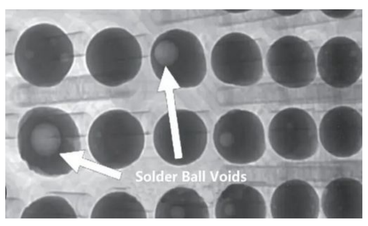
Gaussian Splatting for 3D Solder Ball Reconstruction

Andre Lim

Duke University, Department of Physics
National Taiwan University Department of Computer Science and Information Engineering, Digital Camera and Computer Vision Lab

Abstract

The Printed Circuit Board (PCB) is a critical component in many electronic products. Manufacturers must ensure the quality of every finished unit. This requires inspecting PCBs for potential defects such as open solder joints, short circuits, solder bridges, component misalignment, voids, and more.



Solder ball voids visible as light patches

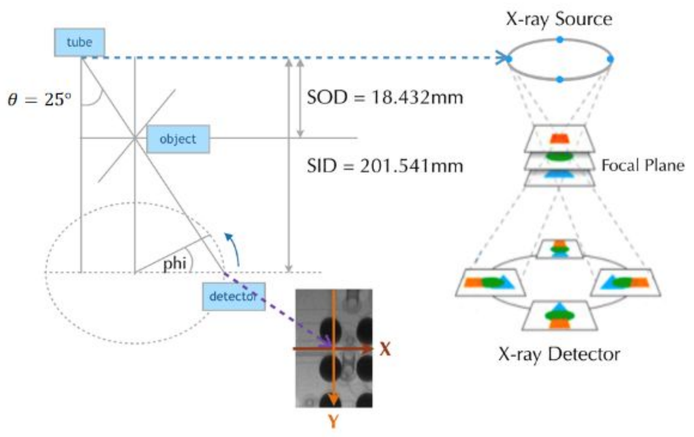
PCB designs are highly precise and many defects are not visible to the naked eye. Therefore, specialized inspection methods—such as Automated Optical Inspection (AOI), Solder Paste Inspection (SPI), and Automated X-ray Inspection (AXI)—are used. PCBs may be single-sided, double-sided, or multi-layered. For multi-layer PCBs, overlapping components make defect detection even more complex and challenging.

Many algorithms are proposed for tomographic reconstruction, such as Simultaneous Algebraic Reconstruction Technique (SART) and NeRF (Neural Radiance Field)

In this project, we decide to implement 3D Gaussian splatting to reconstruct the 3D PCB solder balls according to 128 projections of X-ray images of the solder balls, to inspect the PCB for voids in the solder balls. The X-ray images are obtained by taking several photographs from various angles. From these images, we can reconstruct and observe the inner structure of different levels in solder balls non-invasively.

3D Gaussian Splatting

Before performing the 3D reconstruction, the X-ray projection images are obtained through rotational laminography.



