ABGICP: Photo-realistic Dense SLAM with Gaussian Splatting

https://spla-tam.github.io

Atticus Zhou, Atticus Zhou, Atticus Zhou, Atticus Zhou, Atticus Zhou, Atticus Zhou, Elias D. Striatum

Abstract

We present a dense simultaneous localization and mapping (SLAM) method that uses 3D Gaussians as a scene representation. Our approach enables interactive-time reconstruction and photo-realistic rendering from real-world single-camera RGBD videos. To this end, we propose a novel effective strategy for seeding new Gaussians for newly explored areas and their effective online optimization that is independent of the scene size and thus scalable to larger scenes. This is achieved by organizing the scene into sub-maps which are independently optimized and do not need to be kept in memory. We further accomplish frame-tomodel camera tracking by minimizing photometric and geometric losses between the input and rendered frames. The Gaussian representation allows for high-quality photo-realistic real-time rendering of real-world scenes. Evaluation on synthetic and real-world datasets demonstrates competitive or superior performance in mapping, tracking, and rendering compared to existing neural dense SLAM methods.

1 Introduction

in the image, typically given in pixels.f_x, f_y: Focal lengths of the camera in the x and y directions, respectively, typically given in pixels.d: Depth value at the pixel (u, v) after scaling, given in meters.$I_d(u, v)$: Depth value from the *original depth image* at pixel (u, v), typically given in the units used by the depth sensor.s: Scale factor to convert the depth values from the depth image units to meters.$\mathbf{p}$: 3D point in *camera coordinates*, represented as a vector[1].Keetha and colleagues mention something important [2]. **Concatenate coordinates to form the 3D points with a homogeneous coordinate:** The final array, `points`, is reshaped to $(-1, 4)$ to flatten the point cloud into a two-dimensional array where each row represents a 3D point in homogeneous coordinates.>)

2 Related Work

pixel to point cloud

Variables: u, v: Pixel coordinates in the image plane.c_x, c_y: Coordinates of the principal point (optical center) in the image, typically given in pixels.f_x, f_y: Focal lengths of the camera in the x and y directions, respectively, typically given in pixels.d: Depth value at the pixel (u, v) after scaling, given in meters.$I_d(u, v)$: Depth value from the *original depth image* at pixel (u, v), typically given in the units used by the depth sensor.s: Scale factor to convert the depth values from the depth image units to meters.$\mathbf{p}$: 3D point in *camera coordinates*, represented as a vector[1].Keetha and colleagues mention something important [2]. **Concatenate coordinates to form the 3D points with a homogeneous coordinate:** The final array, `points`, is reshaped to $(-1, 4)$ to flatten the point cloud into a two-dimensional array where each row represents a 3D point in homogeneous coordinates.

3 Method

Variables: u, v: Pixel coordinates in the image plane.c_x, c_y: Coordinates of the principal point (optical center) in the image, typically given in pixels.f_x, f_y: Focal lengths of the camera in the x and y directions, respectively, typically given in pixels.d: Depth value at the pixel (u, v) after scaling, given in meters.$I_d(u, v)$: Depth value from the *original depth image* at pixel (u, v), typically given in the units used by the depth sensor.s: Scale factor to convert the depth values from the depth image units to meters.$\mathbf{p}$: 3D point in *camera coordinates*, represented as a vector[1].Keetha and colleagues mention something important [2]. **Concatenate coordinates to form the 3D points with a homogeneous coordinate:** The final array, `points`, is reshaped to $(-1, 4)$ to flatten the point cloud into a two-dimensional array where each row represents a 3D point in homogeneous coordinates.

Variables: u, v: Pixel coordinates in the image plane.c_x, c_y: Coordinates of the principal point (optical center) in the image, typically given in pixels.f_x, f_y: Focal lengths of the camera in the x and y directions, respectively, typically given in pixels.d: Depth value at the pixel (u, v) after scaling, given in meters.$I_d(u, v)$: Depth value from the *original depth image* at pixel (u, v), typically given in the units used by the depth sensor.s: Scale factor to convert the depth values from the depth image units to meters.$\mathbf{p}$: 3D point in *camera coordinates*, represented as a vector[1].Keetha and colleagues mention something important [2]. **Concatenate coordinates to form the 3D points with a homogeneous coordinate:** The final array, `points`, is reshaped to $(-1, 4)$ to flatten the point cloud into a two-dimensional array where each row represents a 3D point in homogeneous coordinates.

Variables: u, v: Pixel coordinates in the image plane.c_x, c_y: Coordinates of the principal point (optical center) in the image, typically given in pixels.f_x, f_y: Focal lengths of the camera in the x and y directions, respectively, typically given in pixels.d: Depth value at the pixel (u, v) after scaling, given in meters.$I_d(u, v)$: Depth value from the *original depth image* at pixel (u, v), typically given in the units used by the depth

sensor.s: Scale factor to convert the depth values from the depth image units to meters.$\mathbf{p}$: 3D point in *camera coordinates*, represented as a vector[1].Keetha and colleagues mention something important [2]. **Concatenate coordinates to form the 3D points with a homogeneous coordinate:** The final array, `points`, is reshaped to $(-1, 4)$ to flatten the point cloud into a two-dimensional array where each row represents a 3D point in homogeneous coordinates.

Variables: u, v: Pixel coordinates in the image plane.c_x, c_y: Coordinates of the principal point (optical center) in the image, typically given in pixels.f_x, f_y: Focal lengths of the camera in the x and y directions, respectively, typically given in pixels.d: Depth value at the pixel (u, v) after scaling, given in meters.$I_d(u, v)$: Depth value from the *original depth image* at pixel (u, v), typically given in the units used by the depth sensor.s: Scale factor to convert the depth values from the depth image units to meters.$\mathbf{p}$: 3D point in *camera coordinates*, represented as a vector[1].Keetha and colleagues mention something important [2]. **Concatenate coordinates to form the 3D points with a homogeneous coordinate:** The final array, `points`, is reshaped to $(-1, 4)$ to flatten the point cloud into a two-dimensional array where each row represents a 3D point in homogeneous coordinates.

4 RESULTS & DISCUSSION

Variables: u, v: Pixel coordinates in the image plane.c_x, c_y: Coordinates of the principal point (optical center) in the image, typically given in pixels.f_x, f_y: Focal lengths of the camera in the x and y directions, respectively, typically given in pixels.d: Depth value at the pixel (u, v) after scaling, given in meters.$I_d(u, v)$: Depth value from the *original depth image* at pixel (u, v), typically given in the units used by the depth sensor.s: Scale factor to convert the depth values from the depth image units to meters.$\mathbf{p}$: 3D point in *camera coordinates*, represented as a vector[1].Keetha and colleagues mention something important [2]. **Concatenate coordinates to form the 3D points with a homogeneous coordinate:** The final array, `points`, is reshaped to $(-1, 4)$ to flatten the point cloud into a two-dimensional array where each row represents a 3D point in homogeneous coordinates.

[1] Z. Fan *et al.*, InstantSplat: Unbounded Sparse-view Pose-free Gaussian Splatting in 40 Seconds. Accessed: Jul. 10, 2024. [Online]. Available: `http://arxiv.org/abs/2403.20309`

[2] N. Keetha *et al.*, SplaTAM: Splat, Track & Map 3D Gaussians for Dense RGB-D SLAM. Accessed: Jun. 09, 2024. [Online]. Available: `http://arxiv.org/abs/2312.02126`