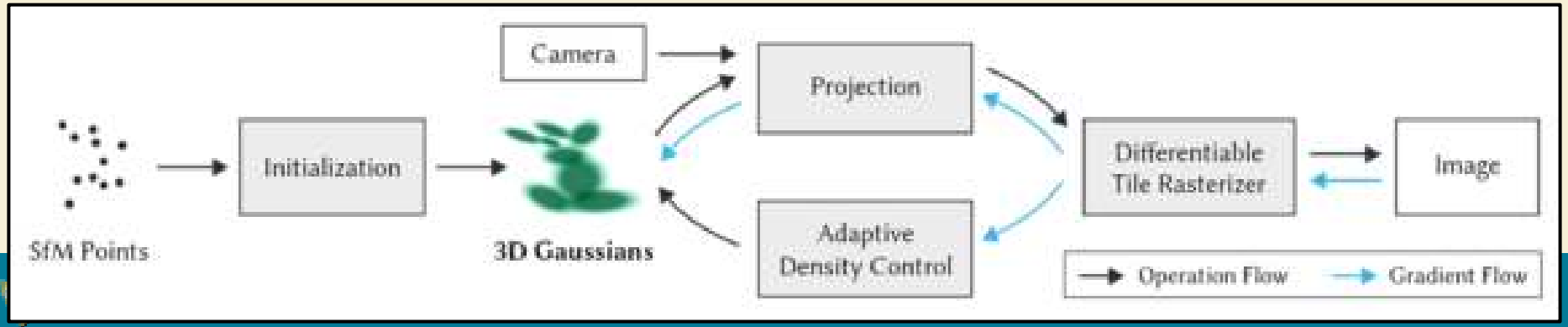
## Gaussian Splatting

- Can get real-time rendering
- Optimizing millions of 3D Gaussians per scene
- High visual quality
- Scales well to large datasets

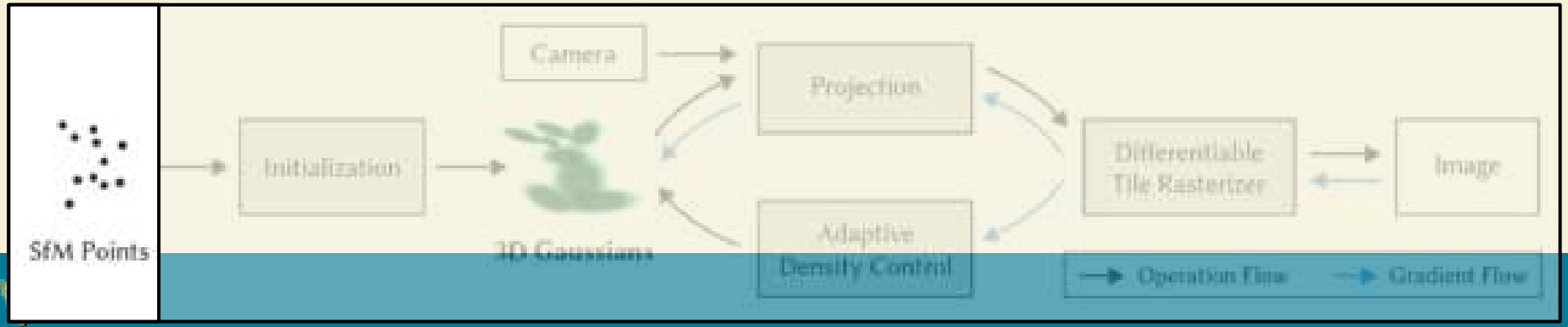
# Gaussian Splats Pipeline



# Gaussian Splats

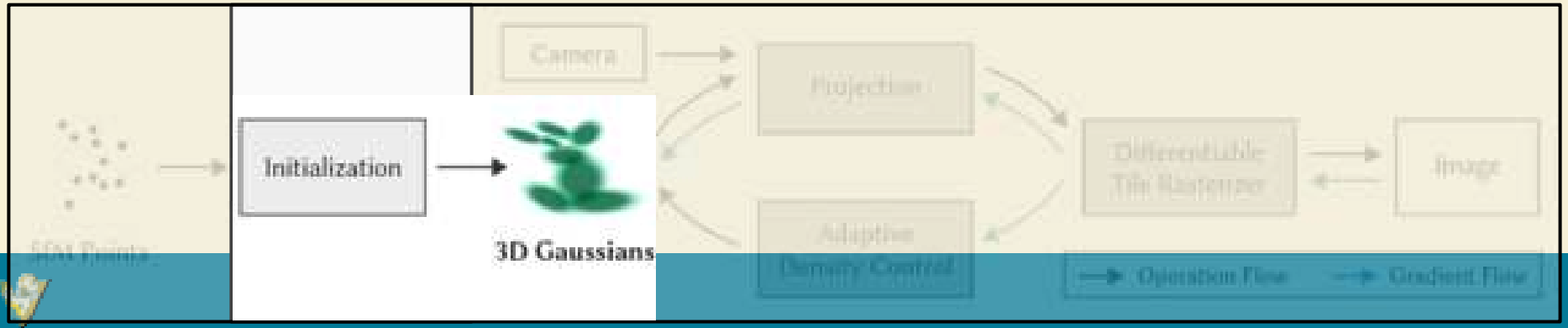


# Gaussian Splats



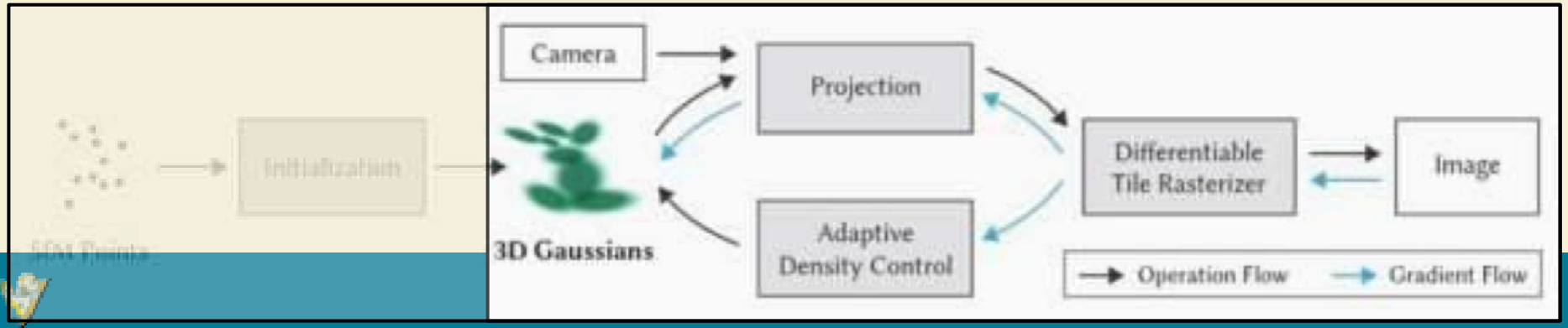
# Gaussian Splats

- Sparse points used to initialize a Gaussian-based scene representation
- Each point becomes 3D Gaussian distribution