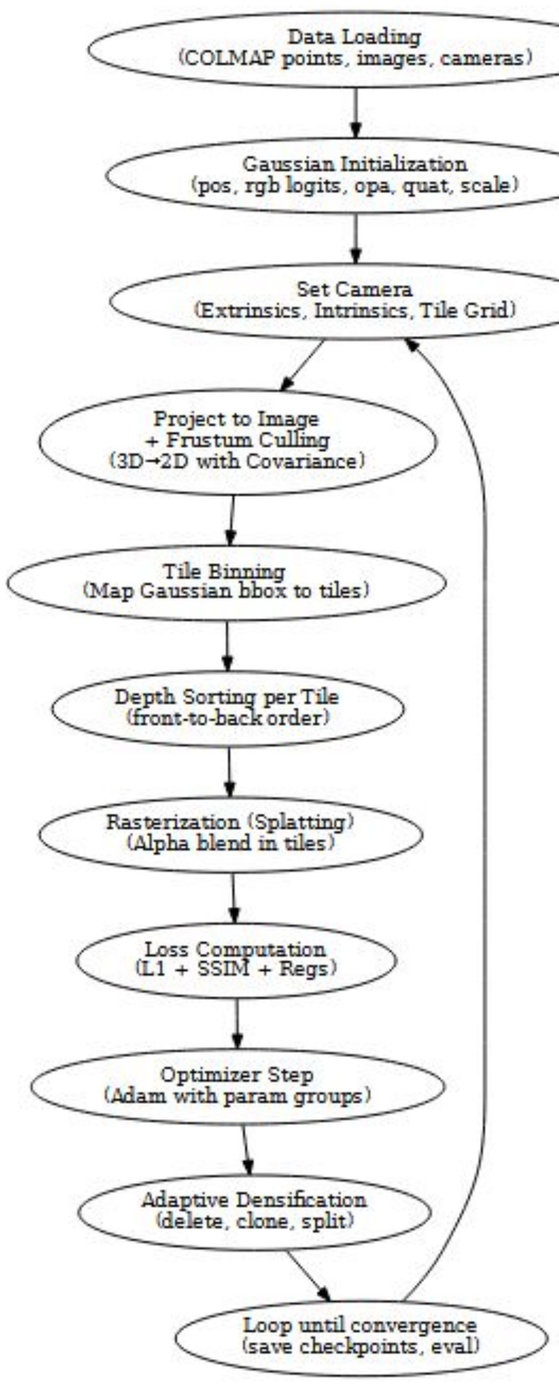
System setup. Theta is the angle between tube to detector and vertical line. SOD is Source to Object Distance (=18.432mm); SID is Source to Image Distance (=201.541mm); and phi is the angle of rotation.

Before performing the 3D reconstruction, the X-ray projection images are obtained through rotational laminography.

Through the relative motion between the source and the detector, the system can obtain a projection image including several objects belonging to different layers after rotating phi degrees (phi = 360° / #projection images).

From the projection images, the method represents the scene as a cloud of ellipsoidal 3D Gaussians derived from COLMAP sparse reconstructions.

Item	Our Environment
CPU	Intel® Core™ i7-8750H @ 2.20 GHz (6 cores, 12 threads)
GPU	NVIDIA® GeForce GTX 1070 (8 GB VRAM, Driver 32.0.15.5585)
Memory (RAM)	16 GB
OS	Windows 11 Home
Programming Language	Python 3.12.7
Main Libraries	PyTorch (CUDA-enabled), OpenCV, NumPy, Einops, PyKDTree



References

Kerbl, B., Kopanas, G., Leimkühler, T., & Drettakis, G. (2023). *3D Gaussian Splatting for Real-Time Radiance Field Rendering*. arXiv. https://doi.org/10.48550/arXiv.2308.04079

Tsan, T.-C., Tung, T.-C., Lin, B.-J., Fuh, C.-S., & Lee, Y.-H. (2018, October). *Solder Ball 3D Reconstruction with X-Ray Images Using Filtered Back Projection*. In Proceedings of TENCON 2018 – 2018 IEEE Region 10 Conference (pp. 2505–2510). IEEE

J. L. Schönberger and J.M. Frahm, *Structure-from-Motion Revisited*, 2016

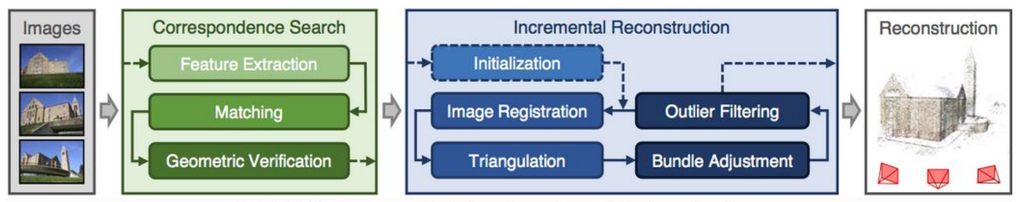
Main Methodology

COLMAP Reconstruction

COLMAP is a Structure-from-Motion (SfM) and Multi-View Stereo (MVS) tool that reconstructs camera poses and sparse 3D points from unordered images. It provides:

- Camera intrinsics (focal length, principal point, distortion model)
- Camera extrinsics (rotation & translation for each image)
- Sparse 3D points with RGB colors

In our pipeline, COLMAP output initializes the Gaussian cloud.



