**✅ System Setup Guide for COLMAP Photogrammetry**

**🖥️ System Requirements**

* OS: Ubuntu 20.04+ (or compatible Linux distro)
* GPU: NVIDIA with CUDA (recommended for faster MVS)
* RAM: ≥8 GB
* Python: 3.8+

**📦 1. Install System Dependencies**

bash

sudo apt update

sudo apt install -y git cmake build-essential \

libboost-program-options-dev libboost-filesystem-dev libboost-graph-dev libboost-system-dev \

libboost-test-dev libeigen3-dev libsuitesparse-dev libfreeimage-dev libgoogle-glog-dev \

libgflags-dev libglew-dev qtbase5-dev libqt5opengl5-dev libcgal-dev

**⚙️ 2. Install COLMAP (with GUI + CLI)**

bash

git clone https://github.com/colmap/colmap.git

cd colmap

mkdir build && cd build

cmake ..

make -j$(nproc)

sudo make install

**🐍 3. Create and Activate Python Virtual Environment (optional but recommended)**

bash

python3 -m venv colmap-venv

source colmap-venv/bin/activate

**📂 4. Clone Test Dataset**

bash

git clone https://gitlab.com/photogrammetry-test-sets/skull-cameramoves-flash-no-background.git

cd skull-cameramoves-flash-no-background

**🧪 5. Run Structure-from-Motion (SfM)**

bash

mkdir sparse

colmap feature\_extractor \

--database\_path database.db \

--image\_path ./

colmap exhaustive\_matcher \

--database\_path database.db

colmap mapper \

--database\_path database.db \

--image\_path ./ \

--output\_path sparse

**🌐 6. Undistort Images for Dense Reconstruction**

bash

mkdir dense

colmap image\_undistorter \

--image\_path ./ \

--input\_path sparse/0 \

--output\_path dense \

--output\_type COLMAP \

--max\_image\_size 2000

**🧊 7. Run Multi-View Stereo (MVS)**

bash

colmap patch\_match\_stereo \

--workspace\_path dense \

--workspace\_format COLMAP \

--PatchMatchStereo.geom\_consistency true

**🔷 8. Fuse Point Cloud**

bash

colmap stereo\_fusion \

--workspace\_path dense \

--workspace\_format COLMAP \

--input\_type geometric \

--output\_path dense/fused.ply

**🔳 9. Mesh Generation**

bash

colmap poisson\_mesher \

--input\_path dense/fused.ply \

--output\_path dense/mesh.ply



✅ If the mesh looks upside down:

* Either rotate in Blender
* Or align using:

bash

colmap model\_aligner \

--input\_path sparse/0 \

--output\_path aligned\_model \

--alignment\_type plane # or try 'custom', 'enu', etc.

**🧪 10. Testing Checklist**

| **✅ Step** | **Outcome** |
| --- | --- |
| Cloned dataset | ✔️ skull images ready |
| Feature extraction | ✔️ SIFT features detected |
| Matching | ✔️ Exhaustive matcher ran successfully |
| SfM Mapping | ✔️ Sparse point cloud created |
| Undistortion | ✔️ Images ready for MVS |
| MVS Patch Matching | ✔️ Dense point cloud estimated |
| Fusion | ✔️ fused.ply generated |
| Meshing | ✔️ mesh.ply generated (can rotate in Blender) |
| Model Alignment | ❗ Need to test best alignment\_type |

**🧱 COLMAP Photogrammetry Pipeline (Enhanced Version)**

**⚙️ 1. Feature Extraction**

bash

colmap feature\_extractor \

--database\_path database.db \

--image\_path images \

--ImageReader.single\_camera 1 \

--SiftExtraction.max\_num\_features 10000 \

--SiftExtraction.estimate\_affine\_shape 1 \

--SiftExtraction.domain\_size\_pooling 1

🧠 Enhancements:

* Enables affine shape and DSP (Domain Size Pooling) for more robust descriptors.
* Increases feature count for richer reconstruction.

**🔗 2. Feature Matching**

Choose one of the following depending on your dataset type:

**a. Exhaustive Matcher (for unordered photos):**

bash

colmap exhaustive\_matcher \

--database\_path database.db \

--SiftMatching.guided\_matching 1

**b. Sequential Matcher (for videos or ordered sequences):**

bash

colmap sequential\_matcher \

--database\_path database.db \

--SiftMatching.guided\_matching 1 \

--SiftMatching.overlap 5

🧠 Guided matching improves robustness by validating geometry.

**🗺️ 3. Sparse Reconstruction (Mapping)**

bash

mkdir sparse

colmap mapper \

--database\_path database.db \

--image\_path images \

--output\_path sparse \

--Mapper.ba\_refine\_principal\_point 1 \

--Mapper.init\_min\_tri\_angle 4.0

✅ Produces a sparse 3D model (point cloud + cameras).

* ba\_refine\_principal\_point = better calibration.
* init\_min\_tri\_angle = avoids degenerate triangulation.

**🧼 4. Undistort Images**

bash

mkdir dense

colmap image\_undistorter \

--image\_path images \

--input\_path sparse/0 \

--output\_path dense \

--output\_type COLMAP \

--max\_image\_size 2000

🔍 Undistorted images are required for multi-view stereo (MVS).

**🧠 5. Dense Stereo (Depth Estimation)**

bash

colmap patch\_match\_stereo \

--workspace\_path dense \

--PatchMatchStereo.geom\_consistency true \

--PatchMatchStereo.window\_radius 5 \

--PatchMatchStereo.min\_triangulation\_angle 2 \

--PatchMatchStereo.max\_image\_size 2000

🧠 Enhanced geometric consistency and higher window radius = better depth maps.

**💠 6. Depth Map Fusion (Point Cloud)**

bash

colmap stereo\_fusion \

--workspace\_path dense \

--output\_path dense/fused.ply \

--StereoFusion.min\_num\_consistent 3

🧪 Filters out noisy points and produces a denser, cleaner point cloud.

**🧱 7. Mesh Generation**

**a. Poisson Meshing (recommended for clean shapes like skulls)**

bash

colmap poisson\_mesher \

--input\_path dense/fused.ply \

--output\_path dense/mesh\_poisson.ply \

--PoissonMeshing.trim 10

**b. Delaunay Meshing (alternative if Poisson fails)**

bash

colmap delaunay\_mesher \

--input\_path sparse/0 \

--image\_path images \

--output\_path dense

**🧭 8. Model Alignment (Optional, fixes orientation or scale)**

bash

colmap model\_aligner \

--input\_path sparse/0 \

--output\_path aligned\_model \

--alignment\_type custom \

--ref\_images "skull\_image\_ref.jpg"

You can also use enu, enu-plane, or similarity depending on your reference type.

**🌐 9. Export to Other Formats**

bash

colmap model\_converter \

--input\_path aligned\_model \

--output\_path output.obj \

--output\_type OBJ

**✅ Output Summary:**

* sparse/: Sparse point cloud & camera poses.
* dense/fused.ply: High-density point cloud.
* dense/mesh\_poisson.ply: Mesh model.
* output.obj: Exported for Blender/CloudCompare.