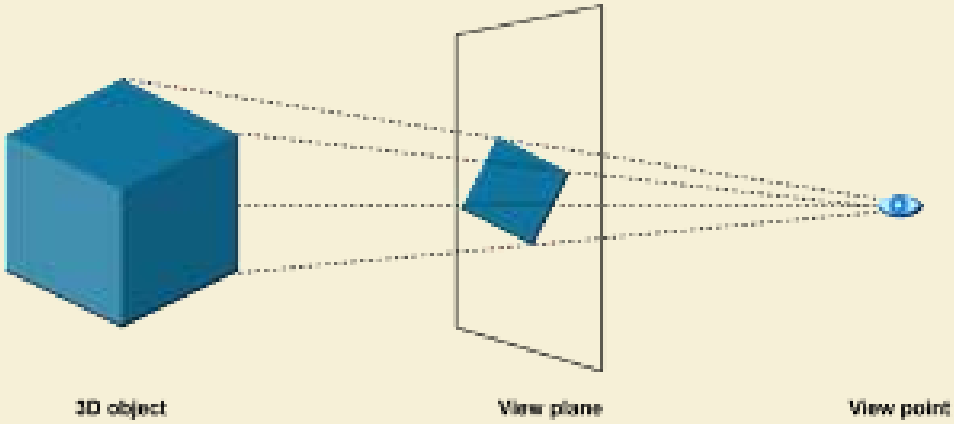


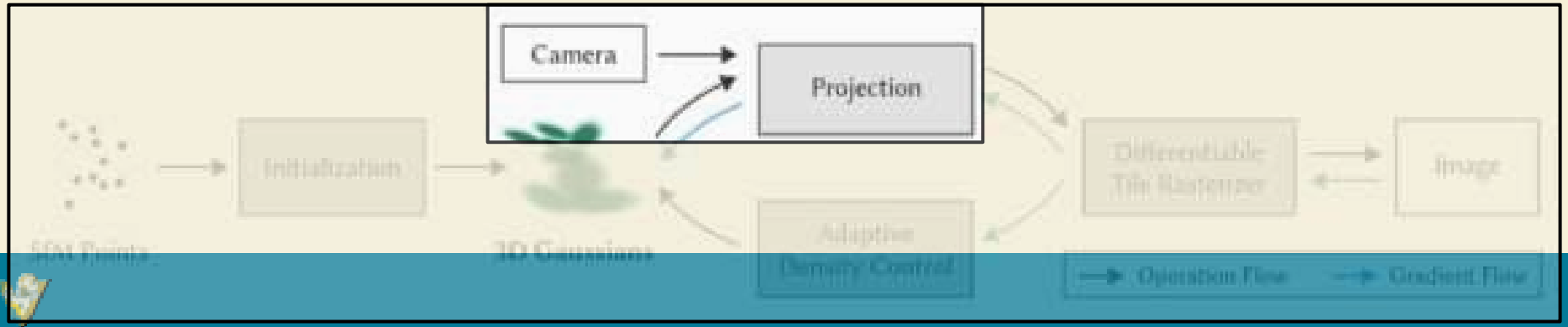
# Gaussian Splats

- Training/Optimizing

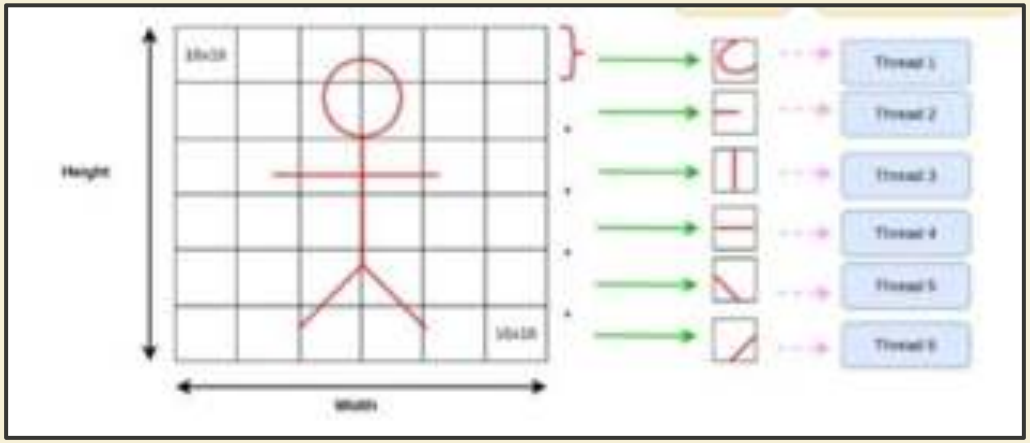


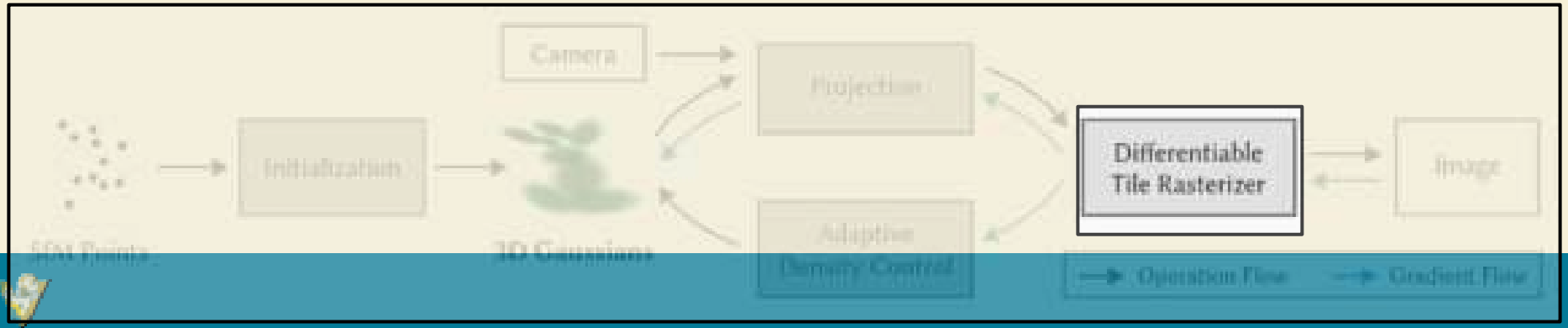
# Gaussian Splats



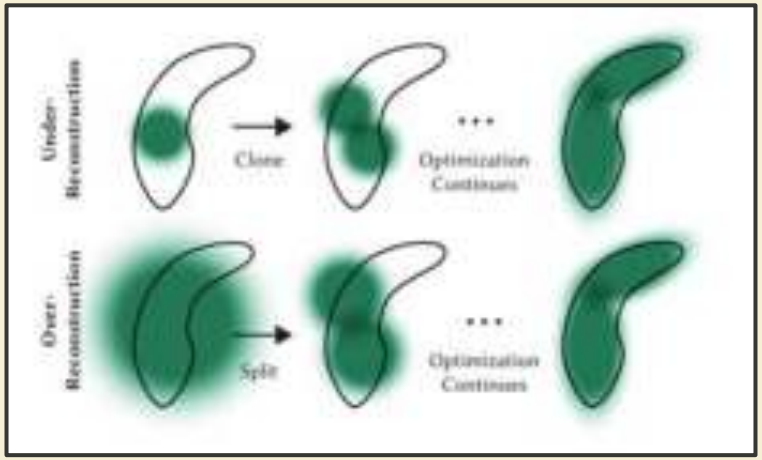



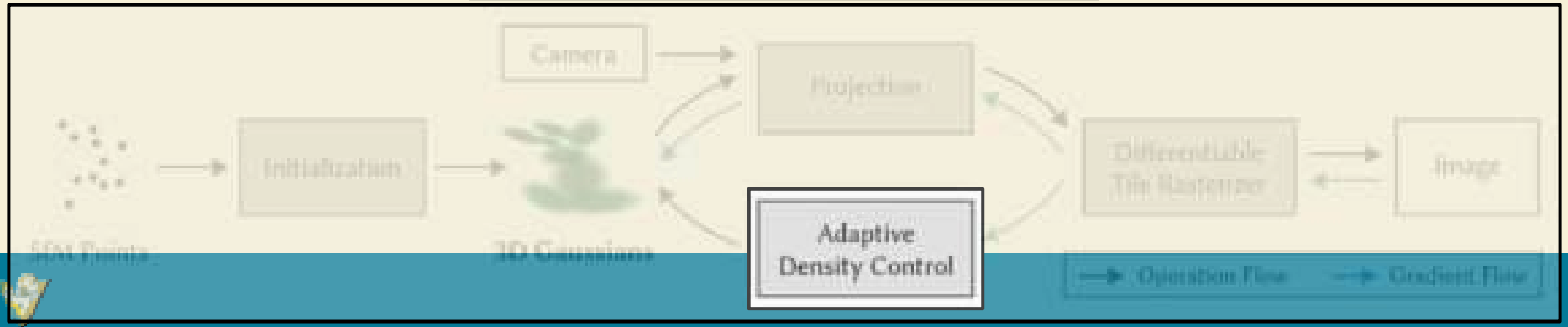
# Gaussian Splats



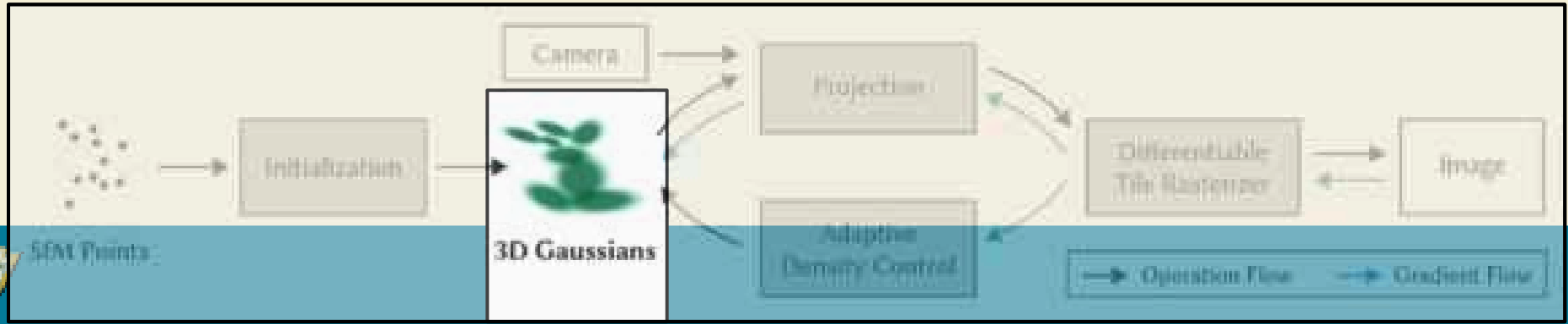



# Gaussian Splats



# Gaussian Splats



# Super Splat

- Load splat files directly in the browser,  clean up noisy splats, crop artifacts, and optimize scenes
- No local software or installation required

# Super Splat - Before and After

Before

After

# GS Example – HFH/MRL

Ground Truth

Render

# GS Example – HFH/MRL

hfh_mrl_dense_filtered by ari — splatter.app

# GS Example – KITP

Ground Truth

Render

# GS Example – KITP

Error

Ground Truth

Render

# GS Example – KITP

# GS Example – KITP

kitp_filtered by ari — splatter.app

# GS Example – LSB

Ground Truth

Render

# GS Example – LSB

lsb_mcmc by ari — splatter.app

# GS Example – Drone East Flight

Ground Truth

Render

# GS Example – Drone East Flight

drone_east_prelim by ari — splatter.app

# Gaussian Splat merging

- Each building is trained as its own separate Gaussian splat
- We merge them into one scene using their GPS coordinates to place each splat in the correct real-world position
- Are there problems with just stacking them together? **YES**
  - Duplicate Gaussians overlap at the boundaries between buildings
- Solution:
  - Voxel grid filter removes duplicate Gaussians at building seams