COLMAP's incremental Structure-from-Motion pipeline.

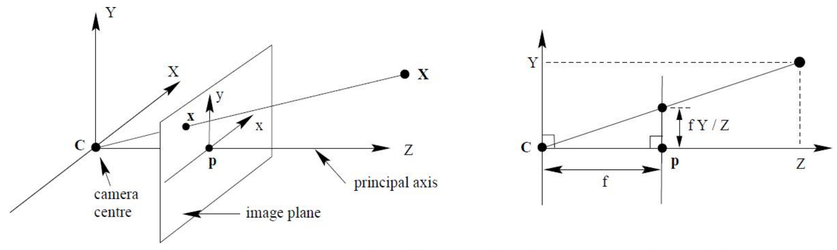
Initialization

The training process begins by parsing COLMAP binary output files — cameras.bin, images.bin, and points3D.bin — into structured Python objects. Each 3D point is converted into an initial Gaussian with parameters: center position, color stored as spherical harmonic coefficients, opacity, orientation quaternion, and scale vector.

Camera Coordinate Transformations

During training, each Gaussian must be rendered from the viewpoint of each training image. This begins with transforming its center from world coordinates to the camera coordinate frame.

The point is then projected to the image plane using the pinhole model. We then perform frustum culling.



A 3D Gaussian is defined by a mean and covariance matrix. To render it, we project this covariance into the 2D image plane via the Jacobian of the projection function. This projection yields an ellipse representing the Gaussian footprint in pixel space. The covariance matrix Σ of a 3D Gaussian is analogous to describing the configuration of an ellipsoid. Given a scaling matrix S and rotation matrix R, we can find the corresponding Σ: $= RSS^TR^T$. Given a viewing transformation W the covariance matrix Σ′ in camera coordinates is given as $\Sigma' = JW\,\Sigma\,W^TJ^T$

Tile Binning and Depth Sorting

For rendering efficiency, the image plane is divided into fixed-size tiles (e.g. 16 x 16 pixels). Each Gaussian's projected ellipse is approximated by a bounding box and assigned to all tiles it overlaps. Within each tile, Gaussians are depth-sorted in ascending order of Z, enabling correct front-to-back alpha blending.

Differentiable Rendering

Rendering proceeds tile by tile in row major order. For each pixel, Gaussians are evaluated in sorted order. The contribution of a Gaussian is computed as:

$$\alpha = o_i \cdot \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_{2D})^T\Sigma_{2D}^{-1}(\mathbf{x}-\mu_{2D})\right)$$

Color is composited using:

$$C_{out} \leftarrow C_{out} + (1-A)\cdot\alpha\cdot c_i \qquad A \leftarrow A + (1-A)\cdot\alpha$$

A is the accumulated alpha. This front-to-back formulation allows early termination once A approaches 1.

Loss Function

To train the Gaussians, we minimize a combination of L_1 loss and SSIM to capture both pixel-wise fidelity and perceptual quality. Regularization terms penalize excessively large scales and encourage low-opacity Gaussians that do not contribute to the image to fade away. We use the Adam optimizer, with separate learning rates for different parameters. Learning rate warmup and cosine decay schedules are applied to stabilize training.

Adaptive Densification

The adaptive Gaussian densification scheme. Top row (under-reconstruction): When small-scale geometry (black outline) is insufficiently covered, we clone the respective Gaussian. Bottom row (over-reconstruction): If small-scale geometry is represented by one large splat, we split it in two.