Overview of the GTSFM Pipeline (end-to-end)

0) What GTSFM is trying to do

GTSFM is a **global, distributed Structure-from-Motion pipeline**. Given a set of images, it estimates:

- **camera poses** (where each photo was taken),
- a **sparse 3D point cloud** of the scene,
- and then refines everything with bundle adjustment.

It's designed to scale to large scenes by using **global SfM formulations** and distributed execution (Dask), and by keeping the pipeline modular. ar5iv+1

1) Inputs & dataset config

Inputs:

- image set
    - Folder of raw images -> this is the actual data -> the pipeline will reconstruct form this images
    - Detect keypoints -> interesting spots in an image that are easy to recognize again from another angle or another photo (corners, sharp edges, textured dots, patterns, places your eye would naturally focus)
    - Match features between pairs of images
    - Estimate relative camera poses
    - Build the cluster graph
    - Reconstruct 3d points
- optional intrinsics/exif
    - This is meta data that describes focal length, sensor size, camera model, distortion parameters
    - GTSFM initializes each camera with a good guess -> enables more stable pose estimation
    - Reduces error in calibration
    - Makes BA converge faster and more accurately
- config YAML / CLI flags

  A configuration file + command-line overrides that control:

    - how clustering happens
    - when merges happen
    - which algorithm version to use

- BA options
- feature extraction parameters
- output directory structure

Examples of what config affects

- threshold for clustering → affects number of clusters
- matcher settings → affects how many points reconstruct
- bundle adjustment settings → affects quality of final model
- choice of pose estimator → affects stability of merges
- multi-view stereo on/off

Outputs you already use in your viewer:

- per-cluster COLMAP-style recon folders (`points3D.txt`, `cameras.txt`, `images.txt`)
- a merge structure (your tree of folders / "who merges into who")

2) Front-end: features + matching

This stage builds all the raw pairwise information.

2.1 Feature extraction

For each image:

- detect keypoints
- compute descriptors
  (think SuperPoint/SIFT/etc depending on config)

2.2 Image retrieval / candidate pairs

To avoid matching every pair:

- retrieve top-K likely overlapping pairs
- using global descriptors / vocab trees / retrieval nets

2.3 Feature matching

For each candidate pair:

- match descriptors
- filter matches using geometric checks (RANSAC)

Output: verified 2D-2D correspondences per image pair.

3) Data association → tracks

GTSFM links pairwise matches into **tracks**:

- A **track** = one real 3D point seen in multiple images.
- Built by unioning pairwise matches across the graph.

Output: multi-view correspondences (tracks).

4) Two-view geometry per pair

For each verified pair:

- estimate relative pose
    - essential/fundamental matrix
    - relative rotation + translation (up to scale)

Output: a graph of relative camera motions.

5) Global SfM core (the “global” part)

This is the mathematical heart.

5.1 Rotation averaging

Solve for **all camera rotations jointly** from relative rotations.
This produces globally consistent orientations. ar5iv

5.2 Translation averaging

Given rotations + relative translations:

- solve for global camera **positions** (up to scale)
- again globally consistent. ar5iv

Output (critical for you):

- one global pose per camera
  This is the “true coordinate frame” your clusters should ultimately align to.

6) Triangulation

Using:

- global camera poses
- tracks

GTSFM triangulates 3D points for each track.

Output: initial sparse 3D point cloud.

7) Bundle Adjustment (BA)

Refines everything by minimizing reprojection error.

Typically:

1. **Global BA** over cameras + points
2. Outlier filtering
3. Possibly another BA pass

Output: final optimized sparse reconstruction.

8) Clustering + merge tree (for large scenes)

This is the part that generates your Gerrard Hall folders and the hierarchy you animate.

High-level idea:

1. **Partition the view graph into clusters**
    - each cluster is a subset of cameras/images that are strongly connected.
2. **Run local/global SfM inside each cluster**
    - produces sub-reconstructions (your `C_1_1`, `C_1_2`, etc.).
3. **Merge clusters hierarchically**
    - align child reconstructions into a parent frame,
    - run BA at merge points,
    - repeat until the final merged reconstruction. ar5iv

Outputs you already see:

- `C_* / ba_output / points3D.txt`
- `C_* / merged / points3D.txt`
- final `merged/points3D.txt`
- the implicit **merge tree hierarchy** (what your timeline plays).

This merge tree is not just UI — it reflects real pipeline decisions about which clusters merge at each step.

9) Where Frank's 3 features "attach" in this pipeline

Feature 1: Thin slab layout

You'll use **cluster metadata from Stage 8**:

- cluster IDs
- merge parent/child relations
- cluster centroids / extents (compute from `points3D.txt`)

Those become inputs to your **slab layout algorithm**.

Feature 2: Grabbable / rotatable clusters

You'll use the same Stage 8 outputs:

- each cluster folder → one rendered THREE.Group
- never touch global poses or point geometry
- only apply **local transforms in the viewer**

So pipeline-wise, this is purely a visualization layer on top of clusters GTSFM already produced.

Feature 3: Animated merges

Stage 8 gives you the discrete merge events.
Your job is to interpolate those events in the viewer:

- child cluster positions → parent position
- fade out children → fade in parent
- timing curves

Again: you're **animating the merge tree outputs**, not changing SfM math.

TL;DR mental model

Think of GTSFM as:

1. **Build view graph** (features/matches/tracks)
2. **Solve cameras globally** (rotation + translation averaging)
3. **Triangulate + BA**
4. **If big scene:** split into clusters, reconstruct each, then merge them in a tree.

5. **Your visualization = Stage 8 made interactive and cinematic.**

If you want, next we can turn this into your formal doc text (with headings exactly how you like), and then move to **Doc 3: Sys Design of Animations + Visualization Implementation** where we explicitly map:

- which files/objects in GTSFM produce **cluster graph + merge events**
- which exact artifacts your merge-tree.html should read to drive slab layout + motion.