# Gaussian Splat Merge of MRL + HFH

# Gaussian Splat Merge of Drone East + MRL

# Next Steps

Gaussian Splatting pipeline, VR/browser integration

# GaussianPlant

How to distinguish between Branch and Leaf

# Why GaussianPlant

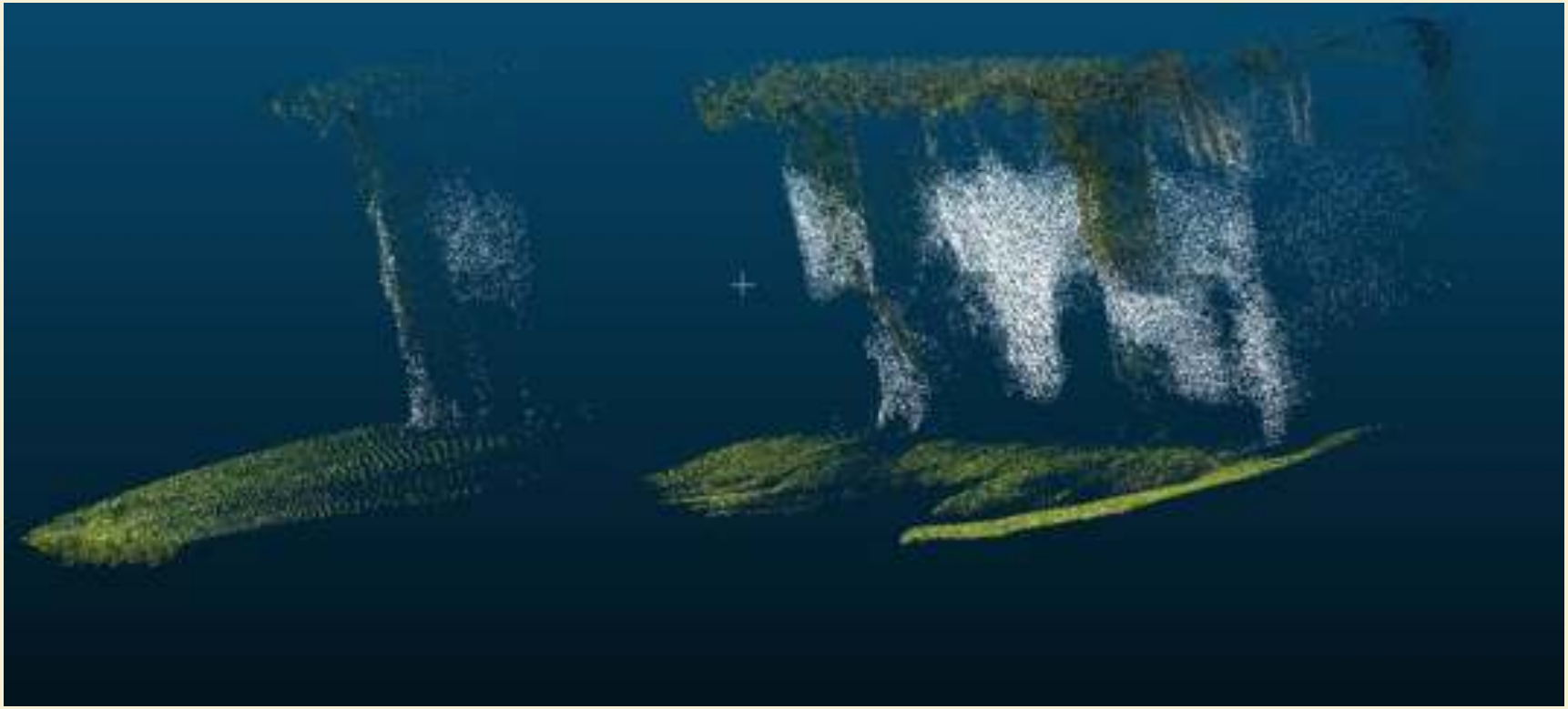
**Why campus 3D is hard？**

- People, cars, and trees move and create artifacts
- Thin structures are easy to miss, like branches, railings, wires
- We often need outputs that are not only pretty, but also measurable and editable

**Why we choose GaussianPlant?**

- Foreground masking and tracking to reduce dynamic noise
- Separate coarse structure from fine appearance for more stable geometry
- Graph based structure extraction to recover tree like topology
- Gaussian to mesh conversion for measurements and downstream tools

# Reconstruction of the KITP wall's ivy



# Reconstruction of the KITP pathway plants

# GaussianPlant + Gaussian Splat Render Example

Ground Truth Image

Result from Desplat

Rendered Result

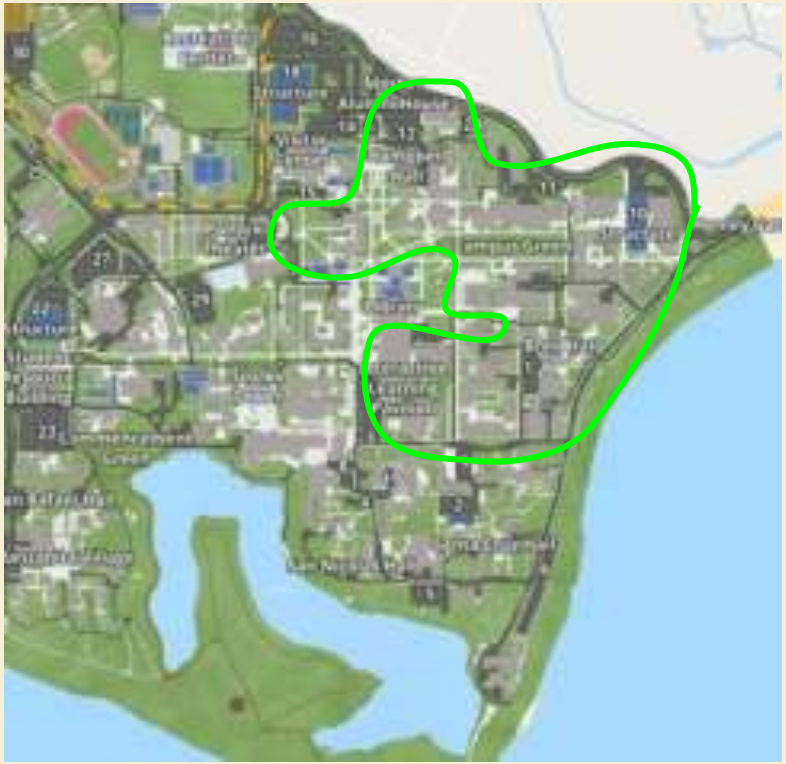
# GaussianPlant + Gaussian Splat Render Example

Ground Truth Image
Result from Desplat

Rendered Result

# Imaging

- Our coverage of campus using the EMLID did not cover all of UCSB
  - In the future, EMLID shots can be collected for the remainder of campus



# Game Engine Integration

- Import rendered Gaussian Splats into Unity
- Implement physics engine so users can “walk” around
- People can add assets and place “temporary” objects in our Gaussian Splat

# Timeline

F25: Basic image collection for testing

F25: DISK & RoMA for initial SfM pipeline

F25: Research other algorithms to include

W26: Using EMLID to gather proper images

W26: Finalize Image Preprocessing Pipeline

W26: Finalize Reconstruction Pipeline

W26: Start Gaussian Splatting Pipeline

S26: Finalize Gaussian Splatting Pipeline

Where we are now

S26: VR and browser implementation

# Acknowledgements


Chandrakanth Gudavalli

Dr. B.S. Manjunath

Charvi Mendiratta

Dr. Yogananda Isukapalli

Dr. Ilan Ben-Yaacov

Karolina Low

G3 Students (Undergrad Volunteers)

Chris Wimmel

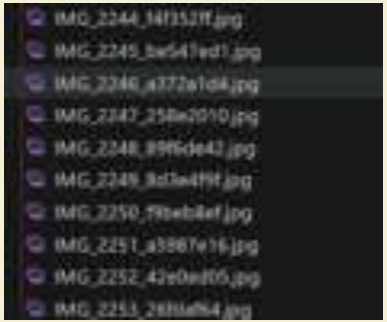
UCSB ECE MLLAB Systems

# Thank You

# Appendix

# Metadata Standardization

- The purpose of this is to have a standardized image set for a reliable metadata foundation
- Convert raw images into a single standard JPG format
- Generate a structured manifest.json to describe each image
- Preserve original metadata for downstream processing

# EMLID GPS Fusion

Establish consistent photo to GPS point connection as well as reference layer

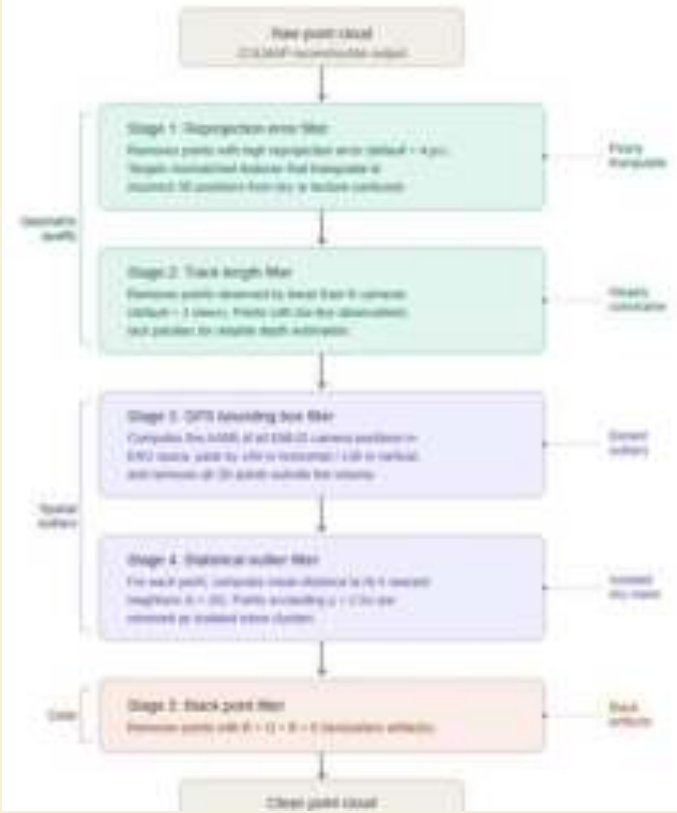
- Merge photo metadata and matched EMLID data via timestamps with time constraint
  - 5 second window for a match
  - Same day constraint
- Output structured XML file for 3D reconstruction
  - Match or no match the photo will show up in the XML file but under different tag

# Captured UCSB Aerial Imagery

# Point Cloud Filtering

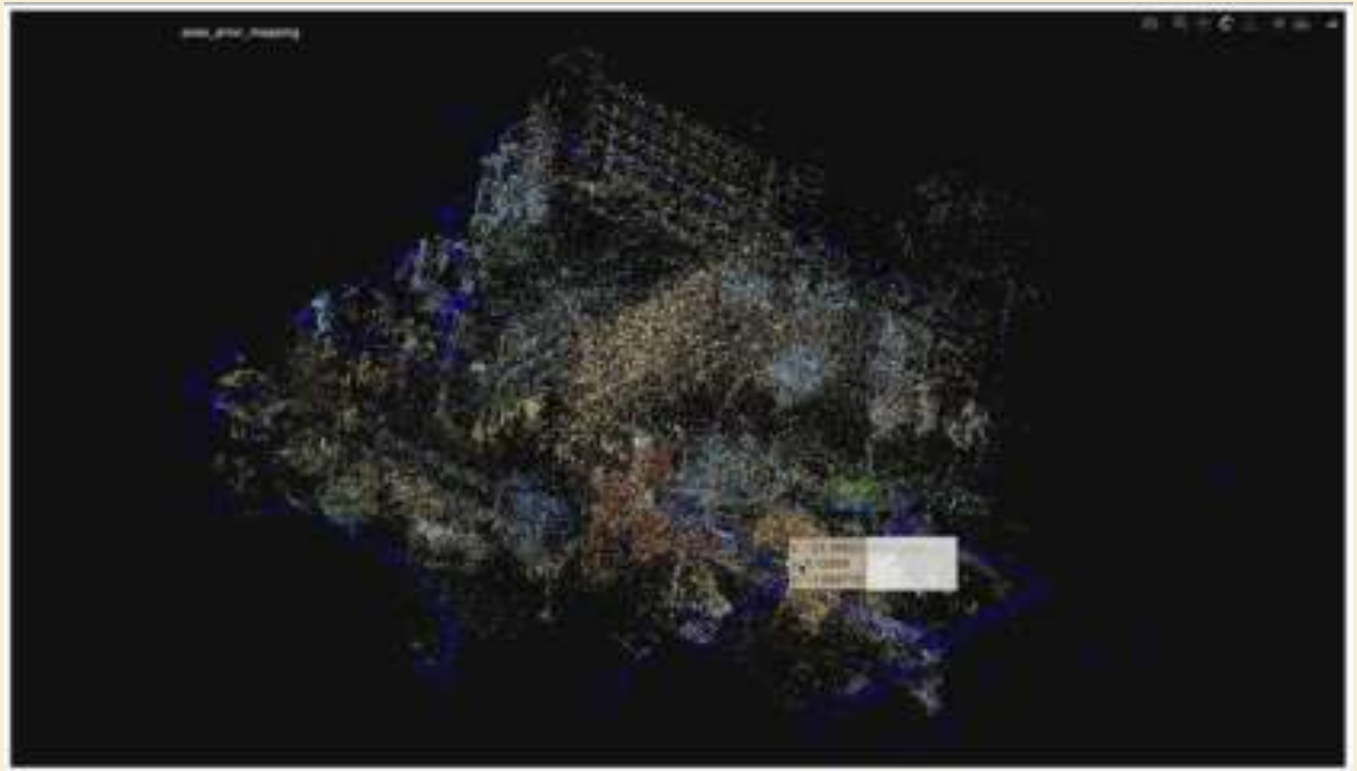
- Point clouds give outliers/errors due to many reasons
  - Snowballs into suboptimal Gaussian placements

- Solution: Filter points after 3D reconstruction



# Sparse Reconstructions

**Ellison Hall**



# Dense Reconstructions

**Ellison Hall + North Hall**

# Gaussian Splats

gsplat:

- Optimized version of 3DGS
- Fast, good for single buildings

# Gaussian Splats

gsplat:

- Optimized version of 3DGS
- Fast, good for single buildings

Octree-GS:

- Hierarchical octree of Gaussians
- Good for large scenes
- Top scores on GS benchmarks

# Gaussian Splats

gsplat:

- Optimized version of 3DGS
- Fast, good for single buildings

Octree-GS:

- Hierarchical octree of Gaussians
- Good for large scenes
- Top scores on GS benchmarks

deSplat:

- Static and distractor splats
- Easier to remove sky gaussians

# Gaussian Splats

gsplat:

- Optimized version of 3DGS
- Fast, good for single buildings

- Can run single buildings, then merge them together
- Ease-of-use allows for faster and easier ways to visualize the splats
- Well-documented and post-processing is easier

# Gaussian Splats

- Can run single buildings, then merge them together
- Useful for buildings with lots of transient objects (e.g. parking lots, bike racks, etc...)

deSplat:

- Static and distractor splats
- Easier to remove sky gaussians

# Gaussian Splats

gsplat:

- Optimized version of 3DGS
- Fast, good for single buildings

Octree-GS:

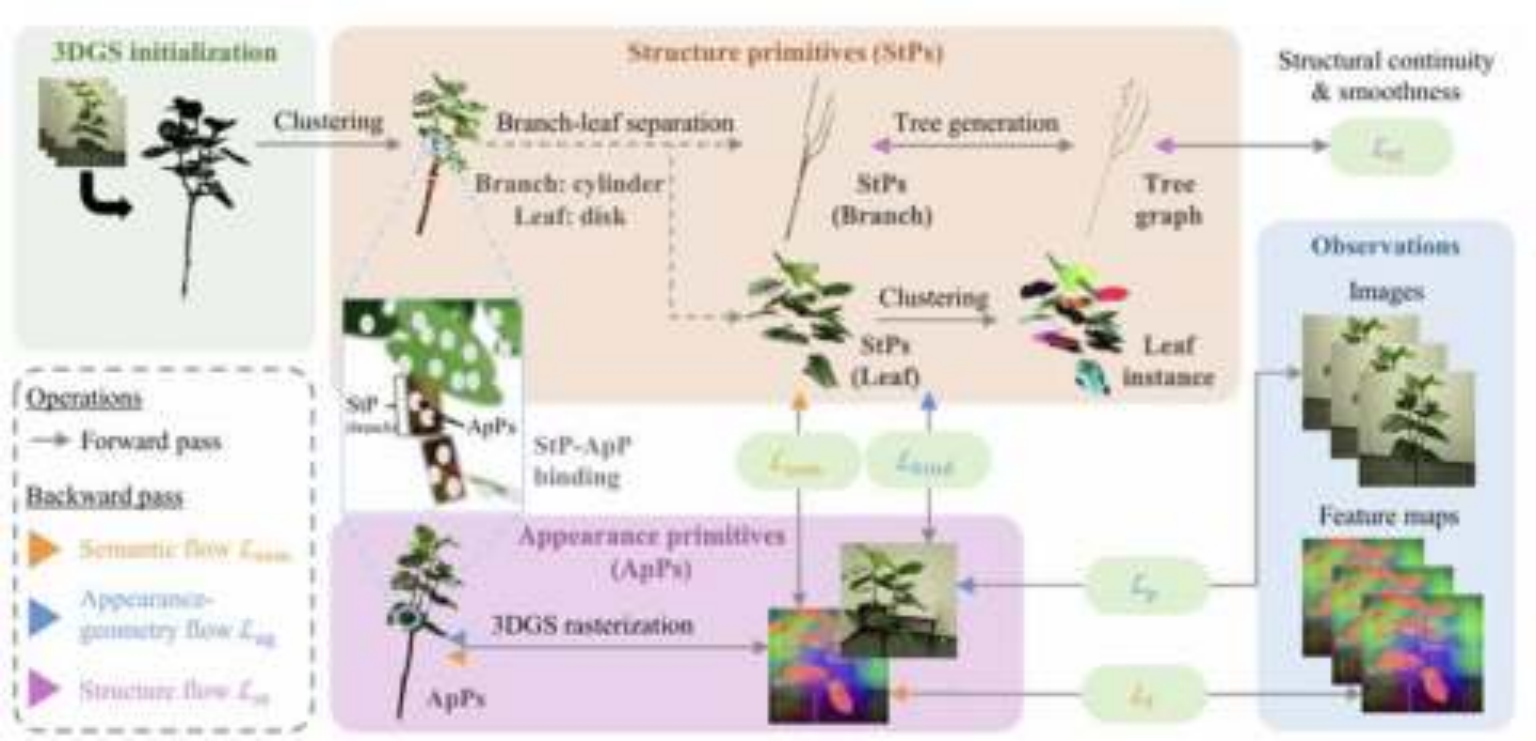
- Hierarchical octree of Gaussians
- Good for large scenes
- Top scores on GS benchmarks

deSplat:

- Static and distractor splats
- Easier to remove sky gaussians

# DeSplat example: KITP

# DeSplat example: KITP


3DGS initialization
Clustering
Structure primitives (StPs)
Branch-leaf separation
Branch: cylinder
Leaf: disk
Tree generation
StPs
(Branch)
Tree
graph
Structural continuity
& smoothness
Clustering
StPs
(Leaf)
Leaf
instance
Observations
Images
StP
ApPs
StP-ApP
binding
Feature maps
Appearance primitives
(ApPs)
3DGS rasterization
ApPs
Operations
Forward pass
Backward pass
Semantic flow
Appearance-
geometry flow
Structure flow

# GaussianPlant Implementation

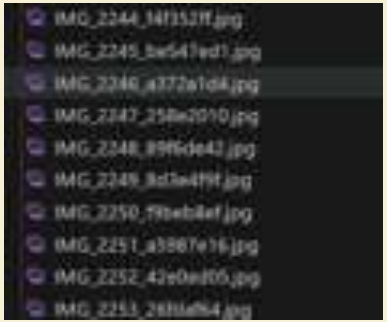
**GaussianPlant: structure-aligned 3DGS for plants**

- Goal: recover **appearance + internal structure**
- Output: realistic rendering
- Output: branch structure (tree-like)
- Output: leaf instances (leaf-by-leaf)
- Designed for plant-oriented tasks

1. Train a standard 3DGS model from multi-view images
2. Cluster Gaussians into groups
3. Use **length vs. width** to decide “branch-like” vs “leaf-like”
4. Build cylinders/disks file, branch file and leaf file
5. Jointly optimize appearance, semantics, and structure graph

# Metadata Standardization

- The purpose of this is to have a standardized image set for a reliable metadata foundation
- Convert raw images into a single standard JPG format
- Generate a structured manifest.json to describe each image
- Preserve original metadata for downstream processing

# EMLID GPS Fusion

Establish consistent photo to GPS point connection as well as reference layer

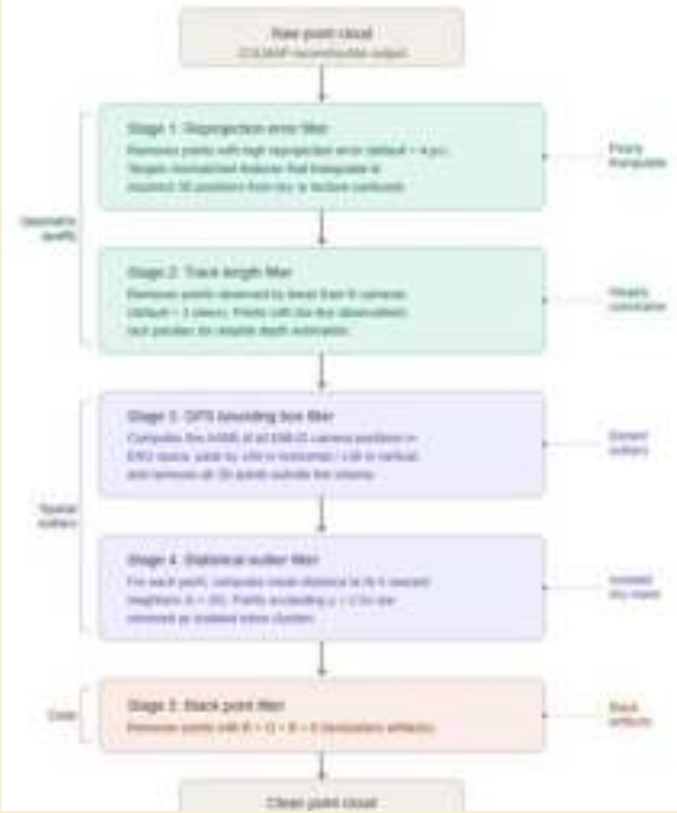
- Merge photo metadata and matched EMLID data via timestamps with time constraint
  - 5 second window for a match
  - Same day constraint
- Output structured XML file for 3D reconstruction
  - Match or no match the photo will show up in the XML file but under different tag

# Captured UCSB Aerial Imagery

# Point Cloud Filtering

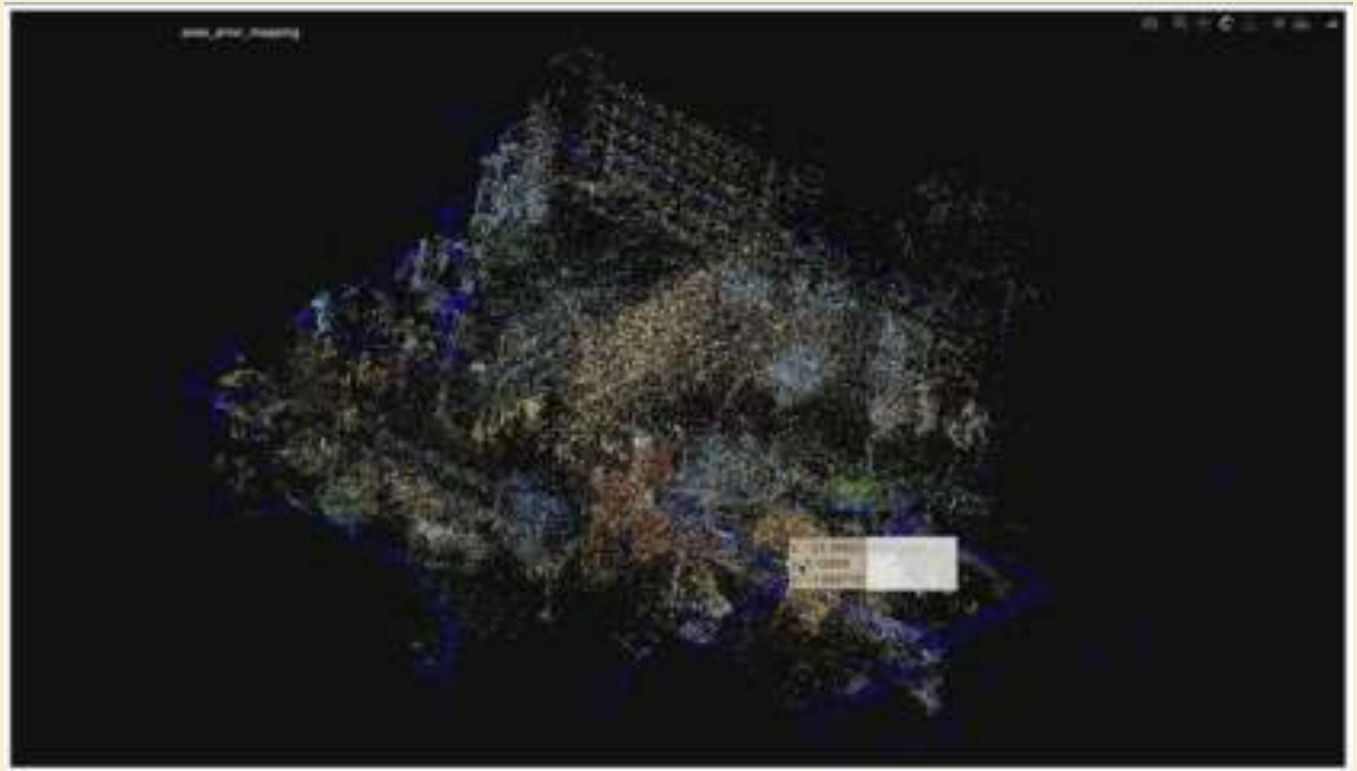
- Point clouds give outliers/errors due to many reasons
  - Snowballs into suboptimal Gaussian placements

- Solution: Filter points after 3D reconstruction



# Sparse Reconstructions

**Ellison Hall**



# Dense Reconstructions

**Ellison Hall + North Hall**

# Gaussian Splats

gsplat:

- Optimized version of 3DGS
- Fast, good for single buildings

# Gaussian Splats

gsplat:

- Optimized version of 3DGS
- Fast, good for single buildings

Octree-GS:

- Hierarchical octree of Gaussians
- Good for large scenes
- Top scores on GS benchmarks

# Gaussian Splats

gsplat:

- Optimized version of 3DGS
- Fast, good for single buildings

Octree-GS:

- Hierarchical octree of Gaussians
- Good for large scenes
- Top scores on GS benchmarks

deSplat:

- Static and distractor splats
- Easier to remove sky gaussians

# Gaussian Splats

gsplat:

- Optimized version of 3DGS
- Fast, good for single buildings

- Can run single buildings, then merge them together
- Ease-of-use allows for faster and easier ways to visualize the splats
- Well-documented and post-processing is easier

# Gaussian Splats

- Can run single buildings, then merge them together
- Useful for buildings with lots of transient objects (e.g. parking lots, bike racks, etc...)

deSplat:

- Static and distractor splats
- Easier to remove sky gaussians

# Gaussian Splats

gsplat:

- Optimized version of 3DGS
- Fast, good for single buildings

Octree-GS:

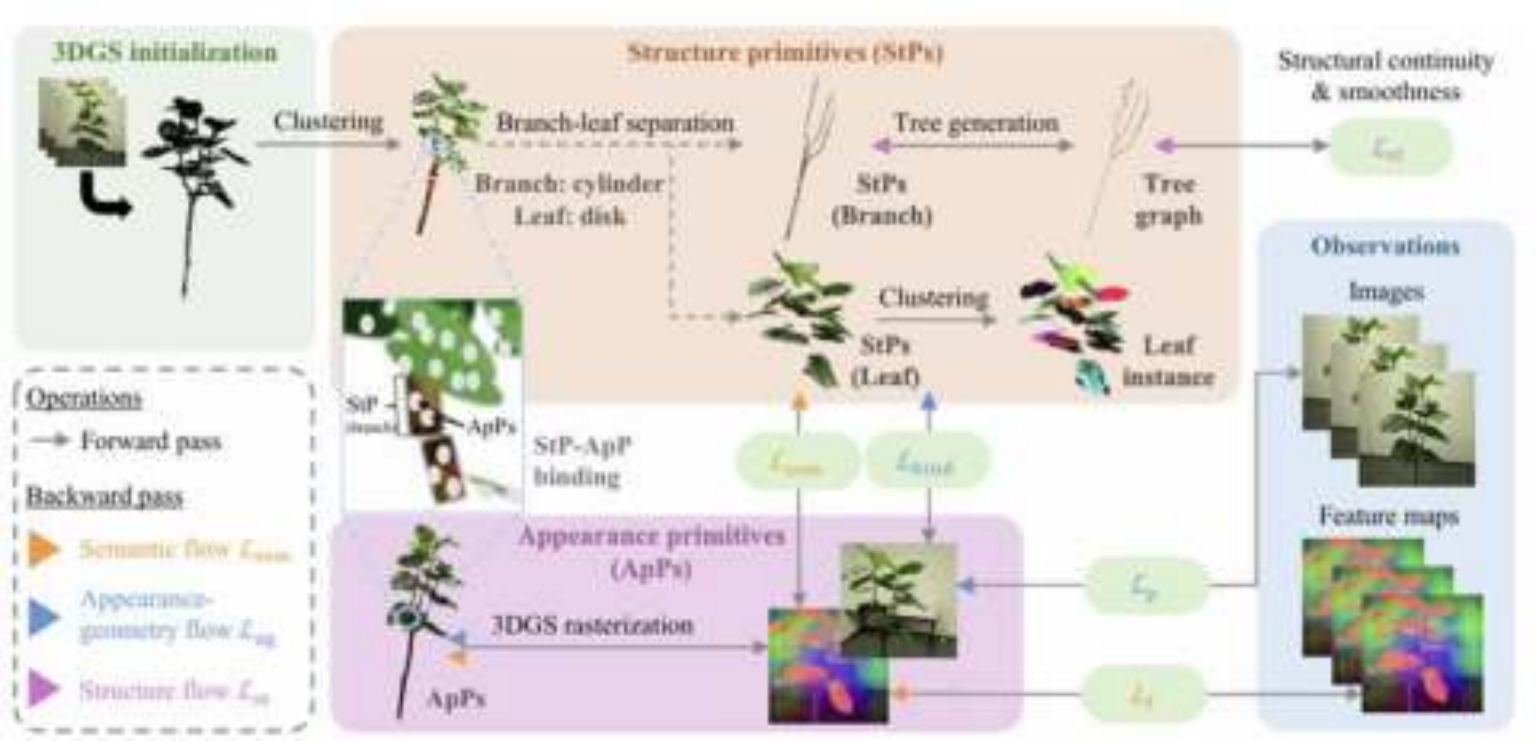
- Hierarchical octree of Gaussians
- Good for large scenes
- Top scores on GS benchmarks

deSplat:

- Static and distractor splats
- Easier to remove sky gaussians

# DeSplat example: KITP

# DeSplat example: KITP


3DGS initialization
Clustering
Structure primitives (StPs)
Branch-leaf separation
Branch: cylinder
Leaf: disk
Tree generation
StPs
(Branch)
Tree
graph
Structural continuity
& smoothness
Clustering
StPs
(Leaf)
Leaf
instance
Observations
Images
StP
ApPs
StP-ApP
binding
Feature maps
Appearance primitives
(ApPs)
3DGS rasterization
ApPs
Operations
Forward pass
Backward pass
Semantic flow
Appearance-
geometry flow
Structure flow

# GaussianPlant Implementation

**GaussianPlant: structure-aligned 3DGS for plants**

- Goal: recover **appearance + internal structure**
- Output: realistic rendering
- Output: branch structure (tree-like)
- Output: leaf instances (leaf-by-leaf)
- Designed for plant-oriented tasks

1. Train a standard 3DGS model from multi-view images
2. Cluster Gaussians into groups
3. Use **length vs. width** to decide “branch-like” vs “leaf-like”
4. Build cylinders/disks file, branch file and leaf file
5. Jointly optimize appearance, semantics, and structure graph

# Pipeline

Imaging / Pre-processing :

1. Raw Image Collection
2. EMLID GPS Fusion
3. YOLO Privacy Filtering
4. Sky Segmentation / Masking

3D Reconstruction :

1. Image Retrieval
2. Feature Detection & Extraction
3. Feature Matching
4. Spatial Matching
5. Structure-from-Motion
6. Multi-View Stereo
7. Point Cloud Filtering
8. Model Alignment
9. Point Cloud Merging

Gaussian Splatting :

1. Input: Point Cloud + Camera Poses
2. Gaussian Initialization
3. Training / Optimization
4. Building Splat Output
5. Voxel Grid Filtering
6. Splat Merging
7. Export / Render

# Pipeline

Imaging / Pre-processing :

1. Raw Image Collection
2. EMLID GPS Fusion
3. YOLO Privacy Filtering
4. Sky Segmentation / Masking

1. Photos from drones and ground cameras
2. Tag each photo with its exact GPS location
3. Blur faces, license plates, and people
4. Mark sky pixels so they're ignored during reconstruction

# Pipeline

Imaging / Pre-processing :
1. Raw Image Collection - Drones + ground cameras
2. EMLID GPS Fusion - Tag each photo with GPS location
3. YOLO Privacy Filtering - Blur faces, plates, people
4. Sky Segmentation / Masking - Mask sky pixels

3D Reconstruction :
1. Image Retrieval
2. Feature Detection & Extraction
3. Feature Matching
4. Spatial Matching
5. Structure-from-Motion
6. Multi-View Stereo
7. Point Cloud Filtering
8. Model Alignment
9. Point Cloud Merging

1. Find which photos overlap with each other
2. Identify distinctive points in each image
3. Link matching points across photos
4. Use GPS to skip pairs that are too far apart
5. Recover camera positions and build a sparse 3D map
6. Fill in the map with millions of points
7. Remove noise and outliers
8. Put the model to real-world GPS coordinates
9. Stitch building clouds into one campus map

# Pipeline

1. Each point becomes a 3D Gaussian
2. Refine color, shape, and opacity per Gaussian
3. Each building trained as its own splat
4. Remove duplicate Gaussians at seams
5. Place each building at GPS-correct position

Gaussian Splatting :

1. Input: Point Cloud + Camera Poses
2. Gaussian Initialization
3. Training / Optimization
4. Building Splat Output
5. Voxel Grid Filtering
6. Splat Merging
7. Export / Render